Deep Learning Methods for Visual Fault Diagnostics of Dental X-ray Systems

Harshit Agrawal

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Otaniemi 18.07.2018

Supervisor

Prof. Simo Särkkä

Advisor

D.Sc. (Tech.) Kalle Karhu

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Harshit Agrawal

Title Deep Learning Methods for Visual Fault Diagnostics of Dental X-ray Systems

Degree programme Life Science Technologies

Major Biomedical Engineering **Code of major** SCI3059

Supervisor Prof. Simo Särkkä

Advisor D.Sc. (Tech.) Kalle Karhu

Date 18.07.2018 **Number of pages** 64 **Language** English

Abstract

Dental X-ray systems go through rigorous quality assurance protocols following their production and assembly. The protocols include tests which address the image quality and find certain errors or artifacts that may be present in the images. Detecting faults from the images require human effort, experience, and time.

Recent advances in deep learning have proven them to be successful in image classification, object detection, machine translation. The applications of deep learning can be extended to fault detection in X-ray systems. This thesis work consists of surveying, applying, and developing state-of-art deep learning approaches for detection of visual faults or artifacts in the dental X-ray systems.

In this thesis, we have shown that deep learning methods can detect geometry and collimator artifacts from X-ray images efficiently and rapidly. This thesis is a precursor for further development of deep learning methods to include detection of wide range of faults and artifacts in X-ray systems to ease quality assurance, calibration, and device maintenance.

Keywords X-ray, CBCT imaging, deep learning, visual fault diagnostics, convolutional neural networks.

Preface

The machine learning and artificial intelligence revolution is moving forward faster than ever. Healthcare is not untouched from it. The latest machine learning advances have potential to redesign healthcare completely. Being a customer service engineer for healthcare machines previously in India, I passionately dream about a world where healthcare is accessible to everyone with the lowest possible time and cost.

This Master's thesis work was done at Planmeca Oy, Helsinki, Finland. I would like to thank Planmeca Oy for the resources and the inspiring environment. I would like to thank thesis supervisor Professor Simo Särkkä for providing feedback and insights for writing this thesis. I thank thesis advisor Dr. Kalle Karhu at Planmeca for his guidance, proofreading, and helping attitude throughout the thesis. I am also grateful to the colleagues at Planmeca for bearing me and helping me out from time to time.

Otaniemi, 18.07.2018

Harshit Agrawal

Contents

Abbreviations

Symbols

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
ADAM	Adaptive moment estimation
AE	Autoencoder
API	Application programming interface
CBCT	Cone-beam computed tomography
CNN	Convolutional neural network
CPU	Central processing unit
CT	Computed tomography
DVT	Digital volume tomography
FDD	Fault detection and diagnostics
FDK	Feldkamp-Davis-Kress reconstruction algorithm
FOV	Field of view
GPU	Graphics processing unit
keV	Kilo electronvolt
kVp	Peak kilovolt
LOO	Leave-one-out
MRI	Magnetic resonance imaging
QA	Quality assurance
R&D	Research and development
RBM	Restricted Boltzmann machine
ReLU	Rectified linear unit
SGD	Stochastic Gradient Descent

List of Figures

List of Tables

1 Introduction

Dentistry has seen tremendous advances over the past three decades. The need for more precise imaging and diagnostics tools has become essential (Shah et al., 2014). X-ray based techniques are playing a major role in the development of these tools. From simple intra-oral periapical X-rays to more advanced extra-oral imaging techniques like fan-shaped X-ray beam based computed tomography (CT) or cone-shaped X-ray beam based cone-beam computed tomography (CBCT) are being used in dental applications (Vandenberghe et al., 201).

Current state-of-art CBCT systems provide excellent high resolution and three dimensional images of oral bony anatomy. CBCT scans are also low cost in comparison to CT scans. However many imaging artifacts are inherently present in CBCT systems (Nagarajappa et al., 2015; Schulze et al., 2011). These artifacts may contribute to image degradation and lead to inaccurate or false diagnostics (Lee, 2008).

Before being delivered to the diagnostics facility, dental X-ray systems go through rigorous quality assurance (QA) protocols following their production and assembly. Many of these tests still require a lot of human intervention. The human operator has to go through many image volumes and decide the quality of the images based on his/her experience, which if automated, could save many work hours and improve the decision making accuracy. In addition, the image quality of the system may change with the wear and tear in various machine parts during the usage at the diagnostics facility. It will be beneficial to have a mechanism to monitor the degradation of image quality and its causes automatically. This would enhance customer experience and reduce number of visits of service engineer to the diagnostics facility.

Recent advances in machine learning methods for image classification and pattern recognition provide excellent precursor for the visual fault detection task. Notably deep learning and large convolutional neural network models (Alex et al., 2012; Simonyan and Zisserman, 2014) have recently demonstrated impressive image classification performance on the ImageNet large scale visual recognition challenge benchmark, an international competition for object detection and classification, consisting of 1.2 million everyday color images (Russakovsky et al., 2015).

The purpose of this thesis is to investigate applicability of state-of-art deep learning methods for the automatic detection of various visual faults in dental X-ray systems in order to ease tests at production or during service. This will provide better and improved quality services to the customer. This work mainly focuses on testing the classification of artificially induced geometry and collimator artifacts in the images generated using skull phantoms. We demonstrate that good accuracies can be obtained for visual faults detection using deep learning methods.

Prior work

Fault detection and diagnostics (FDD) have been an important aspect from shipping (Babaei et al., 2018) to aircraft Glavaski and Elgersma (2001) industries and from building systems (Katipamula and Brambley, 2005b) to medical equipment (López et al., 2009). The primary objective of an FDD system is early detection of faults

and diagnosis of their causes, enabling corrections before additional damage to the system or loss of service occurs as described by Katipamula and Brambley (2005a) for building systems. In the case of medical equipment, safety of patient becomes another important factor. It is better if the faults in medical equipment could be detected during the production phase itself as it minimizes the costs and safety concerns arising from the bug fixing and recalls in later phases (Alemzadeh et al., 2013). The faults may also occur later during the usage at customer sites due to wear and tear in the equipment. The dental X-ray systems are also tightly regulated to avoid radiation overdose to the patient or repeated exams (Hirvonen-Kari, 2013; Rout and Brown, 2013). Thus the fault detection or even fault prediction becomes a crucial aspect for medical equipment.

The FDD systems can be divided into two major categories: model based and data driven. In model based methods, domain knowledge of the system is incorporated to create a model and the measured values of some system parameters are compared with the model values and prediction is made (Isermann, 2005; Oppenheimer and Loparo, 2002). The complexity and noisy working conditions affect the development of such models and most of these models can not be updated with on-line measured data (Zhao et al., 2016b). On the other hand, data driven fault detection models learn a model based on observations of input and output data (Katipamula and Brambley, 2005a). These methods have become more attractive because of availability of big data, large computing power, and breakthrough of deep learning (Zhao et al., 2016b).

The faults in an equipment can be mechanical (Sheng et al., 2011) or software specific (Wallace and Kuhn, 2001, 1999). Depending upon the type of the fault and data available for the model training, different approaches are used. There have been many recent attempts to detect machine faults by applying deep learning methods (Zhao et al., 2016b). Sun et al. (2016) proposed a one layer autoencoder (AE, Goodfellow et al., 2016a) based neural architecture for the classification of faults in induction motor. They classified six types of the motor conditions using vibration signals acquired by an acceleration sensor. Lu et al. (2017) presented a detailed study of stacked denoising autoencoders with three hidden layers for fault diagnosis of rotary machinery components. The input features to those autoencoder models were raw sensory time-series. In Deutsch and He (2017), a restricted Boltzmann machine (RBM, Smolensky, 1986) based method was proposed for the prediction of remaining useful life of a bearing. Ding and He (2017) proposed a deep convolutional network where wavelet packet energy images were used as input for spindle bearing fault diagnosis. Recurrent neural network (RNN Rumelhart et al., 1986b) (RNN) models are used to handle sequential data with its ability to encode temporal information. The RNN models have been applied in various machine health monitoring problems recently (M.Yuan et al., 2016; Zhao et al., 2016a).

The progress in deep learning methods is changing the analysis of natural images (Dong et al., 2016) and videos (Wang et al., 2015). Similar examples are emerging in medical imaging. In the image related problems, convolutional neural networks have been used extensively. Litjens et al. (2017) have stated that out of the 47 papers published on medical image classification in 2015, 2016, and 2017, 36 were using CNNs, 5 AEs, and 6 were using RBMs. The application areas of these methods

were very diverse, ranging from brain magnetic resonance imaging (MRI) to retinal imaging and digital pathology to lung computed tomography (CT). Examples include classification of pulmonary tuberculosis from X-ray images (Lakhani and Sundaram, 2017), X-ray scattering image classification (Wang et al., 2017), differential diagnosis of benign and malignant breast nodules from ultrasound images (Cheng et al., 2016), and identifying diabetic retinopathy in retinal fundus photographs (Gulshan et al., 2016).

Similarly, CNNs have been used recently to detect artifacts in medical images. Meding et al. (2017) implemented a convolutional neural network for automatic detection of motion artifacts in MRI images. The network architecture consisted of two sequential convolutional layers interleaved with ReLU and pooling layers. Küstner et al. (2018) used a patch based approach for detection of motion artifacts from MRI images. They divided image slices to patches and fed the patches to the CNN. Zhang and Yu (2018) used convolutional neural network based approach for metal artifact reduction in CT scan images. Granholm (2018) tried transfer learning based CNNs for the detection of geometry and collimation artifacts from calibration images in CBCT machines.

Structure of the thesis

We start by describing the basics of X-ray production and main principles behind CBCT scanning. We then briefly explain motion and collimation artifacts in CBCT machine. We explain the central concepts in deep learning with focus on convolutional neural networks. Then we describe the neural network architectures and data collection methods used in our work. We also provide the code snippets wherever required. Finally we show the results obtained in our experiments and discuss how the deep learning methods can reduce the workload in the production and enhance accuracy for the visual fault diagnosis in dental X-ray systems.

2 Background

Since the revolutionary invention of X-rays in 1895 by Roentgen (Glasser, 1995), many diagnostics and imaging techniques have been developed such as digital X-ray, computed tomography (CT Scan), mammography, cath-lab. One such technique is cone-beam CT (CBCT), which is also known as digital volumetric tomography (DVT). Commercial development of dental CBCT technology first emerged in second half of the 1990s (Glasser, 1995; Mozzo et al., 1998; Arai et al., 1999; Scarfe and Farman, 2007; Scarfe et al., 2012). CBCT machines have been a widely used tool for several dental applications such as implant planning (implantology), endodontics, oral and maxillofacial surgery, and orthodontics (Scarfe et al., 2007, 2009; Mehr, 2013; Harrel Jr et al., 2018; Doyle et al., 2018; Alamari et al., 2012). Below we describe some basic principles and background physics behind X-ray and CBCT devices.

2.1 Production of X-rays

X-rays are generated in a glass tube containing two oppositely charged electrodes. Positively charged electrode is called anode and the negatively charged electrode is called cathode. Both electrodes are separated by a vacuum. The cathode consists of a filament which when supplied an electric current, gets heated and emits electrons. This process is called thermionic emission. A high voltage of the order of thousands of volts is applied between the two electrodes to accelerate the cathode electrons towards the anode. The area of the anode surface which receives the beam of electrons from the cathode is called focal spot. The focal spot consists of a high density material (e.g. tungsten). When the high speed electrons collide with the target material (focal spot), X-rays are produced (Bushberg et al., 2012a; Hsieh, 2015: pp. 33–42).

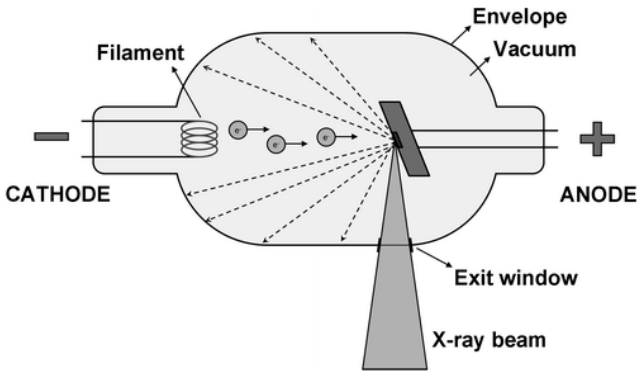

Figure 1: A simple schematic of X-ray tube (Pauwels et al., 2015). The cathode has negative and anode has positive charge. The filament in the cathode emits electrons, when heated. The focal spot makes a small portion of whole anode. The cathode electrons are accelerated towards anode. Whole process is carried out inside a glass envelope with vacuum.

All electron-focal spot collisions do not produce useful X-rays. Around 99% of the

input energy gets converted into heat. There can be three types of the interaction of the speeding cathode electrons with the target atoms as described in Hsieh (2015: p. 35). The high-speed electron travels partially around the nucleus due to the attraction between the positive nucleus and the negative electron. The sudden deceleration of the electron produces Bremsstrahlung radiation and energy of this Bremsstrahlung radiation depends on the kinetic energy of the incident electron which is given off in de-acceleration. As the amount of interaction increases, the resulting X-ray energy increases. This type of radiation covers the entire range of the X-ray energy spectrum. If the electron collides directly with a nucleus, its entire energy appears as Bremsstrahlung. The X-ray energy produced by this interaction represents the upper energy limit in the X-ray spectrum. For example, for an X-ray tube operating at 120 kVp, this interaction produces X-ray photons with 120-keV energy. This kind of interactions are very rare. The third kind of interaction occurs when a high-speed electron collides with one of the inner shell electrons of the target atom and frees the inner shell electron. When the hole is filled by an electron from an outer shell, characteristic radiation is emitted. The energy of the characteristic X-ray is the difference between the binding energies of two shells.

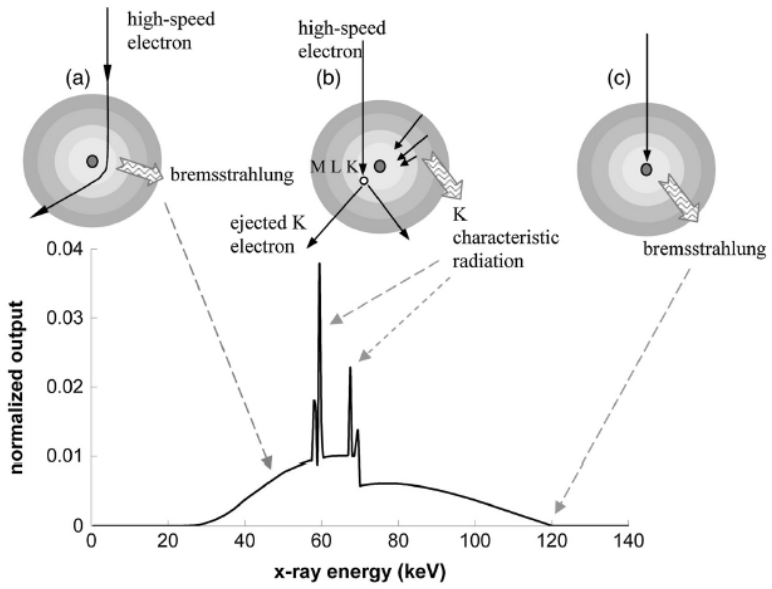

Figure 2: Electron interaction with a target and its relationship to the X-ray tube energy spectrum. (a) Bremsstrahlung radiation is produced when high-speed electrons are decelerated by the electric field of the target nuclei. (b) Characteristic radiation is generated when a high-speed electron interacts with a target electron and ejects it from its electronic shell. When outer-shell electrons fill in the vacant shell, characteristic X-rays are emitted. (c) A high-speed electron directly hits the nucleus and the entire kinetic energy gets converted to X-ray energy. For the X-ray spectrum shown in the figure, the target material is tungsten, and additional filtration is used to remove low-energy X-rays (Hsieh, 2015: p. 36).

2.2 Computed tomography

Computed tomography (CT) is a technique for imaging cross-sections of an object using a series of X-ray measurements taken from different angles around the object. CT uses simultaneously rotating detectors and X-ray tube. The development of the CT technology is often divided into four generations (Goldman, 2007; Bushberg et al., 2012b).

The first-generation of CT consists of X-ray tube rigidly linked to a detector located on the other side of the subject. The tube-detector sweeps horizontally which is called translation. During translation, the measurement is made at each location of the detector using a pencil beam X-ray. Then the system rotates at 1° increments to collect 180 views over 180° rotation.

The second-generation CT used multiple narrow beams of X-rays with multiple detectors and, as in the first generation, used rotate–translate motion. Number of required translations was reduced by a factor of 1/(number of detectors) (Goldman, 2007).

The third-generation CT eliminates translation and motion is purely rotation using slip-ring technology. The X-ray beam is fan-beam. This form of CT technology is most commonly used currently.

The fourth-generation CT consists detectors placed circularly and rapid measurements are taken as the tube sweeps across the field of view of the detector. Because of expensive detector arrays, the fourth-generation CT scans did not gain popularity.

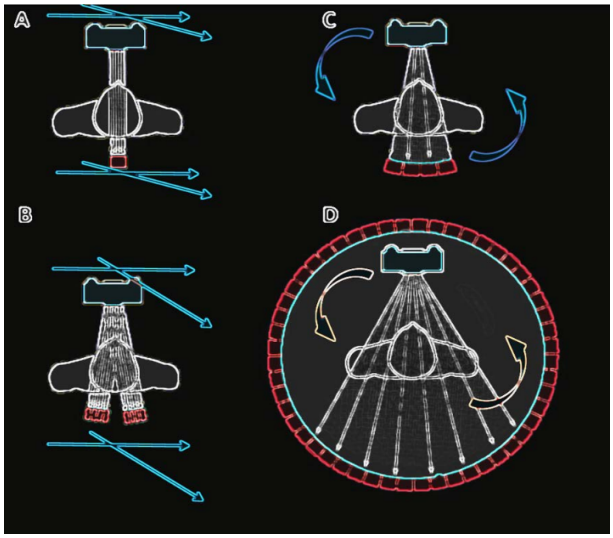

Figure 3: The first generation CT scanners (A) utilized a pencil beam fixed with a single detector moving simultaneously by translation-rotation motion. The second generation scanners (B) were similar to the first generation scanners but used multiple pencil beams coupled with multiple detectors and had a wider angle of rotation. The third generation scanners (C) utilized a fan-beam and an array of multiple detectors rotating simultaneously around the patient. The fourth generation scanners (D) used a wider fan-beam rotating around the patient and a stationary ring of multiple detectors (Mehr, 2013).

2.3 Cone-beam computed tomography

CBCT has revolutionised dental diagnosis by facilitating transition from 2D imaging to 3D imaging, which has added a lot of third party applications. The traditional CT scan systems used a fan-shaped beam and several detectors to acquire individual image slices of the field of view (FOV) and then stack the slices to obtain a 3D representation. CBCT uses a cone-shaped beam and a single detector where X-rays are directed through the middle of the area of interest onto the detector on opposite side. The X-ray source and the detector rotate around the patient and take multiple sequential projection images of the field of view (FOV). Only one rotation sequence of the gantry is necessary to acquire enough data for image reconstruction (Farman and Scarfe, 2018).

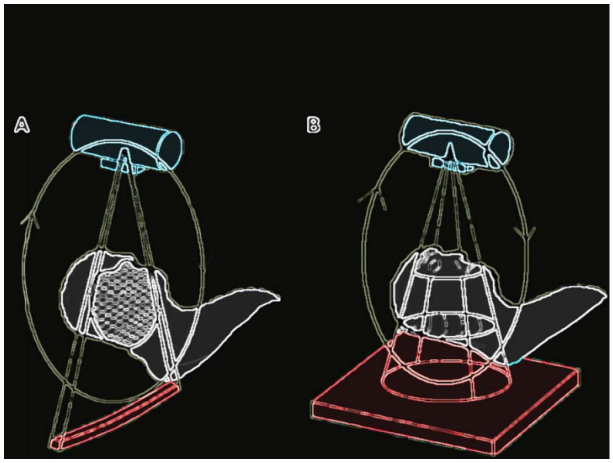

Figure 4: (A) Computed tomography utilizes a fan-shaped X-ray beam. (B) CBCT uses a cone-shaped X-ray beam. (Mehr, 2013).

CBCT scan has smaller size and lower cost in comparison to CT scan. Since CBCT aquires data from whole FOV, it has faster scan time which results in the reduction of image unsharpness caused by the translation of the patient, reduced image distortion due to internal patient movements, and increased X-ray tube efficiency (Gulsahi, 2011). However, because of larger FOV, large amounts of scattered radiation is produced. The detection of large scattered radiation in CBCT causes limitations in image quality related to noise and contrast resolution (Scarfe and Farman, 2008). Nonetheless, CBCT has become a popular tool for dental and maxillofacial imaging (Jacobs, 2011; Mamatha et al., 2015).

A CBCT device is shown in Figure 5. In CBCT device, a C-shaped rotating arm (C-arm) consists of a radiation source (X-ray tube) and a detector mounted on opposite side. The target object (patient or phantom) is placed between the X-ray source and detector. During a scan, the C-arm rotates in a circular trajectory around the target object, collecting hundreds of frames (views) from different angles. The frames are sent to a computer for reconstruction, which computes a 3D reconstruction of the target object.

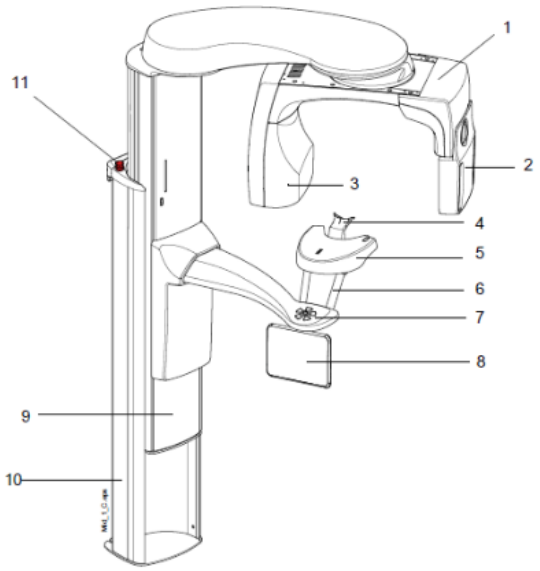

Figure 5: Schematic drawing of Planmeca ProMax 3D CBCT device (Planmeca Oy, 2017). Main parts of the machine are numbered as: (1) C-arm, (2) sensor (detector), (3) tube head, (4) patient supports, (5) patient support table, (6) patient handles, (7) patient positioning controls, (8) touch screen, (9) telescopic column, (10) stationary column, (11) emergency stop button.

Image reconstruction

When an X-ray travels through an object, it attenuates due to effects such as absorption or refraction (McKetty, 1998). The internal material of the object has an absorption coefficient at each point in the space, higher attenuation coefficient implies a denser material.

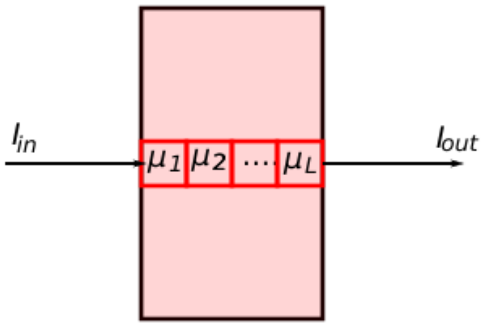

Figure 6: An X-ray beam passing through an object with different attenuation coefficients along a line.

If we assume that X-ray is infinitely thin and travels along a straight line (see Figure 6), the relationship between the incident intensity of X-rays I_{in} and transmitted

intensity I_{out} can be written according to Beer-Lambert law (Swinehart, 1962),

$$I_{out} = I_{in} \exp[-\int_0^L \mu(l)\, dl], \tag{1}$$

where $\mu(l)$ is attenuation coefficient of the object in the beam path. By taking logarithm on both sides, the line integral can be written as,

$$p = \int_0^L \mu(l)\, dl = -\ln \frac{I_{out}}{I_{in}}, \tag{2}$$

where p is the projection measurement and L is the thickness of the object X-ray beam travelled.

This can be extended to a 2D object (image slice) as shown in Figure 7 where X-ray projections are taken along many parallel lines. In a parallel projection geometry, parallel projections are also taken at several angles θ. The source and detector move together in a circular path. For the purpose of CT, the inside of the object can be reconstructed by obtaining attenuation coefficients or the object density information. If the object's densities are represented by a function $f(x, y)$, the line projections can be written as,

$$p(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy, \tag{3}$$

where projection $p(\rho, \theta)$ is also called forward Radon transform of the object function $f(x, y)$, $\rho = x\cos\theta + y\sin\theta$ is line along the projections, and $\delta(.)$ is Dirac delta function (Kak and Slaney, 1988b: p. 50).

The task is to construct the image of the object by obtaining inverse Radon transform using such projection measurements. Two major categories of reconstruction methods are analytical and iterative. In general, analytical reconstruction is considered computationally more efficient while iterative reconstruction can improve image quality (Hsieh et al., 2013). Traditionally, most commonly used analytical reconstruction method in commercial CT scanners has been filtered back-projection (FBP) (Saxena and Kumar, 2018). It applies a 1-D filter on the projection data before back-projecting (smearing) the data onto image space (Hiriyannaiah, 1997; Kak and Slaney, 1988b). Various FBP-type methods have been developed for different generations of CT scans. The detailed explanation of these methods is not in the scope of this thesis but the curious reader can refer to the book by Kak and Slaney (1988a) for fundamentals of image reconstruction and review by Flohr et al. (2005) for multi-slice CT reconstruction.

Feldkamp algorithm

In CBCT the detector is 2D and projections could be considered as 2D forward Radon transforms. While there have been many variants developed for the image reconstruction as described by Turbell (2001), for a simple and fast implementation, the well known Feldkamp algorithm is mostly used either in its original form (Feldkamp et al., 1984) or with some modifications (Xiao et al., 2003; Scherl, 2012). We

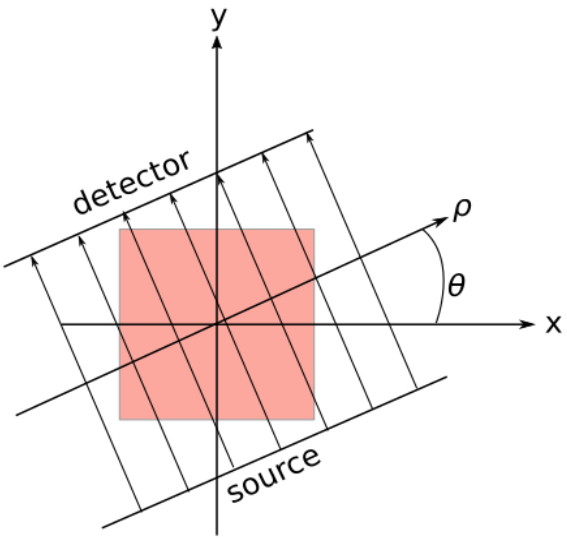

Figure 7: The parallel projection geometry. The source and detector rotate about the center of the object. For each angle θ, the attenuation coefficients are added along the X-ray at the detector. Projections are usually taken for the $\theta \in [0, 180)$.

will explain Feldkamp algorithm in simple form here. For more detailed explanation and derivations, reader can consult (Feldkamp et al., 1984; Turbell, 2001).

The Feldkamp algorithm (FDK) is based upon a circular trajectory (see Figure 8), where the source and detector rotate along an ideal circle. The X-ray source S is at a distance D from the detector center d. The object in the middle is in the world coordinate system (x, y, z) placed at the rotation center O. OX is aligned towards the source and detector center at angle zero. The flat panel detector and source rotate together around the z-axis. The radius of the circular trajectory is R, and u and v are the pixels in the image. For any ray hitting the detector at u and v at a particular angle θ, the projection function can be written as,

$$p(\theta, u, v) = \int_0^\infty f(s(\theta) + \alpha\hat{\gamma})d\alpha, \tag{4}$$

where $s(\theta) = (R\sin\theta, R\cos\theta, 0)$ is the location of the source with respect to rotation center O, $\alpha \in [0, \sqrt{D^2 + u^2 + v^2}]$ is the distance from source along the ray vector and $\hat{\gamma}$ is the unit vector along the direction of ray vector (Scherl, 2012).

Steps: FDK is an approximate reconstruction algorithm. FDK algorithm for circular CBCT with planar detector can be applied in following successive steps:

Step 1 - Filtering: Each measured projection $p(\theta, u, v)$ is filtered in following steps:

(a) Cosine weighting: This step weights the projections according to the cosine of the angle ϕ between the ray hitting the detector at center d and the ray hitting the

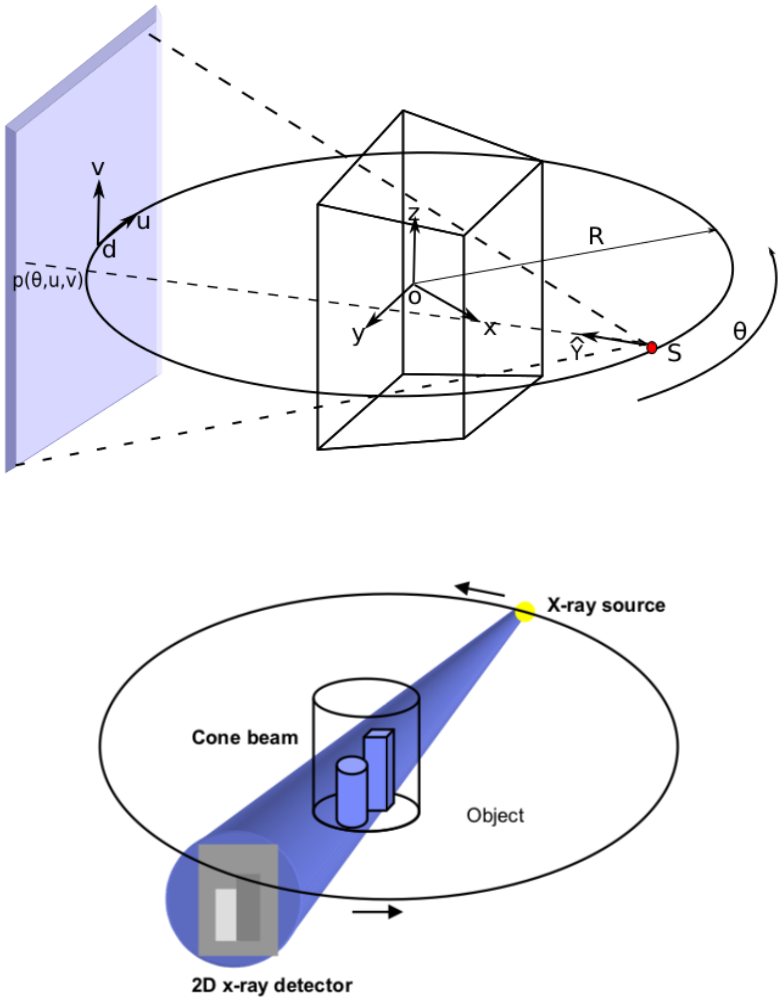

Figure 8: Ideal Feldkamp geometry. The X-ray source S rotates along an ideal circle of radius R with center of rotation at O. In this geometry, the rotation direction is anti-clockwise and given by the angle θ. The detector is on the opposite side and rotates along with the source in the same direction. The lower diagram is a simpler representation of cone-beam geometry (Sakamoto et al., 2005).

detector at (u, v).

$$p_c(\theta, u, v) = p(\theta, u, v) \times \cos\phi, \tag{5}$$

where $\cos\phi = \frac{D}{\sqrt{u^2+v^2+D^2}}$.

(b) Ramp filtering: This step is 1-D convolution along the rows of the detector The cosine-weighted projections are convolved with a ramp filter (Wei et al., 2005).

$$p_f(\theta, u, v) = p_c(\theta, u, v) * h(u), \tag{6}$$

where $h(u)$ is impulse response of ideal ramp filter in frequency domain.

Step - 2 Back-projection: The filtered projections are back-projected (smeared) into the image space to obtain approximated solution as below.

$$\hat{f}(x,y,z) = \int_0^{2\pi} \mu(\theta,x,y,z) p_f(\theta, u(\theta,x,y,z), v(\theta,x,y,z)) d(\theta), \tag{7}$$

where $\mu(\theta,x,y,z)$ is a point-dependent distance weight such as,

$$\mu(\theta,x,y,z) = \frac{R^2}{R + y\sin\theta - x\cos\theta}, \tag{8}$$

and the detector coordinates u and v in terms of x,y,z coordinate system (Xiao et al., 2003) are,

$$u(\theta,x,y) = \frac{D(y\cos\theta + x\sin\theta)}{R + y\sin\theta - x\cos\theta}, \tag{9}$$

$$v(\theta,x,y,z) = \frac{zD}{R + y\sin\theta - x\cos\theta}. \tag{10}$$

2.4 Geometry and collimation artifacts

Geometry artifacts

In reality, the assumption of a circular source-detector trajectory is not essentially true. Motion of the X-ray source and detector differ significantly from a simple circular orbit (Daly et al., 2008) due to, for example, gravity-induced mechanical flex (Jaffray et al., 2002) or due to the tilted detector rows with respect to an ideal Feldkamp-like acquisition (Scherl, 2012). Failing to correct such non-idealities may result in misregistration, loss of details, or image artifacts (Jaffray et al., 2002). A calibration of the imaging system geometry is done for the accurate image reconstruction (Yang et al., 2006).

In geometry calibration, a specific phantom with known dimensions is scanned and various geometry parameters are recorded. During the reconstruction, the geometry parameters already obtained from the calibrations are in the reconstruction algorithm (Cho et al., 2005; Yang et al., 2006). During the patient scan, it is assumed that the patient is stationary and the source and detector are following a predefined trajectory. The motion artifacts can occur from one of the two causes:

1. Patient motion: The exposure time in CBCT ranges from 5.4 to 40 seconds (Nemtoi et al., 2013) and such time is long enough for the patient's movements. Hanzelka et al. (2013) have found that patient with open eyes tend to follow the motion of the C-arm. Patient motion can cause misregistration artifacts within the image (Lee, 2008). If an object moves during the scanning process, the reconstruction algorithm does not account for the movement since no information on this movement is integrated in the reconstruction process.

2. Non-patient motion: If the geometry defined in the geometry calibration files does not repeat itself due to some reason like flex or jitters while rotation, it can

cause motion artifacts in the images (Jaffray et al., 2002). This causes same sort of artifacts as described above. Reconstruction algorithms try to compensate those effects up to some limit. It is not essential that the source and detector follow a perfect circular trajectory, but the trajectory should be repeatable following the geometry calibration (Pauwels et al., 2015). Even minute deviations can cause poor image quality (Nagarajappa et al., 2015).

Motion artifacts can have detrimental impact on the diagnostic images. It is beneficial to be aware of the presence of artifacts and be familiar with their characteristic appearances in order to enhance the extraction of diagnostic information from cone-beam images (Makins, 2014; Nardi et al., 2016).

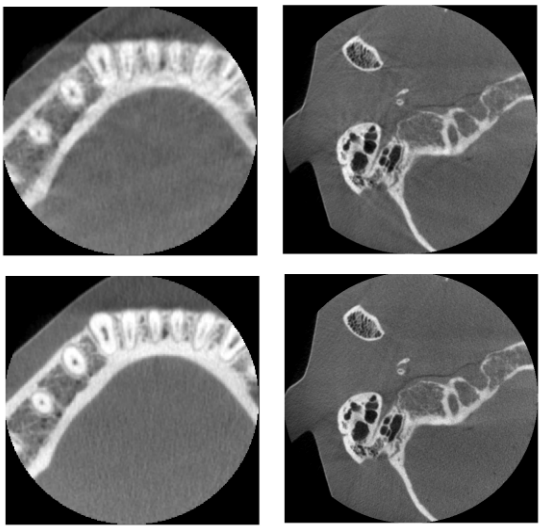

Figure 9: Upper two images show the motion artifact and the lower two images are without any motion artifacts. The images have been acquired on a skull phantom at the R&D facility of Planmeca Oy.

Collimation artifacts

The collimator is essentially a rectangular window made from lead sheets. It controls the field of view for the X-ray exposure. If the collimator window is not open accurately during the scan, it can cause the artifacts in the images. In the production, extended volumes are taken in order to visualise the collimation artifacts more clearly. The worker has to go through all the images in the extended volume manually and decide whether there is artifact. If there is an collimation artifact, worker needs to move the collimator manually and repeat the whole process again.

Collimator calibration pipeline in production: In order to check and do the collimation calibration the CBCT device goes through the work-flow shown in Figure 11. Beam check is done to see whether the collimator is open enough to cover the rectangular detector. The flat field calibration is done for the correction of the pixel

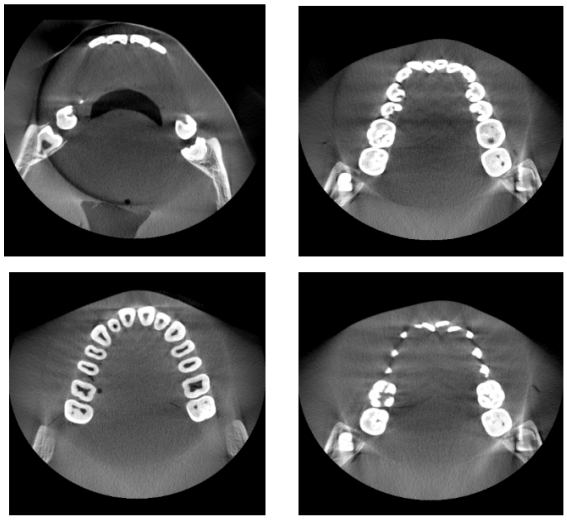

Figure 10: Upper two images show the collimation artifact in the extended volume and the lower two images are without any collimation artifacts. The images have been acquired on a skull phantom at the R&D facility of Planmeca Oy.

gain and dark currents in the detector when no object is present between source and detector. After this, the extended volume scan is taken on a skull phantom to check if any collimator artifact is visible in the images. If the collimator artifact is visible, the collimator blades are moved wider and the whole process is repeated again until the collimator artifacts are not visible. Once there are no collimator artifacts, other scans and checks can be performed.

2.5 Deep learning basics

Machine learning

Classical computer programs are explicitly programmed by hand to perform a task. With machine learning, some part of the human contribution is replaced by an algorithm. In machine learning, computers are fed with data and the computer algorithm tries to learn the hidden relationships in the data (Bishop, 2006; Murphy, 2012).

In supervised machine learning (Hastie et al., 2008a), the program receives a set of inputs and corresponding labels as training dataset and the algorithm tries to learn the relationship between input and output. Once the algorithm is trained, it can predict an output for any new input. In unsupervised machine learning (Hastie et al., 2008b), there is no training dataset, but the task of the computer is to divide the input data into a number of clusters or to find patterns. In supervised learning, we are interested in the prediction of the output class for an input rather than making clusters of inputs whose labels are unknown. For example, in a typical cat-dog classification problem, if a dataset is available with images of dogs and cats and the corresponding labels, supervised learning can be used to learn the relationship

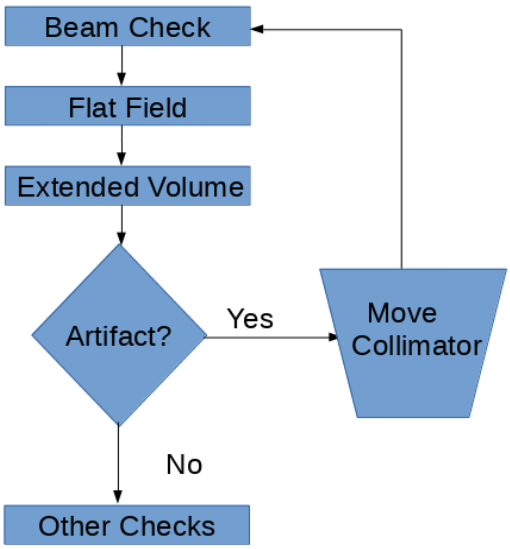

Figure 11: Flowchart of collimation calibration in the production. The process is repeated until the collimator artifacts are not visible any more in the extended volumes.

between the images and labels. The trained network will predict the label for any new image of cat or dog. On the other hand, unsupervised learning will need no labelled data. Unsupervised learning will try to separate the input cat and dog images into two separate clusters, but the label of the both clusters will not be available.

In this work, the focus is in the applications of deep learning (Goodfellow et al., 2016b), a subset of machine learning. Deep learning is a branch of machine learning which has become very popular in recent years for its wide range of applications (Voulodimos et al., 2018; Ker et al., 2018). Below we will describe some central concepts in deep learning which we have utilized in our work.

Neuron model

The basic unit of the neural network is loosely motivated by the basic computational unit of the brain called neuron. The early model of a neuron is termed as McCulloch Pitts neuron (McCulloch, 1943). There are approximately 100 billion neurons in the human brain and each neuron is estimated to have up to 15,000 connections with other neurons (Nguyen, 2013). Each neuron receives input signals from the connected neurons at its dendrites. The input signals get summed and if the total sum is above a certain threshold, the neuron can fire, sending a spike along its axon.

The firing of the neuron is called activation and there are many activation functions which are used in practice nowadays, which are different from the classical (binary) neuronal model. The output can be expressed in mathematical form as,

$$output = \phi(\sum_{m=1}^{m} w_m x_m + b), \tag{11}$$

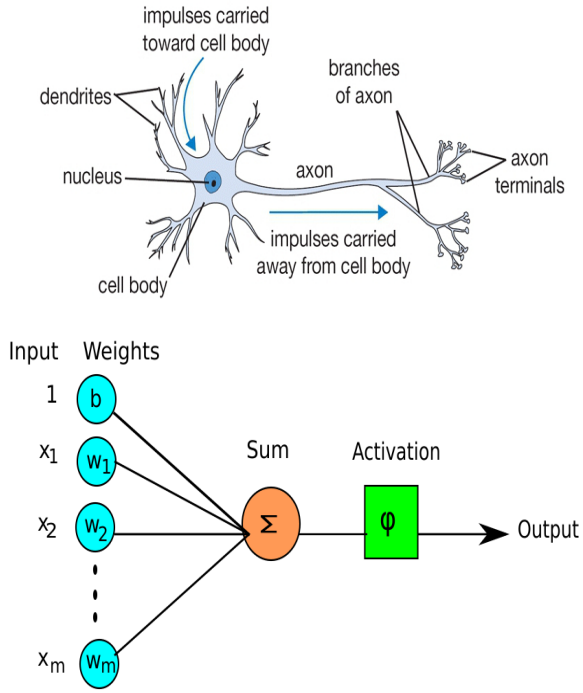

Figure 12: The upper diagram is a biological neuron and the lower diagram is equivalent model of the perceptron (Li, 2018).

where w_m are the weights of the m connected neurons, b is the neuron bias, and ϕ is the activation function.

These neuronal units are essentially not the same as in the brain. There are many different types of neurons with very different characteristics. For example, synapses are not just simple scalar weights but they are complex non-linear and dynamic system (London and Häusser, 2005; Brunel et al., 2014). The McCulloch-Pitts neuron could fire (0,1) or (-1,1). However, this model is so simplistic that it only generates a binary output. Nevertheless, below are some popular activation functions.

Activation functions

The capacity of the neural networks to approximate any function, especially non-convex, is directly the result of the non-linear activation functions. Activation function takes a vector and performs a certain fixed point-wise operation on it. Below are some most used activation functions.

Sigmoid:The sigmoid non-linearity has the following mathematical form,

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{12}$$

The sigmoid function squashes the real valued input number between 0 and 1. Large negative numbers become 0 and large positive numbers become 1. The function is steep in the middle region, that is, any change in the input will change the output

drastically. But at the edges, output values tend to respond less to changes in input, this results in the problem of vanishing gradients during the network training. Also, its output is not zero-centered. In practice, the sigmoid non-linearity has recently fallen out of favour for the use in feed-forward neural networks.

Hyperbolic tangent: The tanh non linearity has the following mathematical form,

$$\tanh(x) = 2\sigma(2x) - 1 \tag{13}$$

The tanh squashes the real-valued input number to the range $[-1, 1]$. Its output is zero centered, but it also has the gradient vanishing problem at the ends.

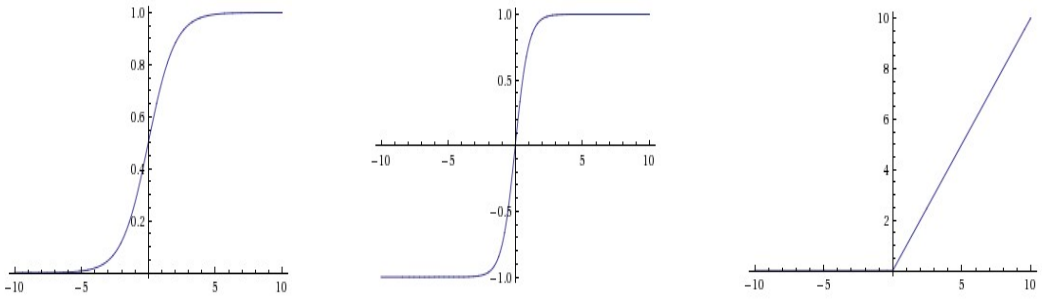

Figure 13: The left diagram is sigmoid, middle is tanh, and right is ReLU non-linearity (Li, 2018).

ReLU: Rectified linear unit computes the function $f(x) = \max(0, x)$. It means that the activations below zero are clipped to zero and the activations above zero remain same. The slope is 1 for the activations above zero. Alex et al. (2012) showed that ReLU accelerated convergence of stochastic gradient descent. It is very simple to calculate in comparison to sigmoid or tanh.

Feed-forward neural network

The individual neuron units are connected in a graph to form a neural network. The network is called feed-forward because the input flows towards the output in forward direction and there is no feedback from the forward branch to backward branch. A simple form of feed-forward neural network is shown in Figure 2.5.

There are three layers in this network. The leftmost layer is called input layer. The middle layer is hidden layer which consists of neuron units and does the calculations. The rightmost layer is called output layer. The output layer is a single linear neuron here, depending upon the task in hand the output layer's size changes.

Let us denote the input features by $\mathbf{x}$ which is a length $\mathbf{m}$ column vector. The weights for the hidden layer and output layers are $\mathbf{W}_1$ and $\mathbf{w}_2$, respectively. The

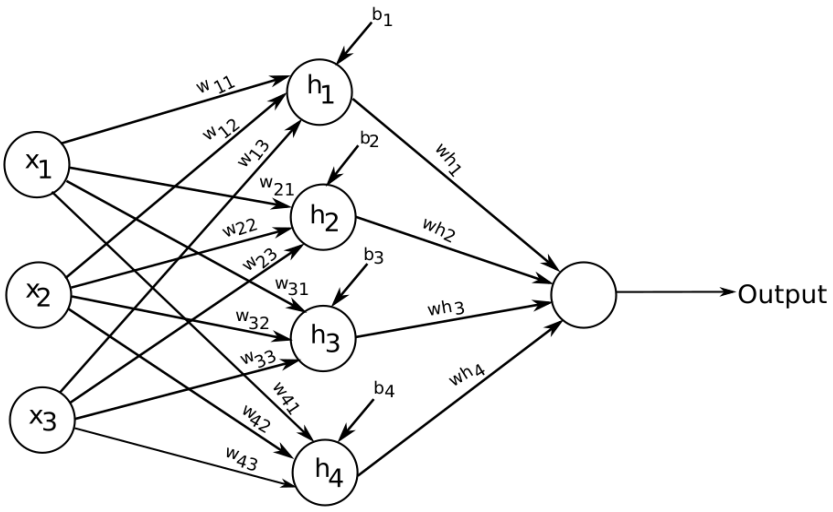

Figure 14: A simple feed-forward neural network with one hidden layer.

biases for hidden layer neuron are $\mathbf{b}$. The output of the feed-forward network can be given as below.

$$\mathbf{y} = \phi(\mathbf{W}_1\mathbf{x} + \mathbf{b})\mathbf{w}_2, \tag{14}$$

where ϕ is an activation function, $\mathbf{x} = [x_1, x_2, x_3]^\intercal$,

$$\mathbf{W}_1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix},$$

$\mathbf{b} = [b_1, b_2, b_3, b_4]^\intercal$, and $\mathbf{w}_2 = [wh_1, wh_2, wh_3, wh_4]^\intercal$.

In modern neural networks a large number of hidden layers are incorporated in a long chain. The output of a hidden layer becomes the input of next hidden layer. The overall length of the chain gives depth of the model. The **deep** term in deep learning comes from this deep chain of hidden layers. The number of hidden layers determine the **depth** and the dimensionality of the hidden layer (number of units in a hidden layer) determines the **width** of the model (Goodfellow et al., 2016b: p. 164).

2.6 Overview of convolutional neural network

Most of the expansion of deep learning is the result of the success of convolutional neural networks (CNN). Although the use of CNNs goes back to 1998 (LeCun et al., 1998), their widespread use is due to much more recent event, when a deep CNN was

used by Alex et al. (2012) to beat a state-of-art level in the ImageNet classification challenge (Russakovsky et al., 2015). Since then, the development in computing power and the access to the large amount of data has brought CNNs at the forefront of rapid progression. For image classification tasks, many variants of CNN architectures have been successfully applied on ImageNet dataset namely VGG (Simonyan and Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), Resnet (He et al., 2016). However, the basic building blocks are very similar. The LeNet-5 (LeCun et al., 1998) is a simple and initial CNN architecture. LeNet network architecture is shown in Figure 15. We will explain different processing layers in a typical convolutional neural network in following section.

Convolutional layer: Images can have millions of dimensions, and it is very expen-

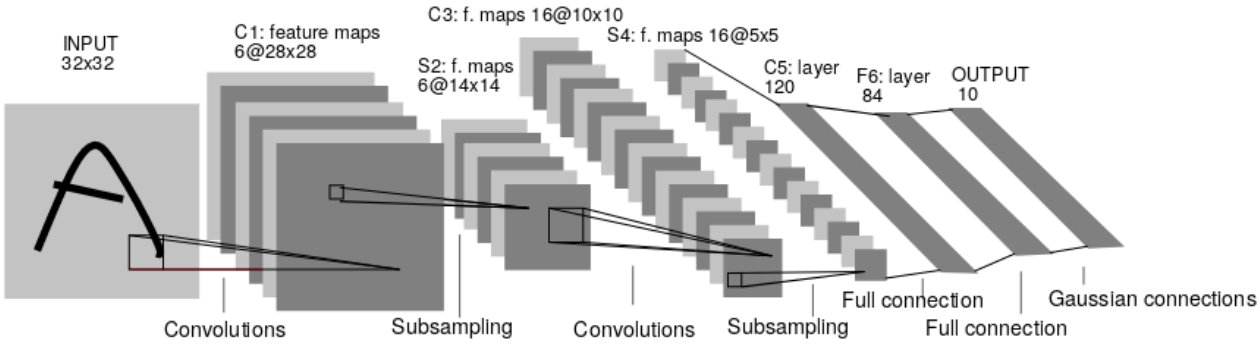

Figure 15: LenNet-5 network (LeCun et al., 1998). The convolution of 6 filter kernels with the input image of 32×32 results in 6 feature maps of size 28×28 (C_1). In next layer S_2, average pooling (subsampling) is performed using a 2×2 kernel. Again a convolution (C_3) and pooling (C_5) was performed. Now the map values are flattened and two fully connected layers (C5, F6) are attached. The output layer has 10 units for 10 classes.

sive to use fully connected layers. For example, if a 32×32 input is connected to a hidden layer of 1000 neurons, total number of parameters will be $32 \times 32 \times 1000 = 1.024$ million, which is huge. Instead, convolutional layers come handy here. Convolution is a linear transformation which preserves the spatial properties of the images. The convolutional layer works as a feature extractor. When the filter weights are convolved with the input image, the output is called feature map. The weights of the filter are learned during the training. Several feature maps are obtained by using many different filter kernels. The idea is to extract important features like edges from the images. Moreover same filter weights can be used for the whole image, which reduces the number of parameters in the network and enhances the computations. Some kind of activation is also applied on the feature maps in order to give the model non-linear functionality. Thus a convolutional layer learns a set of k filters g_1, g_2 .., g_k which are convolved spatially with the image f of size $m \times n$ to produce a set of

k, 2D feature maps h,

$$h_k[x, y] = (f * g)(x, y) = \sum_m \sum_n f[m, n] g[x - m, y - n] \tag{15}$$

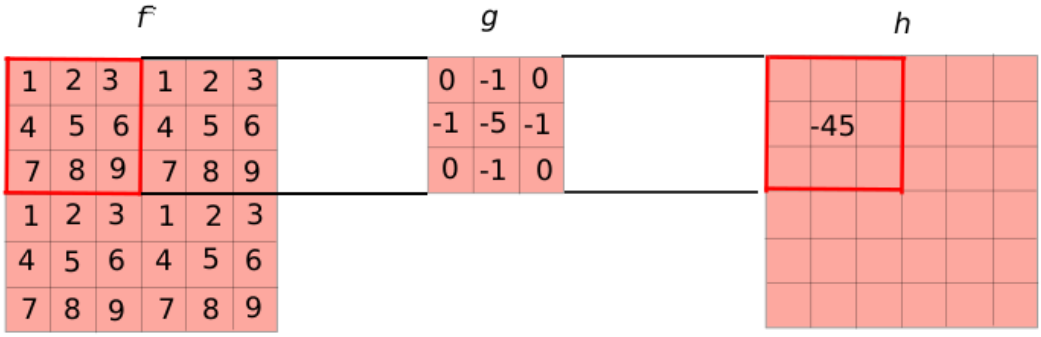

Figure 16: 2-D Convolution. f is a 2-D image which is convolved with the filter kernel g. The center pixel 5 showed inside the red box in the input image gets changed to -45 in the output image h after convolutional operation.

Padding: In order to calculate the center pixel, it is important to consider the type of padding at the boundaries of the image. A common way is to add zero padding along the boundaries, so that the convolved output will be of the same size as of the input image. It is called **same** padding. In the **valid** padding scheme in which we do not add any zero padding to the input, the size of the output would be reduced. Also, depending upon the type of strides and filter size used in the convolution, the output size of the feature map might change. Stride is the units by which the kernel moves left to right or up to down during convolution. The output width and height can be calculated using the following equations.

$$\text{output width} = \frac{w - f_w + 2p}{s_w} + 1, \tag{16}$$

$$\text{output height} = \frac{h - f_h + 2p}{s_h} + 1, \tag{17}$$

where w = input width, h = input height, f_w = filter width, f_h = filter height, p = amount of zero padding, s_w = horizontal strides, and s_h = vertical strides.

Pooling: After the convolutional layers, the feature maps are down-sampled (or subsampled) using pooling. It reduces the spatial size of the features which in turn reduces the amount of parameters and the computations in the network. Pooling can be thought of as a summary statistic of the nearby outputs (Goodfellow et al.,

2016b: p. 330). One form is to apply a 2×2 kernel with a stride of 2 and taking the max value inside the filter box. This kind of max pooling (Zhou and Chellappa, 1988) is shown in Figure 17.

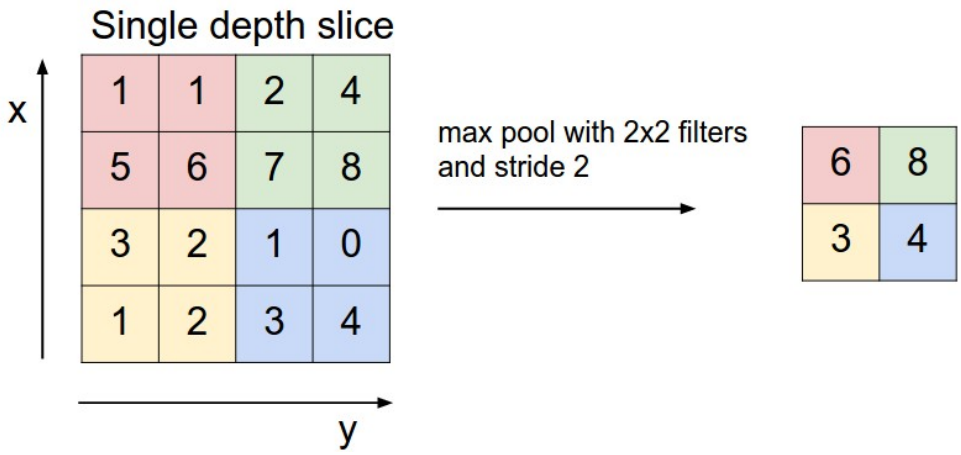

Figure 17: Maxpooling.

Different filter sizes and strides are also used in the literature. Other forms of pooling can be seen as taking the average (shown as sub-sampling in Figure 15) or sometimes taking l^2-norm (also known as euclidean norm or mean-square norm) of the elements inside the filter kernel instead of taking the max (Scherer et al., 2010; Rezaei et al., 2017).

Fully connected layers: The stacks of convolutional and pooling layers are followed by fully connected layers. In the fully connected layer, neurons are connected to all activations in the previous layer, as shown in the case of feed-forward network. Hence, the activations of the neurons are calculated by matrix multiplication followed by a bias offset.

Output layer: The output layer is generally a linear layer which takes the feature maps as input and transforms it into a score or probability. The length of output vector depends upon the number of classes.

Softmax is one type of output layer used in a multi class classification problem. For a given input, it outputs a probability of belonging to each of the class.

Training Process

The training data is usually divided into three sets, namely: training, validation, and test set. The model is trained with the training data and tested on the validation data during training. Once the training has converged, the trained model is tested on unseen test data. The training or learning process has three major steps.

Forward pass: The network takes an input x and produces an output y. This is called forward propagation. Also, a scalar cost L is calculated.

The neural network tries to find a function which maps input to a given output label. The learning of this function is done by providing set of input and output pairs, called training set. In practice, for supervised learning and classification problem, we have a set of inputs and corresponding labels. For example, the input can be a two dimensional matrix such as an $m \times n$ size image, that is, the input is a $m \times n$ dimensional matrix. During the training, the network compares its output y to the real image label and tries to minimise the difference between the predicted output and real label. The idea is to maximise the likelihood of the prediction. In order to do so, the negative log likelihood (loss or cost) is minimised using some optimization techniques. The loss function can be formed depending upon the type of problem in hand.

A typical form of loss function used in convolutional networks for classification tasks is **cross entropy** loss which is the cross entropy between the training data and the model's predictions. The cross-entropy loss for a binary classification problem (class 0 and 1) is calculated as,

$$L = -(y \log(p) + (1 - y) \log(1 - p)), \tag{18}$$

where y and p the correct class label and predicted probability by the model, respectively. The more general form of cross-entropy loss for multi-class problem is,

$$L = -\sum_{i=1}^{C} y_i \log(p_i), \tag{19}$$

where C is number of classes, y_i and p_i are the correct class label and predicted probability by the model for the i_{th} class, respectively.

Backward pass: In this step, the gradient of the cost function is calculated using back propagation. The partial derivatives of cost function L are calculated for all weights w and bias b. Since the information from the cost function flows backward in the network for gradient computation, this process is called back-propagation (Rumelhart et al., 1986a; LeCun et al., 1989) or back-prop.

Computing an analytical expression for the gradients of the cost function with respect to the network parameters is a computationally expensive task owing to the number of parameters in the neural net. Fortunately, we have a method called chain rule of derivatives to calculate the gradient of the scalar loss with respect to any parameter in the network. For more information reader can refer to Nielsen (2018).

Optimization: Optimization is the task of either minimizing or maximizing some function. Here we are interested in minimizing the loss function. Once all gradients are computed, network parameters are updated using an optimization technique. The loss function of CNN is highly non-convex and does not have a global minimum. In **stochastic gradient descent** (SGD) optimization, the loss is reduced by moving in small steps with the opposite sign of its derivative. The small step is called **learning rate**. Learning rate can be fixed or dynamic. The loss is calculated for each training

sample and the process is repeated iteratively until the algorithm seems to converge.

$$\theta \leftarrow \theta - \epsilon \nabla_\theta L(f(x;\theta), y), \tag{20}$$

where θ is the network parameter to be updated, ϵ is a scalar learning rate, and $\nabla_\theta L(f(x;\theta), y)$ is the derivative of the loss function L with respect to θ. It is computationally costly and time taking to iterate over each training sample. In mini-batch SGD, the update is performed for every mini-batch of n training samples. The algorithm has been found to converge faster by this method. It reduces the variance of the parameter updates, which can lead to more stable convergence.

$$\theta \leftarrow \theta - \epsilon \nabla_\theta \sum_{i=1}^{n} L(f(x^i;\theta), y^i). \tag{21}$$

Adaptive moment estimation (Adam) optimizer (Kingma and Ba, 2014) computes adaptive learning rates for each parameter. It stores exponentially decaying average of past squared gradients v_t and past gradients m_t. The calculations are as below,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{22}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{23}$$

where m_t and v_t estimates first moment (mean) and second moment (variance) of the gradients g_t respectively, and β_1 and β_2 are constants (close to 1). The authors found that m_t and v_t are biased towards zero, the bias corrected first and second order moment estimates are,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{24}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{25}$$

Finally, the Adam update is:

$$\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\hat{v}_t} + \delta} \hat{m}_t, \tag{26}$$

where δ is a constant for stabilization in the case $\hat{v}_t$ becomes zero.

Regularization techniques

One of the focus of machine learning is to increase the generalization capability of the model, that is, in addition to the validation set, the model should perform equally well on the unseen data. Following are the major regularization techniques used in this work.

L_2 **regularization**: One regularization approach to overcome over-fitting is the

weight decay, which adds an additional term to the cost function to penalise parameters such that it drives weights closer to the origin. That means it penalises large increase in weights. The new cost function is,

$$J(x, y) = L(f(x; \theta), y) + \lambda \sum_i \theta_i^2, \tag{27}$$

where $\lambda \in [0, \infty)$ is a hyperparameter that weights the contribution of regularization term.

Data augmentation: The best way to make a machine learning model generalise better is to give it more data (Goodfellow et al., 2016b: p. 233). In practice, the amount of training data is limited. One way to solve this problem up to a certain limit is to generate more fake but representative data. For images, many techniques like shift, rotation, contrast change, vertical, and horizontal inversion are some methods to augment the data as demonstrated by Wang and Perez (2017).

Dropout: Dropout (Srivastava et al., 2014) is a recent and simple but effective data regularization method. Dropout refers to dropping (ignoring) some of the neuronal units during the training phase. The set of neurons to be ignored is chosen randomly. At each training stage, individual unites are either dropped out of the network with probability $1 - p$ or kept with probability p, so that a reduced network is left. It has been shown that dropout forces a neural network to learn more robust features and reduces the possibility of over-fitting in convolutional neural networks (Wu and Gu, 2015).

Transfer Learning

In machine learning, transfer learning is, applying models trained on one task on another task. For example, networks trained on ImageNet datasets are widely utilised in medical imaging applications (Shin et al., 2016). A machine learning algorithm trained for the classification of every day color images, such as cat and dogs, could be used to classify radio-graphs. The idea is that all images share similar features such as edges and blobs, which makes transfer learning feasible. In addition, deep neural networks often require large datasets (in the millions) for the proper training. Starting the training with weights from pre-trained networks might perform better than the random initialization if we have small dataset (Tajbakhsh et al., 2016; Zhou et al., 2017). In medical imaging classification tasks, the training data is often limited.

3 Materials and Methods

The experiments involved three major steps: data acquisition, model training, model testing. A number of deep learning frameworks are available to easily build neural networks from existing modules on a high level. Caffe (Jia et al., 2014), Theano (Theano Development Team, 2016), TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2017) and Keras (Chollet, 2018) are some of the frameworks for implementing neural networks. We used Keras for our experiments as Keras comes with Theano and TensorFlow backend. It is easy to use and is continuously being maintained and updated. It runs seamlessly on CPU and GPU. It is very simple with emphasis on rapid model development. It has very good documentation containing many code examples and other resources that help users to get started very quickly. We also used CUDA (parallel computing platform and API model which enables graphics processing unit (GPU) for general purpose processing) and cuDNN (GPU-accelerated library of primitives for deep neural networks) for model development on GPUs.

We worked on two different types of artifacts and our approach deferred for both. We will describe material and methods for the two tasks separately.

3.1 Geometry artifacts

3.1.1 Data acquisition

The images were acquired using six lab phantoms available at the R&D facility of Planmeca Oy using ProMax 3D Mid CBCT device. The data was acquired using a robotic automation script developed at Planmeca Oy. Canine Left and Canine Right sub-programs were used to acquire the training images. We acquired 9 volumes for the training, the size of each volume was $268 \times 268 \times 334$. From each volume, 123 2D axial slices were taken and a total of 1107 images from 9 volumes were used for the training. The motion artifact was synthetically induced in the non-motion images by corrupting the geometry file. The number of motion images were same as the number of non-motion images. A total of 2214 image slices were available for training after taking the motion images into count.

Synthetic motion images: It was easier to produce large dataset of motion images from non-motion images by changing the geometry file and reconstructing the volume from the corrupted geometry file. We used three volumes of real motion images and their corresponding motion artifact corrected volumes to generate a function by fitting a normal distribution on the geometry file difference between motion and non-motion images. We induced motion artifacts in the images by adding randomly sampled values from the fitted function to the original geometry files of the non-motion images.

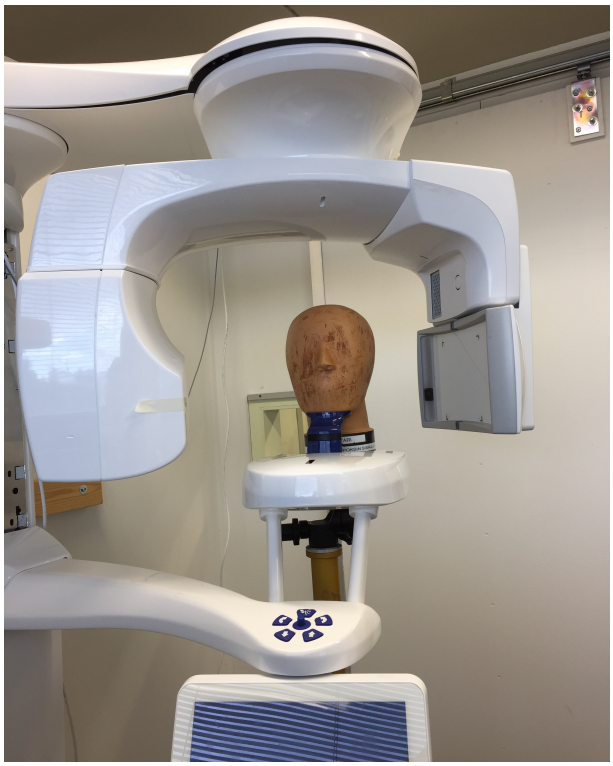

Figure 18: One of the phantom set up used to acquire the training data with Promax mid CBCT device (Courtsey of Planmeca Oy).

3.1.2 Image preparation

Image intensities were normalised from 0 to 1 for the whole 3-D volume by dividing each pixel by the maximum pixel value of the volume. This was the only preprocessing done on the images. Each image slice was divided into 60×60 resolution patches with 50% overlap (see Figure 19). For simplification, it was assumed that all patches in the motion images contain motion artifacts. The total number of patches available were 141,696. The patches from non-motion images were assigned label 0 and the patches from the motion images were assigned label 1. 80% of the patches were used for the training and 20% patches were used for the validation.

3.1.3 Network architecture

We used a method described by Küstner et al. (2018) for the motion artifact detection in MRI images. The network used is called CNNArt by the authors. The network takes image patches as input and outputs the probabilities of belonging to class 0 (non-motion) and class 1 (motion), for each patch. Keras library (Chollet, 2018) was used for the network implementation with TensorFlow backend. Training was performed using two NVIDIA Geforce GTX-780 cards with 3 GB RAM each.

Below is an example of Keras sequential API to create network model. The model object is called using sequential API. Layers and activation can be added in series. Convolutional kernel coefficients were initialized by a Gaussian distributed

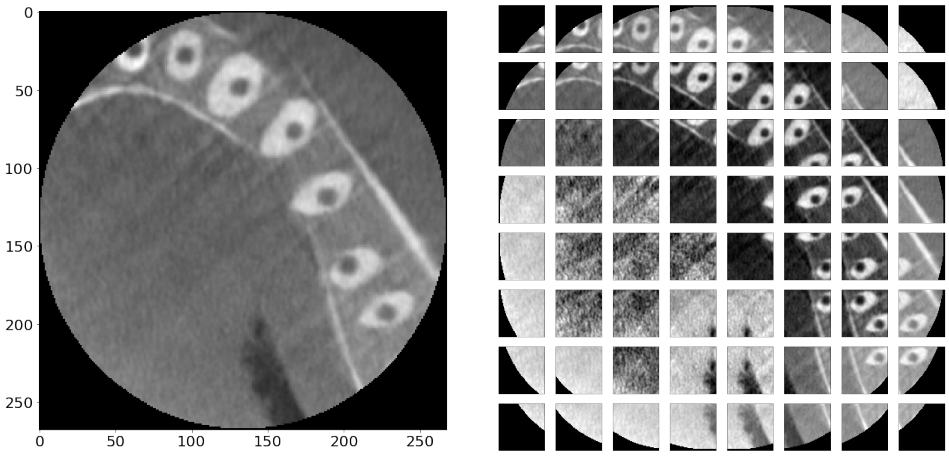

Figure 19: The left image is an input image of size 268×268 and the right image shows 64 patches of size 60×60 made from the left image with 50% overlap.

randomization. For the output predictions, a softmax layer is added in the last.

```
cnn = Sequential()
cnn.add(Conv2D(32, kernel_size=(14, 14),kernel_initializer='he_normal',
               weights=None,padding='valid',strides=(1, 1),
               W_regularizer=l2(1e—6),input_shape=(60,60, 1)))
cnn.add(Activation('relu'))
cnn.add(Flatten())
cnn.add(Dense(output_dim= 2,init = 'normal',W_regularizer='l2'))
cnn.add(Activation('softmax'))
```

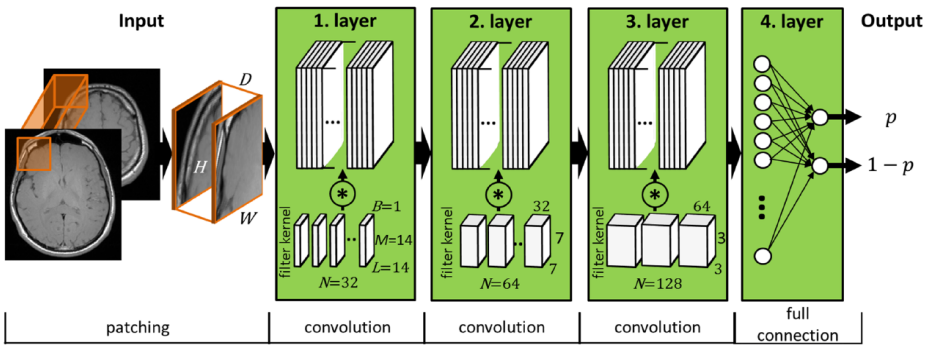

Figure 20: The convolutional neural network with three hidden layers, one fully connected layer and one softmax layer as output. The size of input is $H \times W \times D$, where H is height, W is width and D is depth of the input. In our experiment, the input size is $60 \times 60 \times 1$. The outputs of the network are the probabilities for the two classes, in our case, motion and non-motion (Küstner et al., 2018)

.

CNN filter coefficients were trained by optimizing the categorical cross-entropy loss function for a learning rate of 0.0001 with L_2 coefficient regularization. The cost function was optimized using Adam with corresponding parameters as $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^8$ as used by the authors in Küstner et al. (2018) and described by Kingma and Ba (2014) in original Adam article as good default settings for the tested machine learning problems.

The below Keras code creates the optimizer object with custom parameters.

```
opti = keras.optimizers.Adam(lr=0.0001, beta_1=0.9, beta_2=0.999,
        epsilon=1e−08, decay=0.0)
```

As shown in the below code, model training was run for 500 epochs and the model for the epoch with the lowest validation loss was saved using Keras callbacks object. In Keras, model needs to be compiled before the actual training starts, where the loss, optimizer, and metric (to be used for the training) are specified. Finally the training was run using fit function. The training and validation datasets are given to the fit function as input. "batch_size" determines number of samples taken at a time and "verbose=1" prints the training progress on the screen.

```
iEpochs=500
batchSize=32
callbacks=keras.callbacks.ModelCheckpoint('model.h5',
                monitor='val_loss', verbose=1, save_best_only=True,
                save_weights_only=False, mode='auto', period=1)
cnn.compile(loss='categorical_crossentropy',optimizer=opti,metrics=['accuracy'])
result = cnn.fit(X_train,
                     y_train,
                     validation_data=[X_test, y_test],
                     nb_epoch=iEpochs,
                     batch_size=batchSize,
                     callbacks=[callbacks],
                     verbose=1)
```

For the testing of the trained model, we used canine images from the 6^{th} phantom, which were not used for the training. Also, the performance of the model was tested against volumes from the other sub-programs like temporal bone and jaw. The test slices were first divided into the 60×60 patches and the softmax function gave probabilities for each patch of belonging to both motion and non-motion classes. We considered only the probability of motion class. Patches with probability of greater than 0.5 for motion class were taken as having motion artifacts. Finally, to get percentage of patches in a 2D image slice, below formula was used:

$$motion(\%) = \frac{patches\ with\ motion\ probability{>}0.5}{total\ patches}. \tag{28}$$

To assist the model predictions we dropped all the patches with more than 55% zero pixels. So Equation 28 becomes,

$$motion(\%) = \frac{patches\ with\ motion\ probability{>}0.5\ and\ less\ than\ 55\%\ zero\ pixels}{total\ patches{-}patches\ with\ more\ than\ 55\%\ zero\ pixels}. \tag{29}$$

3.2 Collimator artifacts

3.2.1 Data acquisition

Data for the collimator artifact was collected from the Planmeca Oy's production and R&D facilities. Extended image volumes were acquired from the ProMax Classic CBCT device using teeth sub-program. In the R&D facility, to simulate the collimator artifact, we set the collimator position wrongly which produced artifacts in the extended volume. Data obtained from the production was the data collected during the routine testing protocols. Three different phantoms were used for diversity of information in the images. A total of 48 volumes were used for the training, out of which half of the volumes had collimator artifacts. 65 unseen volumes were used for model testing.

3.2.2 Image preparation

Image pixels were normalised between 0 to 1 for each 3-D volume by dividing each pixel by maximum pixel value of the volume. The volume size was $551 \times 551 \times 551$. 290 2D slices were taken from each of the 48 volumes for training. In total we had 13920 image slices for network training. Each image slice was resized to 224×224 for being used as input to the custom VGG model. Each gray image was stacked to make a 3-channel image to be used with VGG model pre-tuning as pre-trained VGG model accepts only 3-channel input images.

3.2.3 Network architecture

We modified the VGG-16 model shown in Figure 21. VGG-16 has 16 layers in the network. VGG-16 model won ImageNet Challenge 2014 submission, where it secured first and the second places in the localisation and classification tracks respectively (Simonyan and Zisserman, 2014). The VGG article has been cited more than 11,800 times since 2014. We used the original VGG-16 also by using pre-trained weights and fine tuning some of the last layers. We trained modified VGG from scratch.

The training was performed using Elastic compute cloud (EC2 management console on Amazon web services (AWS) platform. A deep learning Amazon machine image (AMI - 5d7c5024), preloaded with Keras, CUDA and CuDNN was used. The testing was performed on a computer system installed with two NVIDIA Geforce

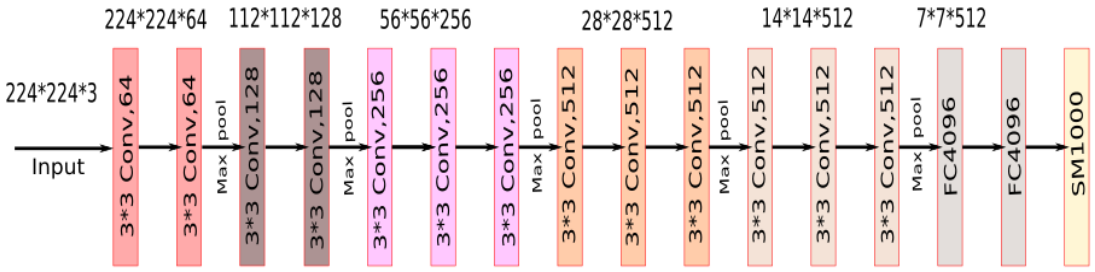

Figure 21: The original VGG-16 network with 16 layers (Simonyan and Zisserman, 2014). The output softmax layer has 1000 units for 1000 classes in ImageNet database.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

Figure 22: The VGG-16 architecture showing different layers and parameters. The softmax layer has 1000 units for 1000 class classification in ImageNet data.

GTX-780 cards with 3 GB RAM each. The pretrained model weights were downloaded from Keras. The code below loads the VGG-16 model with pretrained weights.

```
from keras.applications.vgg16 import VGG16
input = Input(shape=(224, 224, 3))
model = VGG16(input_tensor=input, include_top=True,
              weights='imagenet')
```

Below code is an example of Keras functional API. The output of the last convolutional layer was taken and flattened. The fully connected dense layers (4096

units) were replaced by new smaller layers (512 units) and 1000 units of output softmax layer was replaced by 2 units for binary classification task.

```
last_layer=model.get_layer('block5_pool').output
x=Flatten(name='flatten')(last_layer)
x=Dense(512,activation='relu',name='fc1')(x)
x=Dense(512,activation='relu',name='fc2')(x)
out=Dense(2,activation='softmax',name='output')(x)
finetuned_vgg=Model(input,out)
```

Instead of training all layers, only the last 3 layers' parameters were trained for the input images. The rest of the layer's parameters were frozen. Below is the code for freezing the training of remaining layers of the model.

```
for layer in finetuned_vgg.layers[:-3]:
        layer.trainable = False
```

The loss function was categorical cross-entropy function with a learning rate of 0.0001. The cost function was optimized using Adam with corresponding default parameters as $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^8$. The training was run for 500 epochs and the best epoch model with lowest loss was saved using callbacks in Keras.

The custom VGG model is shown in Figure 23. The implementation of custom model with dropout layers was as following in Keras using functional API. The operations like dropout, convolution, and maxpooling can be imported and applied very easily layer by layer. The ReLu activation is applied after each convolution. The model is built by using "Model" function by providing input tensor and output tensor.

```
from keras.layers import Dense, Dropout, Flatten, Input
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.models import Model

model_input = Input(shape=(224,224,1), name="input_layer")
x = ZeroPadding2D((1,1))                    (model_input)
x = Conv2D(64, (3, 3), activation='relu')     (x)
x = ZeroPadding2D((1,1))                      (x)
x = Conv2D(64, (3, 3), activation='relu')     (x)
x = MaxPooling2D((2,2), strides=(2,2))        (x)

x = ZeroPadding2D((1,1))                      (x)
x = Conv2D(128, (3, 3), activation='relu')    (x)
x = ZeroPadding2D((1,1))                      (x)
x = Conv2D(128, (3, 3), activation='relu')    (x)
x = MaxPooling2D((2,2), strides=(2,2))        (x)

x = ZeroPadding2D((1,1))                      (x)
x = Conv2D(256, (3, 3), activation='relu')    (x)
x = ZeroPadding2D((1,1))                      (x)
```

```
x = Conv2D(256, (3, 3), activation='relu')    (x)
x = ZeroPadding2D((1,1))                      (x)
x = Conv2D(256, (3, 3), activation='relu')    (x)
x = MaxPooling2D((2,2), strides=(2,2)) (x)

x = Flatten()                                 (x)
x = Dense(512, activation='relu')             (x)
x = Dropout(0.5)                              (x)
x = Dense(512, activation='relu')             (x)
x = Dropout(0.5)                              (x)
x = Dense(2, activation='softmax')            (x)

custom_model = Model(inputs=model_input, outputs=x)
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 1)	0
zero_padding2d_1 (ZeroPaddin	(None, 226, 226, 1)	0
conv2d_1 (Conv2D)	(None, 224, 224, 64)	640
zero_padding2d_2 (ZeroPaddin	(None, 226, 226, 64)	0
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0
zero_padding2d_3 (ZeroPaddin	(None, 114, 114, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
zero_padding2d_4 (ZeroPaddin	(None, 114, 114, 128)	0
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 128)	0
zero_padding2d_5 (ZeroPaddin	(None, 58, 58, 128)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	295168
zero_padding2d_6 (ZeroPaddin	(None, 58, 58, 256)	0
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
zero_padding2d_7 (ZeroPaddin	(None, 58, 58, 256)	0
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 256)	0
flatten_1 (Flatten)	(None, 200704)	0
dense_1 (Dense)	(None, 512)	102760960
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 2)	1026

Figure 23: The custom VGG architecture showing different layers and parameters. The softmax layer has 2 units for 2 class classification.

The data augmentation in Keras was done using "ImageDataGenerator" class as shown in the code below. This class generates modified images on the fly during training and the modifications in the images are random. The type and range of modifications can be specified in the arguments.

```
traingen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.5,
    horizontal_flip=True)
train_generator=traingen.flow(X_train, y_train, batch_size=64,seed=42)

testgen = ImageDataGenerator()
test_generator=testgen.flow(X_test,y_test, batch_size=64)

hist = model.fit_generator(train_generator, steps_per_epoch=len(X_train)/64,
        callbacks=[callback], epochs=100, verbose=1,
        validation_data=test_generator)
```

3.3 Metrics

While accuracy is an important metric to measure the model performance, there are some other metrics which provide greater insights into the model performance.

Precision: In a classification task, the precision for a class is the number of items correctly labelled as belonging to the class (true positive) divided by the total number of elements labelled as belonging to the positive class (sum of true positives and false positives). In other words it means how many selected items are correct.

Recall: In a classification task, recall or sensitivity is the number of true positives divided by the total number of elements that actually belong to the positive class (the sum of true positives and false negatives). In other words it means how many correct items are selected.

F_1 **score**: F_1 score is the harmonic average of precision and recall. An F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision+recall}} \tag{30}$$

Cross validation: Cross validation is a method to check the model robustness to the data (Kohavi, 1995). It is particularly useful when the dataset is small. A part of dataset is reserved for testing (validation) and model is trained on the remaining dataset. It can be done in different ways. In leave-one-out cross validation (LOOCV),

we reserve only one data point from the available dataset, and train the model on the rest of the data. In k-fold cross validation the entire dataset is split in k folds. For each fold, model is trained on $k-1$ folds and validated on k^{th} fold. We used 5-fold cross validation using Scikit-learn library (Pedregosa et al., 2011), keeping 20% data for test and 80% data for model training. The dataset division was stratified, means that for each data-folds, the division of classes was balanced. As shown in the code below, "Stratified KFold" cross-validator provides train/test indices to split data in train/test sets and number of folds can be controlled by "n_splits" variable.

```
from sklearn.model_selection import StratifiedKFold
seed=42
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
for train, test in kfold.split(X, Y):
X_train, Y_train =X[train],Y[train]
Y_test,Y_test=X[test],Y[test]
```

4 Results

4.1 Geometry artifacts

The training process curves are shown in Figure 24. The training set accuracy reached 0.9961 and validation set accuracy reached 0.9960. The training and validation losses were 0.0264 and 0.0236 respectively. 5-fold cross validation mean loss and mean accuracy were 0.3683 and 0.9919 respectively for the validation dataset folds.

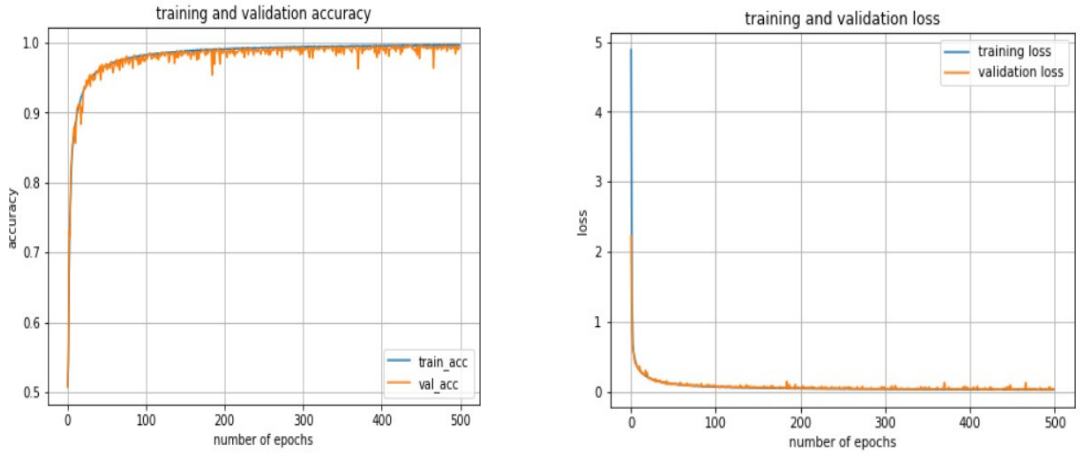

Figure 24: The accuracy and loss curves during model training.

Accuracy

We tested the model trained on the canine images for motion prediction in temporal bone images also. Temporal bone images had higher resolution of (801×801) while the model was trained on low resolution canine images (268×268). The thresholds for prediction motion in a slice has been taken as 10% and 15%, that is, a slice contains a motion artifact if the percentage of patches classified by the model as motion artifact class is more than the threshold. The motion detection accuracies of the model for canine and temporal bone images is shown in Table 1.

Table 1: Slice-wise test accuracy for the canine and temporal bone images. 200 with motion and 200 without motion canine images and 400 with motion and 400 without motion temporal bone images were used for testing of the model. We had 800 with motion and 800 without motion slices from jaw volumes.

	Canine		Temporal bone		Jaw	
Motion	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$
no	100	100	100	100	93	98.75
yes	100	100	98.75	87	94.62	92.62

Below are the volume-wise motion detection accuracies based on taking 100 image slices from the volume for canine and temporal bone. For the canine volumes motion detection accuracy is 100% (see Table 2). For the temporal bone volumes if we put a threshold of 95%, that is, if more than 95% slices in the volume contain artifact, then the volume is considered as motion volume. Then the volume-wise motion detection accuracy for temporal volumes can also be considered as 100% (see Table 3).

Table 2: Volume-wise motion detection accuracies for two canine volumes.

Canine				
	Volume - 1		Volume - 2	
Motion	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$
no	100	100	100	100
yes	100	100	100	100

Table 3: Volume-wise motion detection accuracies for temporal bone volumes.

Temporal bone								
	Volume - 1		Volume - 2		Volume - 3		Volume - 4	
Motion	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$
no	100	100	100	100	100	100	100	100
yes	100	100	100	100	95	52	100	96

The volume-wise motion detection accuracy for jaw volume is also 100% when we consider a volume as having motion if at least 78% of slices have artifact in that volume (see Table 4).

Table 4: Volume-wise motion detection accuracies for each jaw volume.

Jaw								
	Volume - 1		Volume - 2		Volume - 3		Volume - 4	
motion	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$	$> 10\%$	$> 15\%$
no	100	100	93	100	96	100	83	95
yes	78.50	70.50	100	100	100	100	100	100

Scores

Further to examine the model's accuracy, we calculated precision, recall and F_1 score for image slices.

Table 5: Various scores for the canine, temporal bone and jaw image slices. Class 0 slice is without artifact and class 1 slice is with motion artifact.

Canine				
class	precision	recall	F1 - score	slices
0	1	1	1	200
1	1	1	1	200
avg/total	1	1	1	400
Temporal bone				
class	precision	recall	F1 - score	slices
0	0.96	1	0.98	400
1	1	0.96	0.98	400
avg/total	0.98	0.98	0.98	800
Jaw				
class	precision	recall	F1 - score	slices
0	0.95	0.93	0.94	800
1	0.93	0.95	0.94	800
avg/total	0.94	0.94	0.94	1600

Motion maps

We also drew the motion probability maps. An visualization of the motion maps predicted by the model for canine, temporal bone and jaw test images is shown in Figure 25. The map of the motion image contains different colors but the map of the motion-less image is almost completely blue. The percentage of patches with motion artifact as predicted by the model are 96.15% and 0.0%, respectively, for canine motion and non-motion image and 38.39 % and 1.41 %, respectively, for temporal bone motion and non-motion image.

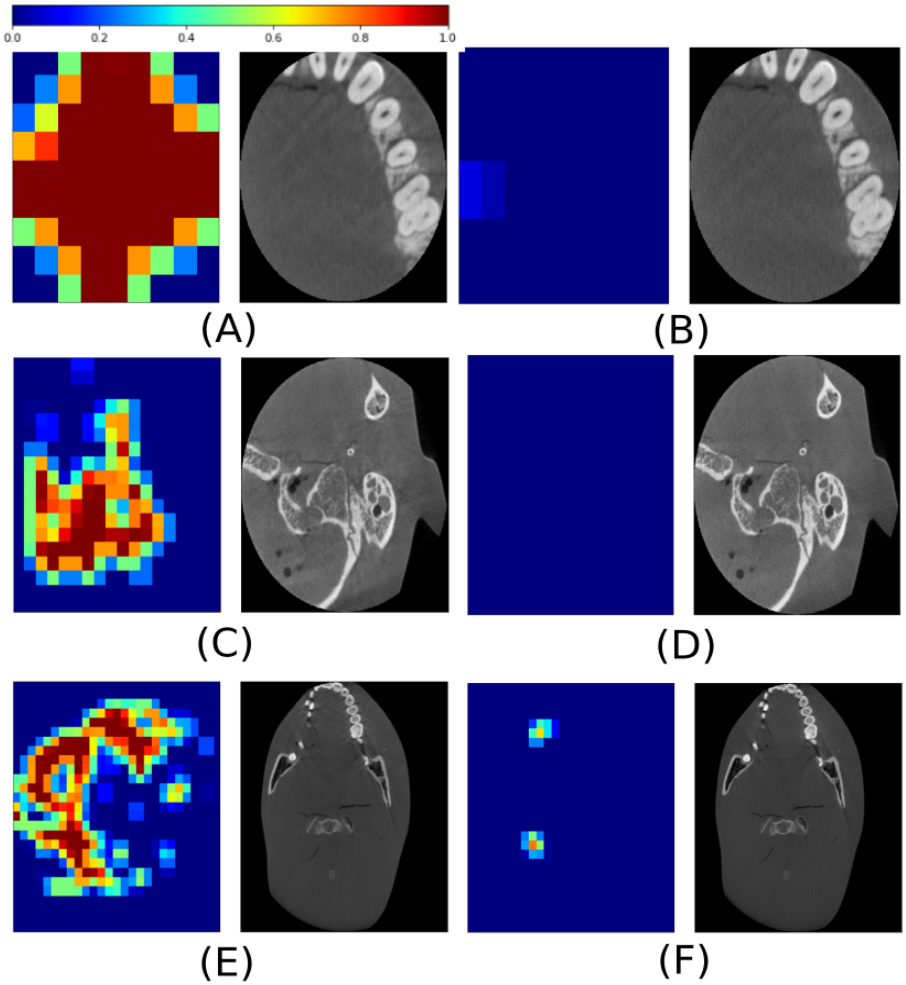

Figure 25: The motion maps for some test images. The left images show the probability of motion for patches of the right images projected back into the image space with an patch overlap of 50%. The patches more than 55% zero pixels have been given 0 probability by default. (A) is canine motion image with 96% motion patches. (B) is canine non-motion image with 0% motion patches. (C) is temporal bone motion image with 32% motion patches. (D) is temporal bone non-motion image with 0% motion patches. (E) is jaw motion image with 38% motion. (F) is jaw non-motion image with 1% motion patches. The colorbar is shown on the top.

4.2 Collimation artifacts

Comparison between different models are shown in Table 6 for 6500 slices obtained from 65 image volumes. Out of which, 5400 slices contained no collimator artifact and 1100 contained collimator artifact. The finetuned VGG model took more time for predictions than the custom model, also the finetuned model had less accurate predictions. For the finetuned VGG model, the gray images needed to be converted into 3-channels both for training and testing, which adds additional time.

Table 6: Accuracy and loss for training, validation, and testing. Testing results are for 6500 slices. "dp" stands for dropout and "aug" is data augmentation.

model	training		validation		testing		
	loss	acc	loss	acc	loss	acc	time (s)
vgg finetuned	0.00001	100	0.00002	100	0.3431	93.29	59.16
vgg finetuned + dp	0.0367	99.09	0.0127	99.75	0.6274	91.72	53.71
custom + dp	0.0101	99.66	0.00001	100	0.1612	96.52	42.36
custom + dp + aug	0.0741	97.05	0.0012	99.96	0.1402	96.84	42.45

We did further analysis for the custom model trained with dropout. Interestingly we observed that the artifact detection accuracy remained almost same for varying number of slices taken from the imaging volume. We took 10 to 100 slices (in increment of 10) from each of the 65 volumes and ran the model prediction. Theses slices were taken by putting the slice number 200 in the center. Time for the prediction reduced greatly, at the same time the prediction accuracy remained almost same (see Figure 26).

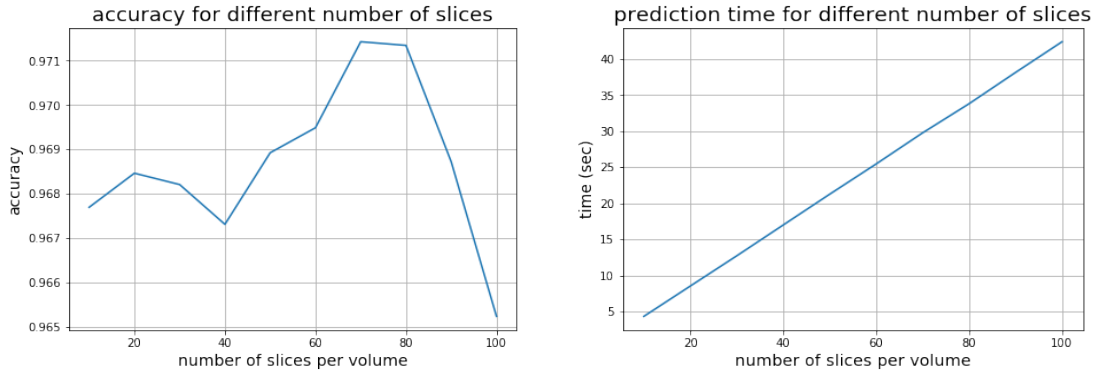

Figure 26: Change in slice-wise accuracy and prediction time for different number of slices. We took different slices from each volume and the model predicted artifact/no artifact for each slice. Even with 10 slices from 65 volumes, the accuracy is 96.7% and the prediction time is only 6.4 seconds.

Volume-wise artifact detection accuracy was 96.29% for 54 good volumes, that is, two volumes were predicted wrong. Also, for 11 bad volumes, the artifact detection

accuracy was 100 %. Volume-wise accuracies also remained same, regardless of the number of slices taken from the volume (see Figure 27).

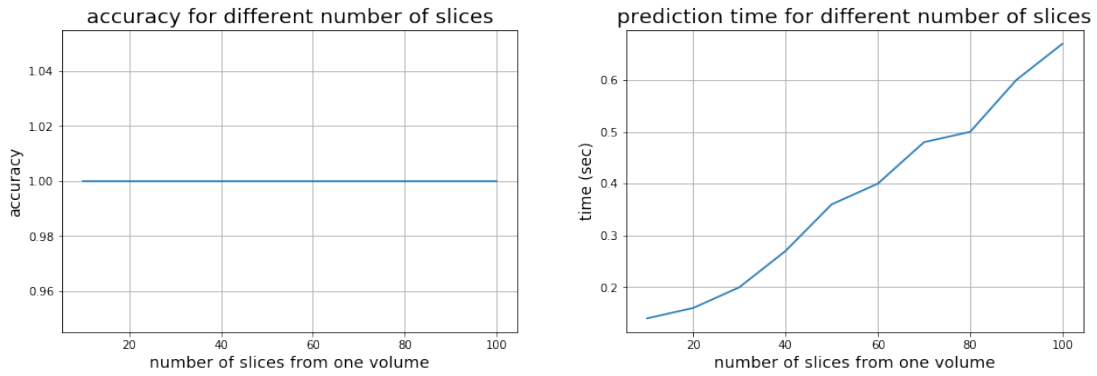

Figure 27: Change in volume-wise accuracy and prediction time for different number of slices. We took different slices from single good volume and the model predicted artifact/no-artifact for single volume. With 10 slices from the volume, prediction time is only 0.13 seconds and accuracy is still 100%.

The training curves of the custom model are shown below. The final loss and accuracy parameters are shown in Table 6 (custom + dp).

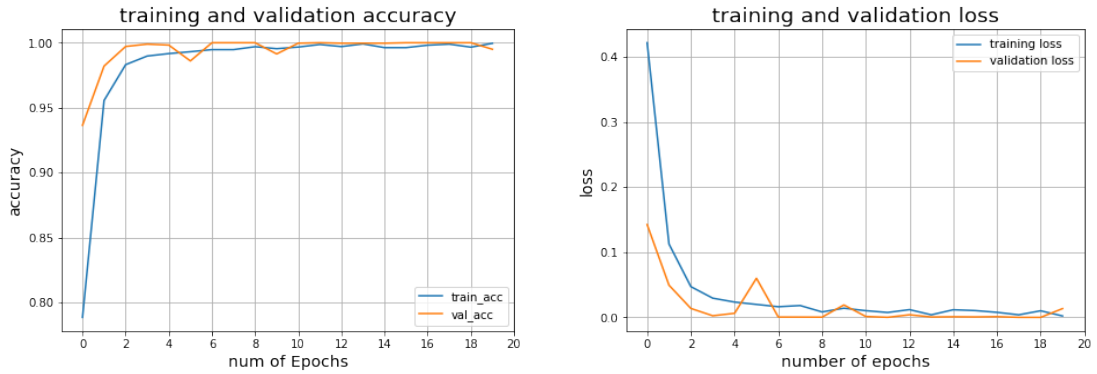

Figure 28: Accuracy and loss curves during training of custom model with dropout.

We visualised the activations of the network during the forward pass. For ReLU networks, the activations usually start out looking relatively blobby and dense, but as the training progresses the activations usually become more sparse and localized.

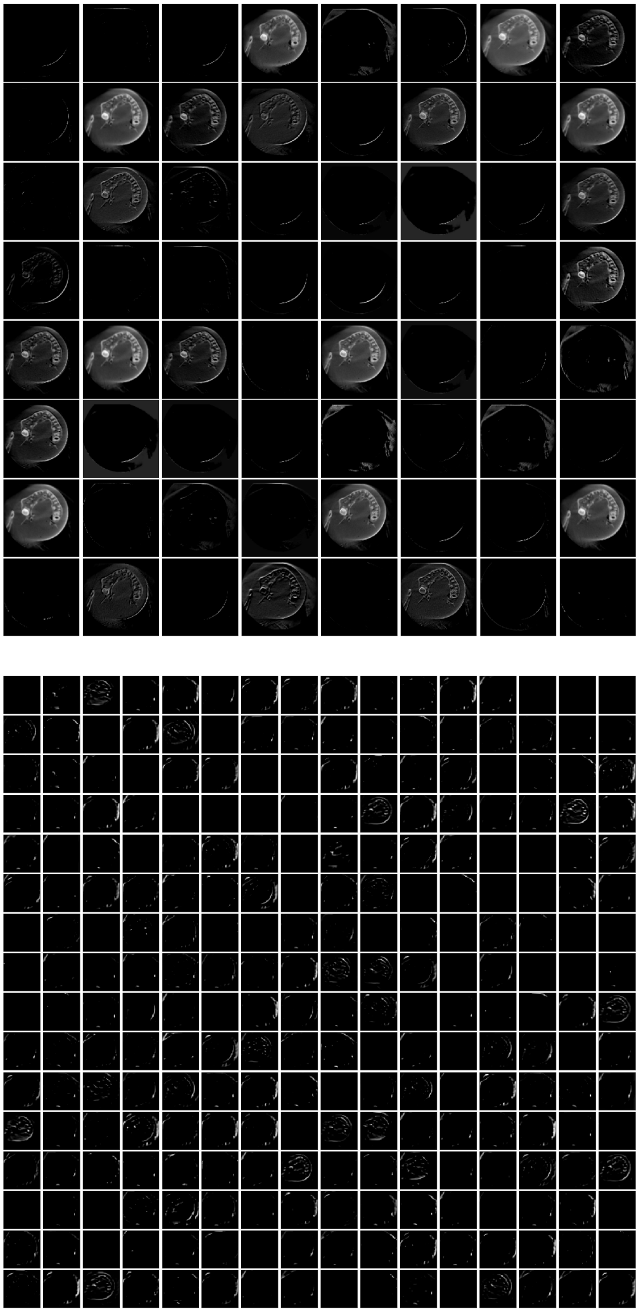

Figure 29: Activations in the first convolutional layer (upper), and the last convolutional layer (lower) of the trained custom model applied on a collimator artifact image. Every box represents an activation map corresponding to some filter. Activations are dense in the first layer but sparse in the last layer capturing arc like artifact.

5 Conclusions

5.1 Summary

Our objective was to test state-of-art deep learning networks for the artifact detection in dental X-ray images.

The motion detection is working well for canine as well as for jaw, and temporal bone images. The volume-wise accuracies are 100 % if the threshold for the motion in a volume is set as 80% slices with motion artifacts. The slice-wise accuracies are also above 90% for all image groups. While the motion detection accuracies of the model are quite impressive, the motion maps do not show the fine resolution motion. The major reason could be simplification in the labelling process. Also, the patch correction (ignoring the patches with many zero pixels) could be the reason of loosing information near the image borders. But at the same time it helps to ignore the black background. The mean F_1 score for canine, temporal bone and jaw slices were 1, 0.98, and 0.94, respectively.

For the collimator artifacts, 96% slice wise accuracy was achieved. The volume-wise accuracy was 98% (96.29% for good volumes and 100% for bad volumes). The convolutional layer activations were visibly picking up the arc-like collimator artifacts. We also demonstrated that with a robust model, the volume prediction accuracies can be obtained much faster (0.14 seconds) by examining only few slices (10) from the volume. The volume-wise prediction was the main objective for implementing the model in production tests.

5.2 Discussion and future work

The training data was acquired in the lab setting on the skull phantom which was a very important step. The results have shown that impressive accuracies can be obtained in X-ray fault detection with relatively limited set of data using deep learning methods. It has been seen that the transfer learning might not perform best in some cases. For us, it might be because of small and different training dataset than the ImageNet dataset. The model accuracies and robustness can be further improved with more data and training analysis but for our objectives, it already is giving pretty good results.

The immediate future objective of this work is to integrate the model with the existing software and further analyse the model robustness. Finally, we list some possible future applications of this work. This thesis is an excellent precursor for the automation of many visual detection based quality tests in the production as well as for the automation of quality assurance tests at the diagnostics facility performed by the customer or service engineer. Many times a field engineer has to do the failure diagnostics on the basis of images. This thesis work can be extended to the applications where the failure diagnostics could be supported using deep learning models. Also, the algorithm could be run at local level on the machine and may be used to monitor the quality of the images during the normal usage. This can help detect the degradation of images which might support proactive diagnosis of

upcoming failures. The fault detection is not limited to only the analysis of visual images. The deep learning applications can easily be extended to other forms of fault data, hidden for example, in signals from motor movements or machine's error logs, which has the potential to completely transform the productivity of the machine and customer satisfaction.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I. J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D. G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P. A., Vanhoucke, V., Vasudevan, V., Viégas, F. B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Alamari, H. M., Sadrameli, M., Alshalhoob, M. A., Sadrameli, M., and Alsheri, M. A. (2012). Applications of CBCT in dental practice: a review of the litrature. *General Dentistry*, 60(5):390–400.

Alemzadeh, H., Iyer, R. K., and Kalbarczyk, Z. (2013). Analysis of safety-critical computer failures in medical devices. *IEEE Security & Privacy*, 11(4):14–26.

Alex, K., Ilya, S., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, volume 1, pages 1097–1105.

Arai, Y., Tammisalo, E., Iwai, K., Hashimoto, K., and Shinoda, K. (1999). Development of a compact computed tomographic apparatus for dental use. *Dentomaxillofacial Radiology*, 28(4):245–248.

Babaei, M., Shi, J., and Abdelwahed, S. (2018). A survey on fault detection, isolation and reconfiguration methods in electric ship power systems. *IEEE Access*, 6:9430–9441.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer, 1^{st} edition.

Brunel, N., Hakim, V., and Richardson, M. J. (2014). Single neuron dynamics and computation. *Current Opinion in Neurobiology*, 25:149–155.

Bushberg, J. T., Seibert, J. A., Leidholt, E. M., and Boone, J. M. (2012a). Computed tomography. In *The essential physics of medical imaging*, chapter 10, pages 312–374. Lipponcott Williams & Wilkins, 3^{rd} edition.

Bushberg, J. T., Seibert, J. A., Leidholt, E. M., and Boone, J. M. (2012b). X-ray production, X-ray tubes and X-ray generators. In *The essential physics of medical imaging*, chapter 6, pages 171–206. Lipponcott Williams & Wilkins, 3^{rd} edition.

Cheng, J. Z., Ni, D., Chou, Y. H., Qin, J., Tiu, C. M., Chang, Y. C., Huang, C. S., Shen, D., and Chen, C. M. (2016). Computer-aided diagnosis with deep learning architecture: applications to breast lesions in us images and pulmonary nodules in ct scans. *Scientific Reports*, 15(24454).

Cho, Y., Moseley, D. J., Siewerdsen, J., and Jaffray, D. A. (2005). Accurate technique for complete geometric calibration of cone-beam computed tomography systems. *Medical Physics*, 32(4):968–983.

Chollet, F. (2018). Keras : The Python deep learning library. https://keras.io/. Last accessed: 15 june 2018.

Daly, M. J., Siewerdsen, J. H., Cho, Y. B., Jaffray, D. A., and Irish, J. C. (2008). Geometric calibration of mobile C-arm for intraoperative cone beam CT. *Medical Physics*, 35(5):2124–2136.

Deutsch, J. and He, D. (2017). Using deep learning based approaches for bearing remaining useful life prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):11–20.

Ding, X. and He, Q. (2017). Energy-fluctuated multiscale feature learning with deep convnet for intelligent spindle bearing fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, 66(8):1926–1935.

Dong, C., Loy, C. C., He, K., and Tang, X. (2016). Image super resolution using deep convolutional networks. *IEEE transcations on Patern Analysis and Machine Intelligence*, 38(2):295–307.

Doyle, S. L., Azevedo, B., Levin, M. D., Gane, D., Farman, A. G., and Scarfe, W. C. (2018). Endodontic applications of CBCT. In *Maxillofacial Cone Beam Computed Tomography*, chapter 22, pages 871–922. Springer.

Farman, A. G. and Scarfe, W. C. (2018). Historical perspectives on CBCT. In *Maxillofacial Cone Beam Computed Tomography: Principles, Techniques and Clinical Apllications*, pages 3–11. Springer.

Feldkamp, L. A., Davis, L. C., and Kress, J. W. (1984). Practical cone-beam algorithm. *Journal of the Optical Society of America*, 1(6):612–619.

Flohr, T. G., Schaller, S., Stierstorfer, K., Bruder, H., Ohnesorge, B. M., and Schoepf, U. J. (2005). Multi detector row CT systems and image reconstruction techniques. *Radiology*, 235(3):756–773.

Glasser, O. (1995). W. C. Röntgen and the discovery of the Röntgen rays. *American Journal of Roentgenology*, 165(5):1033–1040.

Glavaski, S. and Elgersma, M. (2001). Active aircraft fault detection and isolation. In *IEEE Autotestcon Proceesings. IEEE systems Readiness Technology Conference*, pages 692–705.

Goldman, L. W. (2007). Principles of CT and CT technology. *Journal of Nuclear Medicine Technology*, 35(3):115–128.

Goodfellow, I., Bengio, Y., and Courville, A. (2016a). Autoencoders. In *Deep Learning*, chapter 14, pages 493–516. MIT press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016b). *Deep Learning*. MIT press.

Granholm, M. (2018). Utilization of cloud services for visual fault diagnostics of dental X-ray systems. Master's thesis, Aalto University.

Gulsahi, A. (2011). Bone quality assessment for dental implants. In *Implant Dentistry: The Most Promising Discipline of Dentistry*, pages 437–452. InTech.

Gulshan, V., Peng, L., Coram, M., Stump, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., and Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22):2402–2410.

Hanzelka, T., Dusek, J., Ocasek, F., Kucera, J., Sedy, J., Benes, J., Pavlikova, G., and Foltan, R. (2013). Movement of the patient and the cone beam computed tomography scanner: objectives and possible solutions. *Oral Surgery, Oral Medicine, Oral Pathology and Oral Radiology*, 16(6):769–773.

Harrel Jr, W. E., Scarfe, W. C., Pinheiro, L. R., and Farman, A. G. (2018). Applications of CBCT in orthodontics. In *Maxillofacial Cone Beam Computed Tomography*, chapter 18, pages 645–714. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2008a). Overview of supervised learning. In *The Elements of Statistical Learning*, chapter 2, pages 9–41. Springer, 2^{nd} edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2008b). Unsupervised learning. In *The Elements of Statistical Learning*, chapter 14, pages 485–585. Springer, 2^{nd} edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hiriyannaiah, H. P. (1997). X-ray computed tomography for medical imaging. *IEEE Signal Processing Magazine*, 144(2):42–59.

Hirvonen-Kari, M. (2013). Clinical audit and quality assurance in the imaging process. Academic dissertaion, University of Helsinki.

Hsieh, J. (2015). Preliminaries. In *Computed tomography: principles, design ,artifacts and recent advances*, chapter 2, pages 25–32. SPIE press, 3^{rd} edition.

Hsieh, J., Nett, B., Yu, Z., Sauer, K., Thibault, J. B., and Bouman, C. A. (2013). Recent advances in CT image reconstruction. *Current Radiology Reports*, 1(1):39–51.

Isermann, R. (2005). Model base fault detection and diagnostis status and applications. *Annual Reviews in Control*, 29(1):71–85.

Jacobs, R. (2011). Dental cone beam CT and its justified use in oral health care. *Journal of the Belgian Society of Radiology*, 94(5):254–265.

Jaffray, D. A., Siewerdsen, J. H., Wong, J. W., and Martinez, A. A. (2002). Flat panel cone-beam computed tomography for image-guided ratiation therapy. *International Journal of Radiation Oncology, Biology, Physics*, 53(5):1337–1349.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kak, A. and Slaney, M. (1988a). Algorithms for reconstruction with nondiffracting sources. In *Principle of computerized tomographic imaging*, chapter 3, pages 49–112. IEEE Press.

Kak, A. and Slaney, M. (1988b). *Principle of computerized tomographic imaging*. IEEE Press.

Katipamula, S. and Brambley, M. R. (2005a). Methods for fault detection, diagnostics and prognostics for building systems-a review, part I. *International Journal of HVAC & R Research*, 11(1).

Katipamula, S. and Brambley, M. R. (2005b). Methods for fault detection, diagnostics and prognostics for building systems-a review, part II. *International Journal of HVAC & R Research*, 11(2).

Ker, J., Wang, L., Rao, J., and Lim, T. (2018). Deep learning applications in medical image analysis. *IEEE Access*, 6:9375–9389.

Kingma, D. P. and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kohavi, R. (1995). A study of cross validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the* 14^{th} *International Joint Conference on Artificial intelligenceInternational Joint Conference on Artificial intelligence*, volume 2, pages 1137–1143.

Küstner, T., Liebgott, A., Mauch, L., Martirosian, P., Bamberg, F., Nikolaou, K., Yang, B., Schick, F., and Gatidis, S. (2018). Automated reference-free detection of motion artifacts in magnetic resonance images. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 31(2):243–256.

Lakhani, P. and Sundaram, B. (2017). Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324.

Lee, R. D. (2008). Comman image artifacts in cone beam CT. *American Association of Dental Maxillofacial Radiographic Technicians Newsletter*, pages 1–8.

Li, F. F. (2018). Cs231n: Covolutional neural networks for visual recognition. http://cs231a.stanford.edu. Last accessed: 07 june 2018.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Laak, J. A. W. M., Ginneken, B. V., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88.

London, M. and Häusser, M. (2005). Dendritic computation. *Annual Review of Neuroscience*, 28:503–532.

López, B., Meléndez, J., Wissel, H., Haase, H., Laatz, K., and Grosser, O. S. (2009). Towards medical device maintenance workflow monitoring. *International Journal of Health and Medical Engineering*, 3(6):619–625.

Lu, C., Wang, Z. Y., Qin, W. L., and Ma, J. (2017). Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification. *Signal Processing*, 130:377–388.

Makins, S. R. (2014). Artifacts interfering with interpretation of cone beam computed tomography images. *Dental Clinics of North America*, 58(3):485–498.

Mamatha, J., Chaitra, K. R., Paul, R. K., George, M., Anitha, J., and Khanna, B. (2015). Cone beam computed tomography - dawn of a new imaging modality in orthodontics. *Journal of International Oral Health*, 7(Suppl 1):96–99.

McCulloch, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–113.

McKetty, M. H. (1998). The AAPM/RSNA physics tutorial for residents. X-ray attenuation. *RadioGraphics*, 18(1).

Meding, K., Loktyushin, A., and Hirsch, M. (2017). Automatic detection of motion artifacts in MR images using CNNs. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Mehr, M. T. (2013). Usefulness of dental cone beam computed tomography (CBCT) for detection of the anatomical landmarks of the external, middle and inner ear. Master's thesis, University of Iowa.

Mozzo, P., Procacci, C., Tacconi, A., Martini, P. T., and Andreis, I. A. (1998). A new volumetric CT machine for dental imaging based on the cone-beam technique: preliminary. *European Radiology*, 8(9):1558–1564.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. MIT press, 1^{st} edition.

M.Yuan, Y.Wu, and Lin, L. (2016). Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network. In *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, pages 135–140.

Nagarajappa, A. N., Dwivedi, N., and Tiwari, R. (2015). Artifacts: the downturn of CBCT image. *Journal of International Society of Preventive and Community Dentistry*, 5(6):440–445.

Nardi, C., Molteni, R., Lorini, C., Taliani, G. G., Matteuzzi, B., Mazzoni, E., and Colagrande, S. (2016). Motion artefacts in cone beam CT: an in vitro study about the effects on the images. *British Journal of Radiology*, 89(1058):20150687.

Nemtoi, A., Czink, C., Haba, D., and Gahleitner, A. (2013). Cone beam CT: a current overview of devices. *Dentomaxillofacial Radiology*, 42(8):20120443.

Nguyen, T. (2013). Total number of synapses in the adult human neocortex. *Undergraduate Journal of Mathematical Modeling: One + Two*, 3(1).

Nielsen, M. (2018). How the backpropagation algorithmn works. http://neuralnetworksanddeeplearning.com/chap2.html. Last accessed: 07 june 2018.

Oppenheimer, C. H. and Loparo, K. A. (2002). Physically based diagnosis and prognosis of cracked rotor shafts. In *Proceedings SPIE 4733, Component and Systems Diagnostics, Prognostics, and Health Management II*, volume 4733, pages 122–132.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS-Workshop*.

Pauwels, R., Araki, K., Siewerdsen, J. H., and Thongvigitmanee, S. S. (2015). Technical aspects of dental CBCT: state of the art. *Dentomaxillofacial Radiology*, 44(1).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Planmeca Oy (2017). *Planmeca ProMax 3D Plus & 3D Mid user's manual 3D imaging*, 15^{th} edition.

Rezaei, M., Yang, H., and Meinel, C. (2017). Deep neural network with l^2-norm unit for brain lesions detection. In *International Conference on Neural Information Processing*, pages 789–807.

Rout, J. and Brown, J. (2013). Ionizing radiation regulations and the dental practitioner: 2. regulations for the use of X-rays in dentistry. *Dental Update*, 39:248–253.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Sakamoto, M., Nishiyama, H., Satoh, H., Shimizu, S., Sanuki, T., Kamijoh, K., Watanabe, A., and Asahara, A. (2005). An implementation of the Feldkamp algorithm for medical imaging on cell. *White Paper in IBM Technology Group Library (http://www-306.ibm.com/chips/techlib/techlib.nsf). Last accessed: 15 june 2018*.

Saxena, P. and Kumar, R. S. (2018). Restoration of CT images corrupted with fixed valued impulse noise using an optimum decision-based filter. In *Intelligent Multidimensional Data and Image Processing*, chapter 8, pages 220–239. IGI Global.

Scarfe, W. C. and Farman, A. G. (2007). Cone beam computed tomography: a paradigm shift for clinical dentistry. *Austrailian Dental Journal*, 19:92–100.

Scarfe, W. C. and Farman, A. G. (2008). What is cone-beam CT and how does it work? *The Dental Clinics of North America*, 52(4):707–730.

Scarfe, W. C., Farman, A. G., and Sukovic, P. (2007). Clinical applications of cone-beam computed tomography in dental practice. *Journal of Canadian Dental Association*, 72(1):75–80.

Scarfe, W. C., Levin, M. D., Gane, D., and Farman, A. G. (2009). Use of cone beam computed tomography in endodontics. *International Journal of Dentistry*, 634567:440–445.

Scarfe, W. C., Li, Z., Aboelmaaty, W., Scott, S. A., and Farman, A. G. (2012). Maxillofacial cone beam computed tomography: essence, elements and steps to interpretation. *Austrailian Dental Journal*, 57(Suppl 1):46–60.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, pages 92–101.

Scherl, H. (2012). Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction. *Parallel Computing*, 38(3):111.

Schulze, R., Heil, U., Gro, D., Bruellmann, D. D., Dranischnikow, E., Schwanecke, U., and Schoemer, E. (2011). Artifacts in CBCT: a review. *Dentomaxillofacial Radiology*, 40(5):265–273.

Shah, N., Bansal, N., and Logani, A. (2014). Recent advancements in imaging technologies in dentistry. *World Journal of Radiology*, 6(10):794–807.

Sheng, C., Li, Z., Qin, L., Guo, Z., and Zhang, Y. (2011). Recent progress on mechanical condition monitoring and fault diagnosis. *Procedia Engineering*, 15:142–146.

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nougues, I., Yao, J., Mollura, D., and Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *arXiv preprint arXiv:1602.03409*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smolensky, P. S. P. S. P. S. P. S. P. S. P. S. P. (1986). *Information processing in dynamical systems: foundations of harmony theory*, volume 1, chapter Parallel distributed processing: explorations in the microstructure of cognition, pages 194–281. MIT Press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sun, W., Shao, S., Zhao, R., Yan, R., Zhang, X., and Chen, X. (2016). A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement*, 89:171–178.

Swinehart, D. F. (1962). The Beer-Lambert law. *Journal of Chemical Education*, 39(7):333–335.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tajbakhsh, N., Shin, J., Gurudu, S., Hurst, R. T., Kendall, C. B., and andJ. Liang, M. G. (2016). Convolutional neural networks for medical image analysis: full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312.

Theano Development Team (2016). Theano: a Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Turbell, H. (2001). *Cone-Beam reconstruction using filtered backprojection.* PhD thesis, Linköping University.

Vandenberghe, B., Jacobs, R., and Bosmans, H. (201). Modern dental imaging: a review of the current technology and clinical applications in dental practice. *European Radiology*, 20(11):2637–2655.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 7068349.

Wallace, D. R. and Kuhn, D. R. (1999). Lessons from 342 medical device failures. In *Processings 4th IEEE International Symposium on High-Assurance Systems Engineering*, pages 123–131.

Wallace, D. R. and Kuhn, D. R. (2001). Failure modes in medical device software : an analysis of 15 years of recall data. *International Journal of Reliability, Quality and Safety Engineering*, 8(4).

Wang, B., Yager, K., Yu, D., and Hoai, M. (2017). X-ray scattering image classification using deep learning. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 697–704.

Wang, J. and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621.*

Wang, L., Liu, T., Wang, G., Chan, K. L., and Wang, Q. (2015). Video tracking using learned hiearchical features. *IEEE transactions on Image Processing*, 24(4):1424–1435.

Wei, Y., Wang, G., and Hsieh, J. (2005). An intuitive discussion on the ideal ramp filter in computed tomography. *Computers & Mathematics with Applications*, 49(5-6):731–740.

Wu, H. and Gu, X. (2015). Towards dropout training for convolution neural networks. *Neural Networks*, 71:1–10.

Xiao, S., Bresler, Y., and Munson, D. C. (2003). Fast Feldkamp algorithm for cone-beam computer tomography. In *Proceedings 2003 International Conference on Image Processing*, volume 3, pages 819–822.

Yang, K., Kwan, A. L., Miller, D. F., and Boone, J. M. (2006). A geometric calibration method for cone-beam CT systems. *Medical Physics*, 33(6):1695–1706.

Zhang, Y. and Yu, H. (2018). Convolutional neural network based metal artifact reduction in X-ray computed tomography. *arXiv preprint arXiv:1709.01581v2.*

Zhao, R., Wang, J., Yan, R., and Mao, K. (2016a). Machine helath monitoring with LSTM networks. In *IEEE 10th International Conference on Sensing Technology*, pages 1–6.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X. (2016b). Deep learning and its applications to machine health monitoring: a survey. *arXiv preprint arXiv:1612.07640v1*.

Zhou, Y. and Chellappa, R. (1988). Computaion of optical flow using a neural network. In *IEEE 1988 International Conference on Neural Networks*, pages 71–78.

Zhou, Z., Shin, J., Zhang, L., Gurudu, S., Gotway, M., and Liang, J. (2017). Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4761–4772.