Friedrich-Alexander-Universität

Pattern Recognition Lab

Deep Learning-based detection of detector artifacts

Master's Thesis in Data Science

submitted
by

Vrinda Gupta

born 19.06.1999 in Pusad, India

Written at

Lehrstuhl für Mustererkennung (Informatik 5)
Department Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg.

Advisor: Dr.-Ing. Vincent Christlein, Florian Kordon, M.Sc., Prof. Dr.-Ing. habil. Andreas Maier - Department of Computer Science
Dr. Thorsten Ergler, Dr. Bastian Schmidt - Siemens Healthineers

Started: 01.06.2023

Finished: 01.12.2023

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Confidentiality Clause This master thesis contains confidential data of the company Siemens Healthineers AG. This thesis may only be made available to the first and second reviewers and authorized members of the board of examiners. Any publication and duplication of this master thesis – even in part – is prohibited. An inspection of this thesis by third parties requires the expressed permission of the author and company Siemens Healthineers AG.

Erlangen, den 01.12.2023

Vrinda Gupta

Acknowledgements

First and foremost, I express my gratitude to God for the blessings and strength provided throughout this journey.

I am deeply thankful to Dr. Thorsten Ergler for his belief in me and for providing me with the invaluable opportunity to conduct my thesis at Siemens Healthineers. Special appreciation goes to Dr. Bastian Schmidt, not only for his role in onboarding me to the team but also for his exceptional patience in familiarizing me with all the medical terms. His continuous sharing of knowledge and experience has been instrumental to my growth.

I extend my appreciation to Dr. Vincent Christlein for his valuable insights and constant support. A big thanks to Mr. Florian Kordon without whom, this thesis wouldn't have been possible. I thank him for his unwavering cooperation and, guidance, for sharing his immense knowledge, and for promptly addressing my queries. I am grateful to Friedrich-Alexander-Universität, Erlangen-Nürnberg for granting me this opportunity to write my thesis and giving me access to the HPC cluster to facilitate the execution of my experiments.

A warm thank you to my roommates, Ridhima Garg and Nishaat Shaikh, for creating a stress-free environment and providing moments of joy. Heartfelt thanks to my friends and colleagues for their support and well wishes. Special thanks to Rajesh Madhipati, Vatsal Bambhaniya, Raj Bais, and Mijanur Rahman for engaging in discussions and offering their creative insights.

Lastly, I want to dedicate my deepest thanks to my parents. I hope I made them proud. Their constant prayers and support have played an integral role in my journey. Thank you for encouraging me in all my pursuits and inspiring me to follow my dreams. Without you, I could never be where I am today.

Vrinda Gupta

Abstract

Images acquired with an X-ray flat panel detector can show various artifacts, even after the usual processing to correct for typical electronic variations. These artifacts comprise of e.g. failures or deviations of individual lines, columns, segments, or other patterns. Often these artifacts occur only sporadically, which poses a huge challenge in the diagnosis, as they possibly cannot be reproduced by a service technician dispatched to a customer site and associated raw data might no longer be available. To allow for better diagnosis, automation through Artificial Intelligence (AI) for faster identification, and automated system health assessment would significantly enhance the diagnostic process. In this thesis, we use Deep Learning (DL) to classify raw X-ray images for artifacts obtained from a specific flat-panel detector. Our methodology involves employing both image-based and time series-based approaches. The image-based approach utilizes Convolutional Neural Network (CNN) architecture, achieving 97% of validation accuracy. The time series-based approach involves acquiring line profiles from images and using our custom architecture - 1DCNN for classification. This time-series-based approach achieves 96% of validation accuracy with significantly lower computational power and fewer pre-processing steps. We also analyse the effectiveness of a one-class classification approach using generative modeling to classify previously unseen defective images. We enhance interpretability by incorporating Gradient-weighted Class Activation Mapping (Grad-CAM) and saliency maps to gain insights into our model's decision-making process. Additionally, we evaluate the adaptability of the model trained on raw X-ray images for direct classification of post-processed X-ray images without retraining. Overall, our thesis contributes to the advancement of AI-driven quality assurance in X-ray imaging.

Contents

Chapter 1

Introduction

A **radiograph** is an image produced on a radiosensitive surface, such as a detector, by passing X-rays through an object [Stab].

The discovery of X-rays by Roentgen in 1895 transformed the field of medicine and had a significant impact on other scientific fields. Without requiring intrusive procedures, X-ray imaging established itself as a useful diagnostic tool for physicians. It allowed them to view inside the human body. It also found use in a variety of industries, such as non-destructive testing (NDT) of materials and quality control.

Radiography is **bread and butter** of medical imaging and is still the **most widely used imaging modality** [Bou10]. The data presented in Figure 1.1 provides a visual representation of the distribution of different imaging modalities within the *Diagnostic Imaging Dataset*. This dataset encompasses information on diagnostic imaging tests conducted on *National Health Service (NHS)* patients in England [1].

All medical imaging modalities like Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and X-ray (radiography) are subjected to spurious findings, leading to missed or erroneous diagnoses. It is worth emphasizing that the **quality assurance** of X-ray systems holds a position of paramount importance. Ensuring the reliability and accuracy of X-ray equipment is vital due to its widespread usage and diagnostic significance.

The key elements of the radiographic image quality include contrast, noise, spatial resolution, and artifacts [Kru+07]. An **artifact** can be referred to as a feature in an image that either mask, imitates a clinical feature, or degrade it's quality [Wil+04] [JIM+08]. An article by Walz-Flannigan et al.[Wal+18] categorizes artifacts in a structured manner by

[1] Diagnostic Imaging Dataset published by the NHS England. Available from: https://www.england.nhs.uk/statistics/ [accessed November 2023]

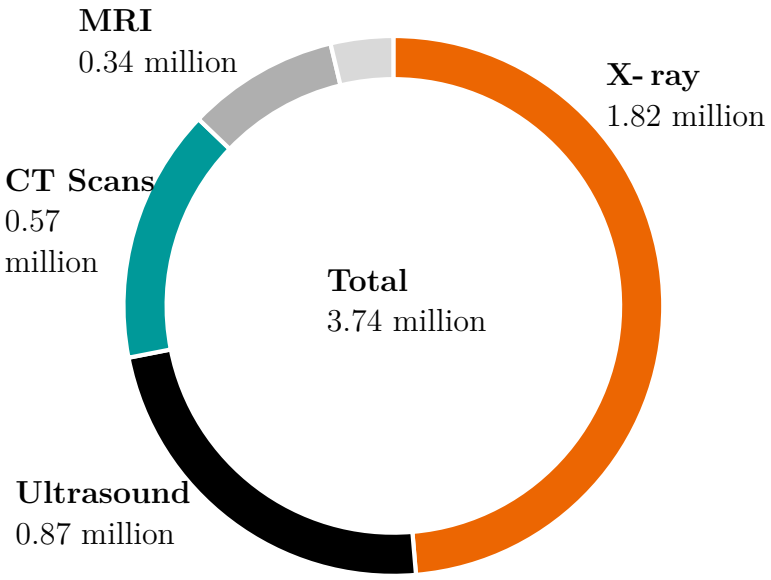


Figure 1.1: From the 3.74 million imaging tests that were taken in England in June of 2023, around 50 percent were X-ray [Staa].

classifying each artifact example based on the specific type of failure mechanism encountered within the digital radiographic imaging chain. The article encompasses a wide range of artifact examples, including those stemming from equipment defects, disruptions caused by accidents or mishandling, issues related to debris or gain calibration, challenges tied to X-ray acquisition techniques, and problems originating from image processing. For more details on artifacts that can occur in radiographic X-ray image acquisition [JIM+08],[DRO+08] can be referred to.

Certain artifacts are readily identifiable in clinical images. They often manifest as obstructions within the image, either obscuring specific areas or interfering with the intended field of view. It is crucial for healthcare professionals, including physicists, radiologists, and radiology technologists, to develop a thorough visual understanding of the diverse range of artifacts [Wal+12]. This familiarity is essential for effectively identifying, preventing, or resolving issues caused by image artifacts in inpatient imaging. These distractions in radiographic images have the potential to compromise the accuracy of diagnoses.

Some types of artifacts can appear uniquely when using digital radiographic systems based on **flat-panel detectors**. These artifacts may encompass those associated with detector gain calibration or interruptions in the readout circuitry, as well as other artifacts linked to image processing. Detectors exhibit variations in sensitivity and uniformity across their surface. The process of flat-field calibration is performed to ensure that the detector

responds consistently to X-rays across its entire surface [Sei+98]. To perform detector flat-field calibration, a uniform and stable radiation source (often an X-ray beam with consistent intensity) is directed at the detector. When the detector is exposed to this uniform radiation, it records a 'flat-field image' or 'calibration image'. Based on the analysis of the flat-field image, a correction map or matrix is generated. This map outlines the areas where sensitivity variations exist and quantifies the degree of correction needed for each region. When acquiring clinical images (e.g., patient X-rays), the correction map is applied to the raw data obtained by the detector. This correction compensates for the sensitivity variations, ensuring that the final diagnostic image represents a more accurate and uniform response of the detector to the radiation. The detector flat-field calibration helps to improve image quality by minimizing artifacts and inconsistencies caused by variations in detector sensitivity. However, it's important to note that **flat-field calibration may not eliminate all artifacts**. In some cases, interpolation techniques are employed to address specific artifacts.

Despite these efforts, certain artifacts may persist even after calibration and interpolation. In this thesis, we are going to deal with the identification of those remaining artifacts.

1.1 Goal of Thesis

Currently in Siemens, when a customer reports a defective X-ray image, a service technician is dispatched to the site. The technician performs a manual diagnosis to determine if the artifact is due to a broken detector or a different component [see Figure 1.2]. For some artifacts, the technician may perform re-calibration to solve the issue. But, when the technician is uncertain about their findings, images are collected and sent to a Siemens service center for further analysis. Ultimately, if the detector is found to be faulty, it is replaced.

This process relies heavily on the expertise and judgment of the service technician, leading to potential delays and increased costs associated with unnecessary service calls and equipment transportation. It is evident that this approach is not the most efficient or cost-effective method for addressing defective X-ray images. This dependence on human intervention results in significant costs, both in terms of time and resources.

However, in many cases, the technician cannot reproduce the issue, because it is highly sporadic and only happens once a week or even less. In these cases, a detector is replaced but no data is available anymore and it is considered fine during analysis at Siemens/supplier.

This thesis work is significant because it has the potential to completely transform quality assurance and diagnostic procedures within Siemens. This work focuses on the **development of AI based algorithm** capable of automatically detecting defective X-ray images. Eventually, the idea is to integrate AI model within the X-ray system at the customer's site that will provide real-time alerts to Siemens when a defect is identified, streamlining the diagnostic process. So, there is the capability to detect recurring defects in X-ray images, signaling the need for detector replacement before customers even have the chance to register a complaint. Also, the trigger would be sent automatically to store (and send to the service center) images when an artifact is detected. In this case, diagnosis can be much simpler and all relevant data is available, especially for artifacts, which happen extremely seldom.

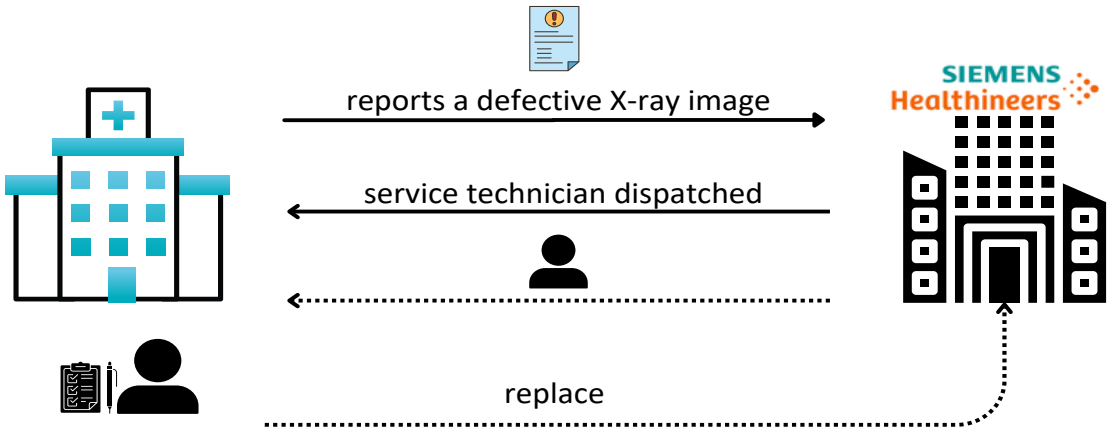


Figure 1.2: Shows manual approach of addressing defective X-ray images in Siemens.

By developing and integrating AI-based algorithms capable of automatically detecting artifacts in X-ray images, this thesis aims to address these critical issues and deliver the following significant benefits:

(1) **Enhanced diagnostic efficiency**: The AI system will eliminate the need for service technicians to perform manual diagnoses. This will result in *faster identification and resolution of issues*, reducing downtime for healthcare facilities and improving patient care.

(2) **Cost saving**: The reduction in service technician visits, recalibrations, and unnecessary repeated site visits will lead to substantial cost savings for both Siemens and its customers. The AI-based solution will enable targeted interventions, minimizing the replacement of detectors when not needed.

(3) **Data-driven insights**: The AI system will trigger storage of defective images. Siemens can use this information to obtain an understanding of system performance trends, enabling proactive maintenance and product improvements.

In summary, this project not only contributes to the advancement of medical imaging technology but also holds the potential to significantly benefit Siemens by improving operational efficiency and reducing costs.

1.2 Problem Statement

This thesis focuses on the analysis of digital radiographic images that have been impacted by **technical artifacts** arising from the **failure or malfunction** of a specific flat-panel detector. Detailed information about the detector and the types of artifacts under consideration can be found in the relevant section of the dataset Chapter 5.

Within the context of this thesis report, an X-ray image containing artifacts is labeled as a 'defective' image, while an image devoid of any artifacts is referred to as a 'good' image.

The following are our study's objectives:

- The development of an AI-based algorithm designed to identify whether a given X-ray image has been affected by an artifact. Additionally, when possible, the algorithm aims to determine the specific type of artifact present.
- Addressing the issue of an imbalanced dataset, where there are more 'good' images than 'defective' images. This will be accomplished by generating synthetic training data through data augmentation techniques and employing one-class classification methods.
- Incorporate an element of explainability into the model to provide insights into the decision-making process, making it more understandable for end-users and stakeholders.
- Assessing the performance of the algorithm on post-processed images to verify if it is effective for both raw (intensity linear) and post-processed X-ray images.

These objectives collectively aim to develop a robust and versatile AI-based solution for artifact detection in X-ray images, addressing challenges related to imbalanced datasets and extending the model's applicability to post-processed images. The inclusion of explainability

ensures not only accuracy but also transparency in the model's decision-making, crucial for building trust and understanding in practical applications.

To the best of our knowledge, our X-ray artifact detection in radiography represents a pioneering approach, being the first to introduce AI-driven techniques for the identification of image artifacts in X-ray images.

1.3 Thesis Organization

This work of thesis covers eight chapters, whose content is presented below. Beginning with primary theoretical Chapter 2, we introduce basics of X-ray image acquisition and digital imaging. Then in Chapter 3, we delve into the basics of DL and introduce all those methodologies that are part of our thesis. We also discuss the literature where deep learning has been applied to X-ray images and for artifact detection in medical imaging in Chapter 4. In the Chapter 5, we discuss in detail the dataset provided to us and the pre-processing steps employed. Chapter 6 focuses on the experiment, its discussion, and evaluation. The Chapter 7 explains the comparative study of all approaches. The thesis concludes in Chapter 8 with a final discussion offering insights on the strengths and limitations of each approach, future approaches, and final conclusion.

Chapter 2

X-ray

X-rays are high energy electromagnetic (EM) radiation, commonly known as Röntgen radiation, named after the German scientist **Wilhelm Conrad Röntgen**, who discovered it in 1895 at the University of Würzburg [Als+11]. Röntgen made this groundbreaking discovery while conducting experiments with cathode rays (electrons) in a vacuum tube when he noticed a faint, mysterious glow produced by a fluorescent screen in his lab, even though the tube was supposed to be shielded. He investigated this phenomenon further and discovered that it was caused by a type of high-energy EM radiation that could penetrate various materials, including human tissue, and create images on photographic plates. He called this newly discovered radiation 'X-ray' because 'X' represented the unknown nature of the radiation at the time [Nov97]. Figure 2.1 shows the energy level in EM spectrum at which X-rays operate.

One of the initial X-ray pictures ever taken depicted the hand of Röntgen's spouse, capturing her wedding ring and the inner structure of her bones is shown in Figure 2.2.

2.1 X-ray Image Acquisation

A basic X-ray setup includes essential components such as an X-ray generator, collimator, the object or patient being examined, film, and an intensifying screen Figure 2.3. An **X-ray generator** consists of a high voltage generator, X-ray tube, control console, and a coolant. An **X-ray tube** is a basic vacuum tube comprising a cathode, which directs a flow of electrons into a vacuum, and an anode made of tungsten to dissipate the heat generated by the electron collision. When these electrons collide with the target, approximately 1% of the energy produced is emitted as X-rays, while the remaining 99% is released as heat,

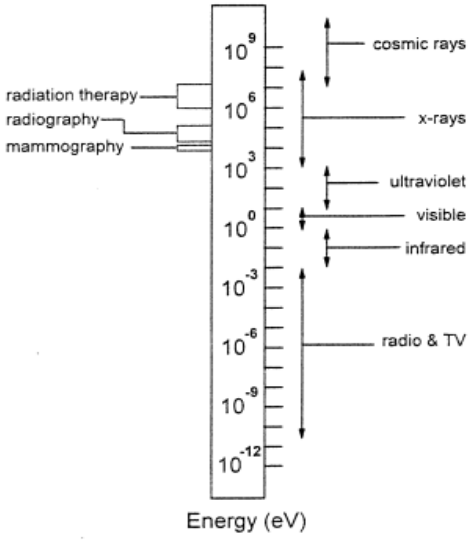


Figure 2.1: EM spectrum showing energy range for radiography Source: [Bou10]

cooled using recirculating systems that involve water or oil. The voltage between the anode and cathode regulates the speed of electrons striking the anode. The higher the voltage, the higher is the energy of X-ray radiation. **Control console** enables precise control over the imaging process, ensuring safety, and facilitating the efficient capture of X-ray images. It controls the voltage, current, and exposure time of the tube.

Figure 2.2: Earliest extand X-ray image of Röntgen's spouse's hand. Courtesy: Deutsches Roentgen Museum, Germany

A **collimator** is an essential component in X-ray imaging systems, and its primary role is to restrict the X-ray beam to a specific size and shape. They ensure that only the desired area or region of interest (ROI) is exposed to radiation. This helps in minimizing unnecessary radiation exposure to surrounding tissues and improves image quality by reducing scattered radiation. **Radiographic film** plays a crucial role in capturing X-ray

and gamma-ray images. It consists of radiation-sensitive silver halide crystals. When X-rays or gamma rays strike these crystals, a latent image is formed.

Until the end of the previous century, the majority of medical imaging procedures relied on film for capturing, displaying, and storing images. However, the digital transformation of medical diagnostic imaging started as early as the 1970s with the invention of the CT scanner. This was followed by the introduction of MRI in the 1980s and digital X-ray acquisition systems like **Computed Radiography (CR)** and **Digital Radiography (DR)** in the 1990s [Age15].

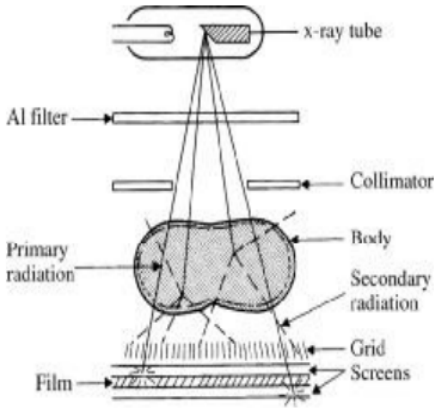


Figure 2.3: Basic X-ray setup showing X-ray tube, collimator, object being examined, film and screen. Source: [Dod+21]

The momentum behind digital medical imaging has grown significantly. This preference is due to the inherent efficiencies in digital image capture, storage, and display, as well as the competitive cost structures of digital systems when compared to alternatives involving traditional film [Age15].

DR directly generates digital images, much like a digital camera. It employs a radiation-sensitive flat panel or a Linear Diode Array (LDA) to capture radiation signals and create digital images. There are two conversion processes: indirect conversion uses scintillators to convert X-ray photons to visible light, which is then captured by electronic arrays [see Figure 2.4], while direct conversion employs a semiconductor layer to directly convert X-ray signals into electronic signals.

In medical diagnostic applications, X-ray imaging operates by detecting the variations in the transmission of X-ray photons through bodily tissues. In this process, an external X-ray source directs photons towards a specific region of the body. Within the body, some of these X-ray photons pass directly through, while others scatter in unpredictable directions,

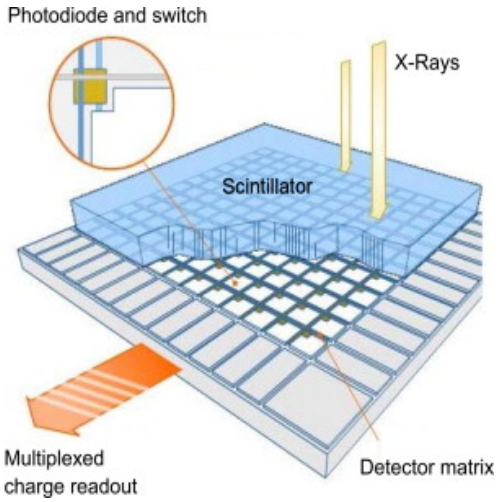


Figure 2.4: Flat panel detectors, indirect conversion. Source: [Spa13]

and some are entirely absorbed by the tissues. A detector situated on the opposite side of the body captures the X-ray photons that remain unabsorbed, as well as some of the scattered ones. This partial ability of the body to allow X-rays to pass through, resulting in varying levels of transmission produces a projection image. Only a brief overview of X-ray imaging was provided here, detailed explanation can be found in [Met+00].

2.2 Digital Imaging in X-ray

Imaging refers to the process of constructing a 2D or 3D representation of a physical object by measuring energy that is either reflected or emitted by the object. Primarily, the objective of medical imaging is to reveal aspects of the body that cannot be observed through external visual inspection or physical examinations. Interestingly, many medical imaging techniques generate visible light images that represent specific physical characteristics of bodily tissues that would otherwise remain hidden from the naked eye.

A **digital image** is a visual representation of data or information in a digital format. It is composed of a matrix of picture elements, known as pixels, where each pixel holds a specific color value or intensity level. It can be represented in both frequency and spatial domain. In the frequency domain, the image is represented as a rate of intensity change using a sinusoidal intensity profile. In the spatial domain, it is represented as a matrix of intensity value in a 2D plane. Radiographic image from an X-ray system specific type of digital image and can be represented as 2D intensity function f(x,y) where f is the average

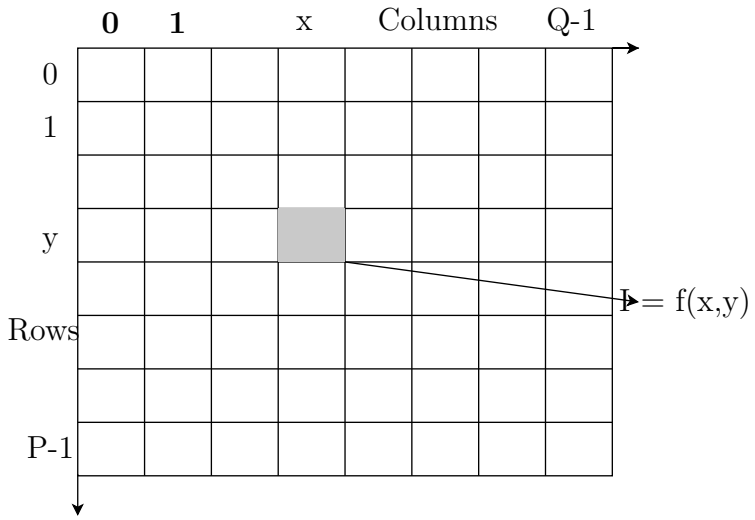


Figure 2.5: P * Q matrix representation

X-ray signal intensity showing the degree of darkness/brightness at x, y coordinate of the X-ray detector. Mathematical representation is

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,Y-1) \\ f(1,0) & f(1,1) & \cdots & f(1,Y-1) \\ \vdots & \vdots & & \vdots \\ f(X-1,0) & f(X-1,1) & \cdots & f(X-1,Y-1) \end{bmatrix} \tag{2.1}$$

with 0 <= f (x, y) <= L, with L = 65,535 for 16 bit image. Figure 2.6 shows a close-up view of the pixel representation of an X-ray image.

In X-ray, we are dealing with 2D grayscale images. A **pixel** in the raw X-ray image data corresponds to the average X-ray signal intensity within a specific volume of space inside the X-ray detector. An image must possess sufficient spatial resolution and the ability of an image to accurately represent and separate distinct objects in space effectively is referred to as **'precision'**. The number of distinct levels, reflecting the highest attainable precision for the stored intensity data, is determined by the number of bits employed for data storage, known as the **'bit depth'**. All imaging data is captured and stored with a level of precision that exceeds the human visual capacity. For instance, most gray-scale image display devices, such as monitors, typically employ an eight-bit depth, allowing them to represent a total of 2^8=256 distinct intensity levels. X-ray images are typically stored in **16 bits** representing all integers from 0 to 65535.

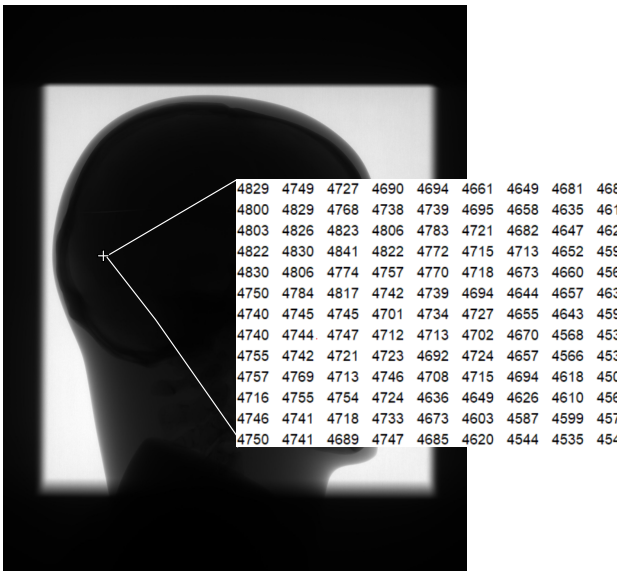


Figure 2.6: Pixel image representation

Image Storage Digital medical images can be stored in various file formats like JPEG, Portable Network Graphics (PNG), Digital Imaging and Communications in Medicine (DICOM) [Bou], Tagged Image File Format (TIF), and Bitmap each designed for specific purposes and types of medical imaging modalities. All of these can store color images too. Some of these file formats are discussed in detail -

- **DICOM** or dcm: is a standard file format for medical images, primarily designed for grayscale images. It is used to store and transmit a wide range of medical images, including X-rays, CT scans, MRIs, ultrasounds, and more. A DICOM file can store both metadata and image data. pixel intensity can be stored with a precision of 8, 12, 16 or 32 bits according to requirement.

- **TIF** or TIFF: is another widely used format to store high-quality medical images. TIF files can store multiple images, making them suitable for multi-frame in a single file. It focuses more on image quality than its size.

- **RAW**: Some medical imaging devices may output raw image data in their proprietary formats. In such cases, the raw data can be stored in its native format or converted to a more standardized format like DICOM for further processing and analysis.

- **Bitmap (BMP)** : BMP is a simple and widely supported image format. While it is not commonly used in medical imaging due to its lack of compression and large file sizes, it can still store medical images.

The file format selected is determined by the particular needs of the medical imaging program, including image quality, compression needs, and the type of imaging modality used. DICOM remains the dominant format for medical imaging due to its comprehensive support for medical metadata and interoperability.

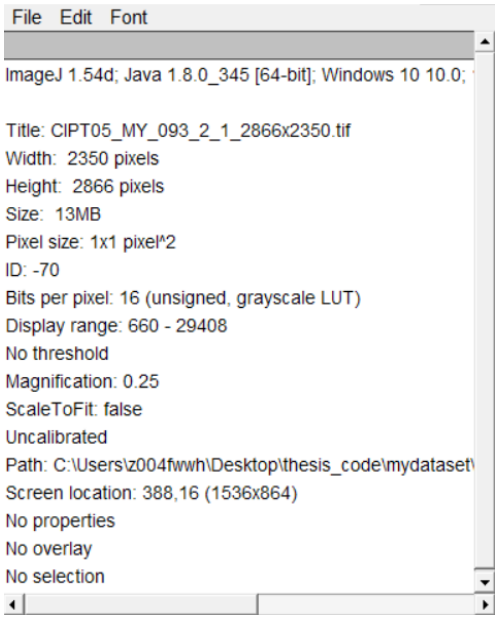


Figure 2.7: ImageMetadata of TIF image visualized through ImageJ, showing image description and value

Image Metadata Along with the sequence of bits representing the pixel intensity of a digital image, it is also important to store other essential information like dimension, size, precision, method of image acquisition, etc. This non-intensity image information is called **image metadata**. An example of typical image metadata stored in TIF format is shown in Figure 2.7.

Digital image visualization There are various digital image visualization tools available, ranging from basic image viewers to more advanced software packages designed for specific tasks. **ImageJ** [1] is an image processing software built on Java, created by the National Institutes of Health in the USA and is compatible with all computer platforms. Figure 2.8 shows ImageJ user interface. More comprehensive details of ImageJ's applications along with Java code can be found in the textbook titled 'Digital Image Processing: An Algorithmic

[1]Available for free download from the ImageJ website: http://rsb.info.nih.gov/ij/index.html

Introduction Using Java' authored by Wilhelm Burger and Mark Burge, published by Springer-Verlag in 2007 [Bur+16].

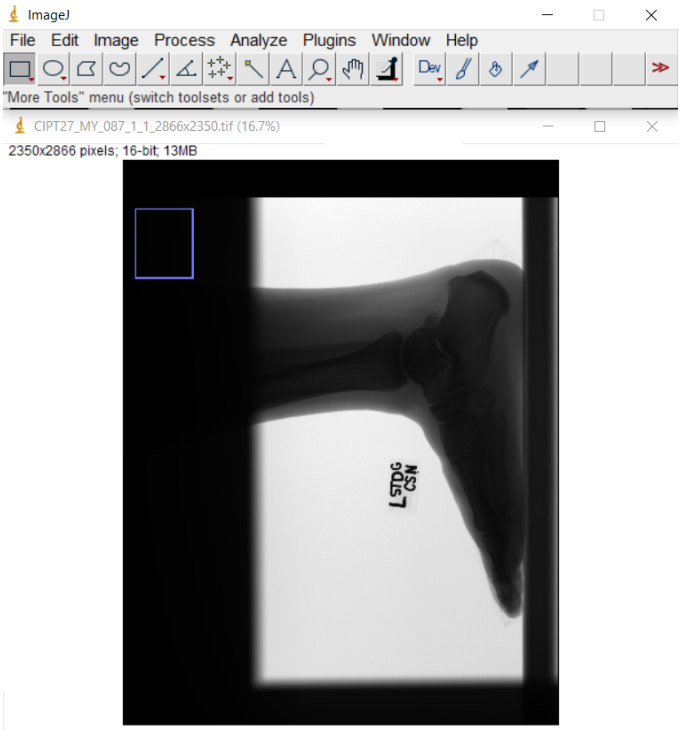


Figure 2.8: ImageJ user interface screenshot of X-ray image

Chapter 3

Deep Learning and its associated concepts

The goal of our thesis is to automate the process of artifact detection through AI. Deep learning can be said as a subset of AI, motivated by working of our biological neurons. The focus of this section is to introduce the core concepts of DL and neural networks. The section also introduces the algorithms for training these neural networks and regularization techniques. More complex artificial neural networks (ANNs) such as CNNs, Recurrent Neural Networks (RNNs), Autoencoder (AE), and Variational Autoencoder (VAE) are additionally introduced. We also discuss the need for explainability and its known techniques. In the end, We also study the designing aspect of deep learning models.

3.1 Feed-Forward Neural Network and Model Training

An ANN adopts features of the biological concept of neural networks. Such a network consists of several units connected to others. All neurons with the same depth inside the network are grouped as a layer. Each neuron connecting link is characterized by a specific weight which is updated by a learning element. A neuron contains a set of input links [Rix14] and a set of output links from other artificial neurons, an activation level, and a method of calculating the activation value for the following neuron. One differentiates between three sorts of neuron units. The input neurons receive signals from the environment and proceed them to hidden neurons between the input and the output layers. The output neurons again transmit the signals as numeric values as output signals. A neural network

with an implicit feature engineering part and multiple hidden layers is called a **deep neural network** [Rus+09]. Figure 3.1 visualizes a schematic example of an ANN.

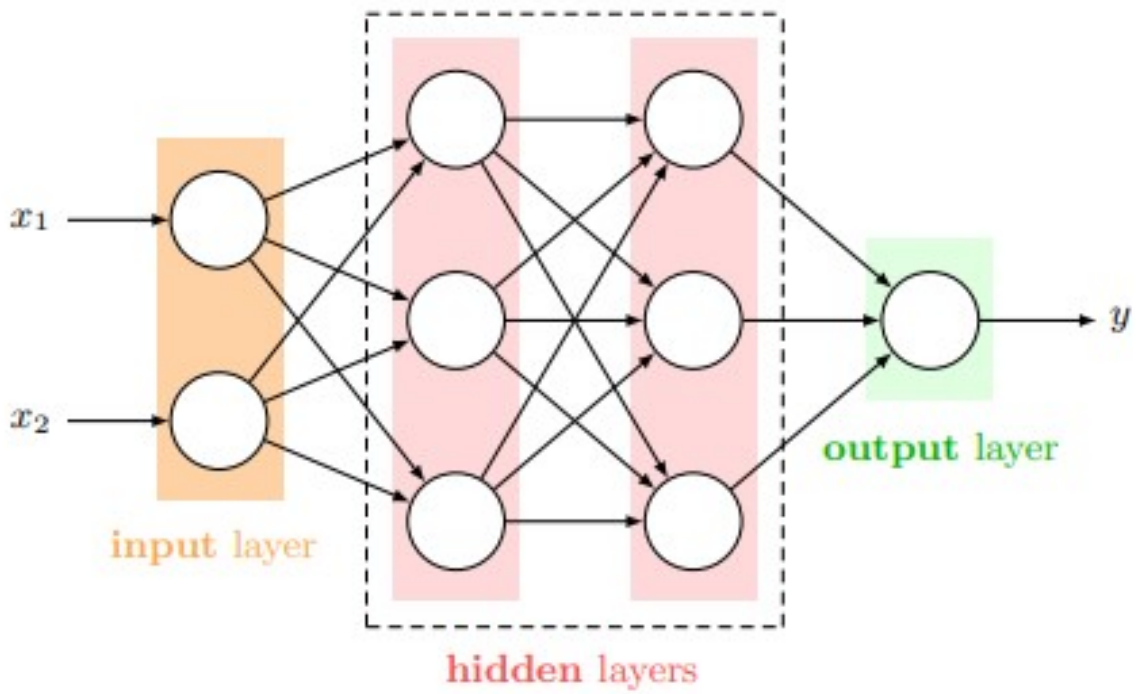


Figure 3.1: **Scheme of a fully connected neural network.** Neurons receive certain inputs such as x1 and x2 for the input neurons and are connected to other neurons of the neighboring layers. Each link has an individual weight whose value is updated during training as described in Section 3.1. This representation is based on Charu Aggarwal et al. [Agg18].

A basic neural network model can be interpreted as a sequence of functional transformations [Gua19]. For a D-dimensional input vector x, the activations a_j are calculated by

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{3.1}$$

where $w_{ji}^{(1)}$ denotes the weights and $w_{j0}^{(1)}$ the biases of the first network layer for $j = 1, ..., M$. The calculated activations are transformed with an activation function $h(\mathring{u})$ which results in $z_j = h(a_j)$. Another linear combination is used to calculate the unit activation outputs,

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{3.2}$$

where the superscript indicates the second layer and $k = 1, ..., K$ (K being the output layer). Lastly, the network outputs y_k are calculated using an appropriate activation function [Bis+24]. Next to other existing activation functions, a logistic softmax function is often used for binary classification tasks and has the form

$$y_k = \sigma\left(a_k\right), \qquad \text{where } \sigma(a) = \frac{1}{1 + \exp(-a)} \tag{3.3}$$

Finally, merging these steps into one function for all output variables x_k with respect to all definitions above lead to

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{3.4}$$

This equation corresponds to matrix multiplication. The values $w_{ij}^{(1)}$ and $w_{kj}^{(2)}$ can be interpreted as the elements of two matrices $W(1)$ and $W(2)$. Together with two bias vectors $b^{(1)}$ and $b^{(2)}$, this leads to

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{3.5}$$

Both representations can be viewed as the forward propagation of information through the network as it evaluates all layers one after another from first to last [Bis+24].

Model Training The aim of training learning models is to learn characteristics from a given set of data. The predefined model should minimize an error function (with respect to w) of the form

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\| \boldsymbol{y}\left(\boldsymbol{x}_n, \boldsymbol{w}\right) - \boldsymbol{t}_n \right\|^2 \tag{3.6}$$

with a set of input and target vectors x_n and t_n. One often uses a cross-entropy function when considering the case of binary classification. The special case of using a logistic function is then given by

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} \tag{3.7}$$

where y_n is an abbreviation for y(x_n, w) and N is the total number of training examples [Bis+24]. The task is now to find a suitable weight vector w to minimize $E(w)$. A common approach is to update the weight vector in the way that

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E \left(\boldsymbol{w}^{(\tau)}\right) \tag{3.8}$$

where $\tau > 0$ is the **learning rate** and η the step index. As the weight vector is moving in the direction of the steepest descent at each step; this method is called **gradient descent**. The gradient of the error function is calculated using the backpropagation principle. As many error functions consist of a sum of terms, where each term is calculated via a data sample from the data set, one can first calculate the error value for an arbitrary summand E_n and generalize this method afterward. The gradient of E_n with respect to a weight $w_j i$ is calculated by

$$\frac{\partial E_n}{\partial w_{ji}} = (y_j (\boldsymbol{x}_n, \boldsymbol{w}) - t_{nj}) x_{ni} \tag{3.9}$$

with $y_k = \sum_i w_{ki} x_i$, where $z_j = h(a_j)$ and Equation (3.6). The computed weighted sum is defined by $a_j = \sum_i w_{ji} z_i$, where $z_j = h(a_j)$ is the activated value of the weighted sum. This refers to the feed-forward principle of networks. Applying the chain rule leads to

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{3.10}$$

The partial derivatives for hidden units in between are again calculated by the chain rule,

$$\delta_j := \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{3.11}$$

This sum considers all k units which are connected with unit j. Finally inserting these terms into one another leads to the *backpropagation* equation

$$\delta_j = h' (a_j) \sum_k w_{kj} \delta_k \tag{3.12}$$

All n terms of $\frac{\partial E_n}{\partial w_{ji}}$, when they are summed up, lead to the gradient of the error function E with respect to the weights w_{ji} (with the assumption that each unit has the same activation) [Bis+24].

If the weights are updated right after each input, the procedure is known as **Stochastic Gradient Descent (SGD)** [Nay+21]. In this case, the gradient is estimated by a randomly picked example x_i at each iteration instead of calculating the gradient of the error function E exactly [Bot10]. In batch training, gradients are averaged over all samples and then updated. The batch size defines the number of averaged gradient samples and is an adjustable hyper-parameter during the training process. All inputs in total are consolidated as an epoch. The total number of epochs consequently defines the number of complete passes through the entire training data [Dud+00].

RMSprop, short for Root Mean Square Propagation [Gra14], is an optimisation algorithm designed to address some limitations of traditional gradient descent methods. It calculates exponentially weighted moving averages of squared gradients, adjusting the learning rates for each parameter individually. The update rule for the weight $w_j i$ in RMSprop is given by:

$$w_{ji} = w_{ji} - \alpha \frac{\frac{\partial E}{\partial w_{ji}}}{\sqrt{v_t} + \epsilon} \tag{3.13}$$

where v_t is the exponentially weighted moving average of the squared gradients.

Adam (short for Adaptive Moment Estimation) [Kin+17] combines ideas from both momentum optimisation and RMSprop. The update rule for the weight w_{ji} using Adam is given by the following equations:

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1 - \beta_1) \frac{\partial E}{\partial w_{ji}} \\ v_{t+1} &= \beta_2 v_t + (1 - \beta_2) \left(\frac{\partial E}{\partial w_{ji}} \right)^2 \\ \hat{m}_{t+1} &= \frac{m_{t+1}}{1 - \beta_1^{t+1}} \\ \hat{v}_{t+1} &= \frac{v_{t+1}}{1 - \beta_2^{t+1}} \\ w_{ji} &= w_{ji} - \alpha \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon} \end{aligned} \tag{3.14}$$

where:

- m_t is the first moment (mean) estimate of the gradient,

- v_t is the second moment (uncentered variance) estimate of the gradient[Lin+23],
- β_1 and β_2 are exponential decay rates for the moment estimates,
- $\hat{m}_{t+1}$ and $\hat{v}_{t+1}$ are bias-corrected moment estimates,
- α is the learning rate,
- ϵ is a small constant to prevent division by zero[Bur+23].

Adam adapts the learning rates of each parameter individually, making it well-suited for a wide range of optimization problems.

Activation function Activation functions play a pivotal role in neural networks by introducing non-linearity into the network, enabling it to learn complex patterns and relationships within the data. If a network uses only linear functions, then making the network deep won't make much of a difference. The final result will be just like combining the inputs, which can be done with a single layer. The rectified linear unit (ReLU) is the most commonly used activation function [Kri+12].

$$ReLU(z) = max(0, z) \tag{3.15}$$

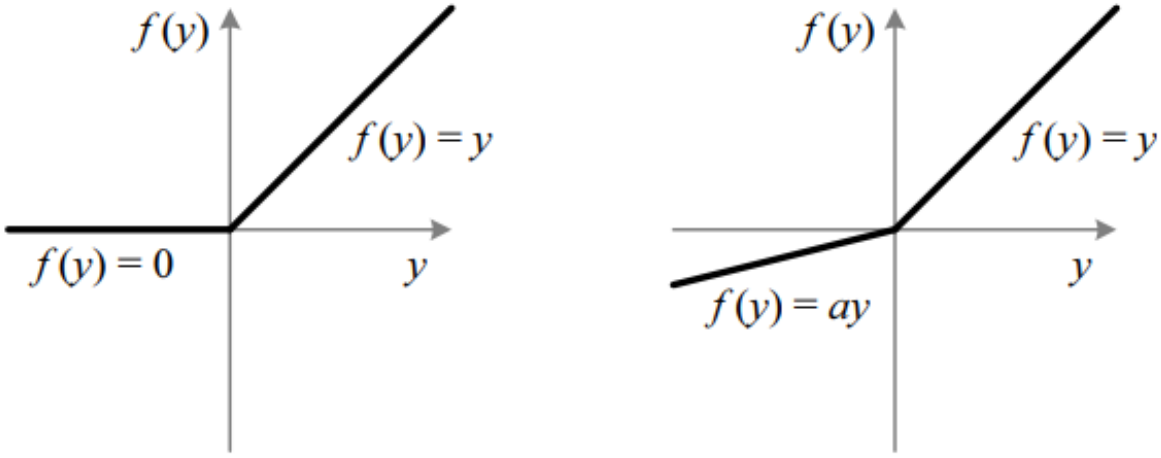


Figure 3.2: ReLU (left) introduces non-linearity by allowing the positive input values to pass through unchanged, while the negative values are clamped to zero vs PReLU (right), an extension of the traditional ReLU activation, introduces a learnable parameter α to control the slope of the negative values [He+15b]

In contemporary models, variants of ReLU such as leaky ReLU, parameterized rectified linear unit (PReLU), and exponential linear unit (ELU) have gained widespread adoption.

PReLU [He+15b]:

$$\phi(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases} \tag{3.16}$$

has α parameter that is learned during training and helps in avoiding the 'dying ReLU' problem.

Schedulers [Nap+23] play a crucial role in optimizing training dynamics for neural networks by adjusting the learning rate during training. The **step-wise learning-rate** scheduler, is a specific type of scheduler that updates the learning rate by a multiplicative factor at fixed time steps. Equation (3.17) shows the formula for this scheduler.

$$lr_{\text{epoch}} = Gamma * lr_{\text{epoch - 1}} \tag{3.17}$$

Regularization Deep neural networks often show the problem of overfitting, meaning that the model only shows a poor generalization ability as it learns the training samples by heart and makes so-called regularization techniques necessary to prevent this effect. The two main methods for regularization are **dropout** and **weight decay**.

Incorporating a dropout layer randomly drops a certain amount of neurons and their connection inside the network. The number of dropped neurons is adjustable via a passed dropout rate for the corresponding layer. The benefit of dropout is reducing the inner complexity of a network [Hin+12]. Weight decay instead limits the growth of the number of network parameters by a passed weight decay value. The weights are therefore chosen to keep on classifying instances correctly without learning samples by heart. The decay is implemented by an additional cost, which penalizes all parameters in the network. A simple example of such an update function is given by

$$E_{wd}(\Theta) = E(\Theta) + \frac{1}{2}\lambda \sum_{i}^{|\Theta|} w_i^2 \tag{3.18}$$

with J representing the cost function to be optimized during training, λ the actual weight decay value and $|\Theta|$ the total number of parameters.

3.2 Convolutional Neural Network

A CNN stands as the neural network architecture specifically designed for processing and analyzing visual data, making it the equivalent of an ANN tailored for images. It consists of multiple sequential stages [LeC+89], [Kri+12], including convolutional layers (conv), non-linearity layers, and pooling layers (pool), followed by additional convolutional and fully connected (FC) layers, discussed in detail below. The input layer takes the raw pixel intensity of the image and the output layer comprises multiple neurons, each corresponding to a specific class [Zha+17a]. The network's weights (W) are fine-tuned by minimizing classification errors on the training set through the backpropagation algorithm [LeC+12].

Convolutional layer The convolutional layer operates by taking local rectangular patches from the input image (or feature maps in subsequent layers) using a specified stride and, if necessary, spatial preservation by padding. Within these patches, a 2D convolution is applied using a filter. The results of these convolutions are summed up to produce an output value, which is then passed through a non-linearity function, which is used to accelerate the training process [Sha+18].

In each layer [Zha+17b], the same filter is shared among the feature maps, while different filters are employed for distinct feature maps. This filter-sharing characteristic in the convolutional layer allows the network to identify the same patterns in various locations within the feature map [Zha+17a].

Pooling layer The pooling layer serves the purpose of downsizing the feature map by condensing the information within non-overlapping local patches [Sha+18]. A common method in this process is **max-pooling**, where the maximum activation value within each local patch is retained. This technique results in features that remain consistent even when there are slight translations in the data. They capture the most important information from the local regions while preserving the general structure of the feature maps. Max-pooling is one of the most widely used methods, but average-pooling is another common option, which calculates the average activation within each patch [Sha+18].

FC layer The FC layer section of the model plays a crucial role in processing feature maps obtained from previous convolutional and pooling layers. These feature maps, which are of smaller dimensions [Zha+17b] compared to the input image, are combined and flattened into a feature vectors. It has a multilayer perceptron (MLP) kind of structure. The final

FC layer is responsible for making class predictions by producing probabilities for each class through a softmax operation.

Figure 3.3 shows the basic architecture of a deep CNN with convolutional, pooling and FC layers.

Batch-normalization (BN) layer As described in [Iof+15], BN layers play a crucial role in the training neural network models. Batch normalization can be conceptualized as a process applied to each batch during training, normalizing each dimension of the input vector x using-

$$x^{\ell}_{\text{norm}} = \frac{x^{\ell} - E\left[x^{\ell}\right]}{\sqrt{\text{Var}\left[x^{\ell}\right]}}. \tag{3.19}$$

In simpler terms, x^{ℓ}_{norm} is the normalized version of the internal layer, adjusting the values based on the mean (E[x]) and variance (Var[x]) of the batch. The resulting normalized values are then used in the internal layer, and the layer y is computed as

$$y^{\ell} = \gamma^{\ell} x^{\ell}_{\text{norm}} + \beta^{\ell}, \tag{3.20}$$

where γ^{ℓ} and β^{ℓ} are parameters learned through training that facilitate scaling and shifting.

The utilization of batch normalization, as highlighted in studies such as [Shi00], has demonstrated its effectiveness in mitigating the internal covariate shift within network parameters, stabilizing and improving the training process by normalizing the input batches, thereby aiding in the overall optimization of neural network models.

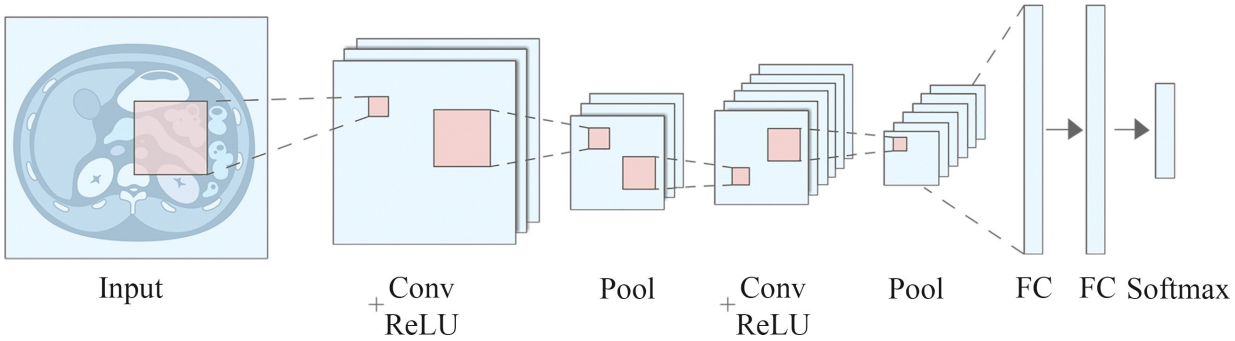


Figure 3.3: Basic architecture of a deep convolutional neural network. Source: [Kwa+17]

3.2.1 Transfer Learning

ImageNet Dataset : ImageNet [Den+09] is a large-scale visual database designed for use in visual object recognition research. Created by researchers at Stanford University, ImageNet originally contained over 14 million labeled images, making it one of the largest datasets of its kind, covering a wide range of object categories, encompassing animals, plants, everyday objects, and more. The diversity of categories allows researchers to evaluate the performance of computer vision models across various domains.

CNN architectures can be pre-trained on large datasets like ImageNet before fine-tuning on specific medical data, known as **transfer learning**. This approach helps overcome the challenge of limited medical data [Bro]. Transfer learning involves transferring knowledge gained from one domain to another, much like how humans apply their knowledge from different fields when faced with a new task.

In the transfer learning process, the pre-trained weights from ImageNet serve as a foundation for the CNN model. The fine-tuning involves retraining the model, with the option to freeze some, all, or none of the layers. This contrasts with non-transfer learning, where all weights are initialized randomly. Transfer learning, especially with ImageNet as a source, has proven instrumental in enhancing the performance of CNNs on specific tasks, such as medical image analysis.

3.3 Time Series Classification

Time series classification refers to the task of assigning predefined labels or categories to sequential data points in a time-ordered manner. In this context, the input data consists of a sequence of observations or measurements, each associated with a specific timestamp. The goal is to develop a model that can learn and recognize patterns, trends, or features within these temporal sequences, enabling accurate classification into predefined classes or categories.

Applications of time series classification are diverse and can be found in various fields such as finance [Wu+20], healthcare, signal processing, and environmental monitoring [Sch14]. For instance, in healthcare, time series classification might involve identifying different health states based on a patient's continuous physiological monitoring data.

Time series classification with deep neural networks Ebrahim et. al in their paper [Ebr+20] discuss different neural network architectures that can be used for time series classification. RNNs, CNNs are commonly employed for this task.

RNN

John Hopfield defined a RNN in 1982 as an extension of a conventional feedforward neural network with connections between nodes that form a directed graph, and depict a time sequence with dynamic temporal behavior.

Originally designed for processing of sequential data, RNN models have found new applications in various classification tasks, like in remote sensing [Ndi+18]. These models can learn and share important features across different positions within the data. The basic structure of RNNs is similar to feed-forward networks despite additionally containing connections backward, which are called **recurrent connections**.

The simplest RNN case is a single recurrent unit, which uses a vector $x_(t)$ and its own scalar output $y_(t-1)$ of the previous time step as the only two inputs. It is also known as a **memory unit** due to the ability to memorize previous state inputs. Several recurrent units are connectable to recurrent layers, where each layer again uses an input vector $x_(t)$ and its own scalar output $y_(t-1)$ of the previous time step. A visualization of a recurrent layer is given in Figure 3.4. Introducing the weight matrices W_x and W_y [Nay+21]for the input and the recurrent input, a bias vector b and activation $\phi(.)$ produces the output

$$\boldsymbol{y}_{(t)} = \phi\left(\boldsymbol{W}_x^T \boldsymbol{x}_{(t)} + \boldsymbol{W}_y^T \boldsymbol{y}_{(t-1)} + \boldsymbol{b}\right) \tag{3.21}$$

However, one significant challenge in training RNNs is the computational cost associated with back-propagating errors at each time step. This can lead to issues like vanishing gradient, which hinders effective learning. To address these challenges, modern RNN architectures often incorporate Long Short-Term Memory (LSTM) units [Ger19].

LSTM

The LSTM architecture was proposed in 1997 with the objective of overcoming the mentioned disadvantages of plain RNNs. This is mainly solved by incorporating a forgetting mechanism. LSTMs are based on a self-looping procedure allowing the network to create paths where gradients can flow for a long time and make them scalable in time and suitable for different sequence lengths. LSTMs follow the structure of RNNs, but each unit is replaced by single

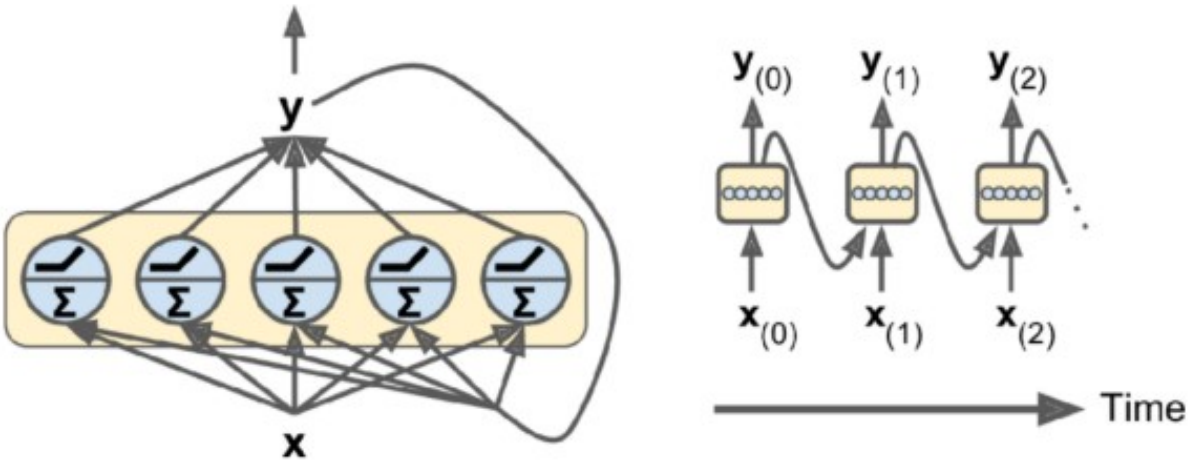


Figure 3.4: Scheme of a recurrent layer [Ger19]. A recurrent layer consists of many recurrent units, each calculating a weighted input sum followed by an activation of the result. The right part of the figure shows the recurrent layer unrolled in time.

LSTM cells with **internal self-loops**, which is visualized in Figure 3.5. The inputs and outputs are the same as normal recurrent units but contain more learnable parameters and gating units to control the information flow [Goo+16]. The three LSTM inputs are given by $x_(t)$, $h_(t1)$ as the short-term state, and $c_(t1)$ as the long-term state, from either the current or the previous time step. The long-term state is forwarded to a forget gate to control memory dropping and receives new information via addition operations. This results in the updated short-term state $c(t)$, which is additionally sent through an output gate and yields the updated short-term state $h_(t)$ and the current output $y_(t)$. Four different fully connected layers determine the new memories using $h_(t1)$ and $x_(t)$ as inputs. The $g_(t)$ layer mainly analyzes the inputs like a typical RNN cell, whereas all other layers are **gate controllers**.

[Smi+18] explores the effectiveness of RNNs, in univariate time series classification. Through an extensive experimental evaluation on 85 (University of California, Riverside) UCR datasets, the study compares RNNs with feedforward baselines and identifies the superiority of LSTM layers over simple recurrent layers in achieving higher classification quality. The results suggest that RNNs remains a viable choice for time series classification, especially when dealing with sequences of varying lengths.

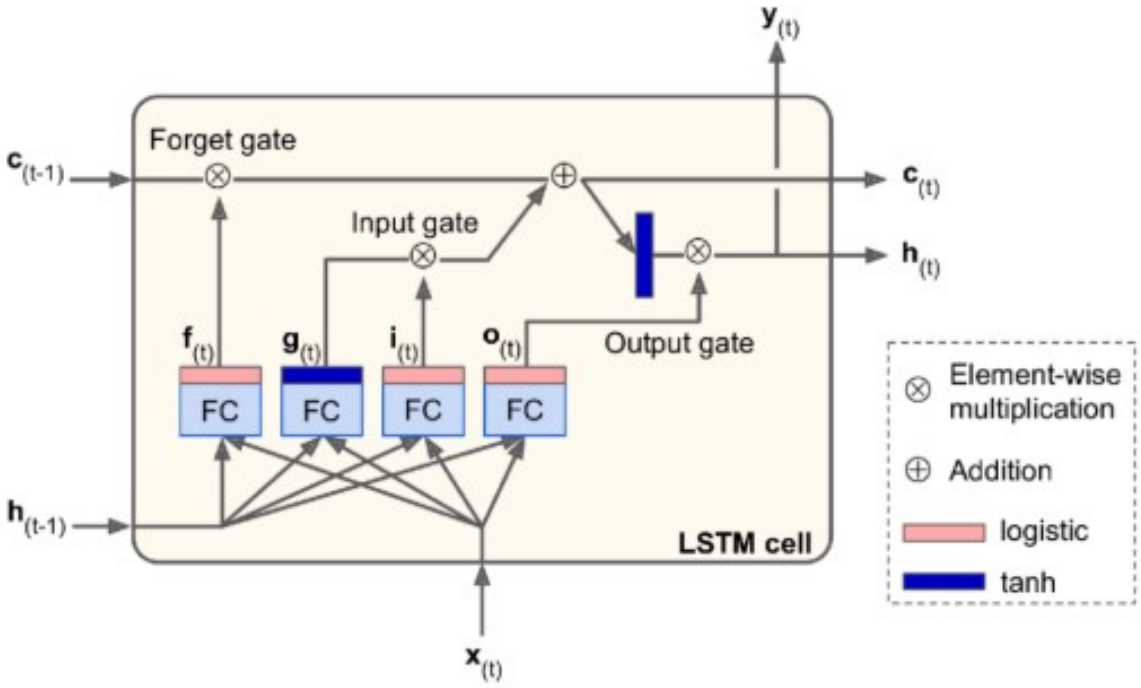


Figure 3.5: Scheme of a LSTM cell [Ger19]. A LSTM is capable of modeling a long and short-term memory, which is each controlled by FC layers (abbreviated with FC). The variables and abbreviations are explained in this section.

1DCNN

A 1DCNN (Convolutional Neural Network) for time series classification is a deep learning architecture specifically designed to analyze and classify sequences of single-variable data over time. This model applies one-dimensional convolutional operations to capture local patterns and features within the time series. Utilizing filters and pooling layers [see Figure 3.6], the 1DCNN automatically learns hierarchical representations, enabling effective feature extraction for improved classification accuracy in tasks such as signal processing, medical diagnosis, and various time series analysis applications. 1DCNNs are effective at capturing local patterns and features in the input sequence through convolutional filters. This is particularly advantageous when the relevant features for classification are present in specific local regions of the sequence.

[Pel+18] in their paper uses a temporal convolutional neural network for Satellite Image Time Series (SITS) classification, and they show 1DCNN outperforms random forest and RNN algorithm by 1 to 3%.

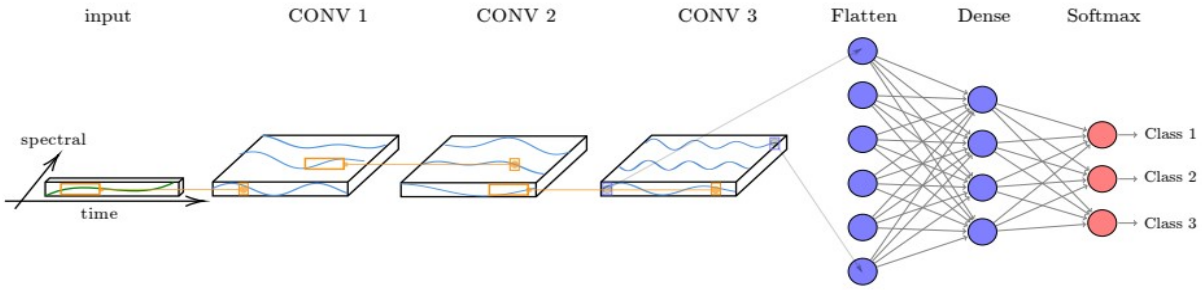


Figure 3.6: Figure shows the architecture of 1DCNN. The network input is a multi-variate time series. One dense layer is applied after three convolutional filters are applied successively, and finally, the softmax layer provides the predicting class distribution. [Pel+18]

It's important to note that the choice between 1DCNNs and RNNs/LSTMs depends on the specific characteristics of the data and the nature of the task. While 1DCNNs offer advantages in certain scenarios, RNNs/LSTMs excel in tasks where capturing temporal dependencies and context over longer sequences is critical.

3.4 One Class Classification

Anomaly or *outlier detection* is a technique used to identify samples in a dataset that stand out or deviate significantly from the expected patterns [Fer+20]. Fernando et. al [Fer+20] provide an overview of all deep learning techniques that can be utilized in the medical field to discover anomalies.

The term 'One class classifier (OCC)', introduced by T.C. Minter in 1975, is a key concept in anomaly detection. OCC differs from traditional binary or multi-class classification as it focuses on a single class during training, lacking instances from other classes. This approach is particularly useful when dealing with imbalanced datasets where one class dominates in terms of examples. This forms the problem of **unsupervised** learning.

Generative approaches, utilizing models like auto-encoders and Generative Adversarial Networks (GANs), are commonly employed in solving OCC problems [Fer+20].

AEs can function as tools for OCC, as evidenced by studies like [Bor+19]. AEs can be trained to reconstruct input with low reconstruction error rates, making them capable of learning latent features. 'Normal' data is used for training, and 'abnormal' data can be identified based on reconstruction errors, serving as anomaly scores. However, AEs are deterministic and lack a probabilistic foundation. To address this, VAE [Kin+13], a stochas-

tic generative model offering calibrated probabilities, has been proposed, demonstrating improved performance in fake image detection.

Autoencoder as OCC A dataset with N samples is denoted as a matrix $X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times D}$, where each sample x is a D-dimensional vector. A typical autoencoder consists of two major components, i.e., an encoder and a decoder, which are both realized by neural networks [Lia+22a]. Let $fe(\cdot; \Theta_e)$ denote the mapping of the encoder parameterized by Θ_e and $fd(\cdot; \Theta_d)$ be the mapping of the decoder parameterized by Θ_d. Accordingly, the encoder maps the input sample x into a lower-dimensional representation z as

$$\boldsymbol{z} = f_e(\boldsymbol{x}; \boldsymbol{\Theta_e}),$$

where $z \in \mathbb{R}^L$ with $L \ll D$ [Lia+22a]. Likewise, the decoder maps the latent representation z back to the original data space as

$$\hat{\boldsymbol{x}} = f_e(\boldsymbol{z}; \boldsymbol{\Theta_d})$$

where $\hat{x}$ is called the reconstruction of the input sample x. Naturally, the learning objective of AE is to maximize the similarity between the reconstruction and original inputs, i.e., minimizing the reconstruction loss [Lia+22a]. A canonical choice of the reconstruction loss is the mean squared error (MSE) loss.

$$\arg\min_{\boldsymbol{\Theta_e}, \boldsymbol{\Theta_d}} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$$

As a consequence, the bottleneck structure in AEs, where $L \ll D$, allows capturing the intrinsic structure of the training data. In other words, only test data from the same distribution as the training data can be effectively reconstructed.

Variational autoencoder as OCC VAE is an improvement over using AEs [Kha+20a]. In contrast to traditional AE, VAE introduces a probabilistic approach to encoding data. In VAE, the encoder maps the input x to a probability distribution in the latent space rather than a specific point. The latent representation is characterized by a mean (μ) and a variance (σ^2). This stochasticity allows the model to generate diverse samples during the decoding process. Figure 3.7 shows the architecture of VAE.

Equations for VAE:

1. Encoder:

$$z \sim \mathcal{N}(\mu, \sigma^2),$$

where μ and σ^2 are the mean and variance, respectively.

2. Reparameterization Trick:

$$z = \mu + \sigma \odot \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 1)$ is a random noise.

3. Decoder:

$$\hat{x} = f_d(z; \Theta_d)$$

4. Loss Function: The loss function for VAE is a combination of a reconstruction term (typically mean squared error) and the Kullback-Leibler (KL) divergence, which regularizes the distribution of latent variables:

$$\mathcal{L}_{\text{VAE}} = \text{MSE}(\hat{x}, x) + \text{KL}(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, 1))$$

The KL divergence term ensures that the learned latent space follows a standard normal distribution.

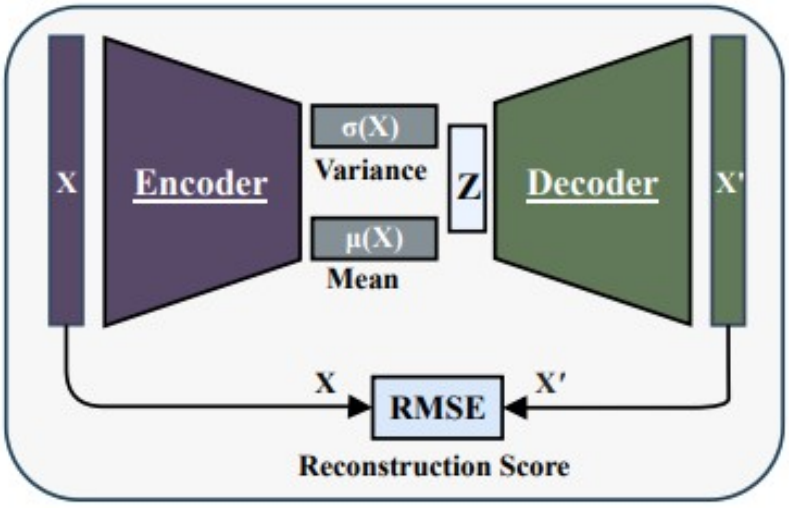


Figure 3.7: Architecture of variational autoencoder for OCC [Kha+20b]

Anomaly Score In OCC, AE is frequently trained onlw with the known class, expecting the trained AE to capture essential structures or latent features specific to that class.

During testing, given a new test sample x_t, the reconstruction score is calculated as:

$$\varepsilon_t = \|x_t - \hat{x}_t\|_2^2 \tag{3.22}$$

Correspondingly, with a predefined threshold ε, the known and unknown classes can be distinguished by:

$$x \in \begin{cases} \text{Good}, & \text{if Reconstruction Score} < Threshold, \\ \text{Defective}, & \text{otherwise} \end{cases}$$

Note that the threshold ε is typically defined based on the tolerable false negative rate (FNR) by users.

3.5 Explainability

Since neural network models are **black boxes**, they are difficult to interpret [Cas16]. Accurate comprehension of deep learning models on how models made their prediction is important in various applications, which are addressed by 'explainability' techniques. These techniques can be broadly categorized into two types: **intrinsic** and **post-hoc**. Intrinsic methods integrate explainability directly into the model architecture, making it more transparent from the start. In contrast, post-hoc methods are applied after a model is trained and aim to provide insights into its decision-making process [Sel+17] [Got+20].

Visualizing Activation Map For image-based models, visualizing an activation map is the simplest post-hoc method, and features are learned by the model [Fer+20]. The activation maps obtained from the last convolution layer are combined with their corresponding weights to create a final activation map for the predicted class. This map is then enlarged to be overlaid onto the original input image. This process helps highlight the specific areas or features in the input that strongly influence the classification decision, providing insights into the model's decision-making process. Figure 3.8 explains this concept.

The Grad-CAM method overcomes the limitation of CAM by using gradient information from the last convolutional layer. For each pixel in the feature map A^k, the gradients of the score for class c, y_c, are computed and averaged. This average is denoted by $\alpha^{c,k}$. The final activation map is generated by combining the weighted feature maps A^k using these averages and passing through a ReLU activation.

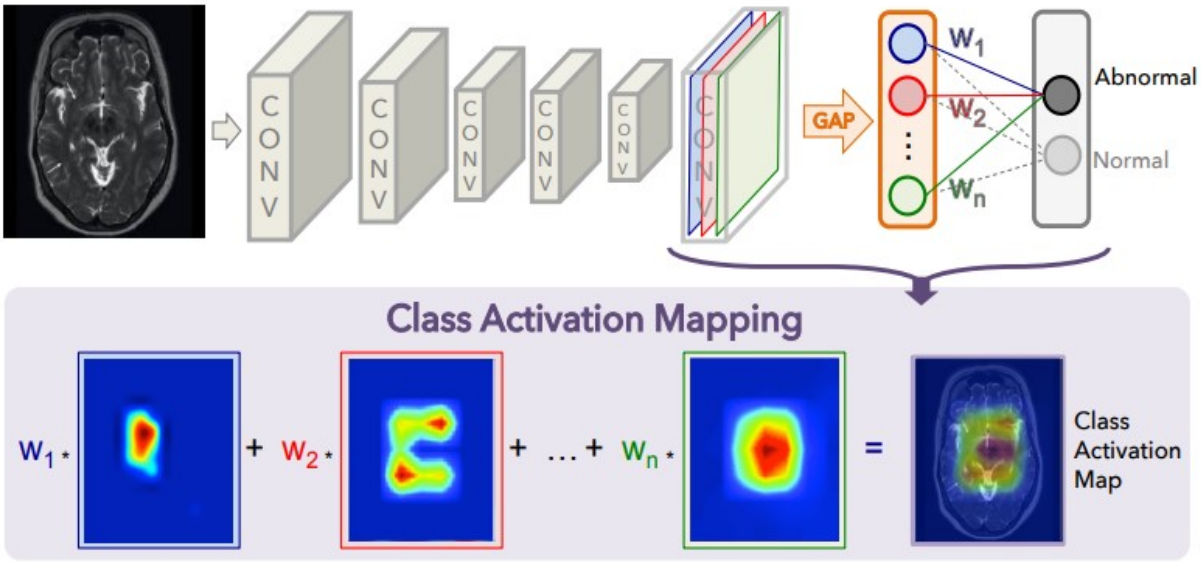


Figure 3.8: Illustration of the process of obtaining class activation maps. Source:[Fer+20]

$$\alpha^{c,k} = \frac{1}{uv} \sum_{i \in u} \sum_{j \in v} \frac{\partial y^c}{\partial A^k_{i,j}} \tag{3.23}$$

$$L_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_{c,k} A_k \right) \tag{3.24}$$

Grad-CAMs is effective for images, enabling domain experts to validate and grasp image-based models. However, these approaches have limitations when applied to arbitrary time series data [Loe+22].

Saliency Maps [Sim+13] introduce the technique of saliency maps. It highlights the regions in the input data that have the most significant impact on the output of a neural network. The gradient of the output with respect to the input is computed to identify these influential regions, providing insights into the model's decision-making process. [Ism+20] use this technique to compare with other interpretability methods for time series prediction.

3.6 Deep Learning Study Design

Creating a deep learning study follows a typical pattern that comprises various stages [see Figure 3.9]. The first step involves shaping a clinical question. Once the clinical question is

established, the appropriate computer vision task and its corresponding metrics are selected. Following this, attention turns to data gathering and data preparation, encompassing the organization of data for training and testing, as well as the annotation of medical data. Subsequently, the software framework and hardware platform are chosen, and the architecture of the neural network is devised. Ultimately, the results are validated using the chosen metrics on the testing dataset. Each of these stages will be further explained in the upcoming sections.

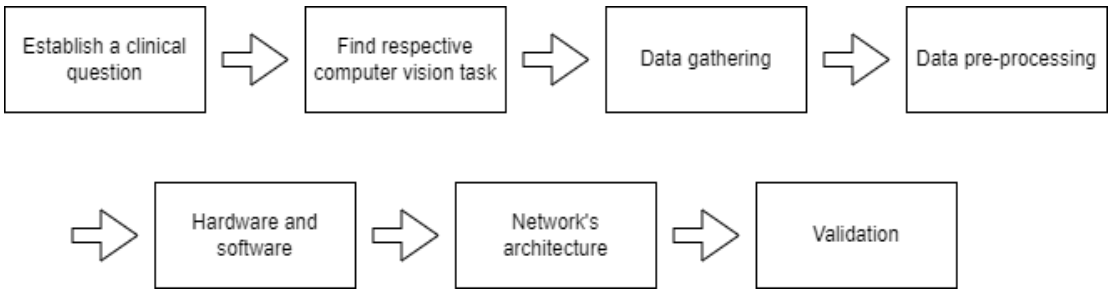


Figure 3.9: Shows steps in deep learning study. The first clinical question is defined, and a suitable computer vision task is chosen. Data is gathered and pre-processed. Software framework and hardware are selected. The network's architecture is defined, trained, and finally validated with testing data.

In our case, deep learning study design consist of following steps:

Clinical question How can we enhance the accuracy and efficiency of X-ray image interpretation by identifying and classifying artifacts present in the images, thereby improving diagnostic reliability and patient care? The goal is to automate the identification process and determine whether deep learning techniques can effectively differentiate between X-ray images with artifacts and those without artifacts, ultimately aiding radiologists in providing accurate diagnoses and improving patient care.

Computer Vision task The computer vision task selected for this study is image classification. Classification is the task of labeling or assigning an image to a particular class. The provided task involves training a deep learning model to classify X-ray images into two categories: those containing artifacts and those free from artifacts, forming binary classification and into different types of artifacts forming multiclass classification.

Data gathering There is a need for large datasets for the training of deep learning models. The available medical imaging datasets are relatively smaller in size when compared

to the extensive non-medical datasets used in computer vision [Lit+17]. For instance, non-medical datasets like ImageNet boast over 14 million annotated images [Den+09], while the CIFAR-10 dataset includes 60,000 annotated images [Hop+17]. The current study is based on a privately collected dataset in Siemens, consisting of both defective and good images. Chapter 5 explains the dataset used in our study in detail.

Data preprocessing An important step of deep learning is data preparation. It involves annotating the data gathered and pre-processing it so that it is in a suitable format to pass it to the deep learning model. Data augmentation techniques can be employed to overcome the problem of a limited training dataset and reduce the overfitting of the model. This step also involves splitting the dataset into training, validation and testing.

Hardware and software In the realm of deep learning research, the synergy between hardware platforms and software frameworks has played a pivotal role in propelling the field forward. Modern deep learning heavily relies on processing complex matrix operations, and the introduction of Graphics Processing Units (GPUs) has been a game-changer. GPUs are highly parallel computing engines, boasting an order of magnitude more execution threads than Central Processing Units (CPUs). As a result, deep learning on GPUs is typically 10 to 30 times faster than on traditional CPUs. The ability to train large neural networks efficiently and rapidly has opened doors to tackling complex problems in various domains, including medical imaging, natural language processing, and computer vision.

Along with hardware, the development of open-source software frameworks paved the way for the popularity of deep learning methods [Sof+19b]. Here are some of the popular software frameworks:

- Caffe [Jia+14]: Developed by the Berkeley Vision and Learning Center, Caffe offers interfaces in C, C++, Python, and MATLAB. It has been widely adopted for its efficiency and versatility.
- TensorFlow [Aba+16]: Created by Google, TensorFlow supports multiple languages, including Python, C++, and R. It has become a cornerstone of deep learning research and is utilized extensively in Google's own research efforts.
- Torch [Col+11]: Utilized by prominent organizations like Facebook, Twitter, and Google, Torch offers a Lua-based deep learning framework. It has gained popularity for its flexibility and extensibility.

For our study, PyTorch [1] framework was used. This is explained in detail in future chapters in Section 6.2

Architecture Architecture defines how the layers of the neural network are structured. The choice of architecture is a critical consideration in this task, and there are two primary approaches to address it. The first approach involves the creation of a new CNN architecture by incorporating a variety of CNN layers. These layers can be selected from established options or even include the proposal of new, custom layers specifically tailored to address architectural challenges. While this approach is ideal for solving specific problems, it demands a substantial amount of labeled data for training. Additionally, it requires extensive effort to demonstrate the efficacy of the newly created architecture.

Conversely, the second approach entails the utilization of pre-existing CNN architectures that have been previously proposed and validated. Well-known architectures like ResNet-50, VGG16, and GoogLeNet are commonly adopted. These architectures have demonstrated high performance for standard image processing tasks and are often pre-trained on datasets with typical image content. Leveraging transfer learning, they can be applied to other problems with minimal data requirements. However, when adapting these architectures to address image processing challenges in different domains, a significant amount of labeled data becomes necessary. Nevertheless, this approach offers advantages such as efficiency and a foundation of proven success. In both approaches, a comprehensive understanding of CNN architectures is essential, including familiarity with the core CNN layers.

Different tasks require different network architectures, and selecting the right one can enhance performance. Researchers can either use existing CNN models or create their own custom architectures. Some popular architectures include:

- **ResNet-18**: The ResNet18 architecture is a variant of the Residual Neural Network (ResNet) introduced by K. He et al. in their paper [He+15a] [see Figure A.1 [Ram+19]]. ResNets are deep neural networks designed to address the challenges of training very deep networks. The key innovation in ResNets is the use of residual blocks, where the input x is transformed into $F(x)$, and this transformation is added to the original input, resulting in $H(x) = F(x) + x$.

- **EfficientNet-B0**: Introduced by Mingxing Tan and Quoc V. Le in 'EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,' [Tan+19] presents

[1] Open source and available to download from https://pytorch.org/

a novel approach to model scaling for convolutional neural networks CNNs. It achieves state-of-the-art performance with significantly fewer parameters compared to traditional methods. The specific EfficientNet variant used for evaluation may vary, as it comes in different sizes (e.g., EfficientNet-B0, EfficientNet-B1, etc.) [see Figure A.2].

- **DenseNet**: introduced by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in 'Densely Connected Convolutional Networks,' [Hua+16] is a deep neural network architecture known for its densely connected structure. DenseNet uses a feed-forward method to link every layer to every other layer. DenseNet-101 is a variant of the DenseNet architecture that specifically refers to a deeper model with 101 layers. The architecture is characterized by its densely connected blocks and transition layers that control the growth of feature maps [see Figure A.3].

.

Validation In machine learning, particularly in deep learning, validation techniques are used to assess a model's ability to generalize to new data. Two common methods are the holdout method and k-fold cross-validation.

- In the **holdout** method, data is randomly split into training, validation, and testing sets.
- In **k-fold cross-validation**, data is divided into k non-overlapping subsets. The model is trained and validated k times, with each subset taking turns as the validation set while the others form the training set in each cycle [refer Figure 3.10].

For any deep learning model, it's essential to validate its performance on an independent test set that the model hasn't seen during training and validation. This step is crucial to ensure the model's generalisability to the broader population.

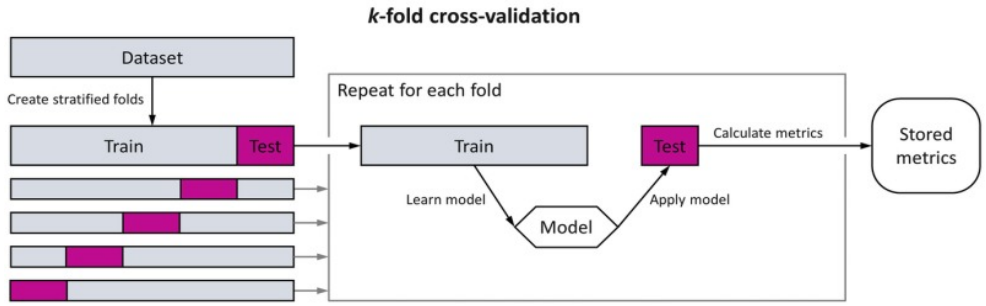


Figure 3.10: K-fold cross-validation schematic overview. The dataset is divided randomly into k-stratified folds. In each iteration, one fold is reserved as the test set, while the remaining k-1 folds are combined to create the training set for model development. Performance metrics are computed and recorded for the test set in each iteration, and this process is repeated for all k folds. [Dan+18]

Chapter 4

Deep Learning in medical imaging

Deep learning is gaining popularity in the field of radiology [Sof+19a]. A key advantage of deep learning lies in its ability to model non-linear relationships effectively. Incorporating non-linearities enhances the model's ability to adeptly capture the complexities inherent in the data. In this chapter, we discuss literature where deep learning has been applied to X-ray images and artifact detection in medical imaging.

Non Destructive Testing Since X-ray can identify inner structures, along with medical imagining, it has found its way into NDT of materials that remain invisible to the naked eye. NDT techniques utilizing X-rays, commonly known as X-ray testing, find extensive applications in various fields, including food product analysis, baggage screening, automotive part inspection, and quality control of welds.

In the automotive industry, the quality of aluminum alloy castings on mechanical products affects its safety performance. To ensure the quality, casting X-ray images can be taken to recognize the defects. [Jia+21] uses the CNN model to detect defective castings automatically. They used weakly supervised learning and introduced novel data augmentation techniques guided by attention maps to deal with fewer data problems. Mutual channel loss and commonly used cross-entropy loss were used so that the network focuses effectively on discriminative features.

[Boa+17] tries to automatically detect and classify defects (porosity, lack of fusion, cracks, slag inclusion) from double wall double image (DWDI) radiographic image of welded joints in the oil and gas industry. Their proposed algorithm consists of the following steps: identifying the ROI, detecting potential defects or discontinuities, extracting relevant features from these detected discontinuities, and ultimately classifying them based on their

characteristics using feed-forward MLP. For each detected discontinuity, the following set of features are extracted: area of discontinuity (A), eccentricity, solidity (A divided by convex hull), extension (ratio between defected area and the area of the smallest rectangle that encloses the discontinuity), ratio 1 (minor axis to A), ratio 2 (major axis to A), ratio 3 (major axis to minor axis), and rounding (perimeter squared divided by 4π times A). The three-layered MLP is trained with eight inputs corresponding to each feature, ten heuristically chosen neurons in the hidden layer, and five outputs(for each defect). Hyperbolic tangent was used as an activation function. The classification accuracy for the test data was determined to be 79.5%.

These paper uses deep learning-based approaches to detect defects through X-ray images for NDT.

Medical Domain Kwasigroch et. al [Kwa+17] in their paper address the challenges of limited data availability while dealing with medical data in training CNN for skin lesions classification. They upsample underrepresented classes by making their copies and using the data augmentation technique of width and height shift (0.1 of the image size), rotations (in range: -30 to + 30 degrees), and horizontal and vertical flips to solve overfitting. They use ResNet50 and VGG19 architectures for the task of classification.

Soffer et. al in their paper [Sof+19a] introduce deep learning technology and outline the stages involved in designing deep learning radiology research. It conducts a survey on the application of CNNs in radiologic imaging for major system organs, discussing challenges, future trends, and implications for radiology. It serves as a valuable guide for radiologists planning research in radiologic image analysis with these networks.

[Gao+17] detect interstitial lung disease (ILD) patterns from CT slice using deep CNNs. Since on single CT slices, multiple ILD diseases could coexist, they replace the single-label softmax-based loss with a multi-label classification loss. They use a variation of AlexNet, called CNN-F [Cha+14], and sum binary logistic loss for each class to get the final loss. As a result, high area under curve values and F1-score were achieved.

Context-encoding Variational Autoencoder (ceVAE), a unique technique in unsupervised learning for identifying abnormalities in medical MRIs, was proposed by Zimmerer et al. CeVAE,[Zim+18] in contrast to conventional techniques, combines density-based anomaly scoring with reconstruction to produce better outcomes in assessments at the sample and pixel levels. They used 140 K images and achieved an ROC-AUC of 0.95 on the BraTS-2017 dataset.

Artifacts and its detection MR imaging is susceptible to the presence of artifacts like chemical shift, radiofrequency and motion. While it is time-consuming and error-prone to detect these artifacts manually, Lim et.al in [Lim+22] use RISE-Net architecture to find these artifacts automatically in fetal MRI. They use two CNNs, first for the classification of artifacts in the image and second to determine the severity of the artifact detected. They achieve state of art results compared to other CNN architecture ResNet-50, Inception and VGG-16.

[Med+17] uses CNNs based algorithm for the automated detection of **motion artifacts** in MRI, eliminating the need for external sensors or hardware. Their architecture comprises a pair of consecutive convolutional layers, alternated with ReLU and pooling layers. Following the convolutional layers, are two fully connected layers, accompanied by a dropout layer, and finally, a classification layer. This algorithm's data-driven approach allows the classification of MR scans as either motion-free or motion-corrupted immediately after acquisition. This automated classification process holds significant potential for improving the quality of MR images in clinical settings, aiding in timely decision-making for further scans or data retention based on motion detection.

In conclusion, this chapter provides a comprehensive overview of deep learning applications in X-ray imaging, spanning non-destructive testing, automotive inspection, medical diagnosis, and artifact detection in MR imaging. The reviewed studies showcase the versatility and effectiveness of deep learning models in addressing challenges across various domains. It's noteworthy that we didn't find any studies focusing on AI-based artifact detection in X-ray imaging.

Chapter 5

Dataset Details

This chapter provides a comprehensive overview of the dataset utilized in this work and outlines the pre-processing steps applied to enhance its quality and for model development.

5.1 Original Dataset

In this work, we utilize an internally collected dataset provided by Siemens, consisting of radiographic X-ray images obtained using the specialized **'MAX wi-D' Detector**. Technical details of this detector are outlined in Table 5.1, offering essential insights into its specifications. Additionally, the physical appearance of the flat panel detector can be observed in Figure 5.1.

The dataset encompasses a wide range of clinical scenarios, including examinations of the chest, head, foot, and knee. The dataset that we are working with, consists of **intensity linear raw images** that underwent a pre-processing pipeline encompassing three key steps:

- **Offset correction**: The first step involved the subtraction of an image captured without radiation. This correction effectively eliminated the inherent offset in the images.
- **Gain correction**: This process involved dividing each image by another reference image with a known constant radiation value. This step ensured that the images were normalized and free from any variations in gain.
- **Defect correction**: The final step addressed any known defects in the images. Interpolation techniques were applied to rectify these defects.

Table 5.1: Technical Parameters

Pixel Matrix	2350 x 2866
Size	35cm * 43cm
Weight	3.3 kg
Pixel Pitch	148 µm
Pixel Matrix	2350 x 2866
Thickness	19mm
System	Wireless

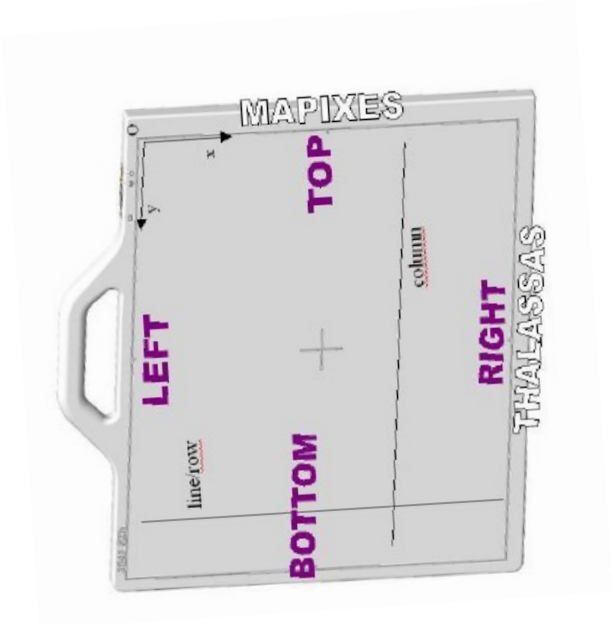


Figure 5.1: Flat panel detector: Source: Siemens

The dataset contains radiological X-ray images, which are distorted and contain artifacts. The artifacts in question are **not anatomical in nature**; instead, they are **technical artifacts** resulting from the failure, malfunction or effect of external disturbance on our detector, affecting the image quality.

Furthermore, these images have been thoroughly inspected by experts for artifacts. The resulting database is organized into **eight folders**, with seven corresponding to specific artifact classes and one dedicated to good images. The database provided has 521 good images and 199 defective images from different anatomies. Each category contains a distinct number of images compared to the other folders. The table shows a detailed count of given images per defect. Figure 5.2 shows the pie chart with percentage of 'defective' vs 'good' X-ray images in our dataset.

Artifact Name	Count of images
Defective images	199
Good images	521

Table 5.2: Artifact Names and Count of images

Each image has **16 bits** per pixel (bpp) resolution means intensity between 0 to 2**16, represented in **gray levels**, and has undergone inspection by experts. Each image has a resolution of **2350** $\times$ **2866** pixels, having a single channel, and is provided in **DICOM** format. Labels or annotations are provided for entire images, indicating whether an image contains a particular artifact class or not. Notably, there is no information specifying the location of the artifact within the image.

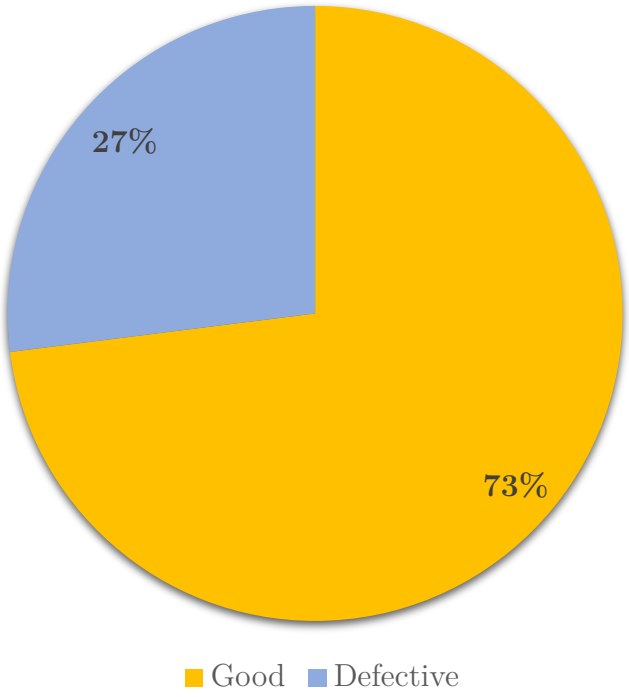


Figure 5.2: Pie chart shows imbalance between 'good' and 'defective' X-ray images present in our dataset

The dataset provided was imbalanced. Formula for Imbalanced Ratio (IR) [Wes+21] is:

$$IR = \frac{|\ \text{majority class}\ |}{|\ \text{minority class}\ |} \tag{5.1}$$

IR for good and defect is:

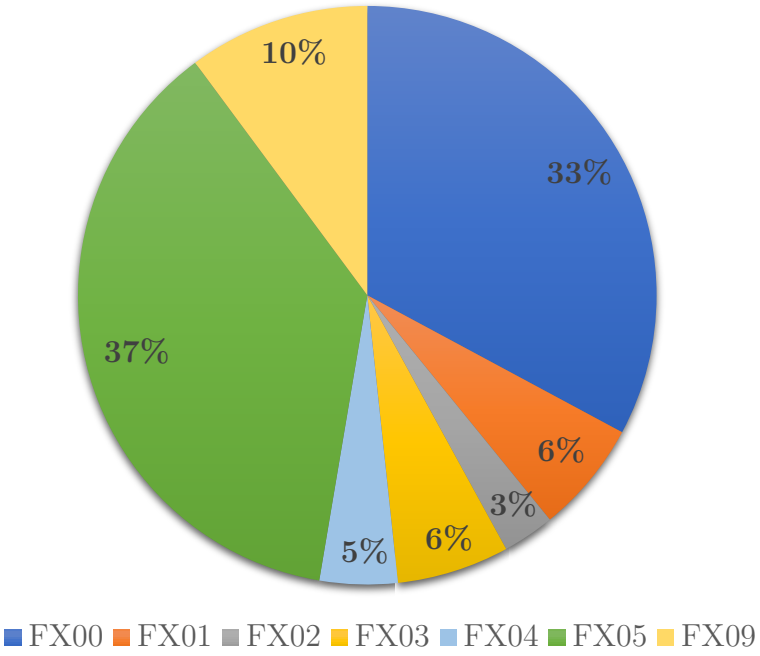
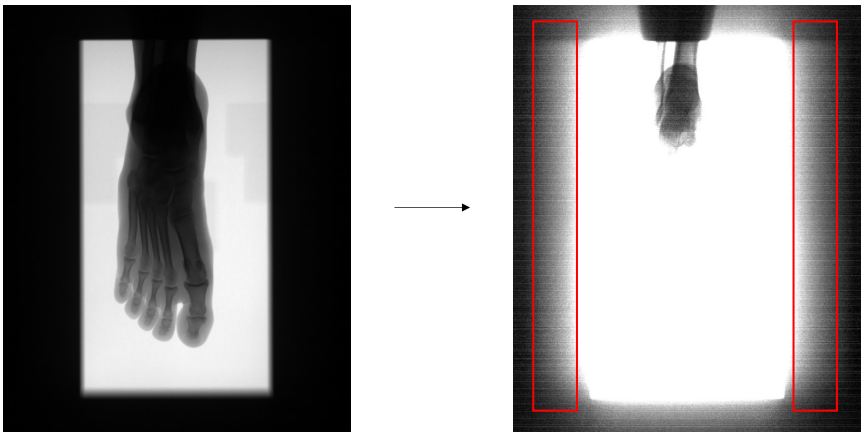
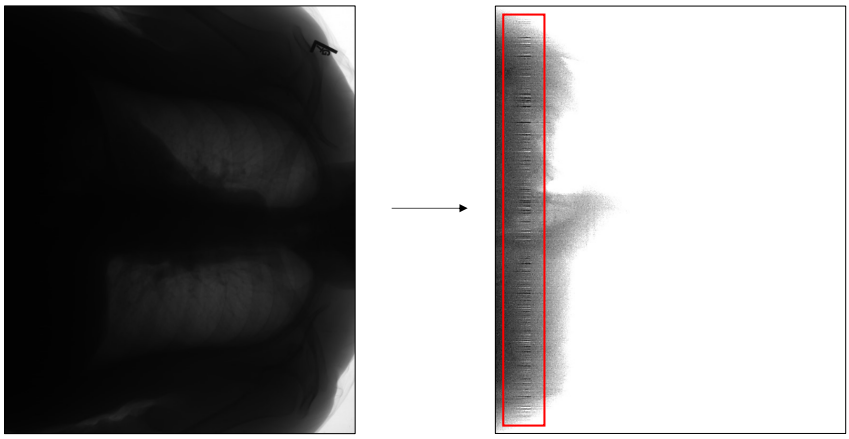


Figure 5.3: Pie chart shows percentage of each artifact in defective images present in our dataset.



(a) Figure shows FX00 sporadic line artifact (left) and adjusted version (right) where continuous line across image can be seen (highlighted in red)



(b) Figure shows FX02 group of line artifact (left) and adjusted version (right) where a continuous line can be seen just along a column (highlighted in red)

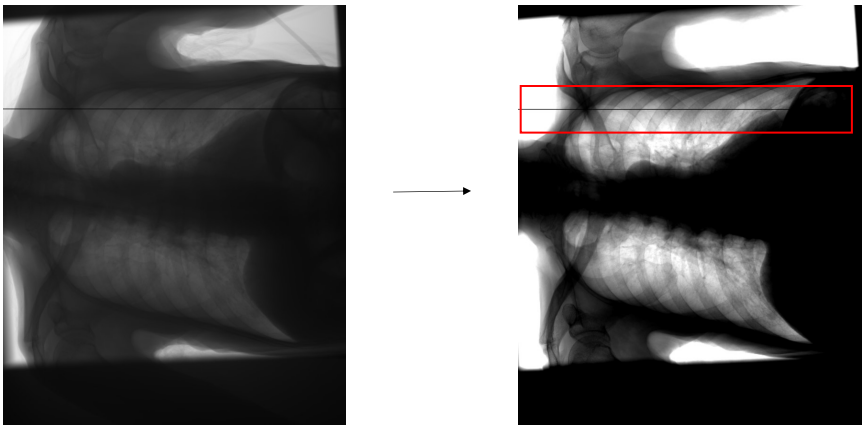
Figure 5.4: Raw X-ray and its contrast adjusted copy

$$IR = \frac{|Good|}{|Defect|} = \frac{521}{199} = 2.62 \tag{5.2}$$

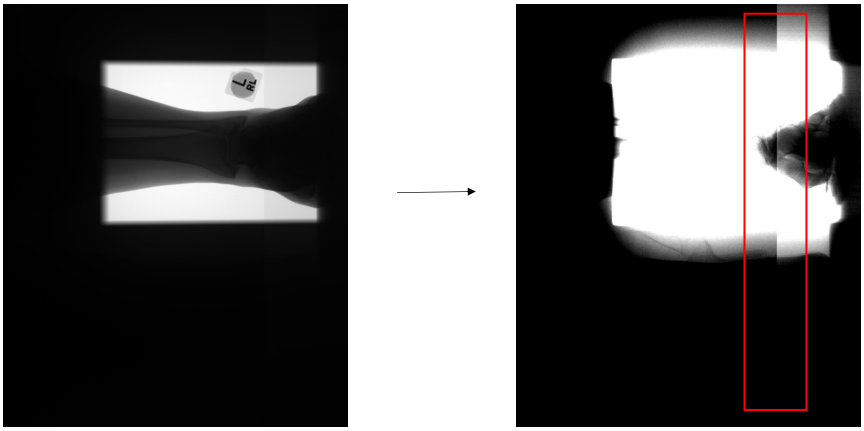
Image parameters, such as brightness, and contrast are adjusted using ImageJ software to enhance the visibility of the artifact.

5.1.1 Artifact Details

The dataset consists of 7 defect classes corresponding to each artifact type. Artifact classes named by experts and their abbreviations used throughout the thesis are - 'FX00_Sporadic_line' as FX00, 'FX01_Line_artifacts' as FX01, 'FX02_Group_of_line_artifacts' as FX02, 'FX03_Partly_brighterdarker' as FX03, 'FX05_Defect_line' as FX05, 'FX04_Group_of_defect_lines' as FX04 , 'FX09_Stripes' as FX09 and 'good' class. Figure 5.4 shows how artifact looks visually. Table 5.4 summarizes in the detailed description and the root cause of each artifact. Table 5.3 summarizes the count of each artifact and their count along with abbreviation used in our thesis report. Figure 5.3 shows percentage of each artifact class in 'defective' images present in our dataset.



(c) Figure shows FX05 defect line (left) and its contrast adjusted version (right) where single defect line is missing (highlighted in red)



(d) Figure shows FX03 partly brighter artifact (left) and its contrast adjusted version (right) where a sudden change, in contrast, can be seen (highlighted in red)

Artifact Name	**Defect/Symptom description**	**Root cause information**
FX00 – Sporadic line artifacts	• Continuous multiple lines • Issue is highly sporadic	• Unknown • Possibly microphony

Artifact Name	**Defect/Symptom description**	**Root cause information**
FX01- Line Artifacts	• Line patterns observed in images • Line artifacts pattern is not stable. It changes from image to image • Effect seems to be sporadic	• Unknown
FX02 - Group of line artifacts	• Line artifacts can be seen in some areas of the image • The line artifacts are localized on the fields of the AEC chamber, or on one of them	• EMI-induced artifacts • Not a defect detector!
FX03 – Partly brighter/-darker	• Some part of the image appears brighter • There is a jump-in signal at the transition	• Flex cable has come loose • Broken soldering
FX05 - Defect line	• One or more lines show low signal magnitude • Failure might be sporadic, and the signal in this line might be different from image to image • Lines in both directions affected (rows and columns) • Sometimes neighboring lines are affected. The signal goes down and up in the adjacent ones	• Flex cable has come loose • Bad/broken soldering • Gate driver failure

Artifact Name	**Defect/Symptom description**	**Root cause information**
FX04 - Group of defect lines	• Multiple lines provide false signals • The course of the signal is always similar - some lines deliver a very low signal, and then there is a jump where further lines deliver a higher signal • Lines in both directions affected (rows and columns)	• Flex cable has come loose • Bad/broken soldering • Gate driver failure
FX09 - Stripes	• Broad stripes visible • Artifact affects almost the complete detector (throughout the X-ray image) • Artifacts are known only in images with pacemaker/heart pumps	• Probably EMI-related issue

Table 5.4: Artifact names, defect description, and their root cause

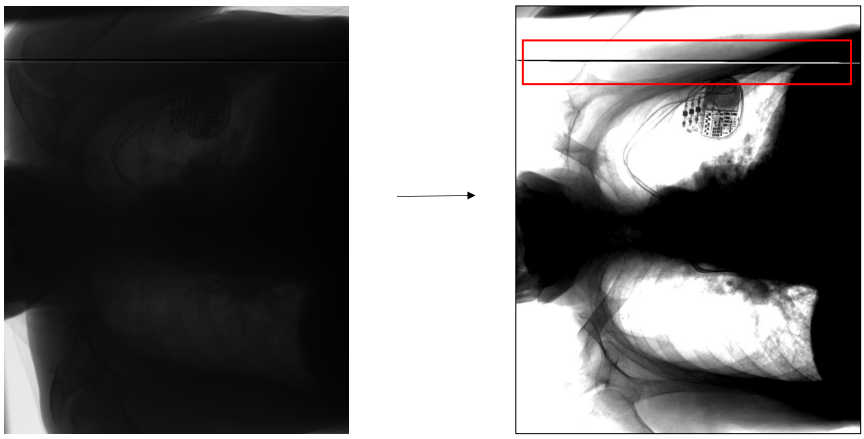
5.1.2 Ambiguity In Dataset

During the examination of the dataset, several instances of ambiguity were identified, leading to the following observations:

(1) Some images exhibited the presence of multiple defects, resulting in a multi-label scenario [see Figure A.4].

(2) Manual inspection revealed instances of mislabeling, where defects of one class were erroneously assigned to another. Most examples that were mis-labeled were FX04_grp_of_defect_line and FX05_defect_line.

(3) Certain images in the dataset were identified as completely 'disturbed' [see Figure A.5].



(e) Figure shows FX04 Group of the defect (left) and contrast adjusted version (right) where a group of defective lines can be seen (highlighted in red)

Artifact Name	Abbreviated As	Count
FX00 – Sporadic line artifacts	FX00	68
FX01- Line artifacts	FX01	13
FX02 - Group of line artifacts	FX02	6
FX03 – Partly brighter/darker image	FX03	13
FX05 - Defect line	FX05	77
FX04 - Group of defect lines	FX04	9
FX09 - Stripes	FX09	21
Good images	Good	521

Table 5.3: Artifact Names, Abbreviations, and Counts

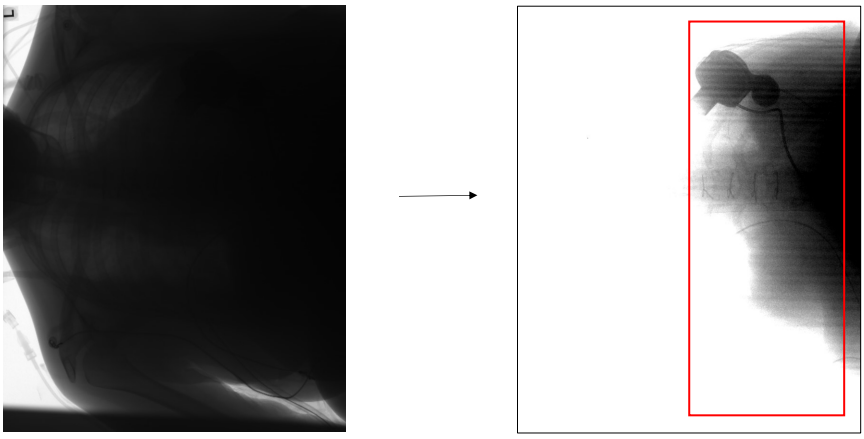
(4) Out of the 21 images in the FX09_stripes class, three were identified as post-processed or post-processed type [see Figure A.7].

(5) Some images in the dataset contained offset images, not containing clinical content [see Figure A.6].

5.2 Dataset Pre-processing

Pre-processing was performed on the provided dataset in the following steps

(1) Images in the folder were reviewed manually, to check for inconsistency. Misclassified images were identified and relocated to their correct folders to ensure accurate categorization.



(f) Figure shows FX09 stripes (left) and contrast adjusted version (right) where stripes pattern can be seen (highlighted in red)

Figure 5.4: Raw X-ray and it's contrast adjusted copy where artifact can be clearly seen (highlighted in red)

(2) Images presenting multiple defects prompted placing them in both respective defect folders, facilitating a multi-class learning approach. It's noteworthy that such instances were relatively rare in the dataset.

(3) Disturbed and offset images were also removed from the dataset.

(4) Post-processed images in the database (mostly in FX09_stripes) were removed because we are dealing with only intensity-linear raw images.

(5) DICOM images were converted into tif format using ImageJ software to create uniform data representation.

(6) Images of a shape other than 2350×2866 were removed because they are most likely from another detector and we are dealing with only images from 'MAX wi-D detector'.

(7) Sometimes images were rotated, and the shape was adjusted so all the images were consistent in shape.

(8) Since FX00 – Sporadic line artifacts and FX01- Line artifacts have similar characteristics and defect appearance. FX00 – Sporadic line artifacts only occur in a single image at the customer site, and if they occur they look like FX01- Line artifacts, but might be gone in the next acquisition. They cannot be differentiated in a single image, so it was decided to combine them into the FX00 – Sporadic line artifacts category.

(9) Data splitting into train and validation was done as discussed in Section 5.2.

Dataset Splitting

Dataset splitting is the process of dividing a dataset into distinct subsets, typically into training, validation, and test sets. This partitioning is essential to assess and improve the performance of machine learning models. The process serves as a foundation for model training and evaluation.

Why is it needed?

- **Model Evaluation:** Dataset splitting enables evaluating the model's performance on unseen data, which is crucial for assessing its ability to generalize to new, real-world cases.
- **Preventing Overfitting:** Reserving a portion of the data for validation helps prevent overfitting, where a model becomes too specialized in the training data and doesn't perform well on new data.

Implementation

- We perform data splitting for all approaches in a similar way.
- Due to the limited size of the dataset, especially for specific artifacts like the FX_03 group of line artifacts (with only 6 images), the test set could not be separated out.
- The dataset was split into 80% for training and 20% for validation.
- The validation set also contains the augmented data [discussed in Section 5.2].
- To ensure unbiased evaluation, we applied a 5-fold cross-validation approach, which repeated this splitting process five times with different subsets each time.

Data Augmentation

Data augmentation is a crucial technique in machine learning and computer vision, particularly for training deep learning models like CNN. It involves applying various transformations to the existing dataset to create new, artificial data while preserving labels. It's noteworthy to mention, that we first split the data and then apply augmentations to it.

Why Data Augmentation is Required?

- **Increased Data Diversity:** Augmentation diversifies the dataset by generating multiple versions of the same image with alterations. This helps the model learn to be more robust and generalize better to different scenarios.
- **Overcoming Limited Data:** In cases where the dataset is small, augmentation can artificially increase the size of the training data, mitigating the risk of overfitting and improving model performance [Kri+12].

This augmentation process was performed in such a way that the original shape and structural characteristics of the images were preserved while introducing diversity into the dataset. In our experiments, data augmentation was carried out offline mode, involving **seven** distinct augmentation techniques: **vertical flip, horizontal flip, flip in both directions, inversion and the combinations including inverted horizontal flip, inverted vertical flip, and inversion with both flips**. We do not augment good class images. Figure A.8 shows all seven augmented images from the original image.

5.2.1 Raw Images

In this section, we deal with the pre-processing steps employed, specifically for image-based models.

Artifact Name	Count of images
FX00 – Sporadic line artifacts	79
FX02 - Group of line artefacts	6
FX03 – Partly brighter/darker image	12
FX05 - Defect line	73
FX04 - Group of defect lines	11
FX09 - Stripes	18
Good images	521

Table 5.5: Artifact Names and image counts after ambiguity resolved and pre-processing

Data Resampling

The provided image size of 2350×2866 is large and can present challenges when using CNNs for analysis. In practice, such large images can lead to significant computational demands and limitations on the batch size during model training. Experiments with large CNN

models could only be performed with a batch size of 1, which means that only one image is processed at a time. However, the model may not have access to the beneficial effects of batch normalization, which negatively impacted training stability and the quality of results. Therefore, it becomes imperative to explore data resizing techniques to address the computational challenges associated with large images, allowing for more efficient model training and improved outcomes in image analysis tasks.

We tried various techniques to decrease the image sizes, such as bilinear, closest neighbor, and other rescaling techniques. Even while these techniques worked well in many situations, there were certain circumstances when they weren't sufficient. For example, the defect line was absent from the resized image of dimension 1024×1024 [see Figure 5.5]. We observed the same for other types of defect categories, too. This experience led us to the conclusion that we need to explore alternative approaches for dealing with large images, as the data resizing technique proved to be ineffective.

Figure 5.5: Original X-ray image with the FX05_defect_line artifact, initially sized at 2350×2866, has been resized to 1024×1024 using the `cv2.resize` function with the `cv2.INTER_NEAREST` flag for interpolation. In the resized image, it is notable that the presence of the artifact is no longer seen.

Data Patching

Data patching played a crucial role in the data pre-processing phase to address the challenges posed by large images (2350×2866) used in CNN models. In data patching, we divide these large images into smaller, more manageable sections or patches and subsequently categorize and organise these patches according to their content. This approach was adopted to enhance the efficiency and effectiveness of our CNN model for image classification.

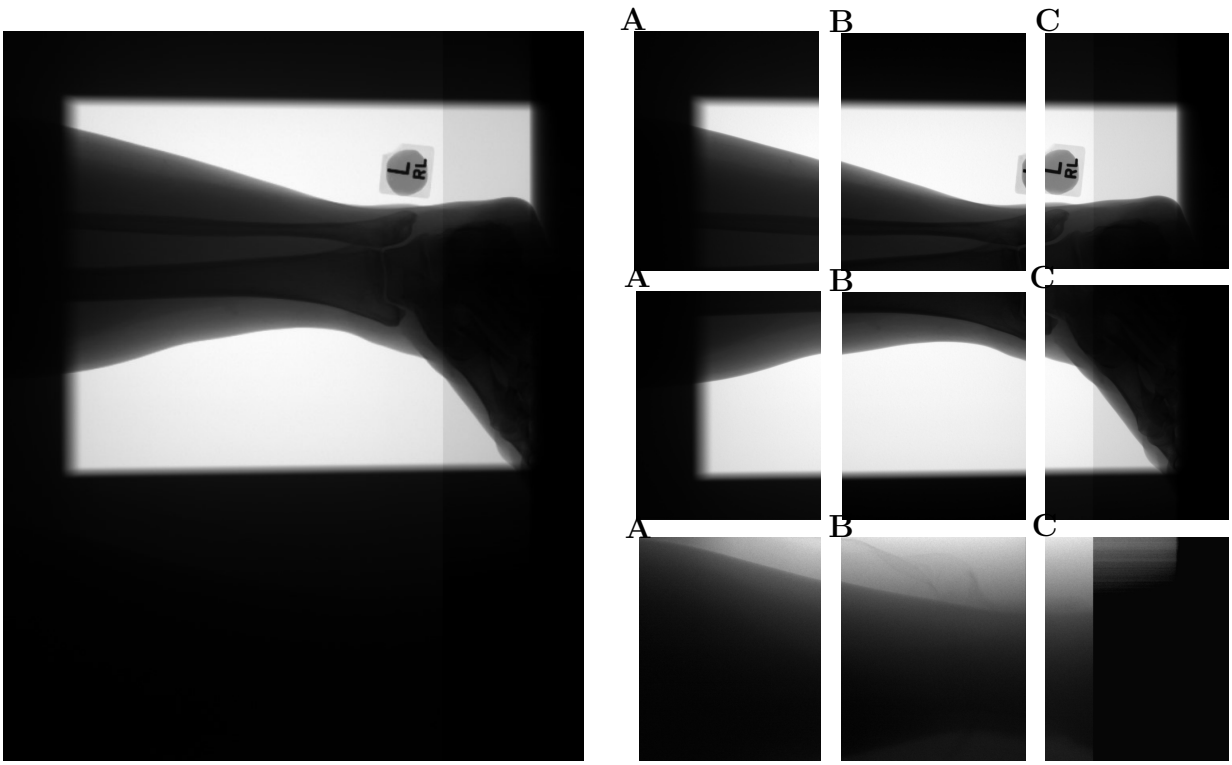


Figure 5.6: Each image of size 2350 × 2866 was patched into 9 parts of shape 783 × 955, then each patch was manually checked for defect and put into the respective defect folder, [A] belongs to FX09, [B] belongs to good and [C] belong to FX03

Rationale: The decision to implement data patching in our experiments was based on several key factors:

- **Computation and Memory Efficiency**: Large images can pose significant computational and memory challenges for deep learning models. By breaking these images into smaller patches, we reduced the resource requirements for processing and training our CNN model.

- **Improved Localization**: Smaller patches facilitated better defect localization. This made it easier for the model to identify and classify defects accurately, instead of attempting to classify the entire large image.
- **Enhanced Generalization**: Data patching created a more diverse and comprehensive dataset. This diversity was essential for improving the generalization capacity of our CNN model, enabling it to recognize defects in varying contexts and scales.
- **Manual Intervention**: Manually categorizing patches into respective folders ensured the integrity of our dataset and the accuracy of labeling. This hands-on approach enabled us to maintain high-quality training data for our CNN model.

Implementation

(1) Image Partitioning: The initial step involved dividing the large 2350×2866 images into smaller, non-overlapping, **nine** patches, each of shape 783×955. This partitioning was performed systematically, ensuring that each patch maintained the structural and contextual information necessary for accurate classification.

(2) Categorization: Each resulting patch was manually categorized into one of several folders based on its content. For instance, patches containing 'good' regions of the image were placed into the 'good image' folder, while patches corresponding to defects were sorted into respective defect folders. This categorization was essential for creating a labeled dataset on which the CNN model could be trained.

Now, the dataset contains a size of image 783×955, with a count of images in each folder given in Table 5.6

Artifact Name	Artifact Count
FX00 – Sporadic line artifacts	714
FX02 - Group of line artifacts	18
FX03 – Partly brighter/darker image	36
FX05 - Defect line	186
FX04 - Group of defect lines	54
FX09 - Stripes	42
Good images	5208

Table 5.6: Artifact and their respective counts after patching images

Line Profile

A **line profile** of an image, often referred to as a *profile plot* or *pixel intensity profile* is a graphical representation that shows the variation in pixel intensities along a specific line. It reveals information like transitions between different regions, the presence of edges, the presence of patterns, and other image characteristics about the image's content along the chosen path. It can be can be represented as a one-dimensional time series. Let's denote the pixel intensities along the specific line or path within the image as $P = p_1, p_2, ..., p_n$, where n is the number of pixels along the line. Each p_i represents the intensity value of the i^{th} pixel. The line profile can be seen as a temporal sequence, where the position along the line corresponds to time steps. This way, the pixel intensity variation forms a time series classification problem. To represent this mathematically, let t be the time variable $t = 1, 2, ..., n$ and X_t represents the intensity value at time t.

Why line profile-based approach be better?

- **Effective for straight line defects**:A line profile-based approach is particularly well-suited in our specific case, where defects are consistently manifested as straight lines originating from the detector. It can excel at capturing and characterizing such defects.
- **Reduced computational complexity**: Processing line profiles can be computationally less intensive than directly analyzing the entire image, especially in cases where images are high-resolution or large in size. Image-based CNN models require GPU to train, but time series-based models can be trained on CPU as well.
- **Noise reduction**: It becomes possible to reduce the impact of noise or irrelevant details e.g. shown in Figure 5.7, in the classification process. This can lead to better classification accuracy, especially in scenarios where images may contain a lot of irrelevant information.
- **Lesser memory requirement**: The line profile of an image would take less memory to store than the image itself. Each image in our dataset, on average, would occupy 15Mb of storage space, but the line profile would only need a couple of Kbs to store.
- **No need for patching**: While images were divided into patches due to computational constraints when passing them to the model, now there is no requirement to do so; the complete line profile of the entire image can be fed to the model all at once.

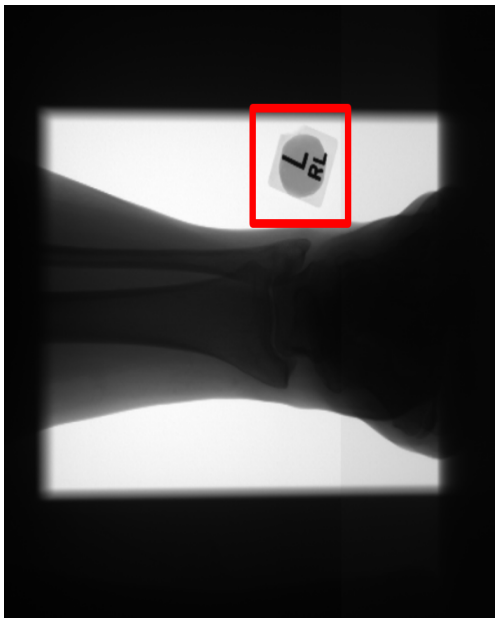


Figure 5.7: Figure shows an X-ray image containing lead marker, which could be noise

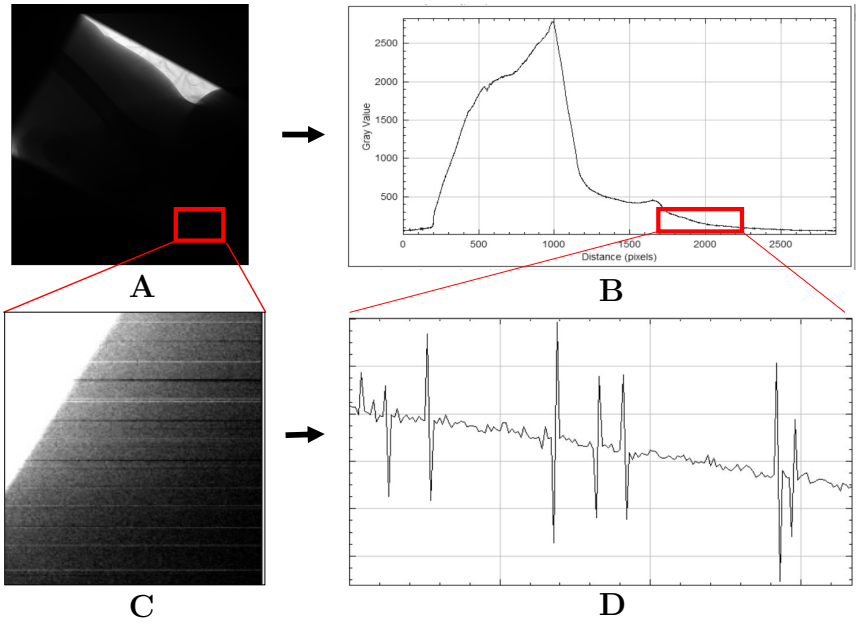
Line Profile Pre-processing

(1) **Profile extraction**: To obtain the line profile, we perform a summation of pixel values along a specific direction, resulting in a profile oriented in that direction. This process effectively transforms the image into a time series.

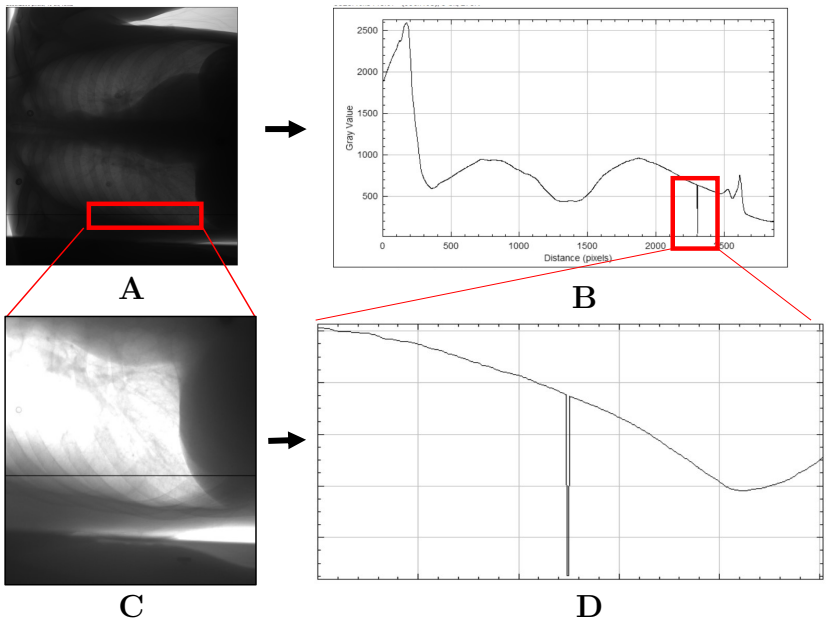
(2) **Profile direction selection**: In our implementation, we focus on vertical and horizontal line profiles by summing along rows and columns, respectively, due to the nature of the artifacts, which always occur in either horizontal or vertical directions.

(3) **Concatenation of profiles**: For certain artifacts, such as FX05 and FX04, artifacts can occur in either vertical or horizontal direction. Consequently, it is necessary to merge both the horizontal and vertical profiles. This fusion results in a single 1D data representation with a dimensionality of 5216, achieved by combining the vertical profile with 2866 values and the horizontal profile with 2350 values.

Characteristics of line profile of each type of defect Line profiles of each kind of artifact look quite distinct from one another. Figure 5.8 shows an X-ray image along with their line profile in a particular direction. The table in appendix Table A.1 shows the distinctive characteristics of each artifact along their line profile directions.

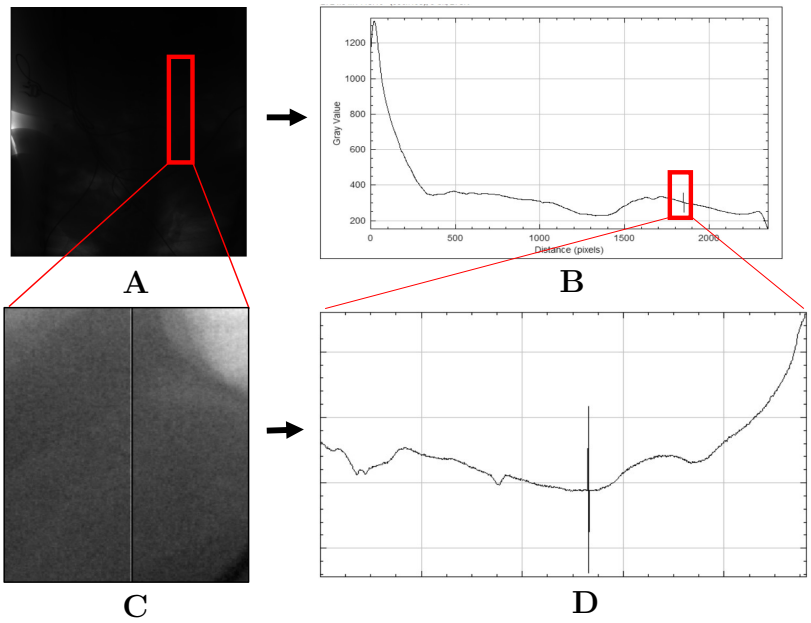


(a) Figure shows [A] X-ray image with FX00_sporadic_line_artifact [B] with its corresponding line profile in horizontal direction [C] brightness, contrast adjusted and zoomed part of the original image in which sporadic pattern is clearly visible [D] Zoomed part of line profile, where the artifact is seen clearly

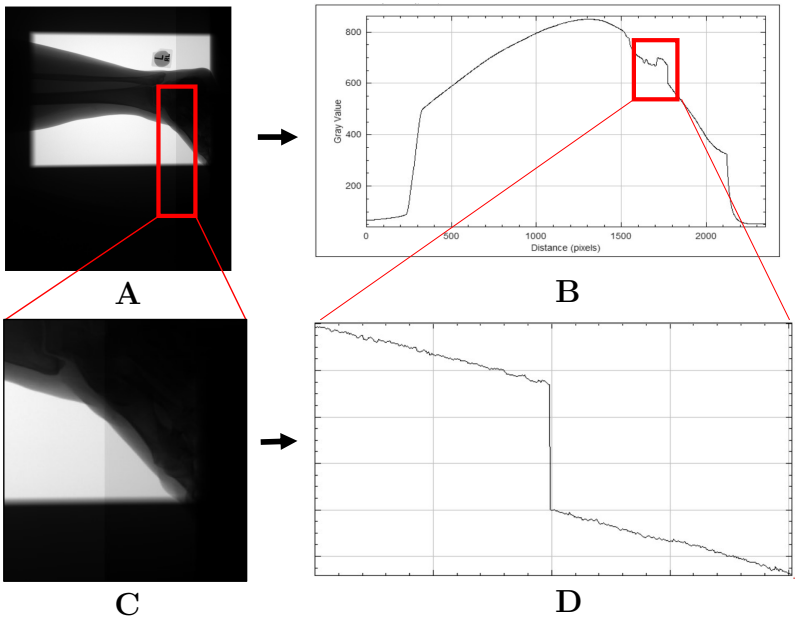


(b) Figure shows [A] X-ray image with FX05_defect_line [B] with its corresponding line profile in horizontal direction [C] brightness, contrast adjusted and zoomed part of an original image in which the artifact is clearly visible [D] Zoomed part of line profile, where defect line is seen clearly

Figure 5.8: Line profiles for each type of artifact

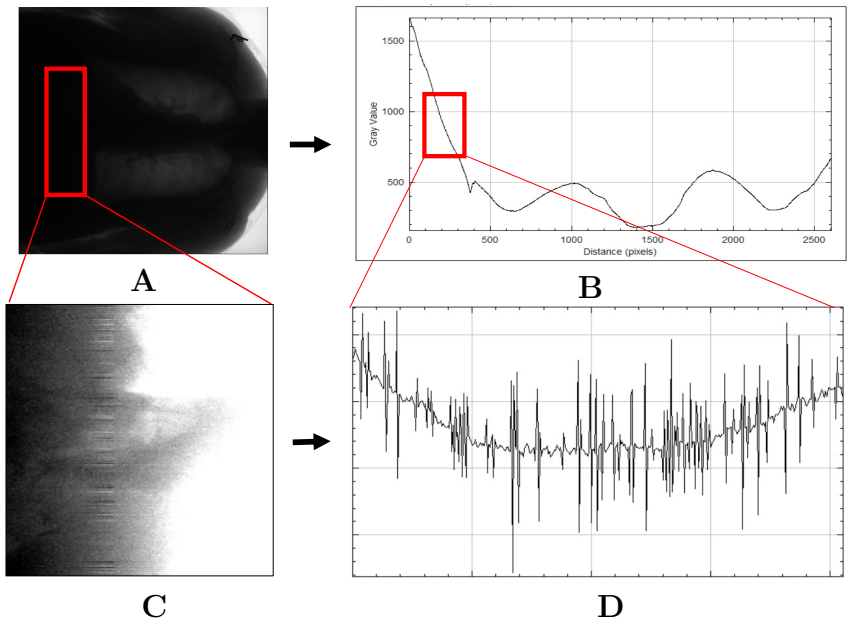


(c) Figure shows [A] X-ray image with FX04_group_of_defect_line [B] with its corresponding line profile in vertical direction [C] brightness, contrast adjusted and zoomed part of the original image in which the artifact is clearly visible [D] Zoomed part of line profile, where a group of defect line is seen clearly

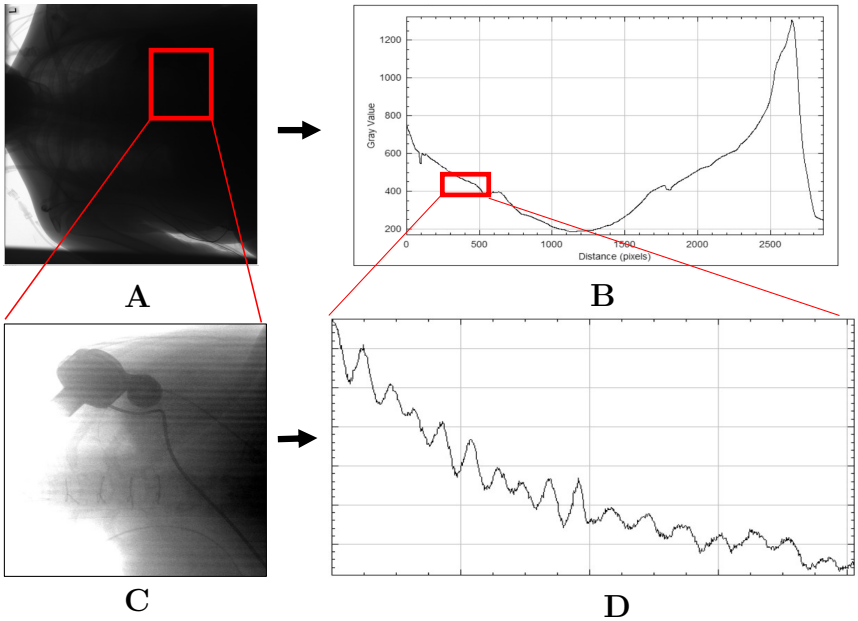


(d) Figure shows [A] X-ray image with FX03_partly_bright [B] with its corresponding line profile in vertical direction [C] brightness, contrast adjusted and zoomed part of an original image in which the artifact is clearly visible [D] Zoomed part of line profile, where a partly bright defect is seen clearly

Figure 5.8: Line profiles for each type of artifact



(e) Figure shows [A] X-ray image with FX04_group_of_defect_line [B] with its corresponding line profile in vertical direction [C] brightness, contrast adjusted and zoomed part of an original image in which the artifact is clearly visible [D] Zoomed part of line profile, where a group of defect line is seen clearly



(f) Figure shows [A] X-ray image with FX04_group_of_defect_line [B] with its corresponding line profile in vertical direction [C] brightness, contrast adjusted and zoomed part of an original image in which the artifact is clearly visible [D] Zoomed part of line profile, where a group of defect line is seen clearly

Figure 5.8: Line profiles for each type of artifact

5.2.2 Post-processed Images

Raw X-ray images obtained through digital image acquisition often come with several challenges and limitations. They contain just the raw data of the detector with minor corrections to have a (usually) artifact-free image. This image is intensity linear, which is very different from the look of a film, radiologists are used to. They are usually not directly used for diagnostics. This is why, before the images are shown to the customer/radiologist, they are subject to so-called post-processing. This changes the appearance to be more suitable for diagnostics, adjusting for the proper contrast, ensuring high image quality, and displaying the overall image impression that the radiologist is used to [Tom+22].

The raw images undergoing steps of *artificial adjustment* are called **'post-processed images'**. These are often easier for radiologists to interpret and can lead to quicker and more accurate diagnoses, ultimately benefiting patient care.

Post-processing steps employed include:

- **Cropping**: Post-processing can involve cropping the image to focus on specific areas of interest to highlight abnormalities or structures of clinical significance.
- **Image enhancement**: Post-processing techniques are used to enhance image contrast, and improve overall image quality. This can involve techniques like *frequency-dependent filtration, contrast normalization, noise reduction, histogram equalization* etc.[Kru+07].

A handful of post-processed images were provided by Siemens. Table 5.7 shows count images of the post-processed type containing different types of artifacts.

Details of this post-processed dataset are as follows:

- The format of the image was DICOM. Each image was 16-bit.
- These images are often cropped, so they are of varied sizes and no longer the same size as raw images.
- Artifacts in these images are clearly visible without adjusting brightness and contrast.
- The dataset was not labeled and sorted for the kind of artifact it contained.

Data Pre-processing

Some pre-processing steps are employed to post-processed images before passing them to our trained model.

Artifact Name	Count of images
FX00 – Sporadic line artefacts	5
FX02 - Group of line artefacts	6
FX03 – Partly brighter/darker image	1
FX05 - Defect line	15
FX04 - Group of defect lines	3
FX09 - Stripes	1
Good images	12

Table 5.7: Table shows the count of post-processed images and their artifact type

- The DICOM format is converted into tif using ImageJ
- Since the dataset was originally not labeled, the images were inspected and sorted into the respective folder. Figure 5.9 show a few examples of post-processed images, each containing one type of artifact.

Image Since the images have varying sizes, a pre-processing step is applied to ensure *uniform patch sizes*. If the size of the post-processed image is smaller than 783×955, the image is padded to reach this size. The padding is performed by replicating the last available values using 'cv2.copyMakeBorder' with the 'cv2.BORDER_REPLICATE' border type.

The number of patches that can be extracted from the image is determined by Algorithm 5.1, which calculates the number of patches necessary to cover the entire image. Each patch can be constructed by either padding the remaining values with zeros or repeating the starting values at the end to ensure that the missing pixels are appropriately filled, resulting in a patch size of 783×955. This approach allows for consistent and uniform processing of images with varying sizes.

```
patch_shape = (783,955)
ndim = postprocessed_image.ndim
num_patches_x, num_patches_y = [
    int(np.ceil(postprocessed_image.shape[i] / patch_shape[i])
       ) for i in range(ndim)
]
```

Listing 5.1: Code snippet for the number of patches in the x and y direction of post-processed images

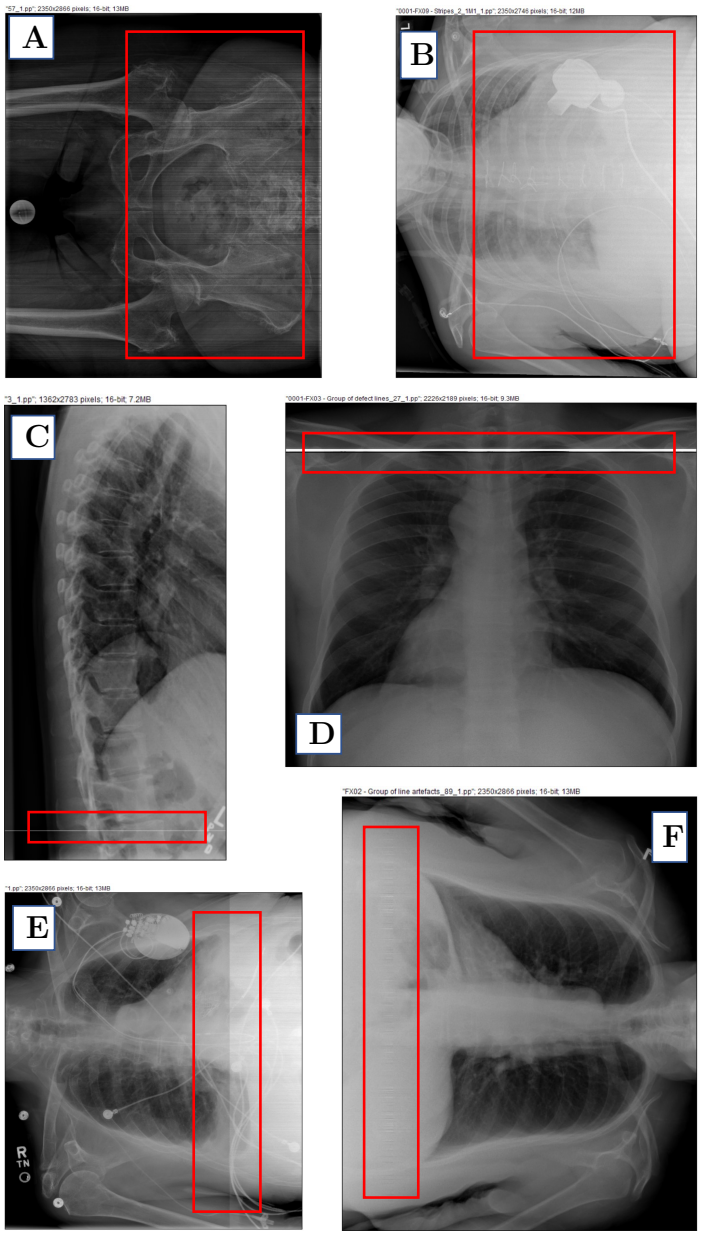


Figure 5.9: Figure shows all post-processed images containing [A]FX00 [B]FX09 [C]FX05 [D]FX04 [E]FX03 [F]FX04 artifacts

In this way, the images are prepared for further analysis and processing, and the patch creation procedure ensures they are suitable for subsequent tasks.

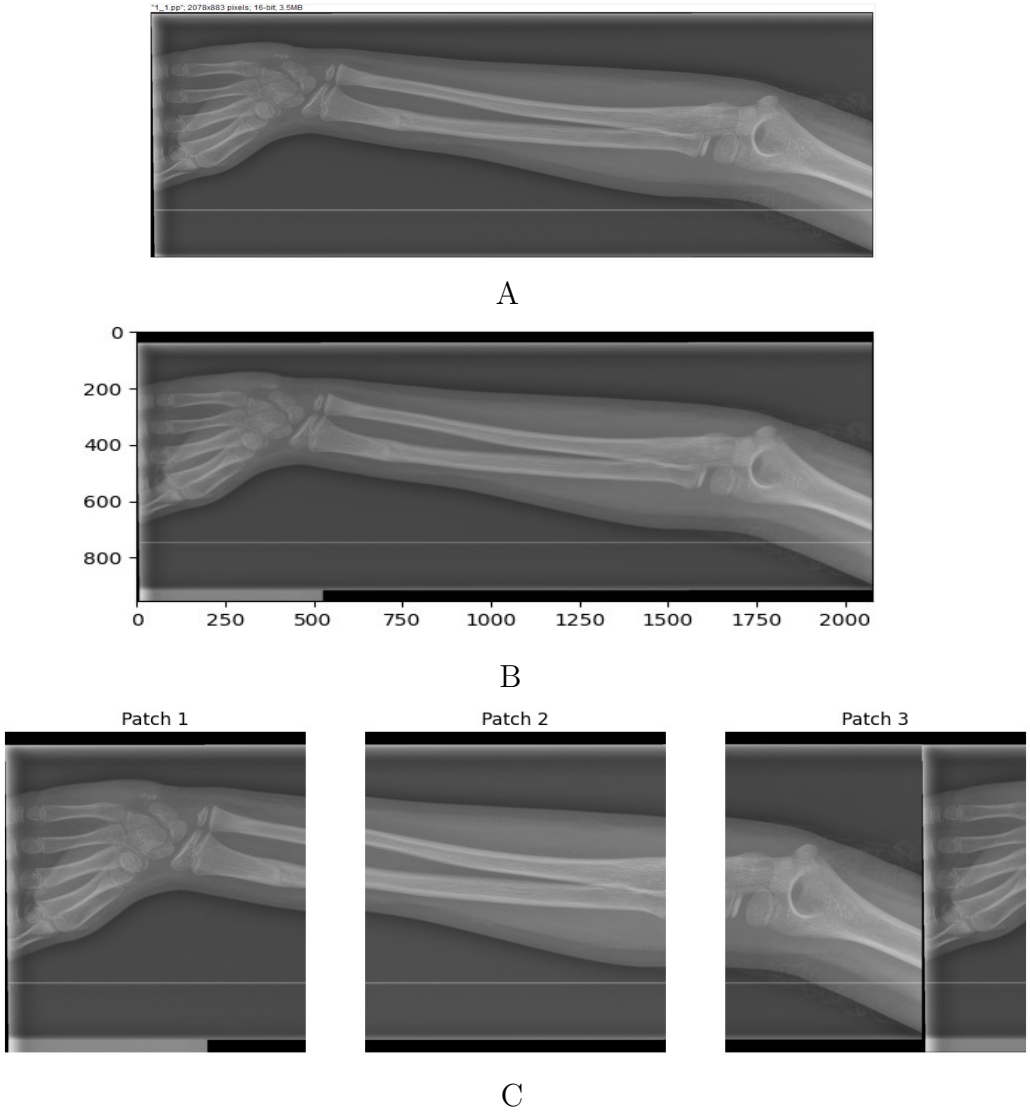


Figure 5.10: [A] shows a post-processed image of 'FX05' type (white line seen at the bottom) of shape 2078×883 [B] Since $883 <$ patch size(955), the image is padded at top and bottom by repeating the first and last values, so shape becomes 2078×955. [C] Using the formula given in 5.1, the number of patches is 3, each having a shape of 783×955

Line Profile This section addresses the pre-processing steps specifically designed to manipulate line profiles extracted from post-processed images. Given the inherent variability in the sizes of these images, our approach involves first extracting both the column and line profiles from the original post-processed image. To create a standardized representation

conforming to a fixed size of 5216, which corresponds to the concatenation size of the horizontal and vertical line profiles of the raw image, a systematic procedure is employed. Recognizing the potential disparity in the lengths of profiles, a meticulous resizing strategy is implemented. The last value of the concatenated profile is iteratively repeated until the desired size of 5216 is reached [see Figure 5.11]. In this way, line profiles of post-processed images, are made ready for further analysis.

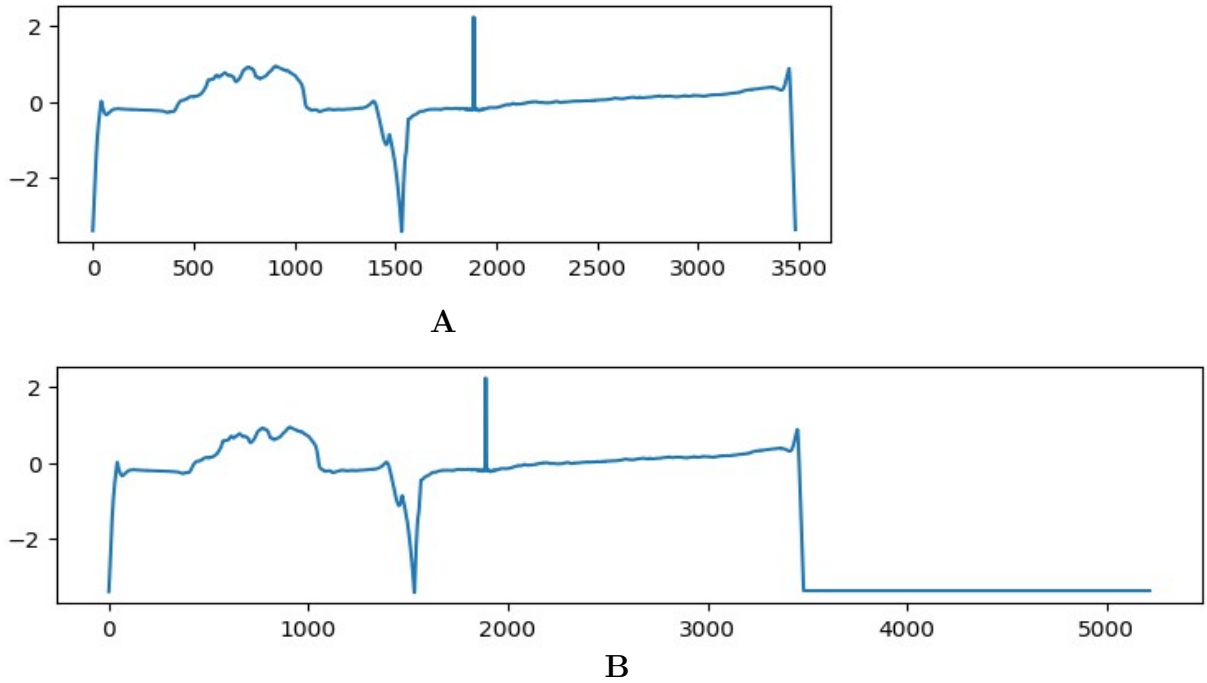


Figure 5.11: [A] shows a line profile of a post-processed image of the 'FX05' type (sudden peaky line seen) of shape 1×3500. [B] Line profile is padded by repeating the last values to make the shape of 1×5216

5.3 Summary

In summary, this chapter comprehensively explores the technical artifacts in the dataset, detailing their types, counts, and the pre-processing steps employed. We discuss details of the line profiles associated with these artifact images and the pre-processing of these line profiles. Additionally, a distinction was made between raw and post-processed images, detailing the specific pre-processing steps applied to each post-processed image and their respective line profiles. This comprehensive discussion sets the stage for an understanding of the dataset and lays the foundation for subsequent analyses and model development.

Chapter 6

Methods and Results

The upcoming chapter is structured to provide a thorough exploration of our experiments, starting with a discussion on the evaluation metrics employed. We then delve into the experimental setup, presenting the methodology, the architecture of our model, and their quantitative results. Additionally, we address the interpretability of our model through the application of explainability techniques, shedding light on the inner workings and decisions made by the model. In this chapter, we also analyze if the model trained on raw images directly works for post-processed images without retraining.

6.1 Evaluation Metrics

Evaluation metrics are quantifiable measures used to assess the performance of AI models. These metrics provide a standardized way to gauge how well a model is performing. Evaluation metrics used in our experiments are selected from previous studies that have been performed on similar datasets of X-ray images.

All the metrics below refer to these kinds of predictions:

⋆ True Positive (TP) = instances of the positive class correctly classified;

⋆ False Positive (FP) = instances of the positive class wrongly classified;

⋆ True Negative (TN) = instances of the negative class correctly classified;

⋆ False Negative (FN) = instances of the negative class wrongly classified;

- **Precision**: Precision measures the accuracy of the positive predictions made by a model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.1}$$

  A high precision indicates that when the model predicts a positive outcome, it is often correct, minimizing false alarms.

- **Recall:** Recall, also called sensitivity or true positive rate, assesses the model's ability to identify all relevant instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.2}$$

  A high recall indicates that the model is effective at capturing most of the actual positive cases, minimizing false negatives.

- **Accuracy:** Accuracy is a more general metric that evaluates the overall correctness of predictions, regardless of the class (positive or negative).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.3}$$

  High accuracy indicates that the model's predictions, both positive and negative, are generally correct.

- **F1 Score**: The F1 score is a metric that combines both precision and recall into a single value, providing a balanced evaluation of a model's performance. It is particularly useful in scenarios with an imbalance between positive and negative classes. The F1 score is calculated using the following formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

  The F1 score ranges between 0 and 1, with higher values indicating a better balance between precision and recall.

Additionally, we also provided the **confusion matrix** because it gives a more holistic and class-specific evaluation of a classification model. In a confusion matrix, the rows represent the true or target classes, while the columns represent the predicted or output classes. Each cell in the matrix contains information about the number of instances that were correctly or incorrectly classified.

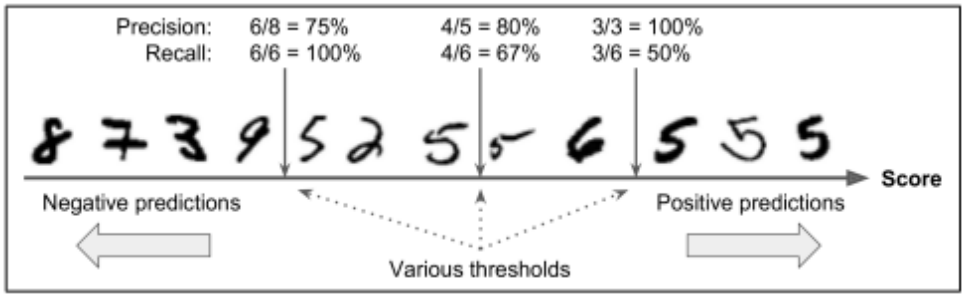


Figure 6.1: precision/recall tradeoff. Source: [Ger19]

A perfectly performed classification would produce zero false positive and false negative predictions (FP=FN=0). It's important to note that precision and recall often have an inverse relationship Figure 6.1. Increasing precision may lead to a decrease in recall, and vice versa. A high F1 score signifies that the model is achieving both high precision and high recall, striking an optimal balance between minimizing false positives and false negatives. This metric is particularly valuable when there is a need to consider both types of errors in classification tasks, offering a comprehensive assessment of the model's overall effectiveness. The choice of which metric to prioritize depends on the specific requirements and constraints of a given task. In medical diagnosis, getting high recall is more critical because we care more about false positives.

6.2 Experimental Setup

The development of deep learning algorithms demands substantial computational power to effectively handle the training of complex deep models and large datasets. Utilizing a standard CPU found in common laptops is insufficient in such scenarios, as it would necessitate an impractical amount of time to complete the tasks. To overcome this limitation, we leverage the power of GPUs. GPUs are specialized in parallel processing, therefore significantly accelerating the training of deep learning models, making it feasible to work in a reasonable timeframe.

For our experiment, we use Intel(R) Xeon(R) Gold 6134 CPU (3.20GHz), 22 GB RAM and NVIDIA GeForce RTX 2080 graphic card with CUDA Version: 12.2; and Python 3.8.5 distributed with Anaconda version: 4.9.2 (64-bit) on a Linux platform to harness the parallel processing capabilities essential for developing and training our deep learning model.

All the models are implemented using the **PyTorch library** [1], a popular and powerful deep learning framework that facilitates the creation and training of intricate neural network architectures. This combination of high-performance GPUs and the PyTorch library provides the computational resources necessary for efficient and effective DL experimentation.

Evaluation We evaluate classification using stratified **five-fold Cross Validation (CV)** [2] on our dataset. In each of the 5 iterations, 4 of 5 folds are used as training data, and the remaining one as validation. We obtain the model's final performance values by averaging results from the 5 validation sets.

6.3 Image-based Approach

In this section, experiments are conducted directly on raw images that have undergone the pre-processing steps discussed in Section 5.2.1.

6.3.1 CNN-based Classication

For multiclass classification of X-ray images, encompassing seven 'defect' classes and a 'good' class, we employ a CNN-based classifier.

Setup

Data To address issues of poor generalization, when applying CNN directly to whole-sized images, a decision was made to patch the images, as discussed in Section 5.2. So, input data to our model comprises of patches of shape 783×955. To increase the diversity of the dataset, we also augment the dataset, as discussed in section **??**. Each grayscale image is converted to **three channels**, through the *merge* command. The single channel is repeated three times, providing images to shape (783,955,3). To facilitate efficient processing, 16-bit unsigned images are converted into float-32 format and normalized using min-max normalization. Additionally, labels are one-hot encoded using the `sklearn.pre-processing.OneHotEncoder` to address the categorical nature of the classes. The data is split into 80-20 k fold times. Splitting is performed in such a way that all patches of an image are either in training or validation datasets.

[1] Open source and available to download from https://pytorch.org/

[2] Using https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Parameter	Value
Training Samples	5988
Validation Samples	1474
Shape of input	(783,955)
Output classes	7
Learning Rate	0.0001
Number of Epochs	30
Batch Size	16
Optimizer	Adam
Weight Decay	0.001
Loss Function	Cross-Entropy
Label Smoothing	0.1
Dropout	0.2
Scheduler	StepLR

Table 6.1: Parameter and hyperparameter setting used for training the ResNet-18 model

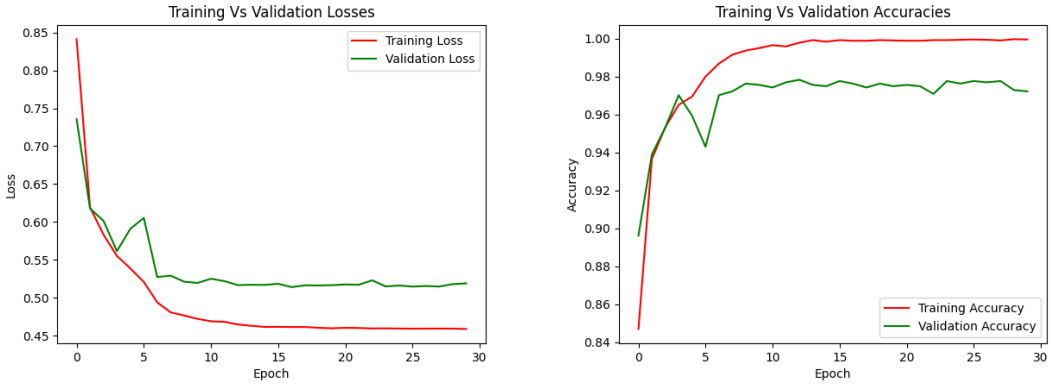
Architecture For our experiment, we utilize a pre-trained ResNet-18 model [3], originally trained on the ImageNet classification dataset (ILSVRC2012), as the backbone of our network. Subsequently, we augment this base with task-specific fully connected layers, initialized with random weights, to facilitate the classification of seven specific classes. The gradient of all learnable parameters is set to true. Parameter and hyperparameter settings for our ResNet-18 model are provided in Table 7.1.

Additionally, we compare the results of the ResNet-18 model with other state-of-the-art CNN architectures- EfficientNet_b0 [Tan+19] [4] and DenseNet-121 [Hua+16] [5]. Both of these architectures are pre-trained on ImageNet and modified for classification into seven classes, similar to ResNet-18. For a detailed explanation of this architecture refer

6.4 sec:architecture

. In our experiments, we use the same hyperparameters for all architectures, except for the batch size.

[3] Available on https://pytorch.org/vision/stable/models.html
[4] Available on https://pytorch.org/vision/main/models/efficientnet.html
[5] Available on https://pytorch.org/hub/pytorch_vision_densenet/



(a) (b)

Figure 6.2: (a) Training and validation loss, and (b) Training and validation accuracy versus number of training epochs for ResNet-18 model. The training curve is in red and the validation curve is in green

Description

We perform experiments k fold times, and results are averaged for k folds. For training our model, the initial learning rate was set to 0.001 and is decreased after a certain number of epochs using *steplr* scheduler with step size 7. To enhance the learning process and mitigate overfitting, momentum, and weight decay [Kri+12] are applied. Through the training process, these weights are continuously adjusted using the gradients of the loss function. This adjustment is carried out using an optimizer over a mini-batch of training samples, with each mini-batch containing 16 samples. The training process stops after a predefined number of epochs which is 30, and the final network is chosen based on the model with the lowest validation loss value. Figure 6.2 the training and validation loss and accuracy curves obtained over 30 epochs for the ResNet-18 model. We also added training, validation loss, and accuracy curve for EfficientNet_b0 model and DenseNet-121 model in Appendix [see Figure A.9 and Figure A.10].

Quantitative Results

For the comprehensive evaluation of ResNet-18, EfficientNet_b0, and DenseNet-121 models, a diverse set of metrics from Section 6.1 is employed. The confusion matrix for the ResNet-18 model averaged over 5 folds and covering 7 classes, is presented in Table 6.2. Corresponding matrices for the EfficientNet and DenseNet models are available in the appendix [refer Table A.2 and Table A.4]. Class-specific statistics for the ResNet-18 model, averaged over 5

folds, are detailed in Table 6.3, while statistics for EfficientNet_b0 and DenseNet-121 are provided in the appendix Table A.3 and Table A.5.

The overall classification performance for all three models is compared in Table 6.11, utilizing weighted values of metrics. The results indicate similar performance among the three models.

		Predicted Class						
		FX00	FX02	FX03	FX04	FX05	FX09	Good
Target Class	FX00	135	0	0	0	0	0	1
	FX02	0	32	0	0	0	0	0
	FX03	0	0	42	0	1	0	0
	FX04	0	0	0	86	0	0	1
	FX05	0	0	0	1	76	0	1
	FX09	0	0	0	0	0	63	5
	Good	1	0	0	0	0	1	1039

Table 6.2: Confusion Matrix from ResNet-18 model, averaged over 5 folds

	Accuracy	**Precision**	**Recall**	**F1 Score**
FX00	94.54%	0.9747	0.9454	0.9600
FX02	98.75%	1.0000	0.9875	0.9937
FX03	98.15%	0.9860	0.9815	0.9837
FX04	99.08%	0.9887	0.9908	0.9897
FX05	96.93%	0.9715	0.9693	0.9704
FX09	92.66%	0.9594	0.9266	0.9429
Good	99.75%	0.9870	0.9975	0.9922

Table 6.3: Statistics of ResNet-18 model for all 7 classes

Architectures	**k fold CV**	**Accuracy%**	**Precision**	**Recall**	**F1**
ResNet-18	5	98%	0.9867	0.9866	0.9865
EfficientNet_b0	5	98.06%	0.9896	0.9890	0.9451
Densenet-121	5	97.75%	0.9821	0.9820	0.9818

Table 6.4: Weighted overall performance comparison of ResNet-18, EfficientNet, and Densenet-121 models, averaged over 5 folds.

Architectures	ResNet-18	DenseNet-121	EfficientNet$_b$0
Batch-size	16	4	8
Model parameters	1177538	6961031	4016515
Training Duration	147 min. 23sec.	310 min. 23 sec.	187 min. 40 sec.

Table 6.5: Comparison of training batch size, number of model parameters, and training time for 30 epochs, averaged over 5 folds of ResNet-18, EfficientNet$_b$0$and DenseNet-121 models$

Computational Speed

A detailed comparison of key training-related metrics for ResNet-18, EfficientNet, and DenseNet-121 models is presented in Table 6.5. The table includes information on the training batch size, the number of model parameters, and the overall training time for each architecture. From the table, it is observed that the training time for DenseNet-121 models is the highest due to dense connections between layers [Hua+16]. However, ResNet-18 requires the least training time due to fewer connections between layers, so was preferred for subsequent tasks. The training duration values provided are averaged over five folds for each model.

Explainability

The interpretability of the ResNet-18 model was improved by incorporating the Grad-CAM technique [6]. Grad-CAM generates detailed heatmaps, highlighting the regions in an input image that significantly contribute to the model's predictions [Sel+17]. Figure 6.3 illustrates examples from each of the six defect categories, accompanied with their respective Grad-CAM heatmaps, providing insights into the areas where the model focused while making decisions.

Inference phase

As our model was designed for patches, predicting entire images requires a specific approach. During inference, unseen raw X-ray images of size 2350×2866 are split into 9 patches, each measuring 783×955. These patches are then fed into the model, producing individual predictions for each patch [see Figure 6.4]. The final prediction is determined by combining predictions from all patches using a set of conditions. The pseudo-code for the 'classify_patch' function is outlined in Algorithm 1.

[6] Code available at: https://github.com/MECLabTUDA/M3d-Cam

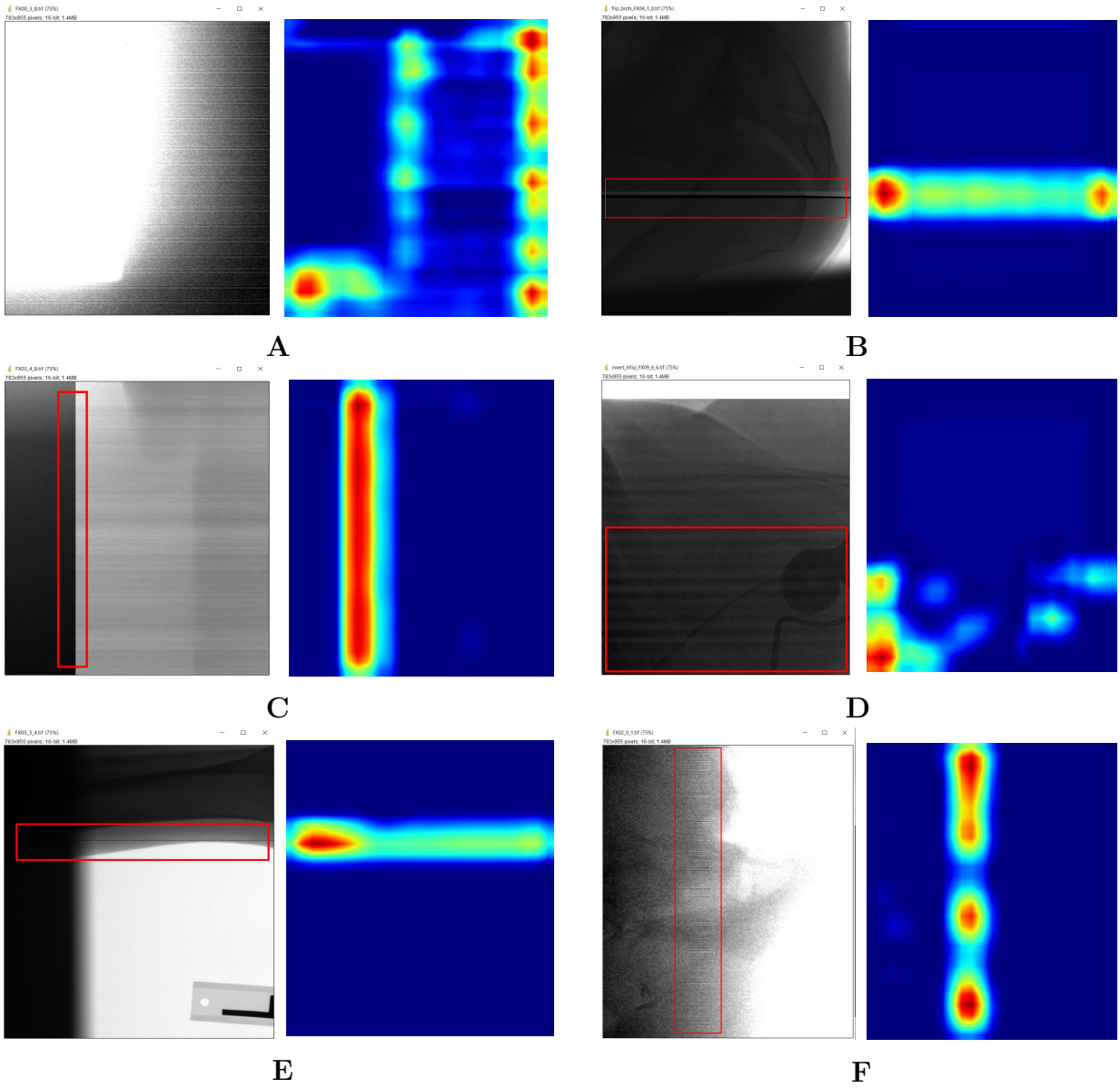


Figure 6.3: shows an example of a heatmap generated using Grad-CAM, indicating the regions on which the model focuses when making predictions. The orange-red area indicates the region where the attention of the model is high. The left image shows the image of [A]FX09 [B]FX04 [C]FX03 [D]FX00 [E]FX05 [F]FX02 and its corresponding to the heat map shown on the right

The function first identifies unique values within the list of predictions. If only one unique value is present, it is returned as the final prediction. However, if there are multiple unique values, a set of rules is applied. For instance, if the count of 'FX02' in the predictions is 3, the function returns 'FX02' as the final prediction. Similarly, if the count of 'FX03', 'FX04', or 'FX05' is 3, the respective value is returned. This is because these defects span the entire image in a straight manner, and if three patches indicate a defect, it is considered a defect. If 'Good' is present, it is excluded, and the function returns the most frequently occurring value among the remaining predictions. If none of these conditions are met, the function returns the value with the highest frequency among the predictions.

To assess the model's ability to make predictions for entire images, we utilized the same validation dataset originally designed for patched data. Instead of providing individual patches as input, the entire images were used. The final confusion matrix for the entire images using ResNet-18 is presented in Table 6.6, and class-wise weighted precision, accuracy, recall, and F1 scores are summarized in Table 6.7.

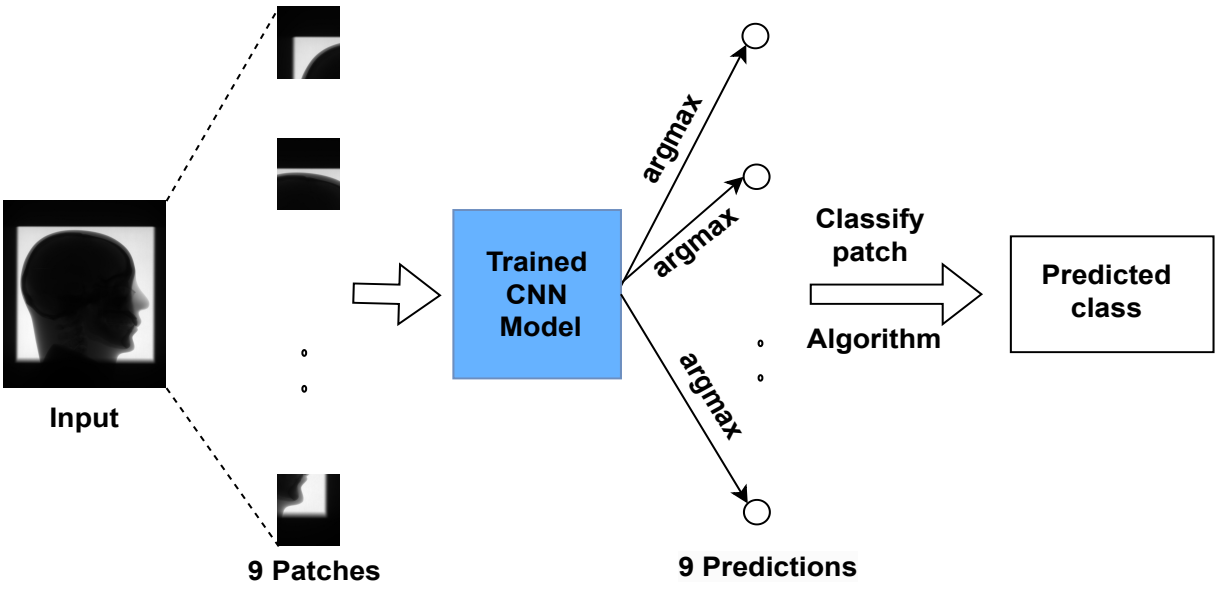


Figure 6.4: Design architecture for prediction of entire image using trained CNN model on patched images

Algorithm 1 classify_patch(predictions)

unique_values ← set of unique values in *predictions*
if length of *unique_values* == 1 **then**
    return the only unique value
else if $FX01$, $FX02$, $FX03$, or $FX04$ appear exactly 3 times in *predictions* **then**
    return the value among $FX01$, $FX02$, $FX03$, and $FX04$ that appears 3 times
else if *Good* is in *predictions* **then**
    Remove all occurrences of *Good* from *predictions*
    return the most frequent value in the modified *predictions*
else
    return the most frequent value in *predictions*

		Predicted Class						
		FX00	FX02	FX03	FX04	FX05	FX09	Good
Target Class	FX00	115	0	0	0	0	2	3
	FX02	1	7	0	0	0	0	0
	FX03	0	0	16	0	0	0	0
	FX04	0	0	0	23	0	0	1
	FX05	0	0	0	0	103	0	1
	FX09	0	0	0	0	0	20	4
	Good	1	0	0	0	0	1	107

Table 6.6: Confusion Matrix for full-sized validation images based on `classify_patch` using ResNet-18 model trained on patched images

	Accuracy	**Precision**	**Recall**	**F1 Score**
FX00	99.27%	0.9829	0.9583	0.9705
FX02	99.75%	1.0000	0.875	0.9333
FX03	100%	1.0000	1.0000	1.000
FX04	97.87%	1.0000	0.9583	0.9975
FX05	99.52%	1.0000	0.9904	0.9975
FX09	98.27%	0.8696	0.8333	0.8511
Good	97.28%	0.9224	0.9817	0.9511

Table 6.7: Class-wise statistics for entire image classification using the ResNet-18 model. The table provides precision, recall, accuracy, and F1 score for each artifact class.

Summary

In summary, experiments demonstrated the effectiveness of the CNN-based classifier in classifying X-ray images for artifacts. The performance of ResNet-18, EfficientNet, and DenseNet-121 was compared. The models exhibited comparable performance, with ResNet-18 showing superior computational speed, so was used as a model for subsequent tasks. The explainability of the model was enhanced through Grad-CAM, providing insights into the decision-making process. The testing phase demonstrated the model's adaptability to predict entire images based on patch-wise predictions.

6.4.1 One-Class Classification

In this section, we explore an unsupervised learning approach to address the challenge of imbalanced data in our dataset. Given more number of 'good' images than 'defective' ones, we delve into the concept of 'One-Class Classification' using generative modeling. Unlike well-known binary and multi-class classification problems, where classes are assumed to be known and fixed during both training and inference, OCC is trained with an absence of explicit class labels [Lia+22b]. The objective of a one-class classifier is to learn the patterns and distribution of normal data during training. Subsequently, during inference, samples that significantly deviate from this learned normal distribution are identified as anomalies. In our case, normal data are 'good' images and those with anomalies are 'defective' images. We conduct a comparative analysis of two generative models - AE and VAE. In brief, AE models operate by encoding input data into a lower-dimensional latent space using an encoder and then reconstructing the input from this encoded representation using a decoder. The key distinction with VAE models is their utilization of a probabilistic approach, where the encoder produces not only a specific encoding but also a distribution in the latent space.

Setup

Data We use a patches dataset of raw X-ray images containing 5205 good images and 1129 defective images with each image of size 783×955. Images are grayscale. Only good images are used in training, 90% images of which used in train and 10% used in test Table 6.8. All defective images form part of the test dataset. We don't do k-fold validation here. Images are normalized using the min-max method. For introducing variation in our dataset, we use online augmentation. We augment training data with random horizontal, and vertical flips and color glitter by changing brightness and contrast with a probability of

Data	Count
Train	
Good	4684
Test	
Good	521
Defective	1129

Table 6.8: Table shows count of images used in training and testing of OCC models

0.5. We use augmentation from torchvision library [7]. Augmentation of the test dataset is not performed.

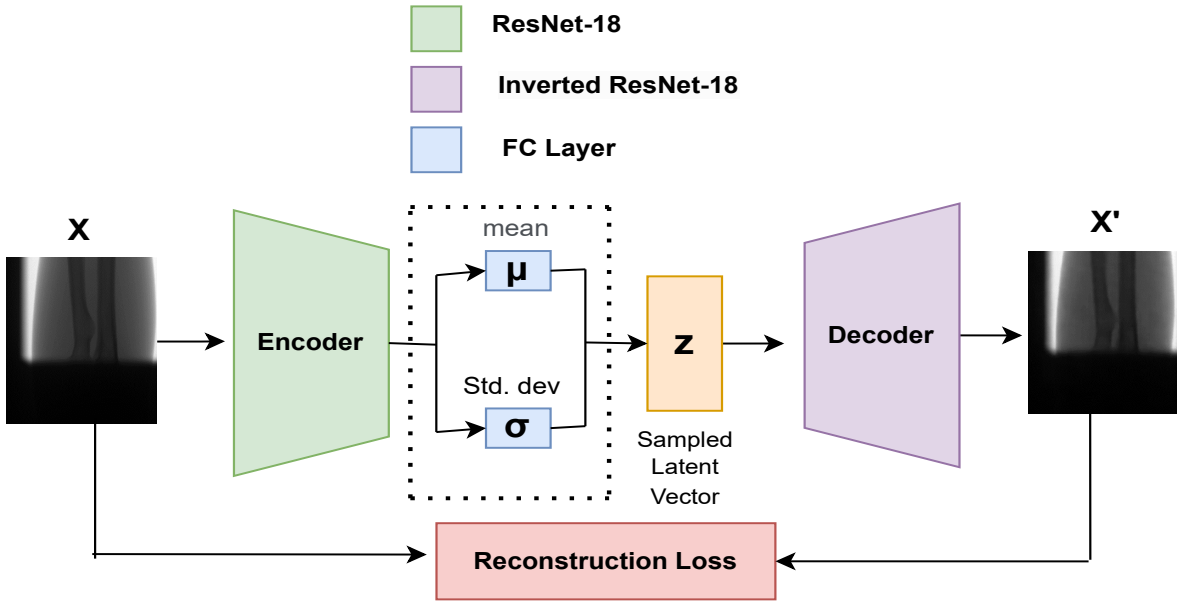


Figure 6.5: OC-VAE architecture using ResNet-18 backbone where X is input image and X' is reconstructed image

Architecture and training description We use both AE and VAE models and in our thesis, we call OCC with AE structure as OC-AE and with VAE structure as OC-VAE.

We employ a ResNet-18 encoder-decoder architecture with customized modifications to accommodate our image dimensions of 783×955, ensuring that the output from the

[7] Available at: https://pytorch.org/vision/stable/transforms.html

decoder is also of the same dimension. The architecture of OC-VAE is given in Figure 6.5. The latent dimension was chosen to be 256. The encoded dimension from an encoder is of shape 25×30.

Parameter	**Value**
Input size	(783,955)
Initial Learning Rate	1e-4
Batch Size	8
Optimizer	Adam
Latent Dimension	256
OC-AE	
Number of Epochs	1860
Training time	88 hours
Loss Function	MSE[8] loss with reduction 'sum'
OC-VAE	
Number of Epochs	1640
Training time	78 hours
λ	0.005
Loss Function	MSE loss with reduction 'sum' + λ * Kullback–Leibler (KL) divergence

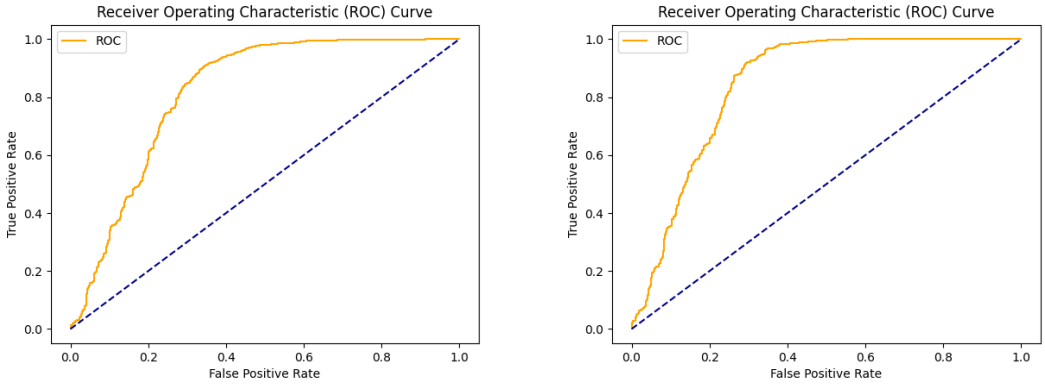
Table 6.9: Parameters and hyperparameters settings used for training of OC-AE and OC-VAE

Description

For training of our OC-AE and OC-VAE models, we use parameters and hyper-parameters as given in Table 6.9. We use Adam optimizer with an initial learning rate of 1e-4. The learning rate was reduced to 1e-6 after 1000 epochs. A batch size of 8 was used. We trained our model till loss stops to decrease i.e. OC-AE model after 1860 epochs and of OC-VAE after 1640 epochs. We use mean squared error with reduction as 'sum' for training OC-AE and additional KL term scaled with λ value of 0.005 for training in OC-VAE.

Experimental result

Reconstructed images for good and defective images are shown in Figure 6.7. To distinguish between good and defective images, it is required to evaluate them using the same metric. We calculate the reconstruction score for each image using root mean squared error (RMSE) between input and output images.



(a) (b)

Figure 6.6: Figure shows AUC-ROC curve for (a) OC-AE, and (b) OC-VAE

$$\text{RMSE} = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(X_i' - X_i)^2} \tag{6.5}$$

where X is the input image to the encoder and X' is its reconstructed image from the decoder.

We calculate reconstruction scores by comparing the input and the output from the decoder. The static threshold value is determined using the AUC-ROC score to distinguish between good and defective images. If the reconstruction score is below the threshold, the images are labeled as 'good'; otherwise, they are considered defective. The AUC-ROC curve of OC-AE and OC-VAE is given in Figure 6.6. The threshold value for OC-AE was found to be 22.4 and for OC-VAE to be 41.72. In Figure 6.7 it can be seen that 'good' images are reconstructed well, while the reconstruction of defective images doesn't depict the defects, resulting in higher reconstruction scores. Figure 6.6 shows and compares the AUC-ROC curve for both OC-AE and OC-VAE models.

As an example, in Figure 6.8 we also present the histogram of 200 good and 200 defect images to show the differences between two distributions of 'good' and 'defective' images where the X-axis is the calculated reconstruction score from OC-VAE. A similar histogram for OC-AE is given in the appendix [See Figure A.11].

Performance comparison of OC-VAE and OC-AE is given in Table 6.10. We see OC-VAE perform better with F1 score of 0.84 than OC-AE which has F1 score of 0.82.

We present and compare model performances of OC-AE and OC-VAE in Table 6.10 using class-wise and overall precision, recall, F1 score, and accuracy.

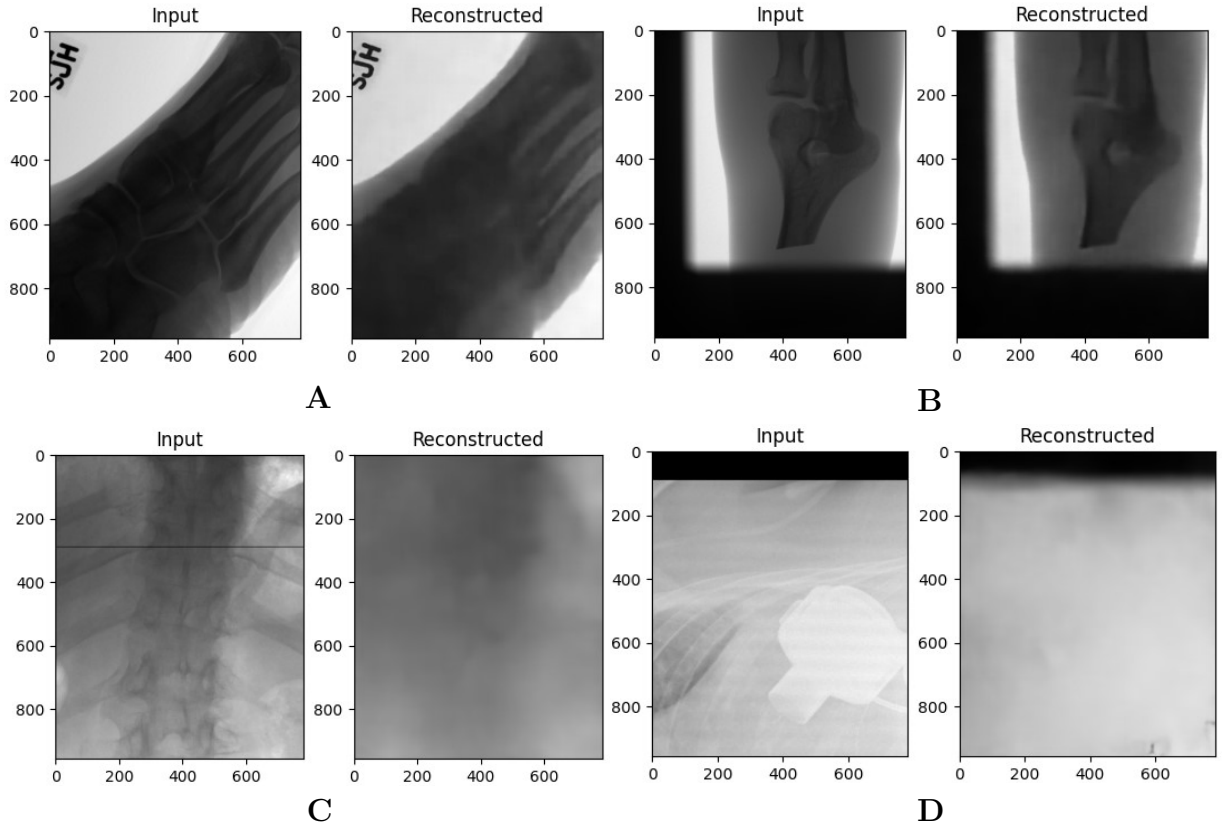


Figure 6.7: shows input (left) and reconstructed (right) images from the trained OC-VAE model: [A] and [B] represent good images, while [C] and [D] depict artifacts of FX05 and FX09. The reconstruction quality is evident, with good images accurately reconstructed, while images with artifacts exhibit poor reconstruction

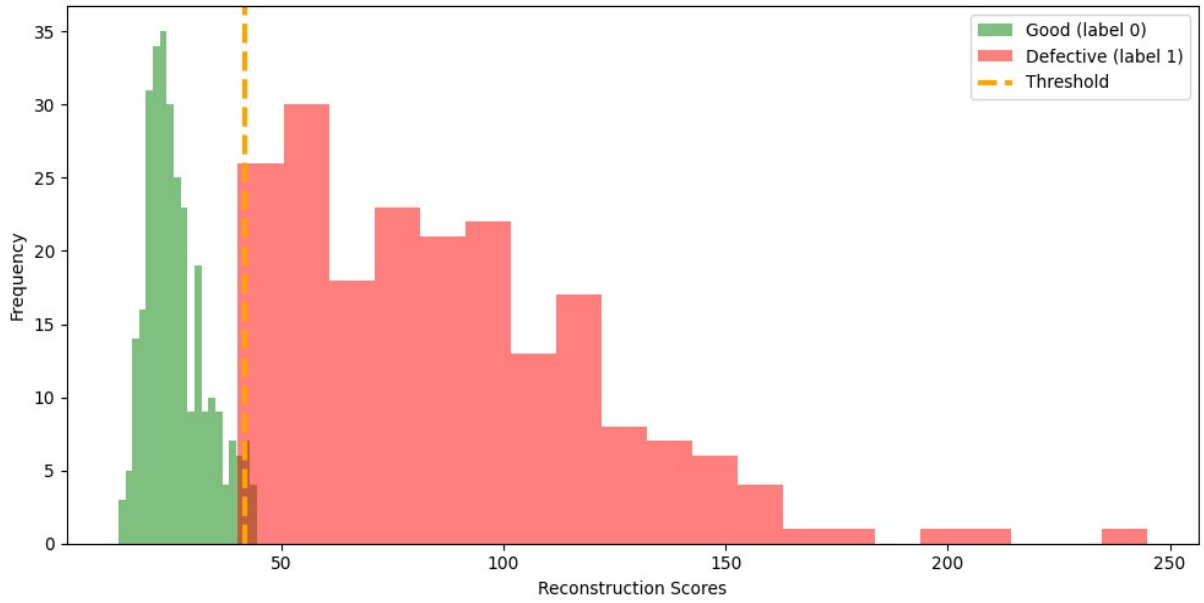


Figure 6.8: Histogram of reconstruction score of good (green) and defective (red) and statistical threshold (orange) with 200 good and 200 defective raw X-ray images from OC-VAE model

Inference phase

As the OCC model was trained on patches of X-ray images, the classification of the entire test image required a specific approach. We segmented the test image into patches, generated predictions for each patch, and determined the final predicted class based on the highest occurrence among the predicted classes. Table 6.11 show performance on full-sized image using OC-VAE model.

Summary To summarize, we train generative models for one class classification. Here, we learn in an unsupervised setting with only 'good' images and during testing use reconstruction loss to discriminate between 'good' and 'defective' class. The images with loss value higher than the threshold are 'defective', else 'good'. We compare the results of AE and VAE. VAE seems to work better than AE.

6.5 Time Series-based Approach

This section delves into the development of a univariate time series classification model using a pre-processed line profile dataset obtained from raw X-ray images, as detailed in

Model	Data	Accuracy	Precision	Recall	F1
OC-AE	Good	76.8%	0.7680	0.6545	0.7067
	Defective	85.1%	0.8507	0.9087	0.8788
	Overall	81.7%	0.8246	0.8284	0.8244
	Threshold				22.40
OC-VAE	Good	78.9%	0.7880	0.7063	0.7449
	Defective	87.1%	0.8706	0.9123	0.8910
	Overall	**84.4%**	**0.8445**	**0.8472**	**0.8448**
	Threshold				41.72

Table 6.10: Performance metrics for OC-AE and OC-VAE model for one class classification. The threshold value is obtained based on the reconstruction score for each real and fake image of the testing data using AUC-ROC curve

Architecture	Accuracy%	Precision	Recall	F1
OC-VAE	83.82%	0.8104	0.7386	0.7629

Table 6.11: Weighted performance of OC-VAE model on entire image

Section 5.2.1. The line profiles extracted from the images display a temporal nature, to leverage the temporal characteristics of these profiles, we conduct experiments using RNN and 1DCNN models for time-series classification.

Setup

Data The dataset utilized in this section is derived from vertical (V) and horizontal (H) line profiles, computed by summing the intensity values of raw X-ray images along rows and columns, respectively. These profiles are combined, taking into account the bidirectional nature of artifacts (i.e., they can manifest in both vertical and horizontal directions), as detailed in Section 5.2.1. The input data has a shape of $(1, H + V)$. In our case, with an image size of 2350x2866, the input becomes (1, 5216), maintaining the channel dimension as 1. This representation allows us to interpret the input data as a univariate time series, where the number of features corresponds to the number of time steps.

To ensure uniformity and comparability across datasets, we applied normalization using the standard deviation and mean of the data. Furthermore, the labels were one-hot encoded, facilitating their utilization in classification tasks.

To rigorously evaluate the robustness of our models, we employed a 5-fold CV approach. In each fold, the dataset was partitioned into an 80-20 split for training and validation,

respectively. This methodological choice ensures that our models are thoroughly tested on diverse data subsets, enhancing our finding's generalisability.

Architecture

RNN The RNN architecture employed in this study comprises a single layer of LSTM cells. The LSTM layer is configured with a hidden size of 1024, offering a substantial capacity for capturing temporal dependencies within the input time series data. A dense layer follows the LSTM, with the output layer containing neurons equal to the number of classes in our classification task i.e. seven in our case.

To mitigate the risk of overfitting and enhance the generalisation capability of the model, a dropout rate of 0.2 is applied specifically to the LSTM cells.

Parallel CNN LSTM The architecture consists of 2 parallel branches of convolution block and LSTM block [see Figure 6.9]. The actual parameters used in the convolution block are: conv1d(1,64, 3, 1)[9], BN1d(64), conv1d(64,128,3,1), BN1d(128), MP(2,2)[10] and in LSTM block consists of 1 layer of LSTM cell with hidden size 128 and linear layer. BN is the batch-norm layer and MP is the max pooling layer. The output of convolution branches is flattened are merged with the output of the LSTM block. This concatenated output is given as input to the classifier network - consisting of FC layers with 256 units. The final linear layer produces the model's output with seven units corresponding to the number of classes.

1DCNN We developed 1DCNN model consisting of 1D convolution and pooling operations [see Figure 6.10]. The model architecture comprises two parallel branches, each contributing distinctive aspects to the overall feature extraction process.

Each branch consists of a network of 2 consecutive layers of 1D convolution, each followed by a layer of batch norm and a max-pooling layer. The actual parameters used in the first branch are: conv1d(1,32, 8, 1) [11], BN1d(32), conv1d(32,32, 8, 2), BN1d(32), MP(4,2)[12]. The actual parameters used in the second branch are: conv1d(1, 32, 8, 1), BN1d(32), conv1d(32,32, 8, 2), BN1d(32), MP(4,2). The outputs of these two branches are flattened and concatenated with input. This concatenated output of feature extraction is

[9] input channel 1, output channel 64, kernel size 3, stride 1
[10] kernel size 2, stride 2
[11] input channel 1, output channel 32, kernel size 8, stride 1
[12] kernel size 4, stride 2

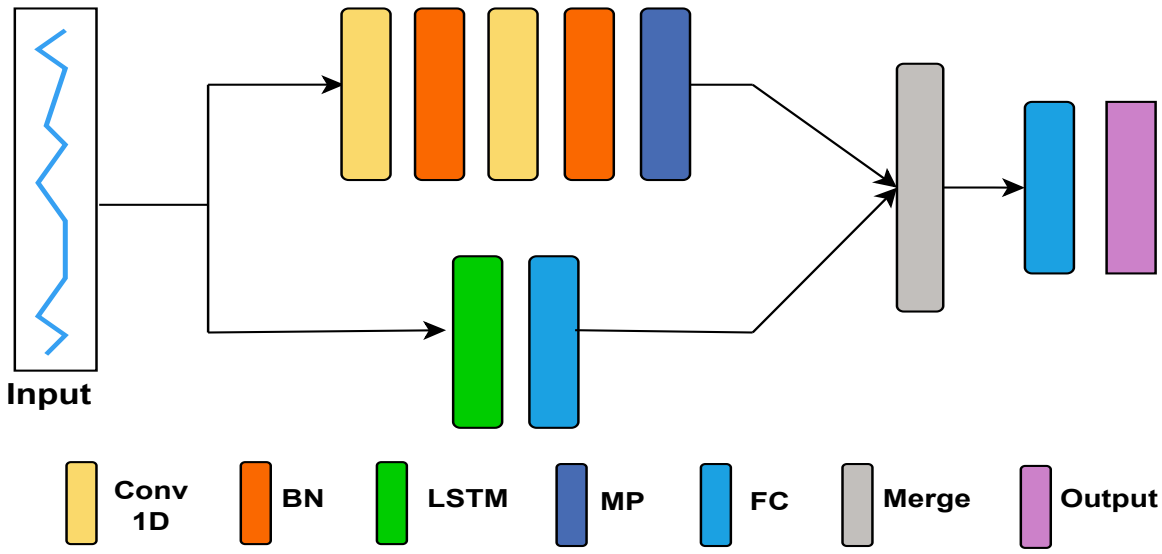


Figure 6.9: Parallel CNN LSTM architecture

then used as input to a classifier network - consisting of fully connected layers with 1024 units and a dropout rate of 0.2. The final linear layer produces the model's output with seven units corresponding to the number of classes. Activation function ReLU is used in feature extractor branches and PReLU is used in fully connected layers.

Description

First, we performed our multiclass classification on the line profile dataset with the RNN model. The model is trained using Adam optimizer with a batch size of 32, weight decay of 1e-3, the initial learning rate of 1e-2, scheduled using steplr learning rate scheduler with a step size of 7. Table 6.12 details a number of training and validation samples used and hyperparameters used in training the RNN model. We use the same hyper-parameters and parameters for 1DCNN model.

Results We conducted a comparative study of the results of 1DCNN,parallel CNN LSTM and RNN-based models in Table 6.14 on the validation dataset of line profiles. The table compares the overall weighted precision, weighted recall, weighted f1, and accuracy of all the models. It can be seen from the table that the 1DCNN model performed better than other architectures. In Table 6.15 gives Confusion Matrix of validation dataset using 1DCNN model. whereas in Table 6.13 detailed statistics of each class is given for 1DCNN.

Hyperparameter	Value
Training Samples	1617
Validation Samples	405
Shape of input	(1,5216)
Number of output class	7
Learning Rate	0.001
Number of Epochs	30
Batch Size	32
Optimizer	Adam
Weight Decay	0.0001
Loss Function	Cross-Entropy

Table 6.12: Hyperparameters used in training of RNN model

	Accuracy (%)	Precision	Recall	F1 Score
FX00	97.6%	0.969	0.988	0.978
FX02	96.0%	0.939	0.920	0.929
FX03	86.9%	0.921	0.894	0.907
FX04	89.9%	0.944	0.877	0.910
FX05	96.5%	0.970	0.960	0.965
FX09	95.9%	0.967	0.991	0.979
Good	98.0%	0.981	0.981	0.981

Table 6.13: Statistics of the 1DCNN model for all 7 classes

Model	Accuracy	Precision (Weighted)	Recall (Weighted)	F1 Score (Weighted)
1DCNN	**96.6**%	**0.963**	**0.961**	**0.961**
Parallel CNN LSTM	71%	0.693	0.707	0.698
RNN	57%	0.576	0.568	0.562

Table 6.14: Weighted Precision, Recall, F1 Score, and Accuracy for 1DCNN, Parallel CNN LSTM and RNN on the validation dataset

Explainability In order to comprehend the decision-making process of our 1DCNN-trained model, we explore the field of explainability. For this, initially, we experimented with Grad-CAM. However, the results were not satisfactory, failing to provide clear insights

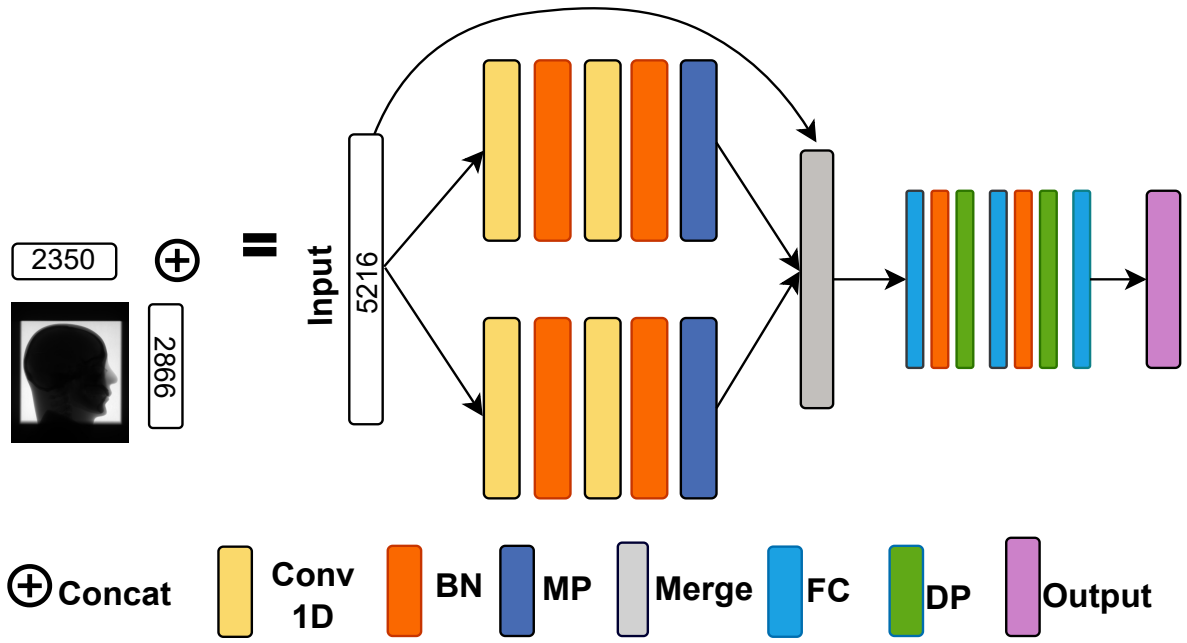


Figure 6.10: Our proposed 1DCNN architecture for line profile classification

into the model's decision process. The limitations of Grad-CAM led us to explore alternative explainability methods. Notably, the failure examples of Grad-CAM are documented in the appendix for reference [see Figure A.13].

To address the shortcomings encountered with Grad-CAM, we turned to saliency maps generated through gradient-based techniques. A detailed explanation of the working of saliency maps is given in Section 3.5. Through saliency maps, we aim to highlight areas in line profiles that significantly influence the model's predictions. Figure 6.11 and Figure 6.12 gives illustrative examples FX04_group_of_defect_line and FX03_partly_brighter defects. Examples of explainability using saliency maps for other defect classes are provided in Appendix [see Figure A.14 to Figure A.16].

Summary

In this section, we perform experiments with line profiles for artifact detection. We convert images into time series. We compare results with RNN, parallel LSTM CNN model, and 1DCNN. Through 1DCNN, we achieve the highest accuracy of 96.6%. Additionally, we also give an explainability of the 1DCNN model, using saliency-based techniques. It is

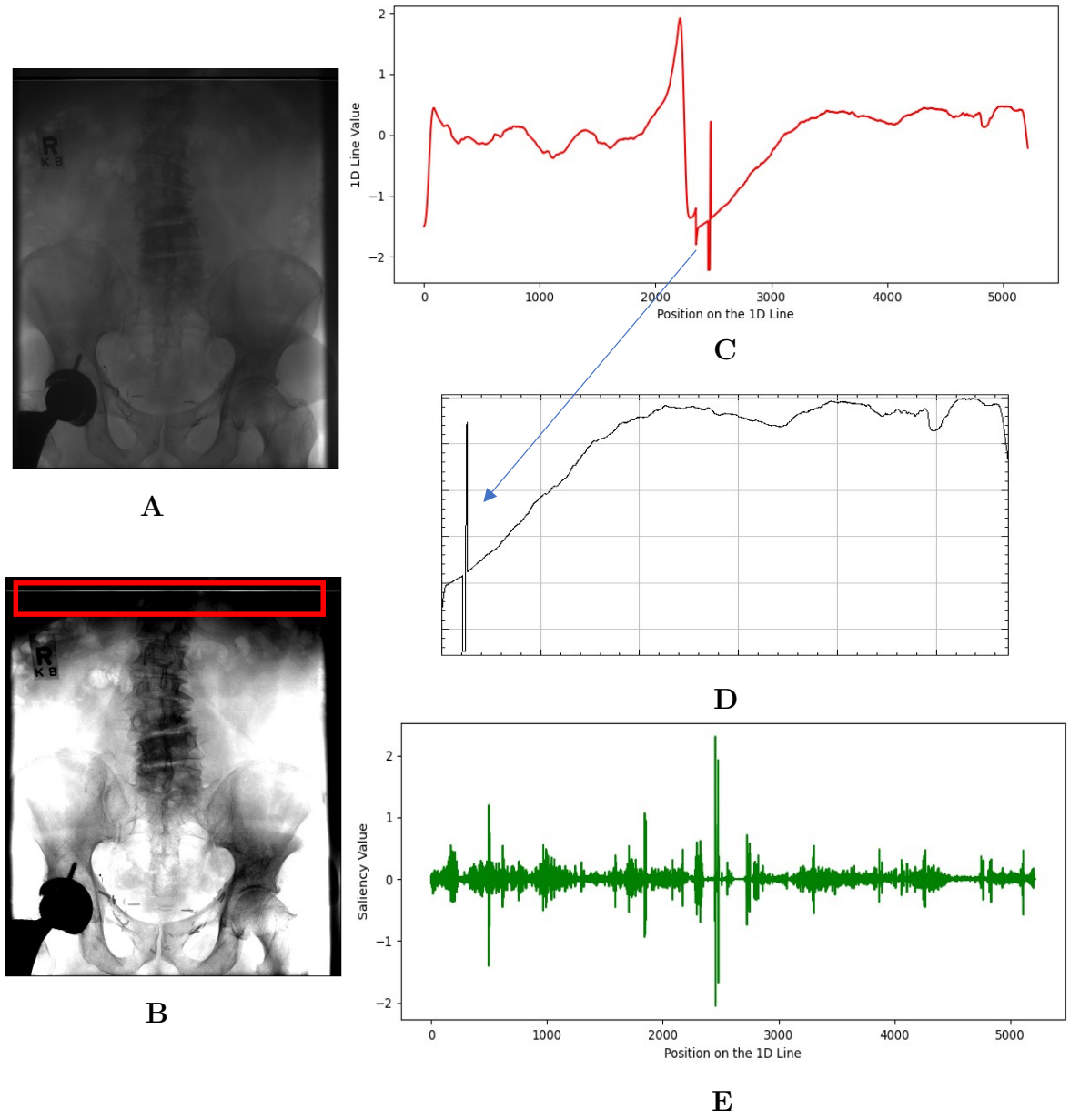


Figure 6.11: Explainability of group of defect line artifact: [A] Original 2350×2866 raw X-ray image featuring the FX04_group_of_defect_line artifact, with enhanced defect visibility in [B] through ImageJ adjustments; [D] Represents the input to the 1DCNN model, a concatenation of row and column profiles; [C] Zoomed-in segment of the line profile, highlighting the visible defect; [E] Saliency maps from the explainability analysis of the 1DCNN model, indicating elevated attention values in regions corresponding to the defect presence

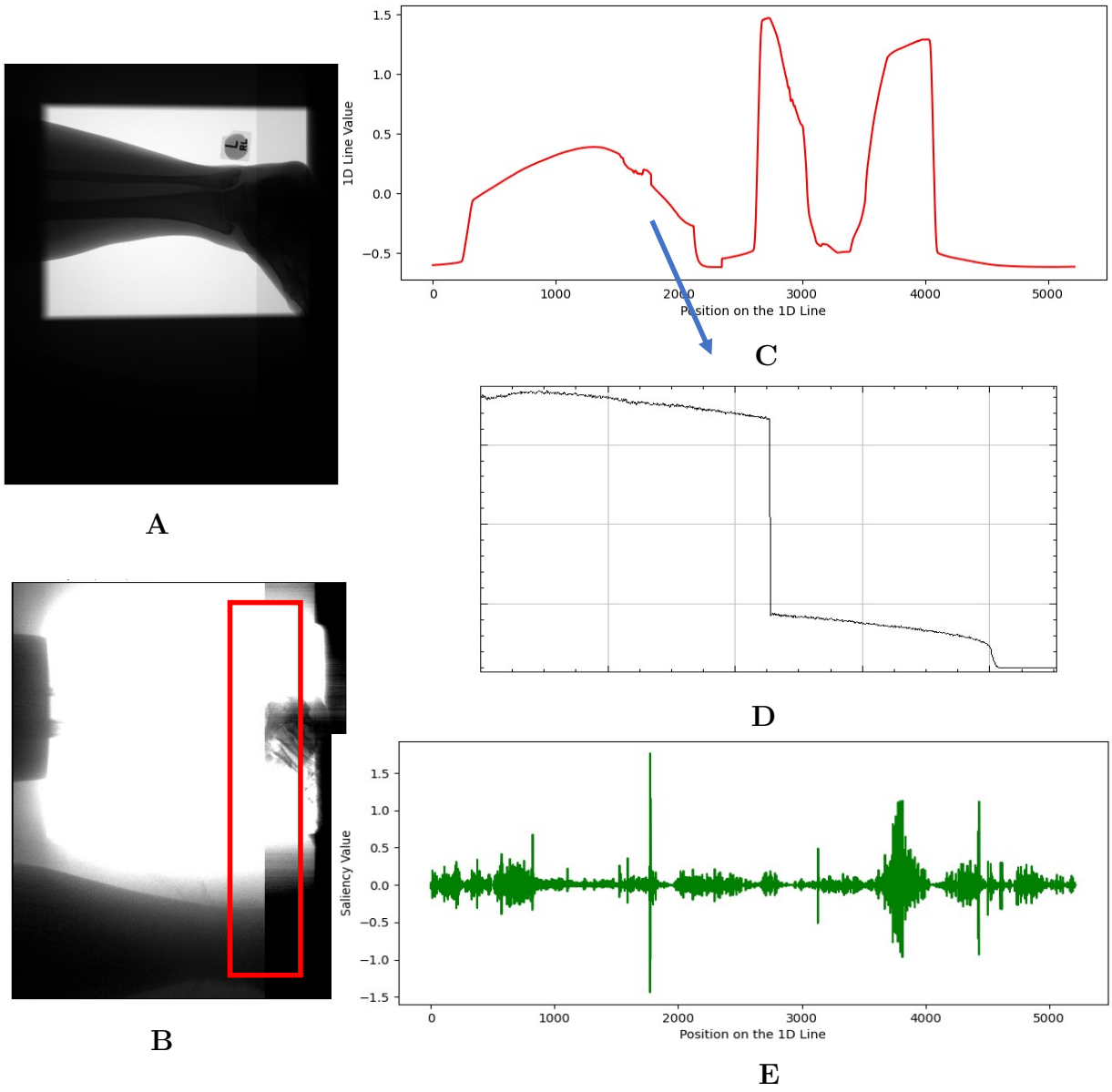


Figure 6.12: Explainability of partly brighter artifact: [A] Original raw X-ray image; [B] Enhanced visualization using ImageJ to highlight areas of artifact; [D] Input to the 1DCNN model, consisting of concatenated row and column profiles; [C] Zoomed-in section of the line profile revealing details where FX03_partly_brighter defect exists; [E] corresponding saliency map indicating elevated attention values corresponding to the partly brighter features.

		Predicted Class						
		FX00	FX02	FX03	FX04	FX05	FX09	Good
Target Class	FX00	123	0	0	0	0	0	2
	FX02	0	10	0	0	0	0	0
	FX03	1	0	11	0	1	0	0
	FX04	0	0	0	20	2	1	0
	FX05	1	0	1	1	102	0	1
	FX09	0	0	0	1	0	21	0
	Good	2	0	0	0	1	0	102

Table 6.15: Confusion Matrix of validation dataset using 1DCNN model

evident that the model is focused on relevant input neurons where the defect is, to obtain its prediction.

6.6 Post-processed Analysis

In this section, we discuss the adaptability of a model originally trained on intensity-linear raw X-ray images to the distinct domain of post-processed images. The primary question is whether the model can effectively make predictions on post-processed images without undergoing additional training or modifications.

To achieve this, we use the softmax layer to obtain pseudo-probability scores for each class, allowing us to identify the class with the highest probability as the model's prediction. When the model's prediction aligns with the ground truth (i.e., it correctly identifies the class of the image) and the model expresses a high level of confidence in its prediction (i.e., a significant difference in probabilities among classes), this indicates that the model is performing effectively in this context.

Conversely, when the model exhibits low confidence by assigning similar probabilities to multiple classes, it raises concerns about the model's reliability and suitability for this specific task. In these instances, the model's output lacks decisiveness and may not be a reliable basis for practical decision-making. Therefore, it becomes apparent that the model developed originally for linear intensity raw X-ray images may not seamlessly transition to the analysis of post-processed images without the need for enhancements or adjustments. These enhancements could encompass techniques such as fine-tuning the model to adapt it specifically to the characteristics and nuances of post-processed images. This suggests

that further refinement and adaptation are necessary to ensure the model's competence and reliability when applied to post-processed X-ray images.

Another approach would also be manually checking the results, but above mentioned approaches have the benefit of being faster/more automized. Analysis was only performed on defect classes and not on good classes since the post-processed dataset was not labeled. And, it would need an expert's opinion on know whether an image is artifact-free.

6.6.1 Image-based Analysis

In this section, we investigate the effectiveness of a ResNet-18 model originally trained on intensity-linear raw images when applied directly to post-processed images.

Setup

For this investigation, we employ a pre-trained ResNet-18 model, initially trained on raw images. Considering the potential variation in size and cropping of post-processed images, we generate patches of size 783×955, as detailed in Section 5.2.2. We standardize these images using min-max standardization for consistency.

Experiments and Results

Predictions are generated for each patch, and the average prediction is computed for the entire image. The softmax function is applied to obtain probabilities from the model's output. These probabilities represent the model's confidence in its predictions, and the argmax is taken to determine the final predicted output.

As illustrated in Figure 6.13, the model consistently assigns higher probabilities to the FX00 sporadic line artifact class for each patch, as all patches contain sporadic line patterns. Additional examples of successful predictions are provided in the appendix [see Figure A.17 to Figure A.20]. Consequently, the direct application of the pre-trained ResNet-18 model to post-processed images, without further retraining, demonstrates promising results, making it a viable approach for defect classification in such images.

6.6.2 Time Series-based Analysis

In this subsequent section, we conduct a comprehensive analysis of the time series model's efficacy when applied to post-processed images.

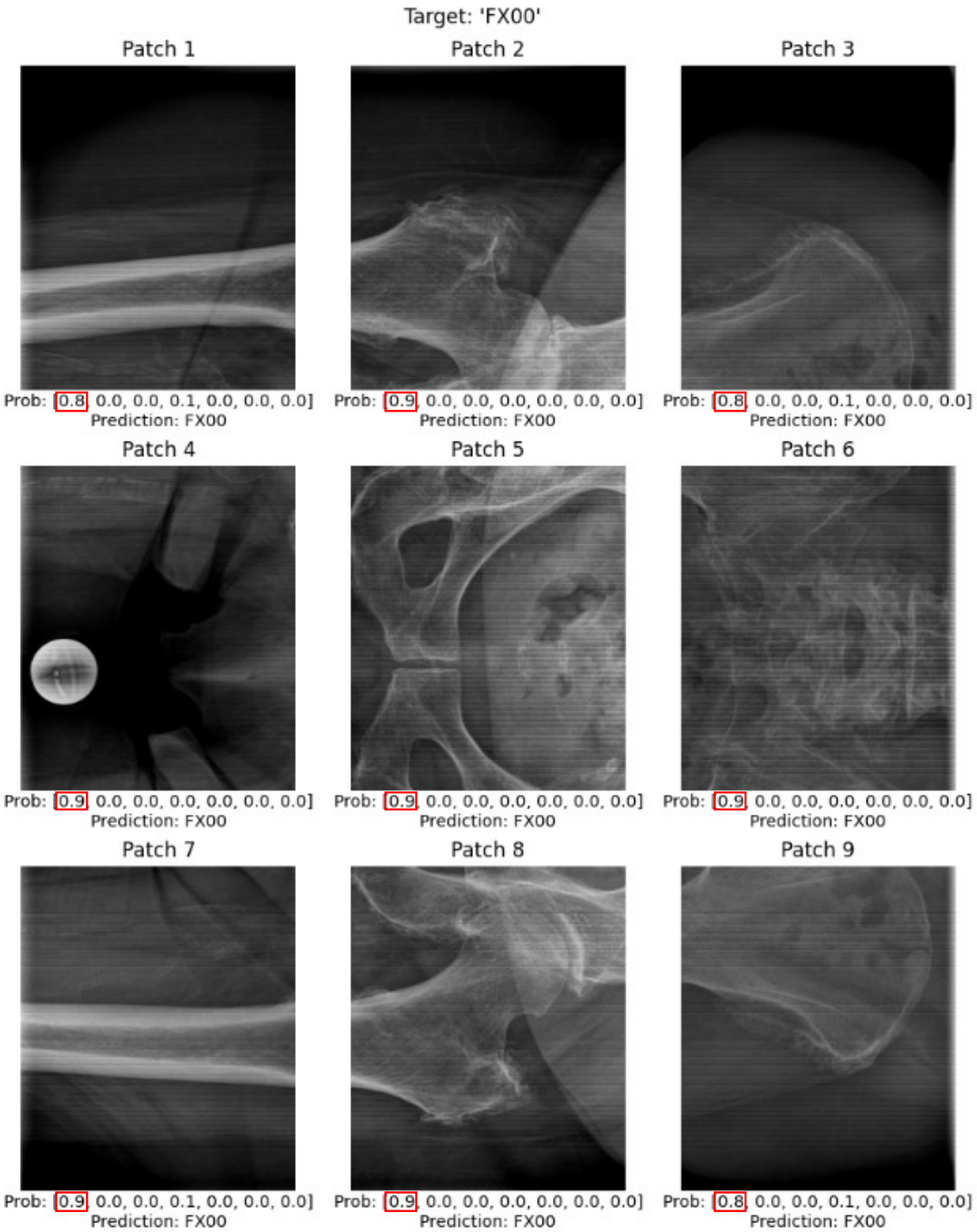


Figure 6.13: Analysis of post-processed image containing FX00 sporadic artifact. It can be seen from the figure that each patch predicts the artifact class correctly with high confidence.

Setup

Line Profile

Line Profile A 1DCNN trained on the line profile of raw images for 30 epochs serves as the basis for this analysis, as discussed in Section 6.5. To address data size mismatches between raw and post-processed images, we employ a resizing strategy, repeating the last values after concatenating row and column profiles (Section 6.6.2). Additionally, post-processed images are standardized before inputting into the model.

Experiment and Results

We conducted zero-shot predictions on post-processed images, revealing noteworthy discrepancies in the model's performance. One distinctive case is illustrated in Figure A.21, showcasing a concatenated line profile containing sporadic line artifacts. The model exhibited a high probability prediction, assigning the image to the class FX00_sporadic_line_artifact.

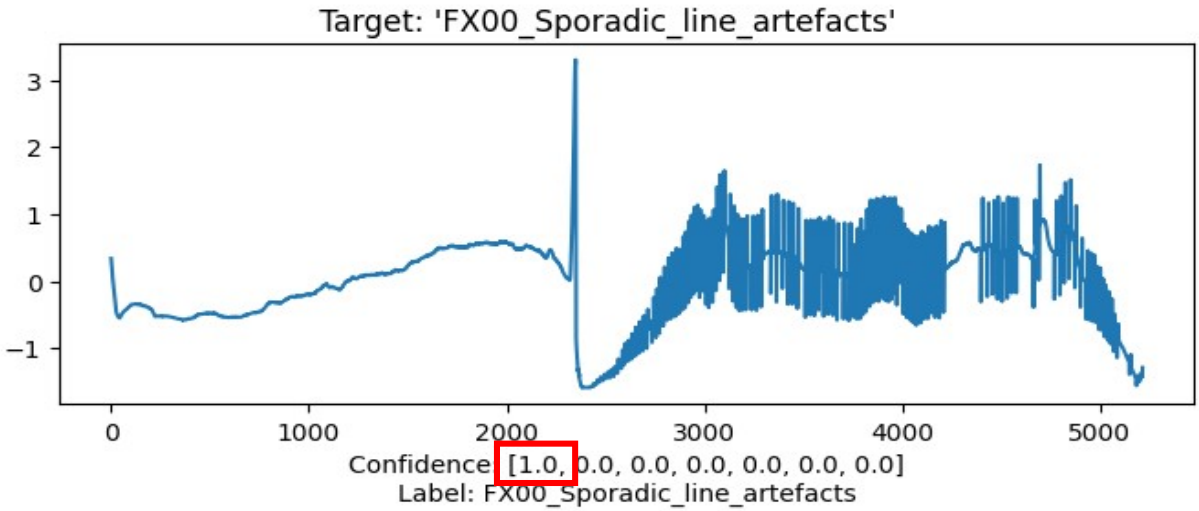


Figure 6.14: Figure shows line profile of a post-processed image containing defect of FX00_sporadic_line_artifact, with very high confidence, the model predicts that it belongs to sporadic artifact

Contrastingly, Figure 6.15 presents an example of an image with grouped line defects. The model assigned a 59% probability to the FX09_Stripes class and a relatively lower probability of 21% to FX04_group_of_defect_line. This case represents a failure scenario,

indicating the model's struggle to distinguish between classes in the presence of grouped defects. Additional examples are provided in the appendix [refer Figure A.21 to Figure A.22].

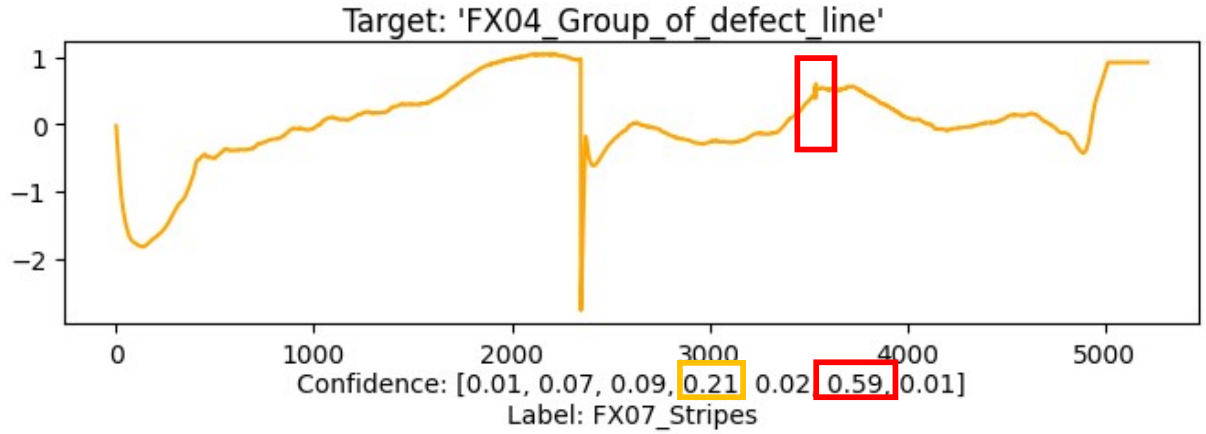


Figure 6.15: Figure shows failure case of a post-processed image containing FX04_group_of_defect_line artifact, the model gives higher probability to stripes artifact

Summary

In summary, the direct application of the image-based model shows promising results while the time-series model yields inconsistent results. This analysis emphasizes the need for further investigation and potential model adjustments to the time-series model to enhance its adaptability for post-processed images.

Chapter 7

Comparative Result Study

In this section, we conduct a comprehensive comparative study of the various approaches outlined in the methodology Chapter 6.

7.1 Image-based vs One-Class Classification vs Time Series-based Approach

Table 7.1 provides a general comparison between image-based, one-class, and time series approaches.

Note: The table provides a comparative analysis of key aspects, including architecture, shape of input, pre-processing steps, and training parameters for all approaches. Further details on specific parameters and hyper-parameters are available in the respective sections.

The asterisk (*) denotes that, at present, we have pre-processed the data by extracting line and column profiles, concatenating them, and saving the result in a CSV file. During the training of the 1DCNN model, this CSV file is loaded. However, it's worth noting that this pre-processing can alternatively be performed in an online fashion by directly passing the original image to the model and extracting the profiles on the fly. While this approach may slow down the model slightly, it offers the advantage of saving manual pre-processing time.

7.2 As Binary Classifiers

We compare the results of CNN and 1DCNN model, along with one class classifier model in the context of binary classification between 'good' and 'defective' image [refer Table 7.3]. Notably, the CNN model outperforms the others, achieving an accuracy of 98%, while the one-class classifier attains an accuracy of 84%.

7.3 Summary

In this section, we provide a comprehensive qualitative comparison, using factors such as training time, computational requirements, chosen approach, architecture, input shape, and preprocessing steps for image-based, one-class classification, and time series-based approaches. Additionally, we perform quantitative comparisons across all our approaches, evaluating multiclass classification metrics including precision, recall, F1 score, and accuracy.

Approach	**Image-based**	**One-class-classification**	**Time series-based**
Architecture	CNN with ResNet-18 backbone	Variational Autoencoder with ResNet-18 backbone	1D CNN
Method	Supervised	Unsupervised	Supervised
Does	multiclass classification	binary classification	multiclass classification
Works on	Image patches	Image patches	Line profiles
Preprocessing	Making patches of image, label each patch manually	Making patches of image, label each patch manually	Extract row and column profile from image and concatenate
Can pre-processing be done online?*	No	No	Yes
Epochs trained for	30	1640	30
Shape of input	(783,955)	(783,955)	(1,5216)
Training needs	GPU	GPU	GPU or CPU
Training time (GPU)	147 min 23 secs	78 hours	3min 2secs
Training time (CPU)	-	-	14min 36secs
Model for raw images works directly on post-processed images?	Yes	-	No

Table 7.1: Qualitative comparative analysis of image-based, one-class-classifier and time-series-based approach. Training time is averaged for 5 folds for respective number of training epochs.

Models	**Accuracy**	**Precision**	**Recall**	**F1-score**
ResNet-18	**98%**	0.9867	0.9866	0.9865
1D-CNN	96.6%	0.96	0.96	0.96

Table 7.2: Comparison of evaluation metrics of both image and time-series approaches

Model	**Class**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
CNN	Good	96.3%	0.922	0.981	0.951
	Defective	96.3%	0.993	0.969	0.981
	Overall	98.1%	0.963	0.993	0.969
1DCNN	Good	98%	0.9774	0.9800	0.9787
	Defective	93.46%	0.9419	0.9346	0.9382
	Overall	96.8%	0.9683	0.9683	0.9683
OC_VAE	Good	83.82%	0.7671	0.5333	0.6292
	Defective	83.82%	0.8537	0.9439	0.8966
	Overall	83.82%	0.8104	0.7386	0.7629

Table 7.3: Comparison of Precision, Recall, F1 Score, and Accuracy for 'Good' and 'Defective' Classes, and Overall Metrics for CNN, OC-VAE, and 1DCNN Models

Chapter 8

Discussion and Conclusion

8.1 Discussion

In medical imaging, all imaging modalities can be prone to spurious findings caused by hardware malfunctions or operator errors. Artifacts can mask clinical content or affect its quality degrading the accuracy of diagnosis. Detecting these artifacts in X-ray imaging is paramount due to their widespread use and diagnostic importance. Currently, Siemens X-ray systems rely on manual inspection by service technicians to ensure that X-ray images are devoid of artifacts. This manual process is time-consuming, costly, and dependent on the technician's expertise, leading to potential delays and increased expenses.

Our goal is to develop an AI based system that can automatically and accurately classify X-ray images that are affected by artifacts. We focused on artifacts originating from a specific type of flat panel detector. The artifacts in question are not anatomical in nature; instead, they are technical artifacts resulting from the failure or malfunction of this particular type of X-ray machine component. These anomalies may manifest as irregular patterns, unusual distortions, or inconsistencies in the X-ray images that can lead to misleading diagnostics. This classification is essential for quality control and for ensuring that only high-quality diagnostic images are used for clinical evaluation and decision-making.

Challenges in Dataset and Data Imbalance The dataset used in this research posed several challenges, ranging from the overall limited quantity of available data and data imbalance between classes to the large size of images and difficulties in resizing. To address resizing challenges, a patching strategy was employed, dividing the image into patches, and manually labeling them individually. While this resolved resizing issues, it introduced

additional efforts in manual labeling, with challenges such as potential inaccuracies and variations in labeling across different patches. Additionally, ambiguity in the dataset, including cases of multi-labeling and incorrect labeling, introduced complexities that needed careful consideration. Augmentation techniques were applied to address data scarcity with careful consideration to ensure their appropriateness and avoid introducing additional artifacts or distorting the intrinsic characteristics of the images. Some defective classes had only 6 and 9 images, presenting challenges for robust model training. Cho et. al's findings [Cho+15] emphasize the importance of an adequate number of training samples, suggesting a minimum of 4092 samples per class for achieving a 99.5% accuracy in medical imaging models. Because of this presented data constraint, the test dataset could not be separated out. We could access the performance of our model only validation evaluation metrics and exact performance on an unseen test dataset remains unknown.

We employed supervised image-based and time series-based methods, and unsupervised approaches to solve our artifact classification problem. Each approach comes with its own set of results and considerations, as discussed below -

Image-Based Approach We initially utilized CNNs on image patches, and a comparative analysis was performed using ResNet-18, EfficientNet_b0, and DenseNet-121, revealing comparable accuracies of 98%, 98.06%, and 97.7%, respectively. Despite their similar performance, ResNet-18 exhibited a faster training time, leading to its selection for subsequent experiments. The inference process on the validation dataset involved segmenting the whole-sized images into nine patches, making predictions on each patch, and averaging the results. The validation accuracy of 97% was achieved on whole-sized images. For model interpretability, the Grad-CAM technique was employed, illustrating that the trained ResNet-18 model accurately focused on the regions containing defects.

However, the image-based approach's drawback lies in the manual labeling process, which is prone to errors and subjectivity, particularly when conducted by non-experts. This limitation becomes more pronounced as the dataset size increases, making the labeling task both time-consuming and potentially less accurate. Despite this disadvantage, the success of the ResNet-18 model demonstrates the viability of the image-based approach in artifact detection.

Time series-based Approach : We explored an alternative by converting the images into time series data using their line profiles. This approach involved obtaining pixel intensities along specific paths, which could potentially capture defect-specific patterns.

However, the performance with a single-layer LSTM cell achieved only 57% accuracy. To enhance results, we designed a custom architecture with parallel branches, leading to an improved accuracy of 96%. The interpretability of this model was aided by saliency maps.

The advantage of the time series model lies in its ability to operate on the entire X-ray image without the need for patching and its lesser computational requirement. The pre-processing step only involves obtaining line profiles. However, this approach hinges on the assumption that the artifacts primarily manifest as straight lines and exist predominantly in horizontal and vertical directions. Given the electronic design of detectors, artifacts along straight lines are the most common ones. If artifacts exhibit more complex patterns or orientations, this approach may face challenges in capturing their characteristics accurately. Nevertheless, for scenarios where artifacts align with the anticipated directional patterns, the time series model provides a straightforward and efficient solution, offering a holistic analysis of the entire X-ray image.

One-Class Classification Acknowledging the dataset's class imbalance, we explored unsupervised learning using generative modeling for binary classification. The model aims to learn the patterns and distribution of good images. During inference, samples that deviate significantly from the training distribution are flagged as defective. An AE and a VAE approaches were compared, achieving 82% and 84% accuracy, respectively. With more training images, results of OCC can be improved. Both CNN-based and time series-based approaches are limited to classifying defects present in the training data, while OCC has the capability to classify novel defects. However, it is essential to acknowledge OCC challenges. These models demand a substantial amount of training data, introducing computational complexity and longer training times. Despite this drawback, the OCC model becomes a valuable tool in scenarios where novel and visually distinct artifacts may occur.

Post-Processed Images Analysis A significant aspect of the research was assessing the generalization of models trained on intensity linear raw X-ray images to post-processed X-ray images directly, without further retraining. To assess this generalization, we conducted an evaluation based on the model's confidence scores. We observed image-based CNN approach displayed resilience and reasonable generalization, while the time series model faced challenges. This could be because

- **Distribution Discrepancy**: Raw X-ray images and their post-processed counterparts often exhibit different statistical distributions. The post-processing steps can introduce

variations that alter the characteristics of the image data. CNNs, especially in image-based tasks, are adept at capturing structural information. In the case of defect detection in X-ray images, the structural characteristics of defects often remain consistent between raw and post-processed images. The deeper layers of a CNN can learn hierarchical representations that encapsulate such structural details, enabling the model to generalize across different data distributions.

- **Line Profile Model Sensitivity**: The Line profile model is particularly sensitive to variations in data distribution. The precise alignment of profiles and the inherent sensitivity to pixel-wise changes make them more susceptible to variations introduced. Learning domain-invariant features is a challenging task, and the Line Profile model might struggle to adapt to the variations introduced during image post-processing.

- **Unseen Resizing Technique**: The fact that post-processed images are smaller and need to be resized to match the dimensions of raw images introduces an additional layer of complexity. Resizing involves interpolating pixel values, and in our case for the line profile, by extending the last pixel value to fill the additional space. The time series model has not been exposed to this specific resizing technique during its training phase. As a result, the model may struggle to generalize effectively to post-processed images.

8.2 Future Approaches

The path forward in this research opens up several exciting possibilities for enhancing the current model's capabilities and expanding its applicability. There's a need to address stated data challenges. Expanding the dataset with a focus on balanced representation across defect classes for training and forming a comprehensive test set for further analysis of our model is crucial. There is an opportunity to explore innovative data resizing techniques as an alternative to our patching data approach. This exploration aims to enhance computational efficiency while preserving data integrity, particularly for handling large images.

Another avenue for investigation involves adapting a model that is initially trained with a specific set of defects to accommodate new defect types. This challenge necessitates techniques such as transfer learning and fine-tuning to seamlessly integrate the detection of new anomalies without the need for extensive retraining. Moreover, the research could delve into cross-detector generalization, where we assess whether a model originally trained

for one detector can be effectively applied to a different detector. This exploration is key to expanding the model's versatility and real-world applicability.

To improve the model's initialization and performance, instead of pre-training CNN models with ImageNet, pre-training with dedicated X-ray datasets like MURA [1], could be explored.

Moreover, a transition from single-label to multi-label classification could be considered for images containing multiple artifacts. This shift would offer a more comprehensive analysis of image content, providing valuable insights for diagnosis and quality control. Furthermore, there is an opportunity to broaden the scope of the research by delving into artifact segmentation. This approach enables the precise localization of defects within images, a valuable aspect of image analysis.

Other one-class classification models with GANs architecture like AnoGAN and Efficient GAN (E-GAN) can be tried out [Cek23].

Future work can also be performed on post-processed image analysis of the time series model. It can include finding a conclusive reason for its failure and experimenting replacement of extra pixels by mean, or median or introducing such extending of pixels while training raw images. Conducting a comprehensive analysis to understand the specific failure modes of the model on post-processed images is a crucial avenue for future work. Insights gained from this analysis can guide targeted improvements. One could also try to expose the model to varied spatial transformations, including resizing methods, during training, which may develop a more robust internal representation. Investigating transfer learning techniques and domain adaptation methods may also be fruitful.

These future directions offer a holistic roadmap for advancing artifact detection and classification in X-ray images, addressing various aspects of model performance and applicability. With these avenues of exploration, we aim to further refine the model's capabilities and contribute to the field of medical image analysis.

8.3 Conclusion

In our study, we present AI-based methods for detecting artifacts in raw X-ray images obtained from flat panel detectors. Addressing challenges of dataset scarcity, ambiguity, and class imbalance, we explore diverse approaches for artifact identification. We have shown

[1] MURA is a widely recognized dataset for musculoskeletal radiographs. Available on: https://stanfordmlgroup.github.io/competitions/mura/

the capability of supervised learning methods, using image-based and time series-based approaches; and an unsupervised one-class classification approach using generative models. We identified major strengths and challenges associated with each approach and understood the inner workings of the models using interpretability techniques. The image-based approach attains higher accuracy, but with considerable pre-processing requirements. On the other hand, the time-series-based approach utilizing line profiles emerges as the optimal choice in our scenario, particularly since artifacts consistently manifest as straight lines. This approach offers the advantage of lower computational demands and reduced pre-processing steps. We have also shown the successful classification of post-processed images using a CNN-based model trained on raw X-ray images without requiring retraining.

In implementing our first version, we achieved promising results, paving the way for further evaluation with additional data. The potential integration of our proposed approach with Siemens X-ray systems could streamline the process of obtaining high-quality X-ray images. Identifying certain artifacts in imaging would serve as a reliable indicator of failed detectors, signaling the need for replacement. Our work, thus, lays the groundwork for a future where AI becomes an integral part of quality control in X-ray medical imaging, paving the way for more efficient and accurate diagnostics.

Appendix A

Appendix

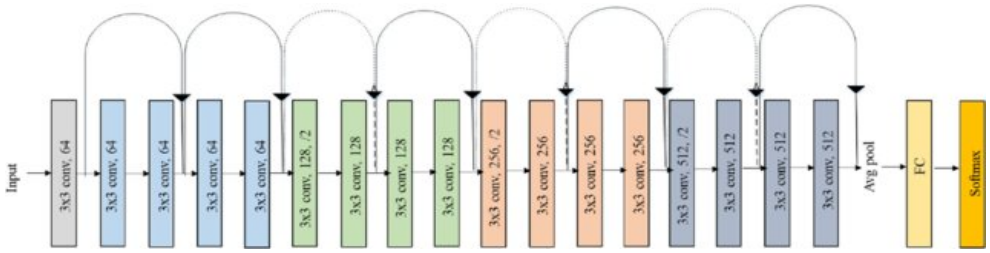


Figure A.1: Original ResNet-18 Architecture Source:[Ram+19]

Figure A.2: The architecture for EfficientNet-B0 Source:[Tan+19]

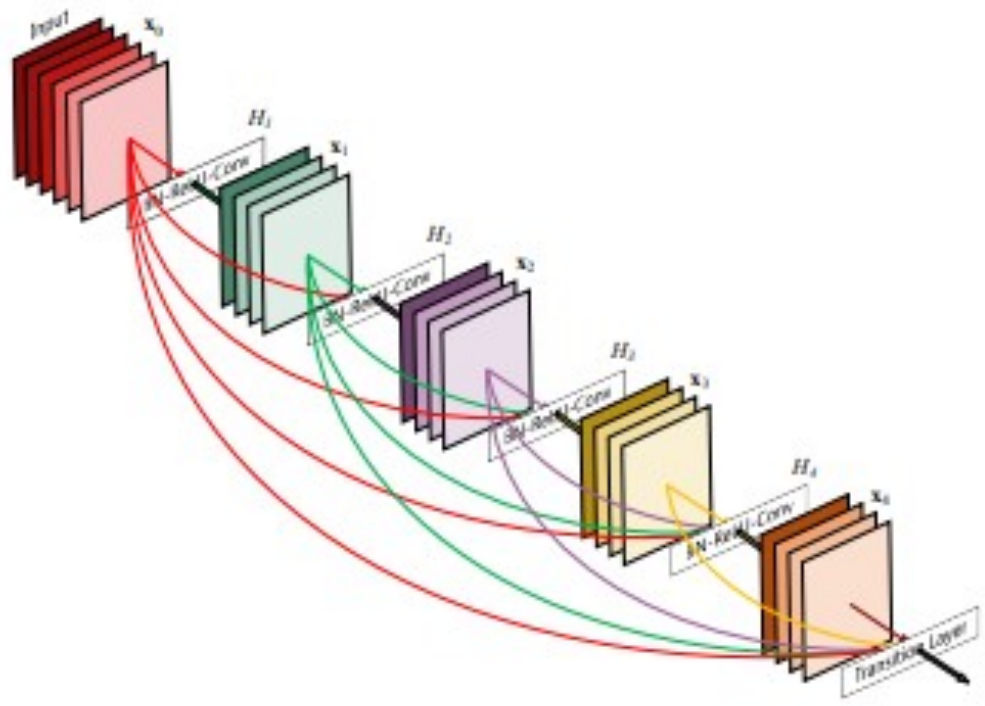


Figure A.3: The architecture for DenseNet Source:[Hua+16]

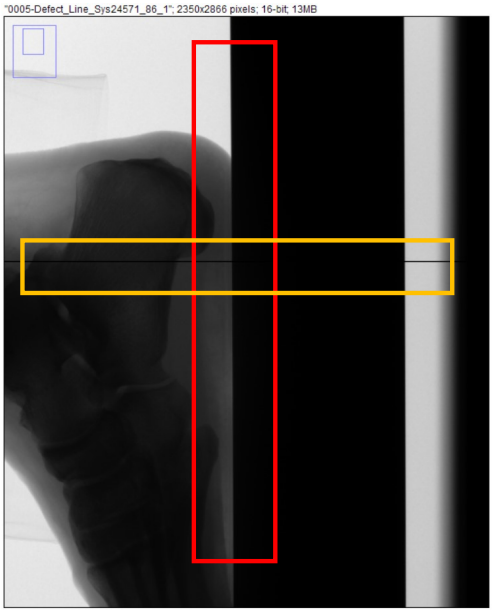
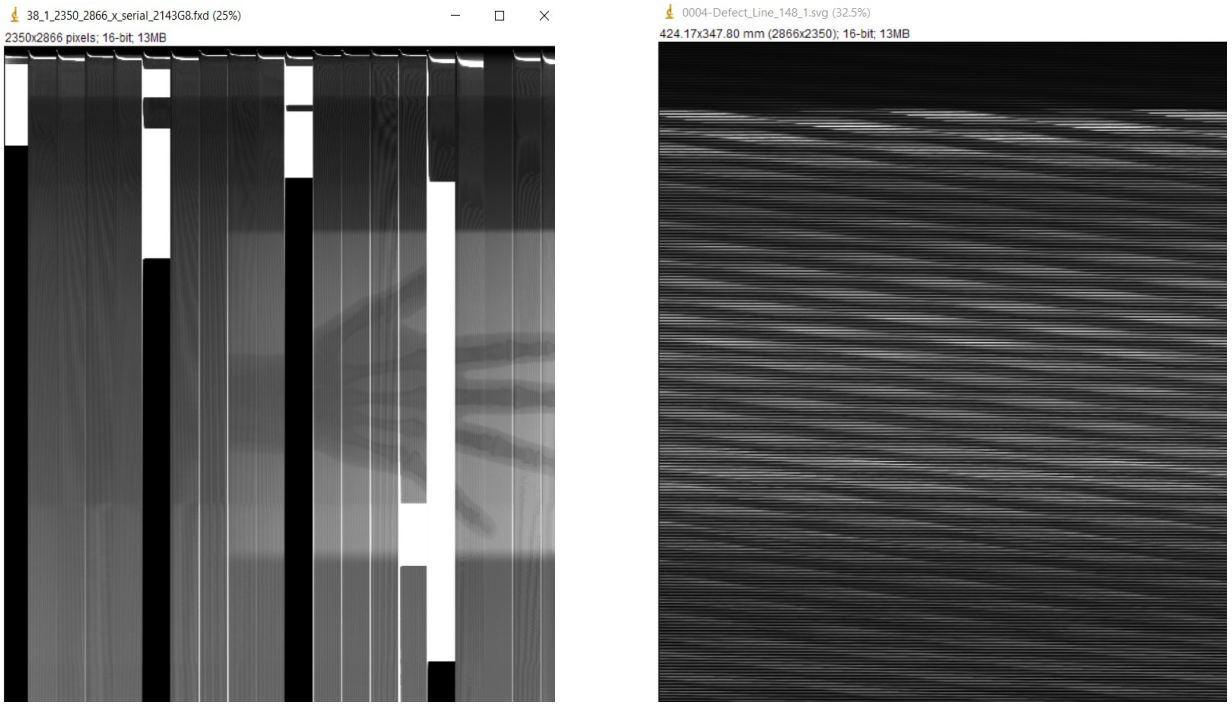


Figure A.4: Example of multi-label case, where single X-ray image contains FX05 (highlighted in orange) and FX03 (highlighted in red) artifact



A **B**

Figure A.5: disturbed images

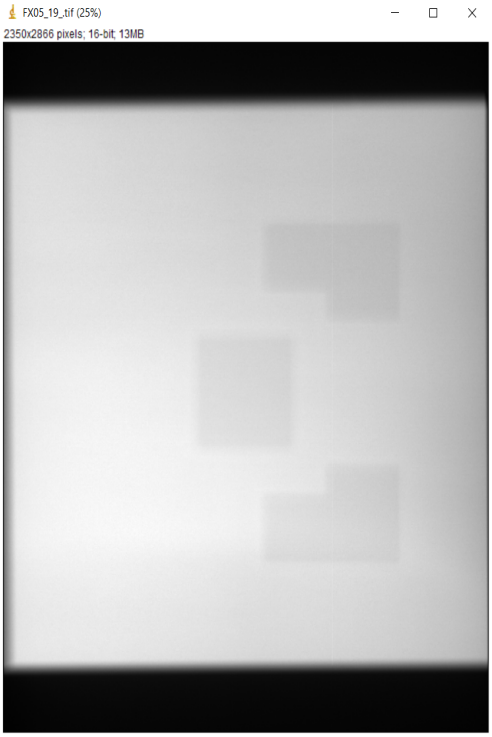


Figure A.6: shows offset image not containing any clinical content

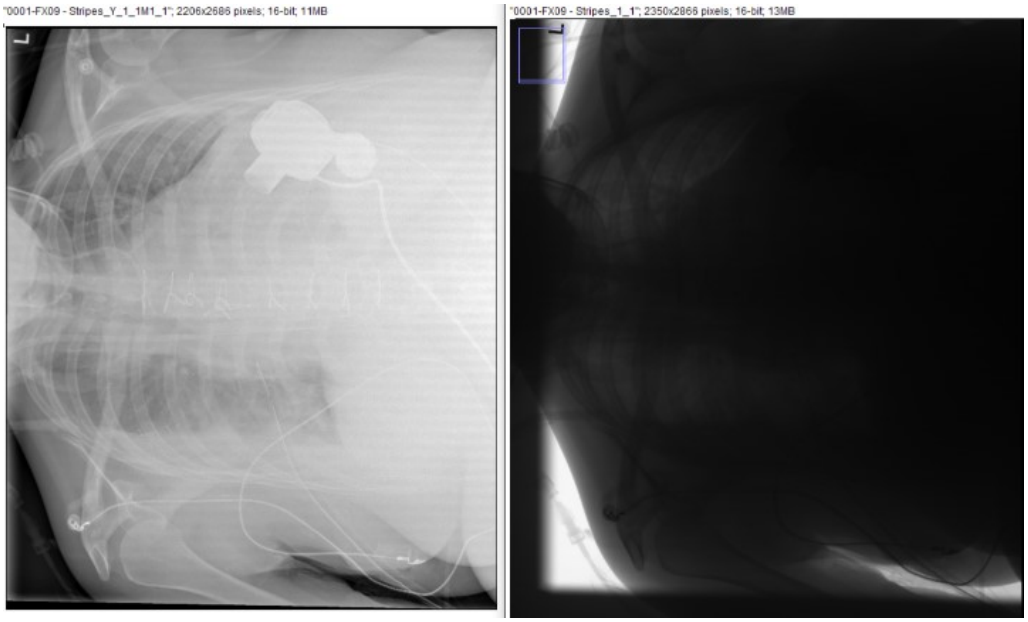


Figure A.7: Figure shoes both raw image (right) and it's post-processed image (left) containing FX09_Stripes artifact, present in dataset

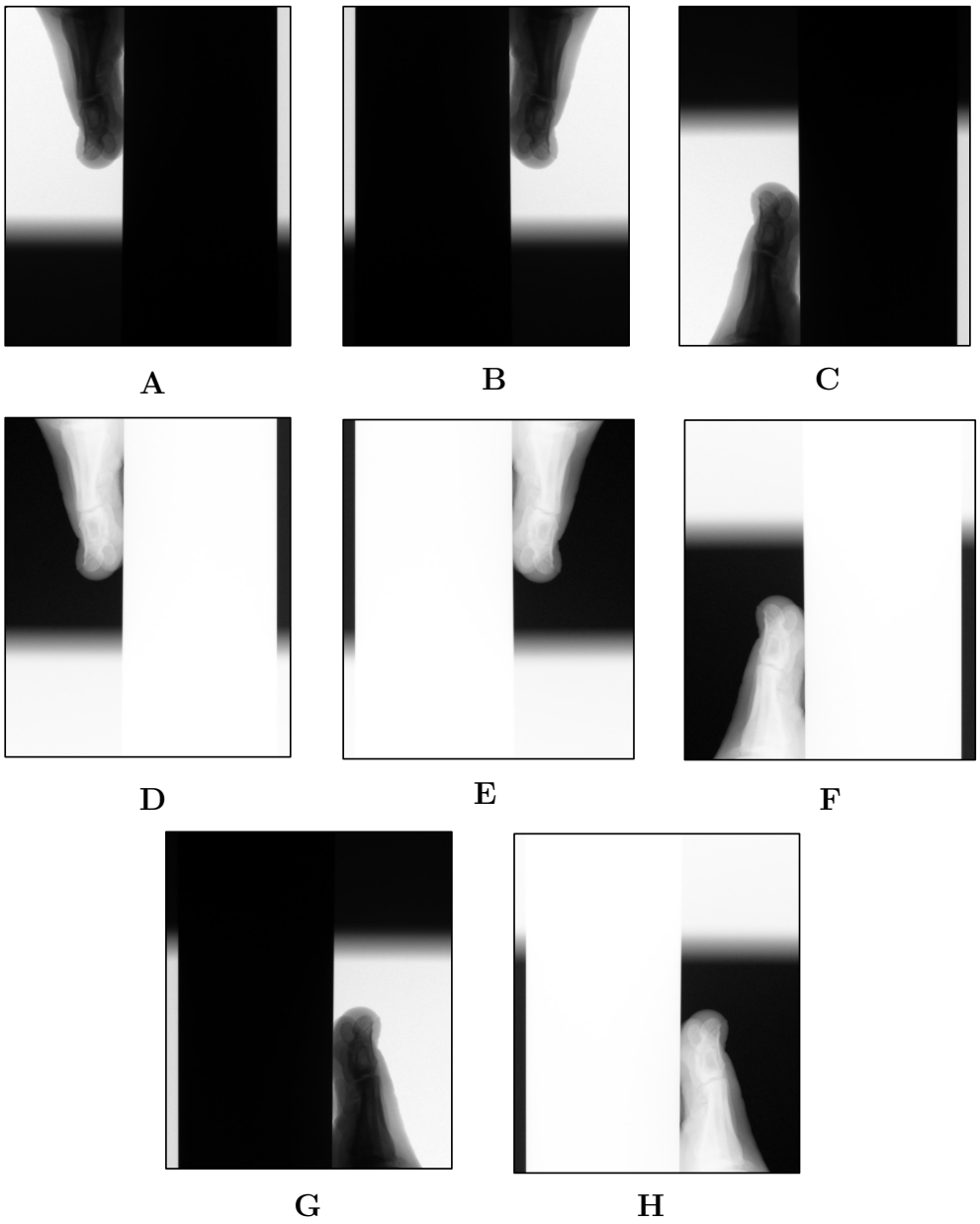


Figure A.8: Figure shows all [A] original image and [B-G] 7 augmented views of the original image with horizontal-flip, vertical-flip, flipboth, invert, invert horizontal flip, invert vertical flip, invert flipboth

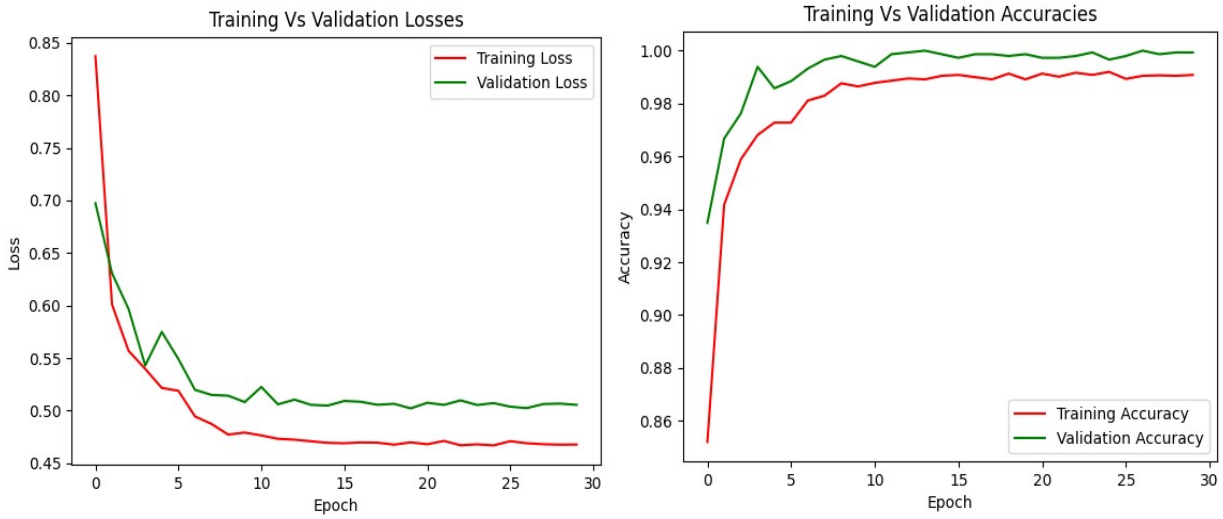
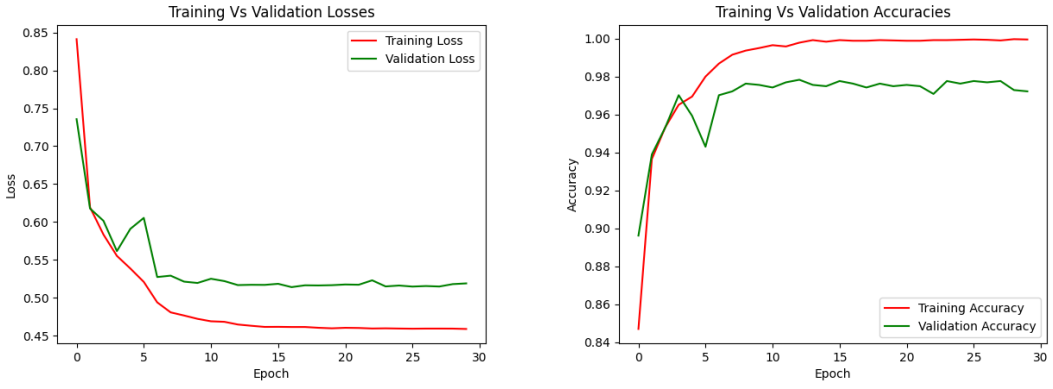


Figure A.9: (a) Training and validation loss, and (b) Training and validation accuracy versus number of training epochs for EfficientNet model



(a) (b)

Figure A.10: Training and validation loss(left), and Training and validation accuracy(right) versus number of training epochs for DenseNet model

Artifact	**Line profile direction**	**Characteristics**
FX00 Sporadic line artifacts	Horizontal	A sequence of sharp peaks and troughs is observed at farther intervals.
FX02 Group of line artifacts	Horizontal	A sequence of sharp peaks and troughs observed at closer intervals.
FX03 Partly brighter	Vertical	A sharp sudden fall in intensity when there is a change in contrast of the image.
FX04 Group of defect lines	Can happen in both vertical and horizontal	Sharp spikes followed by deep troughs.
FX05 Defect line	Can happen both vertically and horizontally	A sudden drop is observed when the line is black, and a sudden peak when the line is white.
FX09 Stripes	Horizontal	A recurrent undulating pattern, where the line undergoes systematic cycles of elevation and descent.

Table A.1: Artifact Names, Profile direction, and Characteristics

Table A.3: Statistics of the efficient net model for all 7 classes

	Accuracy (%)	**Precision**	**Recall**	**F1 Score**
FX00	99.53%	0.9927	0.9577	0.9749
FX02	100%	1.0000	1.0000	1.0000
FX03	100%	1.0000	1.0000	1.0000
FX04	100%	1.0000	1.0000	0.9825
FX05	99.93%	1.0000	0.9870	0.9935
FX09	99.73%	0.9701	0.9701	0.9701
Good	99.19%	0.9914	0.9971	0.9942

		Predicted Class						
		FX00	FX02	FX03	FX04	FX05	FX09	Good
Actual Class	FX00	136	0	0	0	0	0	6
	FX02	0	32	0	0	0	0	0
	FX03	0	0	43	0	0	0	0
	FX04	0	0	0	86	0	0	0
	FX05	0	0	0	0	76	0	1
	FX09	0	0	0	0	0	65	2
	Good	1	0	0	0	0	2	1037

Table A.2: Confusion Matrix from EfficientNet model

		Predicted Class						
		FX00	FX02	FX03	FX04	FX05	FX09	Good
Actual Class	FX00	135	0	0	0	1	0	6
	FX02	0	31	0	0	0	0	1
	FX03	0	0	42	0	0	0	0
	FX04	0	0	0	84	0	0	2
	FX05	0	0	0	1	74	0	1
	FX09	0	0	0	0	0	60	7
	Good	1	0	0	0	0	1	1038

Table A.4: Confusion Matrix from DenseNet model, averaged over 5 folds

Table A.5: Statistics of the Densenet-121 model for all 7 classes

	Accuracy (%)	**Precision**	**Recall**	**F1 Score**
FX00	99.46%	0.9926	0.9507	0.9712
FX02	99.93%	1.0000	0.9688	0.9841
FX03	100%	1.0000	1.0000	1.0000
FX04	99.80%	0.9882	0.9767	0.9825
FX05	99.80%	0.9867	0.9737	0.9801
FX09	99.46%	0.9836	0.8955	0.9375
Good	98.73%	0.9839	0.9981	0.9909

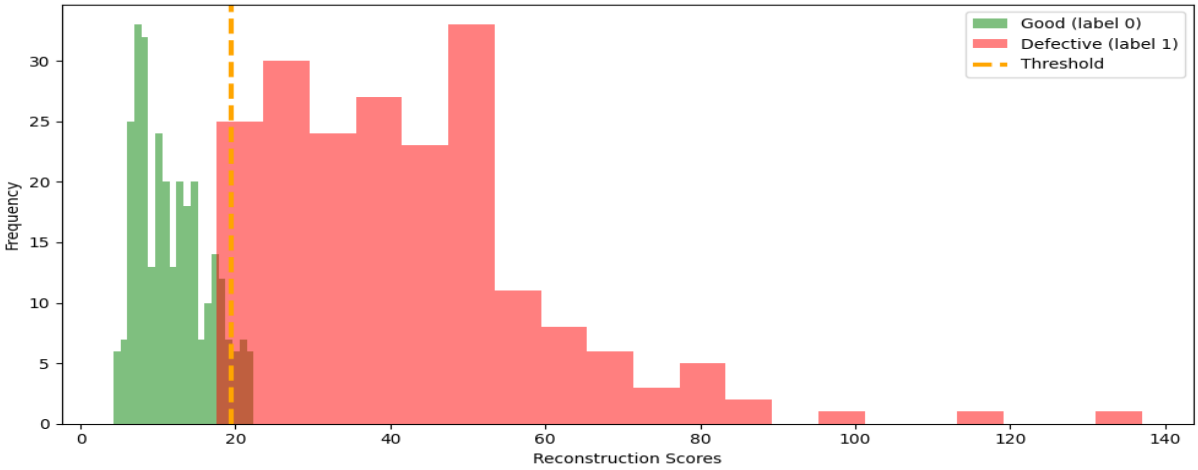
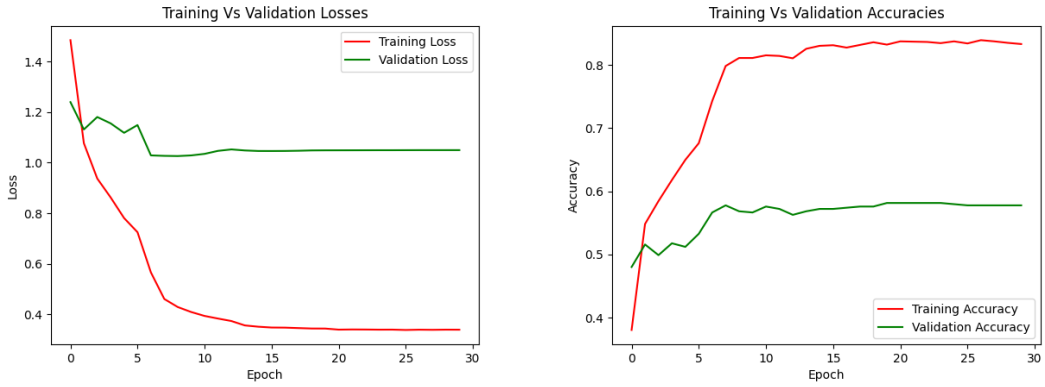
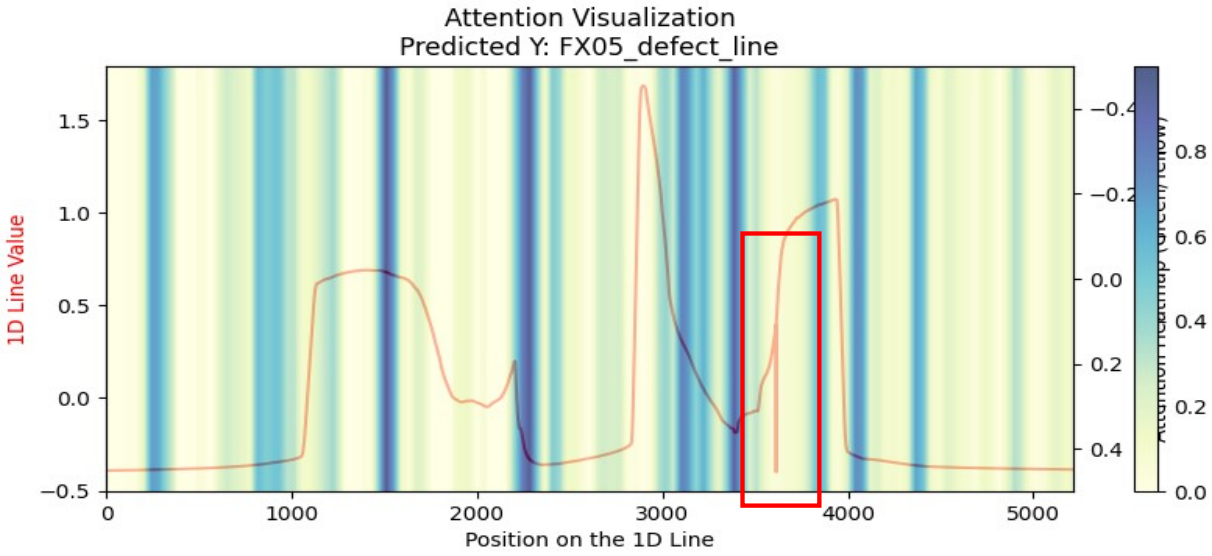


Figure A.11: Histogram of reconstruction score of good (green) and defective (red) and statistical threshold (orange) with 200 good and 200 defective raw X-ray images from OC-AE model

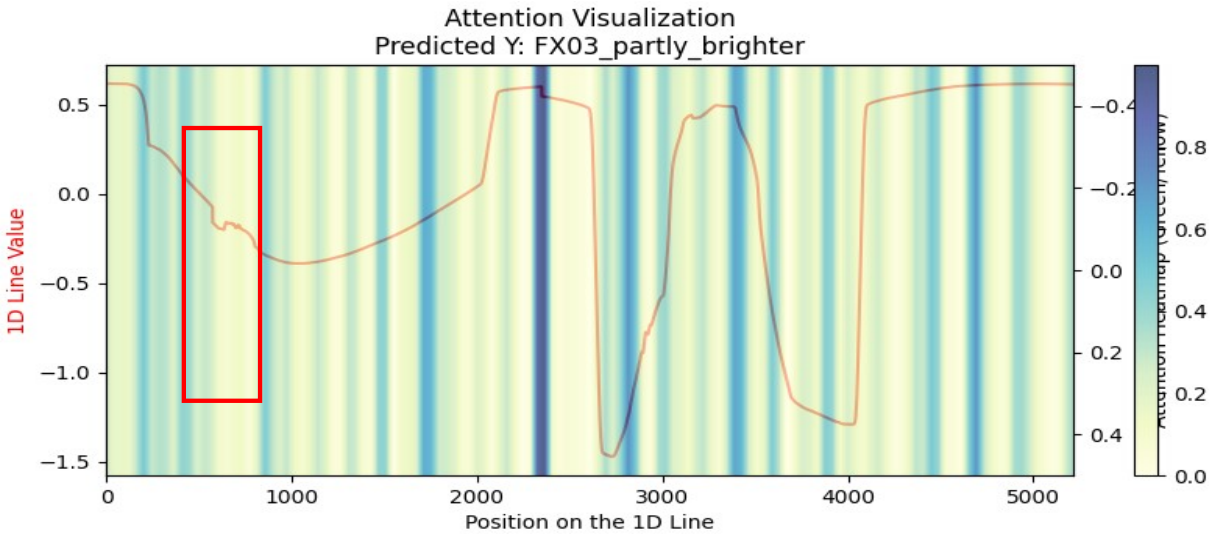


(a) (b)

Figure A.12: (a) Training and validation loss, and (b) Training and validation accuracy versus number of training epochs



A



B

Figure A.13: shows failure cases of Grad-CAM on [A] FX05 [B] FX03, the red box shows the region where the model should focus on, but instead, it focuses on the region with blue color

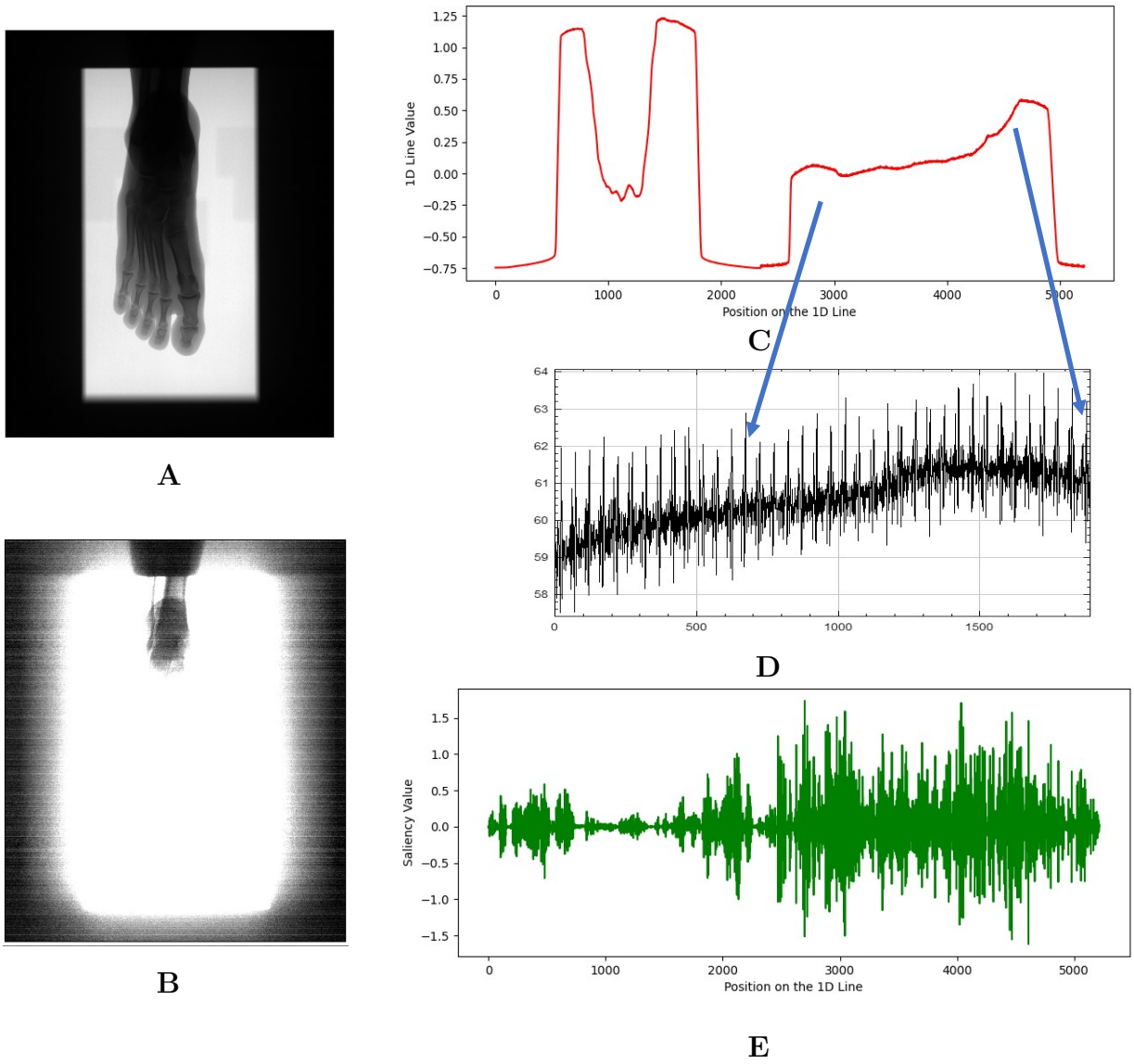


Figure A.14: Explainability of sporadic artifact: [A] Original raw X-ray image; [B] Enhanced visualization using ImageJ to highlight areas of artifact; [D] Input to the 1DCNN model, consisting of concatenated row and column profiles; [C] Zoomed-in section of the line profile revealing details where FX00_sporadic_line_artifact exists; [E] corresponding saliency map indicating elevated attention values corresponding to the sporadic line features.

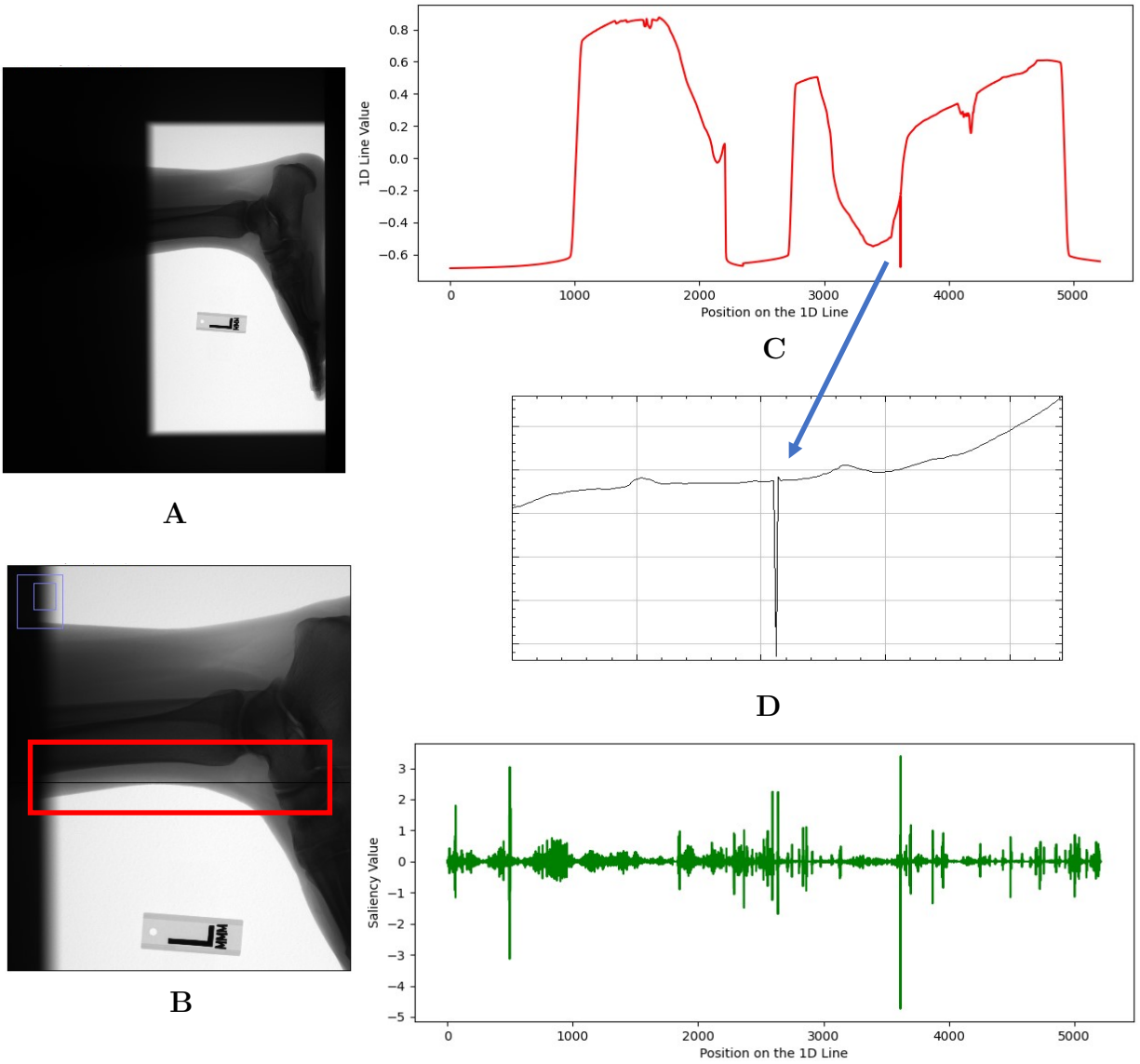


Figure A.15: Explainability of defect line artifact: [A] Original raw X-ray image; [B] Enhanced visualization using ImageJ to highlight areas of artifact; [D] Input to the 1DCNN model, consisting of concatenated row and column profiles; [C] Zoomed-in section of the line profile revealing details where FX05_defect_line exists; [E] corresponding saliency map indicating elevated attention values corresponding to the defect line.

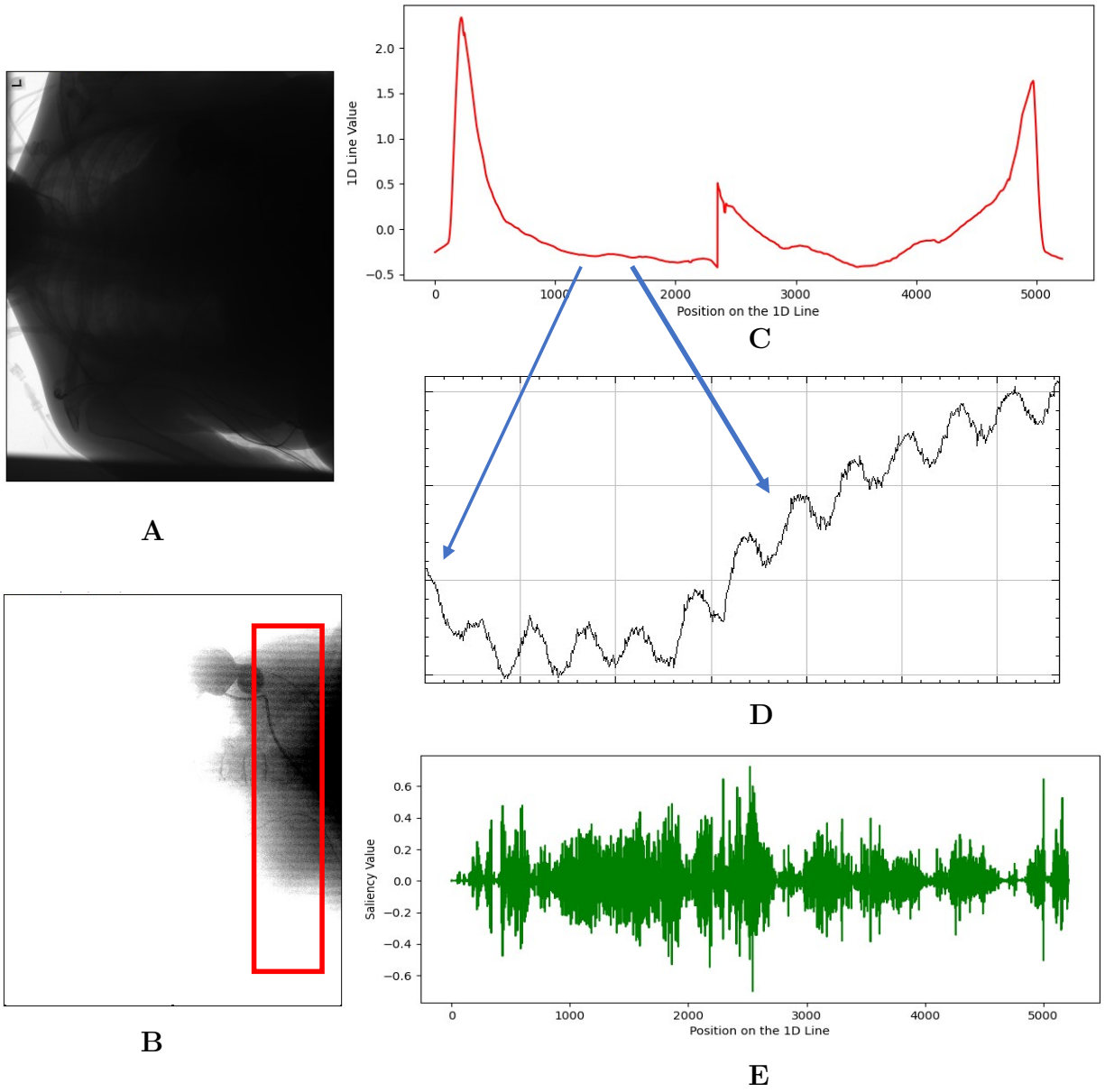


Figure A.16: Explainability of stripes artifact: [A] Original raw X-ray image; [B] Enhanced visualization using ImageJ to highlight areas of artifact; [D] Input to the 1DCNN model, consisting of concatenated row and column profiles; [C] Zoomed-in section of the line profile revealing details where FX09_stripes artifact exists; [E] corresponding saliency map indicating elevated attention values corresponding to the stripes pattern.

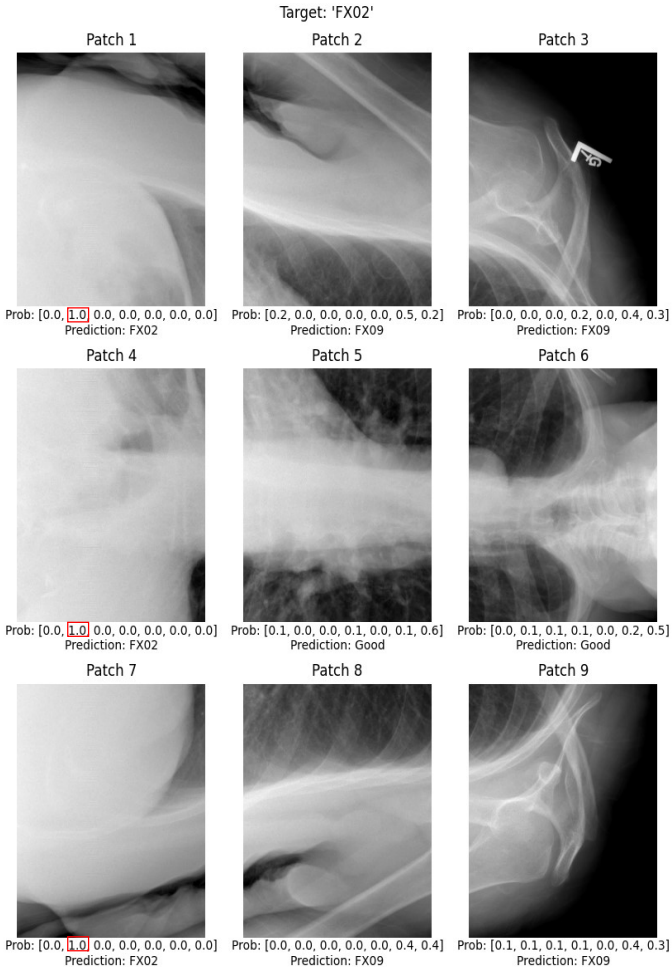


Figure A.17: Analysis of post-processed image containing FX02 group of line artifact, it can be seen from the figure that each patch predicts correctly with high confidence

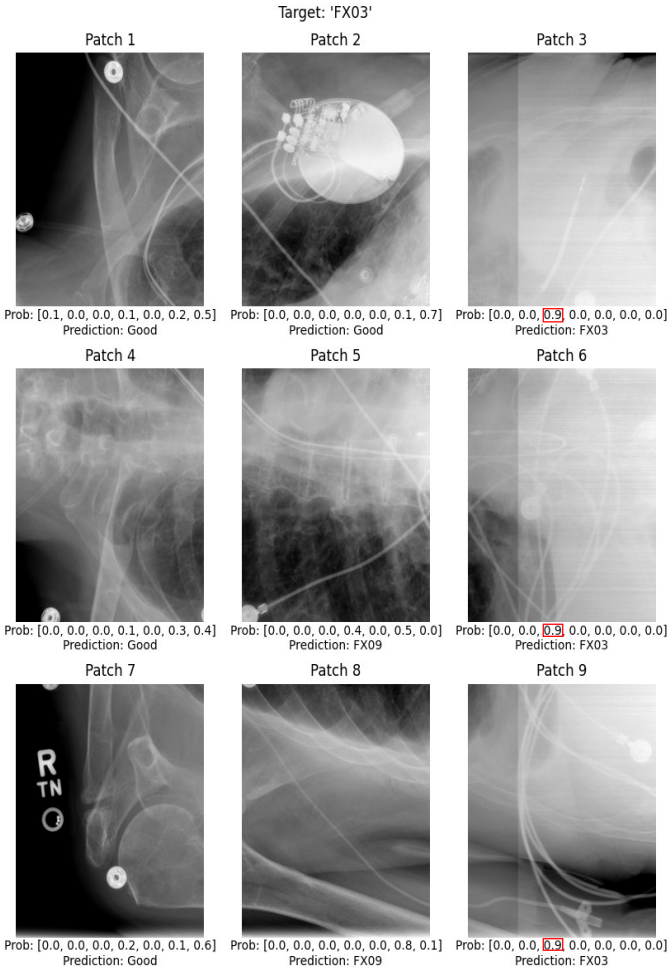


Figure A.18: Analysis of post-processed image containing FX03 partly brighter artifact, it can be seen from the figure that each patch predicts correctly with high confidence

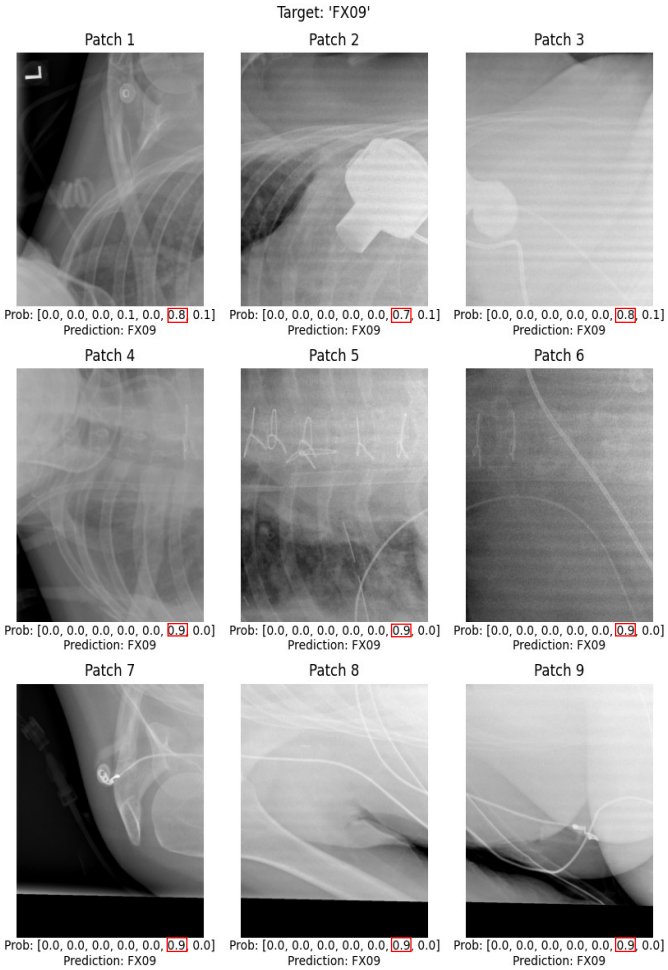


Figure A.19: Analysis of post-processed image containing FX09 stripes artifact, it can be seen from the figure that each patch predicts correctly with high confidence

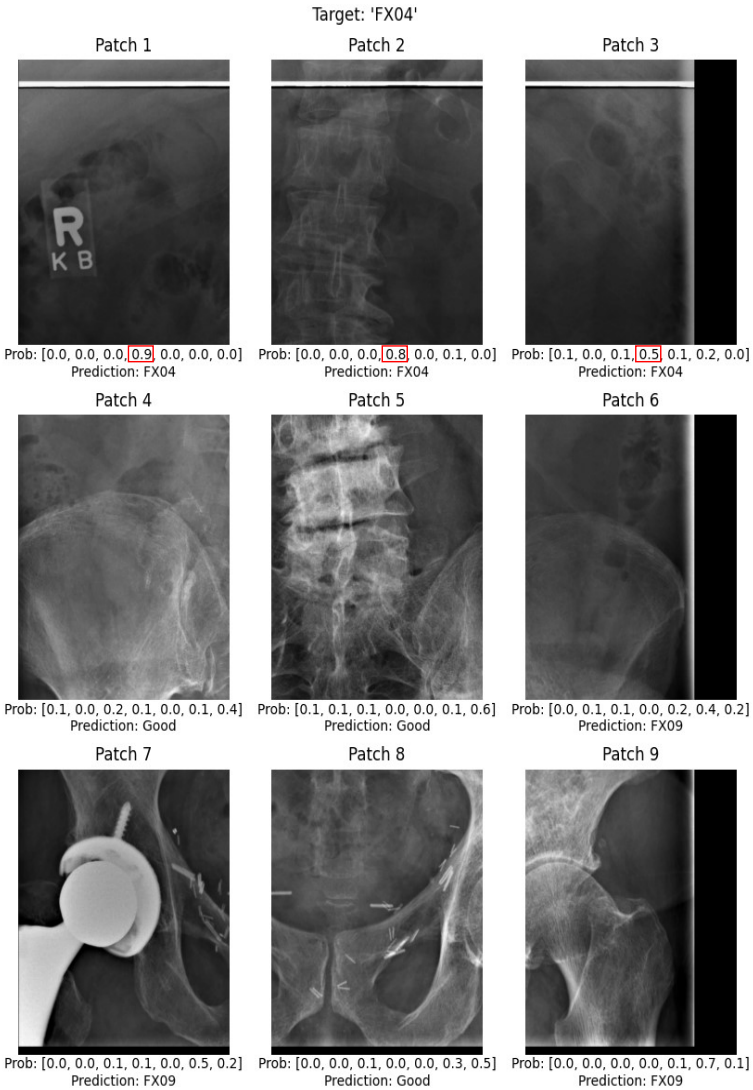


Figure A.20: post-process analysis of image containing FX04 group of defect line artifact, it can be seen from the figure that each patch predicts correctly with high confidence

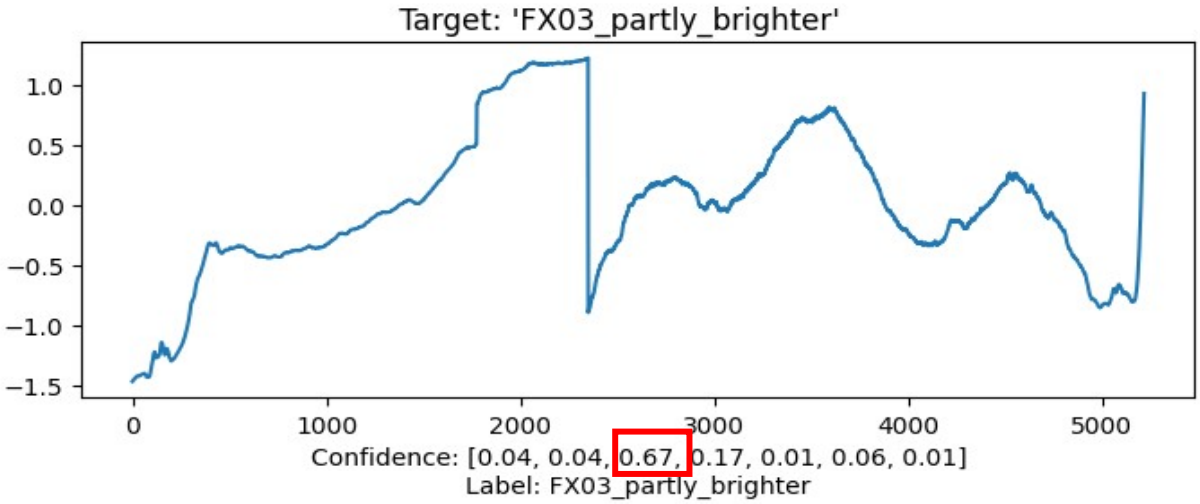


Figure A.21: Figure shows line profile of a post-processed image containing defect of FX03_partly_brighter artifact, with very high confidence, the model predicts that it belongs to FX03 artifact class type

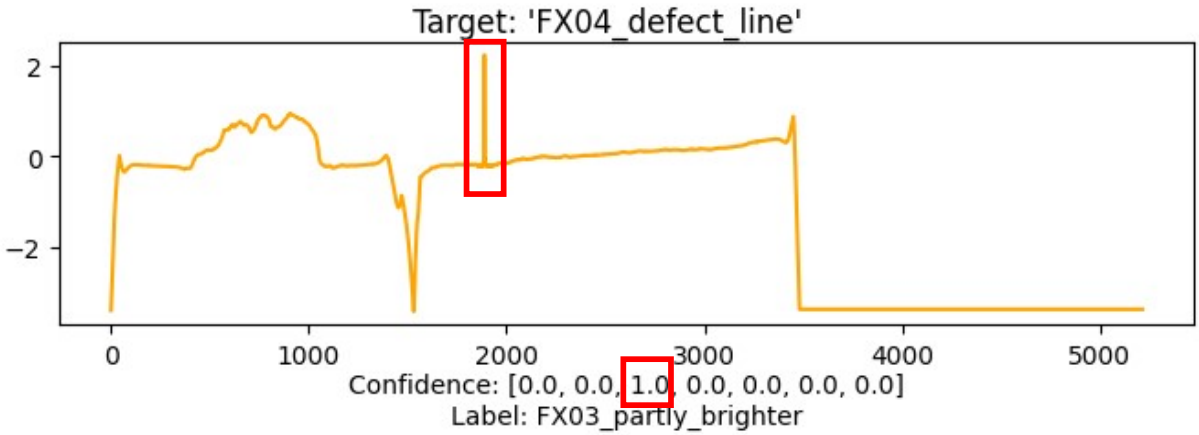


Figure A.22: Figure shows failure case of a post-processed image containing FX04_defect_line artifact, the model gives higher probability to FX03_partly_brighter artifact

List of Abbreviations

IR Imbalanced Ratio

AE Autoencoder

VAE Variational Autoencoder

MRI Magnetic Resonance Imaging

CT Computed Tomography

AI Artificial Intelligence

TIF Tagged Image File Format

PNG Portable Network Graphics

DICOM Digital Imaging and Communications in Medicine

BMP Bitmap

CR Computed Radiography

DR Digital Radiography

NDT non-destructive testing

MLP multilayer perceptron

EM electromagnetic

ROI region of interest

IP Imaging Plate

GPU Graphics Processing Unit

CPU Central Processing Unit

CNN Convolutional Neural Network

RNN Recurrent Neural Network

CV Cross Validation

ReLU rectified linear unit

PReLU parameterized rectified linear unit

TempCNN Temporal 1D-CNN

Grad-CAM Gradient-weighted Class Activation Mapping

BN Batch-normalization

GRU Gated Recurrent Unit

LSTM Long Short-Term Memory

XAI Explainable artificial intelligence

ANN artificial neural network

OCC One class classifier

DL Deep Learning

FC fully connected

RMSE root mean squared error

KL Kullback–Leibler

Bibliography

[Aba+16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: large-scale machine learning on heterogeneous distributed systems, 2016. DOI: 10.48550/ARXIV.1603.04467. URL: https://arxiv.org/abs/1603.04467 (cited on p. 34).

[Age15] I. A. E. Agency. *Worldwide Implementation of Digital Imaging in Radiology: A Resource Guide*. IAEA human health series. International Atomic Energy Agency, 2015. ISBN: 9789201134196. URL: https://books.google.de/books?id=ADvNzgEACAAJ (cited on p. 9).

[Agg18] C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018. ISBN: 9783319944647. URL: https://books.google.de/books?id=AsTswQEACAAJ (cited on p. 16).

[Als+11] J. Als-Nielsen and D. McMorrow. *Elements of Modern X-ray Physics*. Wiley, March 2011. DOI: 10.1002/9781119998365. URL: https://doi.org/10.1002/9781119998365 (cited on p. 7).

[Bis+24] C. M. Bishop and H. Bishop. *Deep Learning - Foundations and Concepts*. Springer, 2024. ISBN: 978-3-031-45467-7. DOI: 10.1007/978-3-031-45468-4. URL: https://doi.org/10.1007/978-3-031-45468-4 (cited on pp. 17–19).

[Boa+17] N. Boaretto and T. M. Centeno. Automated detection of welding defects in pipelines from radiographic images DWDI. *NDT &amp E International*,

86:7–13, March 2017. DOI: 10.1016/j.ndteint.2016.11.003. URL: https://doi.org/10.1016/j.ndteint.2016.11.003 (cited on p. 39).

[Bor+19] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini. Anomaly detection using autoencoders in high performance computing systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9428–9433, July 2019. DOI: 10.1609/aaai.v33i01.33019428. URL: https://doi.org/10.1609/aaai.v33i01.33019428 (cited on p. 28).

[Bot10] L. Bottou. *Large-scale machine learning with stochastic gradient descent*. In *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, 2010, pages 177–186. ISBN: 9783790826043. DOI: 10.1007/978-3-7908-2604-3_16. URL: http://dx.doi.org/10.1007/978-3-7908-2604-3_16 (cited on p. 19).

[Bou] R. Bourne. *Fundamentals of Digital Imaging in Medicine*. Springer London. ISBN: 978-1-84882-086-9. DOI: 10.1007/978-1-84882-087-6. URL: https://doi.org/10.1007/978-1-84882-087-6 (cited on p. 12).

[Bou10] R. Bourne. *Fundamentals of Digital Imaging in Medicine*. Springer London, 2010. ISBN: 9781848820876. DOI: 10.1007/978-1-84882-087-6. URL: http://dx.doi.org/10.1007/978-1-84882-087-6 (cited on pp. 1, 8).

[Bro] J. Brownlee. A Gentle Introduction to Transfer Learning for Deep Learning - MachineLearningMastery.com — machinelearningmastery.com. https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Accessed 20-11-2023] (cited on p. 24).

[Bur+16] W. Burger and M. J. Burge. *Digital Image Processing*. Springer London, 2016. DOI: 10.1007/978-1-4471-6684-9. URL: https://doi.org/10.1007/978-1-4471-6684-9 (cited on p. 14).

[Bur+23] C. M. Burlacu, A. C. Burlacu, M. Praisler, and C. Paraschiv. Harnessing deep convolutional neural networks detecting synthetic cannabinoids: a hybrid learning strategy for handling class imbalances in limited datasets. *Inventions*, 8(5), 2023. ISSN: 2411-5134. DOI: 10.3390/inventions8050129. URL: https://www.mdpi.com/2411-5134/8/5/129 (cited on p. 20).

[Cas16] D. Castelvecchi. Can we open the black box of ai? *Nature*, 538(7623):20–23, October 2016. ISSN: 1476-4687. DOI: 10.1038/538020a. URL: http://dx.doi.org/10.1038/538020a (cited on p. 31).

[Cek23] M. Cekić. Anomaly detection in medical time series with generative adversarial networks: a selective review. In D. V. K. Parimala, editor, *Anomaly Detection - Recent Advances, AI and ML Perspectives and Applications*, chapter 7. IntechOpen, Rijeka, 2023. DOI: 10.5772/intechopen.112582. URL: https://doi.org/10.5772/intechopen.112582 (cited on p. 107).

[Cha+14] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: delving deep into convolutional nets, 2014. arXiv: 1405.3531 [cs.CV] (cited on p. 40).

[Cho+15] J. Cho, K. Lee, E. Shin, G. Choy, and S. Do. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?, 2015. eprint: arXiv:1511.06348 (cited on p. 104).

[Col+11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: a matlab-like environment for machine learning. In *BigLearn, NIPS workshop*, number CONF, 2011 (cited on p. 34).

[Dan+18] F. J. W. M. Dankers, A. Traverso, L. Wee, and S. M. J. van Kuijk. Prediction modeling methodology. In *Fundamentals of Clinical Data Science*, pages 101–120. Springer International Publishing, December 2018. DOI: 10.1007/978-3-319-99713-1_8. URL: https://doi.org/10.1007/978-3-319-99713-1_8 (cited on p. 37).

[Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. DOI: 10.1109/CVPR.2009.5206848 (cited on pp. 24, 34).

[Dod+21] P. S. Dodamani and A. Danti. Assesment of bone mineral density in x-ray images using image processing. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 905–908, 2021 (cited on p. 9).

[DRO+08] W. T. DROST, D. J. REESE, and W. J. HORNOF. DIGITAL RADIOGRAPHY ARTIFACTS. *Veterinary Radiology Ultrasound*, 49:S48–S56, January 2008. DOI: 10.1111/j.1740-8261.2007.00334.x. URL: https://doi.org/10.1111/j.1740-8261.2007.00334.x (cited on p. 2).

[Dud+00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. en. A Wiley-Interscience publication. John Wiley & Sons, Nashville, TN, 2nd edition, October 2000 (cited on p. 19).

[Ebr+20] S. A. Ebrahim, J. Poshtan, S. M. Jamali, and N. A. Ebrahim. Quantitative and qualitative analysis of time-series classification using deep learning. *IEEE Access*, 8:90202–90215, 2020. DOI: 10.1109/ACCESS.2020.2993538 (cited on p. 25).

[Fer+20] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Deep learning for medical anomaly detection – a survey, 2020. eprint: arXiv:2012.02364 (cited on pp. 28, 31, 32).

[Gao+17] M. Gao, Z. Xu, L. Lu, A. P. Harrison, R. M. Summers, and D. J. Mollura. Holistic interstitial lung disease detection using deep convolutional neural networks: multi-label learning and unordered pooling. *CoRR*, abs/1701.05616, 2017. arXiv: 1701.05616. URL: http://arxiv.org/abs/1701.05616 (cited on p. 40).

[Ger19] A. Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition, 2019. ISBN: 1492032646 (cited on pp. 25–27, 71).

[Goo+16] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org (cited on p. 26).

[Got+20] K. Gotkowski, C. Gonzalez, A. Bucher, and A. Mukhopadhyay. M3d-cam: a pytorch library to generate 3d data attention maps for medical deep learning, 2020. eprint: arXiv:2007.00453 (cited on p. 31).

[Gra14] A. Graves. Generating sequences with recurrent neural networks, 2014. arXiv: 1308.0850 [cs.NE] (cited on p. 19).

[Gua19] X. Guan. *Predict next location of users using deep learning*. en. PhD thesis, 2019 (cited on p. 16).

[He+15a] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (cited on p. 35).

[He+15b] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: surpassing human-level performance on imagenet classification, 2015. arXiv: 1502.01852 [cs.CV] (cited on pp. 20, 21).

[Hin+12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. DOI: 10.48550/ARXIV.1207.0580. URL: https://arxiv.org/abs/1207.0580 (cited on p. 21).

[Hop+17] T. Hope, Y. S. Resheff, and I. Lieder. *Learning tensorflow: A guide to building deep learning systems*. Ö'Reilly Media, Inc.", 2017 (cited on p. 34).

[Hua+16] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993 (cited on pp. 36, 73, 76, 110).

[Iof+15] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift, 2015. arXiv: 1502.03167 [cs.LG] (cited on p. 23).

[Ism+20] A. A. Ismail, M. Gunady, H. C. Bravo, and S. Feizi. Benchmarking deep learning interpretability in time series predictions, 2020. eprint: arXiv:2010.13924 (cited on p. 32).

[Jia+14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: convolutional architecture for fast feature embedding, 2014. DOI: 10.48550/ARXIV.1408.5093. URL: https://arxiv.org/abs/1408.5093 (cited on p. 34).

[Jia+21] L. Jiang, Y. Wang, Z. Tang, Y. Miao, and S. Chen. Casting defect detection in x-ray images using convolutional neural networks and attention-guided data augmentation. *Measurement*, 170:108736, January 2021. DOI: 10.1016/j.measurement.2020.108736. URL: https://doi.org/10.1016/j.measurement.2020.108736 (cited on p. 39).

[JIM+08] D. A. JIMNEZ, L. J. ARMBRUST, R. T. O'BRIEN, and D. S. BILLER. ARTIFACTS IN DIGITAL RADIOGRAPHY. *Veterinary Radiology Ultrasound*, 49(4):321–332, July 2008. DOI: 10.1111/j.1740-8261.2008.00374.x. URL: https://doi.org/10.1111/j.1740-8261.2008.00374.x (cited on pp. 1, 2).

[Kha+20a] H. Khalid and S. S. Woo. Oc-fakedect: classifying deepfakes using one-class variational autoencoder. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2794–2803, 2020. DOI: 10.1109/CVPRW50498.2020.00336 (cited on p. 29).

[Kha+20b] H. Khalid and S. S. Woo. Oc-fakedect: classifying deepfakes using one-class variational autoencoder. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2794–2803, 2020. DOI: 10.1109/CVPRW50498.2020.00336 (cited on p. 30).

[Kin+13] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. eprint: arXiv:1312.6114 (cited on p. 28).

[Kin+17] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization, 2017. arXiv: 1412.6980 [cs.LG] (cited on p. 19).

[Kri+12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012 (cited on pp. 20, 22, 54, 74).

[Kru+07] E. A. Krupinski, M. B. Williams, K. Andriole, K. J. Strauss, K. Applegate, M. Wyatt, S. Bjork, and J. A. Seibert. Digital radiography image quality: image processing and display. *Journal of the American College of Radiology*, 4(6):389–400, June 2007. DOI: 10.1016/j.jacr.2007.02.001. URL: https://doi.org/10.1016/j.jacr.2007.02.001 (cited on pp. 1, 63).

[Kwa+17] A. Kwasigroch, A. Mikolajczyk, and M. Grochowski. Deep neural networks approach to skin lesions classification — a comparative analysis. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, August 2017. DOI: 10.1109/mmar.2017.8046978. URL: http://dx.doi.org/10.1109/mmar.2017.8046978 (cited on pp. 23, 40).

[LeC+12] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient backprop*. In *Neural Networks: Tricks of the Trade: Second Edition*. G. Montavon, G. B. Orr, and K.-R. Müller, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pages 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3 (cited on p. 22).

[LeC+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. DOI: 10.1162/neco.1989.1.4.541 (cited on p. 22).

[Lia+22a] Y. Liao and B. Yang. To generalize or not to generalize: towards autoencoders in one-class classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. DOI: 10.1109/IJCNN55064.2022.9892812 (cited on p. 29).

[Lia+22b] Y. Liao and B. Yang. To generalize or not to generalize: towards autoencoders in one-class classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2022. DOI: 10.1109/ijcnn55064.2022.9892812. URL: http://dx.doi.org/10.1109/IJCNN55064.2022.9892812 (cited on p. 80).

[Lim+22] A. Lim, J. Lo, M. W. Wagner, B. Ertl-Wagner, and D. Sussman. Automatic artifact detection algorithm in fetal MRI. en. *Front. Artif. Intell.*, 5:861791, June 2022 (cited on p. 41).

[Lin+23] X. Lin, Y. Cheng, G. Chen, W. Chen, R. Chen, D. Gao, Y. Zhang, and Y. Wu. Semantic segmentation of chinarsquo;s coastal wetlands based on sentinel-2 and segformer. *Remote Sensing*, 15(15), 2023. ISSN: 2072-4292. URL: https://www.mdpi.com/2072-4292/15/15/3714 (cited on p. 20).

[Lit+17] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017 (cited on p. 34).

[Loe+22] C. Loeffler, W.-C. Lai, B. Eskofier, D. Zanca, L. Schmidt, and C. Mutschler. Don't get me wrong: how to apply deep visual interpretations to time series, 2022. DOI: 10.48550/ARXIV.2203.07861. URL: https://arxiv.org/abs/2203.07861 (cited on p. 32).

[Med+17] K. Meding, A. Loktyushin, and M. Hirsch. Automatic detection of motion artifacts in mr images using cnns. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, March 2017. DOI: 10.1109/icassp.2017.7952268. URL: http://dx.doi.org/10.1109/icassp.2017.7952268 (cited on p. 41).

[Met+00] R. L. V. Metter, J. Beutel, and H. L. Kundel, editors. *Handbook of Medical Imaging, Volume 1. Physics and Psychophysics*. SPIE, February 2000. DOI: 10.1117/3.832716. URL: https://doi.org/10.1117/3.832716 (cited on p. 10).

[Nap+23] M. A. Napoli Spatafora, A. Ortis, and S. Battiato. Glr: gradient-based learning rate scheduler. In G. L. Foresti, A. Fusiello, and E. Hancock, editors, *Image Analysis and Processing – ICIAP 2023*, pages 269–281, Cham. Springer Nature Switzerland, 2023 (cited on p. 21).

[Nay+21] J. Nayak, B. Naik, and A. Abraham, editors. *Understanding COVID-19: The role of computational intelligence*. en. Studies in computational intelligence. Springer Nature, Cham, Switzerland, 1st edition, July 2021 (cited on pp. 19, 25).

[Ndi+18] E. Ndikumana, D. H. T. Minh, N. Baghdadi, D. Courault, and L. Hossard. Deep recurrent neural network for agricultural classification using multitemporal SAR sentinel-1 for camargue, france. *Remote Sensing*, 10(8):1217, August 2018. DOI: 10.3390/rs10081217. URL: https://doi.org/10.3390/rs10081217 (cited on p. 25).

[Nov97] R. Novelline. *Squire's Fundamentals of Radiology*. Harvard University Press, 5th edition, 1997. ISBN: 0-674-83339-2 (cited on p. 7).

[Pel+18] C. Pelletier, G. I. Webb, and F. Petitjean. Temporal convolutional neural network for the classification of satellite image time series, 2018. eprint: arXiv:1811.10166 (cited on pp. 27, 28).

[Ram+19] F. Ramzan, M. U. G. Khan, A. Rehmat, S. Iqbal, T. Saba, A. Rehman, and Z. Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of Medical Systems*, 44(2), December 2019. ISSN: 1573-689X. DOI: 10.1007/s10916-019-1475-2. URL: http://dx.doi.org/10.1007/s10916-019-1475-2 (cited on pp. 35, 109).

[Rix14] J. G. Rix. *Transportation Optimization in Tactical and Operational Wood Procurement Planning*. PhD thesis, École Polytechnique de Montréal, December 2014. URL: https://publications.polymtl.ca/1622/ (cited on p. 15).

[Rus+09] S. Russell and P. Norvig. *Artificial intelligence*. en. Pearson, Upper Saddle River, NJ, 3rd edition, December 2009 (cited on p. 16).

[Sch14] P. Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, September 2014. ISSN: 1573-756X. DOI: 10.1007/s10618-014-0377-7. URL: http://dx.doi.org/10.1007/s10618-014-0377-7 (cited on p. 24).

[Sei+98] J. A. Seibert, J. M. Boone, and K. K. Lindfors. Flat-field correction technique for digital detectors. In *Medical Imaging*, 1998. URL: https://api.semanticscholar.org/CorpusID:128680776 (cited on p. 3).

[Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017 (cited on pp. 31, 76).

[Sha+18] N. Sharma, V. Jain, and A. Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.198. URL: https://www.sciencedirect.com/science/article/pii/S1877050918309335. International Conference on Computational Intelligence and Data Science (cited on p. 22).

[Shi00] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, October 2000. ISSN: 0378-3758. DOI: 10.1016/s0378-3758(00)00115-4. URL: http://dx.doi.org/10.1016/S0378-3758(00)00115-4 (cited on p. 23).

[Sim+13] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps, 2013. eprint: arXiv:1312.6034 (cited on p. 32).

[Smi+18] D. Smirnov and E. Mephu Nguifo. Time series classification with recurrent neural networks. In September 2018 (cited on p. 26).

[Sof+19a] S. Soffer, A. Ben-Cohen, O. Shimon, M. M. Amitai, H. Greenspan, and E. Klang. Convolutional neural networks for radiologic images: a radiologist's guide. *Radiology*, 290(3):590–606, March 2019. ISSN: 1527-1315. DOI: 10.1148/radiol.2018180547. URL: http://dx.doi.org/10.1148/radiol.2018180547 (cited on pp. 39, 40).

[Sof+19b] S. Soffer, A. Ben-Cohen, O. Shimon, M. M. Amitai, H. Greenspan, and E. Klang. Convolutional neural networks for radiologic images: a radiologist's guide. *Radiology*, 290(3):590–606, 2019 (cited on p. 34).

[Spa13] M. Spahn. X-ray detectors in medical imaging. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 731:57–63, 2013. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2013.05.174. URL: https://www.sciencedirect.com/science/article/pii/S0168900213007961. PIXEL 2012 (cited on p. 10).

[Staa] Statistics. Statistics — england.nhs.uk. https://www.england.nhs.uk/statistics/. [Accessed 27-11-2023] (cited on p. 2).

[Stab] Statistics. Statistics &xBB; Diagnostic Imaging Dataset — england.nhs.uk. https://www.england.nhs.uk/statistics/statistical-work-areas/diagnostic-imaging-dataset/. [Accessed 20-11-2023] (cited on p. 1).

[Tan+19] M. Tan and Q. V. Le. Efficientnet: rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946 (cited on pp. 35, 73, 109).

[Tom+22] A. Tompe and K. Sargar. *X-Ray Image Quality Assurance*. StatPearls Publishing, October 2022 (cited on p. 63).

[Wal+12] A. Walz-Flannigan, D. Magnuson, D. Erickson, and B. Schueler. Artifacts in digital radiography. *American Journal of Roentgenology*, 198(1):156–161, 2012. DOI: 10.2214/AJR.11.7237. eprint: https://doi.org/10.2214/AJR.11.7237. URL: https://doi.org/10.2214/AJR.11.7237. PMID: 22194492 (cited on p. 2).

[Wal+18] A. I. Walz-Flannigan, K. J. Brossoit, D. J. Magnuson, and B. A. Schueler. Pictorial review of digital radiography artifacts. *RadioGraphics*, 38(3):833–846, May 2018. DOI: 10.1148/rg.2018170038. URL: https://doi.org/10.1148/rg.2018170038 (cited on p. 1).

[Wes+21] E. Westphal and H. Seitz. A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural networks. *Additive Manufacturing*, 41:101965, 2021. ISSN: 2214-8604. DOI: https://doi.org/10.1016/j.addma.2021.101965. URL: https://www.

sciencedirect.com/science/article/pii/S2214860421001305 (cited on p. 45).

[Wil+04] C. Willis, S. Thompson, and S. Shepard. Artifacts and misadventures in digital radiography. *Applied Radiology*, 33:11–20, January 2004 (cited on p. 1).

[Wu+20] D. Wu, X. Wang, J. Su, B. Tang, and S. Wu. A labeling method for financial time series prediction based on trends. *Entropy*, 22(10):1162, October 2020. ISSN: 1099-4300. DOI: 10.3390/e22101162. URL: http://dx.doi.org/10.3390/e22101162 (cited on p. 24).

[Zha+17a] L. Zhang, L. Lu, I. Nogues, R. M. Summers, S. Liu, and J. Yao. Deeppap: deep convolutional networks for cervical cell classification. *IEEE Journal of Biomedical and Health Informatics*, 21(6):1633–1643, November 2017. ISSN: 2168-2208. DOI: 10.1109/jbhi.2017.2705583. URL: http://dx.doi.org/10.1109/JBHI.2017.2705583 (cited on p. 22).

[Zha+17b] L. Zhang, L. Lu, I. Nogues, R. M. Summers, S. Liu, and J. Yao. Deeppap: deep convolutional networks for cervical cell classification. *IEEE Journal of Biomedical and Health Informatics*, 21(6):1633–1643, 2017. DOI: 10.1109/JBHI.2017.2705583 (cited on p. 22).

[Zim+18] D. Zimmerer, S. A. A. Kohl, J. Petersen, F. Isensee, and K. H. Maier-Hein. Context-encoding variational autoencoder for unsupervised anomaly detection, 2018. arXiv: 1812.05941 [cs.LG] (cited on p. 40).