

FPGA-Accelerated Deep Learning for Denoising Low-Dose PET Scans

by

Eric-Khang Dao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Eric-Khang Dao 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Positron Emission Tomography (PET) is an essential imaging technique used in clinical settings for diagnosing conditions such as cancer and neurological disorders. However, its dependence on radiopharmaceuticals poses potential radiation exposure risks. Lowering the administered dose can help improve patient safety, but it also results in imagery with a reduced Signal-to-Noise (SNR), which can impact diagnostic accuracy. The trade-off between minimizing radiation exposure and maintaining image quality remains a key challenge in PET imaging. Recently, deep learning-based denoising techniques, such as Denoising Convolutional Neural Network (DnCNN), have been proven effective in restoring noisy images to standard quality. However, traditional implementations relying on Central Processing Units (CPU) and Graphics Processing Units (GPU) are often constrained by high power consumption and hardware overhead, limiting their feasibility in edge-compute applications.

To address these challenges, this thesis explores Field-Programmable Gate Array (FPGA)-based acceleration for PET image denoising. A dataset is constructed using PET scans from 10 Alzheimer’s disease patients from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database, with only 0.5% of the original radiotracer dose used. A software-based implementation is first developed using a proposed U-Net-like architecture. The model is then ported to an FPGA using OpenVINO and Intel’s FPGA AI Suite for hardware emulation. Experimental results show that the FPGA implementation offers a 77% improvement in the performance-to-watt ratio compared to the GPU-based solution, and a 2x reduction in latency compared to the CPU-based solution.

Acknowledgements

I would like to thank everyone who made this thesis possible. Firstly, God, who has given me with wisdom, strength, and perseverance to complete this work, in hopes of serving lives of those who may build upon it.

I would like to thank my supervisor, Dr. Vincent Gaudet. His generosity, kindness and guidance towards allowed me to pursue this opportunity, further developing my technical skills in this field.

I would like to thank my parents, who at a young age, decided to leave their home country of Vietnam to pursue opportunities in a foreign land for their children, including myself. Even working 70 hour weeks, my father managed to find the time to drive me to and from Toronto to Waterloo before I had my full driver's license. My mother, as busy as she is taking care of the house, always found time to prepared weeks worth of home cooked meals for me to take to school. Their sacrifice, dedication, and hard work will never be taken for granted. I am a lucky individual to have received these blessings.

I would like to thank my partner, Mimi Nguyen, who has given me nothing but love and positive encouragement throughout this journey. Her endless support and understanding gave me the necessary clarity to push forward and keep moving.

Finally, my time in this program would have not been possible without my good friend Thanushon Sivakaran. He gave me a place to stay for majority of the program, and offered me countless advice on machine learning, a new field to me at the time.

Dedication

To my parents, who have nurtured me with love and sacrifice.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	ix
List of Tables	xii
List of Abbreviations	xiii
1 Introduction	1
1.1 Motivation and Objectives	1
1.2 Thesis Objectives	2
1.3 Contributions	2
1.4 Thesis Organization	3
1.5 Publications	3

2	Background & Technical Foundation	4
2.1	Positron Emission Tomography	4
2.1.1	Physical Principles	4
2.1.2	Radiotracers for PET Imaging	7
2.1.3	Imaging Technology	9
2.1.4	Reconstruction	10
2.2	Fully Convolutional Network Architectures	11
2.2.1	Fully Convolutional Networks	12
2.2.2	Fully Convolutional Residual Networks	13
2.3	Intel OpenVINO and FPGA AI Suite	14
2.4	Machine Learning Model Quantization	15
2.4.1	Numerical Formats used for Quantization	15
2.4.2	Quantization Principles	18
2.5	Conclusion	20
3	Literature Review	21
3.1	FPGA Acceleration for Medical Image Processing	21
3.2	Deep Learning Low-Dose PET denoising	23
3.3	Research Gaps and Problem Definition	25
3.4	Conclusion	25
4	Software Implementation of Proposed Model	26
4.1	Dataset Selection and Preprocessing	26
4.1.1	Data	26
4.1.2	Preprocessing	27
4.2	Proposed Model Architecture	28
4.3	Training Environment and Hyperparameters	30
4.4	Evaluation Metrics	30
4.5	Conclusion	31

5	FPGA Deployment with OpenVINO and FPGA AI Suite	32
5.1	Preprocessing the Trained Model	33
5.2	Hardware Architecture Generation	33
5.3	Inference on Emulated FPGA Architectures	34
5.4	Power Consumption Measurements	34
5.5	Conclusion	34
6	Experimental Results and Evaluation	35
6.1	Visual Representations of Experimental Results	35
6.2	Comparison of Inference Accuracy	41
6.3	Comparison of Inference Latency	42
6.4	Comparison of Power Consumption	43
6.5	FPGA Resource Utilization	44
6.6	Conclusion	44
7	Discussion and Conclusions	45
7.1	Evaluation of Results	45
7.2	Conclusions and Future Directions	46
	References	48

List of Figures

2.1	Alpha Decay, adapted from [6]	5
2.2	Beta Decay, adapted from [6]	5
2.3	Gamma Emission, adapted from [6]	5
2.4	Electron Capture, adapted from [6]	6
2.5	Positron Emission, adapted from [6]	6
2.6	Positron Electron Annihilation, adapted from [23]	7
2.7	Diagrammatic Representation of a Cyclotron	8
2.8	Axial Diagrammatic View of PET Scanner	9
2.9	General Structure of a CNN Classification Network	11
2.10	General Structure of a Fully Convolutional Network	12
2.11	Original U-Net Architecture as Proposed by [31]	13
2.12	Original DNCNN Architecture as Proposed by [45]	14
2.13	Packing Structure of FP32	15
2.14	Packing Structure of FP16	16
2.15	Packing Structure of INT8	16
2.16	Packing Structure of INT7BFP	17
2.17	Visualization of symmetric quantization; floating-point values (orange line) are mapped onto the integer grid (blue line). s denotes scale factor, β denotes maximum of floating-point input.	19
2.18	Visualization of asymmetric quantization; floating-point values (orange line) are mapped onto the integer grid (blue line). s denotes scale factor, z denotes the zero-point, $[\alpha, \beta]$ denotes minimum and maximum of floating-point input, respectively.	20

4.1	Preprocessing Workflow	28
4.2	Proposed Model Architecture	29
5.1	FPGA Deployment Workflow	32
6.1	PET Scan of Patient 1. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	36
6.2	PET Scan of Patient 2. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	36
6.3	PET Scan of Patient 3. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	37
6.4	PET Scan of Patient 4. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	37
6.5	PET Scan of Patient 5. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	38
6.6	PET Scan of Patient 6. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	38
6.7	PET Scan of Patient 7. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	39
6.8	PET Scan of Patient 8. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	39
6.9	PET Scan of Patient 9. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	40
6.10	PET Scan of Patient 10. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth	40

6.11 Comparison of Average Inference Accuracy Metrics	41
6.12 Inference Latency Comparison	42
6.13 Average Power Consumption Comparison	43

List of Tables

6.1	Average Inference Accuracy	41
6.2	Inference Latency Per Image	42
6.3	Average Power Consumption and Performance per Watt	43
6.4	Resource Utilization of FPGA Configurations	44

List of Abbreviations

- AD** Alzheimer’s Disease [1](#), [26](#)
- ADNI** Alzheimer’s Disease Neuroimaging Initiative [iii](#), [26–28](#), [31](#)
- ALM** Adaptive Logic Module [33](#), [44](#)
- ASIC** Application-Specific Integrated Circuit [15](#)
- AUQ** Asymmetric Uniform Quantization [19](#), [20](#)
- BMP** bitmap [27](#)
- BRAM** Block RAM [22](#)
- CNN** Convolutional Neural Networks [1](#), [11](#), [21–23](#), [25](#), [46](#)
- CNR** Contrast-to-Noise [24](#)
- CPU** Central Processing Units [iii](#), [1–3](#), [13](#), [31](#), [44–47](#)
- CT** Computed Tomography [21](#), [23](#), [24](#)
- DICOM** Digital Imaging and Communications in Medicine [27](#)
- DnCNN** Denoising Convolutional Neural Network [iii](#)
- DRF** Dose Reduction Factor [24](#), [27](#)
- DSP** Digital Signal Processor [14](#), [22](#), [33](#), [44](#)
- FBP** Filtered-Back Projection [10](#), [24](#)

FCN Fully Convolutional Networks 1, 3, 4, 12, 13, 20, 22, 28, 31

FCRN Fully Convolutional Residual Network 13

FDG Fluorodeoxyglucose 8, 9

FF Flip-Flops 22, 44

FPGA Field-Programmable Gate Array iii, 1–4, 13–15, 20–22, 25, 27, 32–35, 41, 44–47

GAN Generative Adversarial Network 23

GPU Graphics Processing Units iii, 1–3, 13, 25, 44–47

HDL Hardware Description Language 22

HLS High Level Synthesis 14, 22

IFAS Intel FPGA AI Suite 14, 32–34, 45

IoU Intersection over Union 22

IPTC Intel Power and Thermal Calculator 34

IR Intermediate Representation 14, 33, 34

LOO Leave-One-Out 23

LOR Line of Response 10

LOR-RAMLA Line-of-Response Row-Action Maximum-Likelihood Algorithm 10, 27

LUT Look-Up Tables 21, 22

M20K M20K Memory Block 33, 44

MAC Multiply-and-Accumulate 18, 22

MCI Mild Cognitive Impairment 23

MLEM Maximum Likelihood Expectation Maximization 24

MO Model Optimizer 33

MRI Magnetic Resonance Imaging [21–24](#)

MSE Mean Squared Error [24](#)

NRMSE Normalized Root Mean Squared Error [23, 30, 45](#)

ONNX Open Neural Network Exchange [33, 34](#)

PET Positron Emission Tomography [iii, 1–4, 7–10, 20, 21, 23–25, 27, 31, 34, 35, 46, 47](#)

PSNR Peak Signal-to-Noise Ratio [23, 30, 45](#)

RAPL Running Average Power Limit [31](#)

ResNet Residual Networks [28](#)

RTL Register-Transfer Level [22](#)

SATP Standard Ambient Temperature and Pressure [34](#)

SNR Signal-to-Noise [iii, 24](#)

SSIM Structural Similarity Index [23, 31, 45](#)

SUQ Symmetric Uniform Quantization [18–20](#)

SUV Standard Uptake Values [23](#)

UQ Uniform Quantization [18](#)

Chapter 1

Introduction

1.1 Motivation and Objectives

PET is widely used in medical imaging for detecting and monitoring diseases such as cancer, and neurological disorders such as **Alzheimer’s Disease (AD)**. A fundamental component of **PET** imaging is the use of radiotracers, such as fluorodeoxyglucose (^{18}F -FDG), which are injected into the patient to facilitate imaging, which increases the risk of radiation exposure [26].

Although the radiation dose is generally within medically accepted limits according to the FDA [44], cumulative exposure poses potential health risks, particularly for vulnerable populations such as pediatric patients, and those participating in long-term monitoring studies. A viable approach to mitigate radiation exposure is to reduce the administered radiotracer dose. However, this results in images with increased noise, lower contrast, and diminished structural details, potentially affecting diagnostic accuracy.

Recent advancements in deep learning offer a promising alternative for reconstructing high-quality **PET** images from low-dose acquisitions. **Convolutional Neural Networks (CNN)** and **Fully Convolutional Networks (FCN)** have demonstrated an exceptional ability to enhance image quality by reconstructing images. However, the practical deployment of such models on **CPU** and **GPU** presents challenges, including efficient inference on resource-constrained hardware, edge computing capabilities, and power efficiency. **FPGA** architectures provide a viable platform that more readily meets these criteria, delivering acceptable performance, accuracy, and reduction in power consumption.

This thesis is motivated by the need to investigate deep learning accelerators that enable **PET** image denoising from low-dose acquisitions while supplying adequate performance and lower power consumption. By addressing these challenges, this research aims to improve patient safety, reduce diagnostic computing overhead, and encourage broader adoption of low-dose **PET** imaging in clinical and diagnostic settings.

1.2 Thesis Objectives

This thesis aims to develop software and hardware accelerators for denoising low-dose **PET** images to a quality comparable to a standard dose. The research begins with the implementation of a software-based model, accelerated both on **CPU** to establish a baseline for evaluating performance. To enhance inference speed, the deep learning model will be then deployed onto **GPU**, which serves as a basis of comparison to accelerators currently in use for academic and clinical settings.

Following the **GPU**-accelerated implementation, the focus will shift towards adapting the deep learning model for hardware compatibility. This involves refining the architecture to fit the constraints of specialized hardware platforms, particularly **FPGA**. This hardware-adapted model is then deployed for emulation using a bit-accurate model. The results demonstrate the progression and trade-offs between speed, accuracy, and power efficiency between acceleration architectures.

1.3 Contributions

This thesis explores, for the first time to our knowledge, the acceleration of low-dose **PET** image denoising using **FPGA** hardware. While deep learning models have been widely used for **PET** image enhancement, their deployment on resource-efficient hardware remains largely unexplored.

This research presents a structured pipeline, beginning with a software-based model evaluated on **CPU** and **GPU**, followed by an optimized hardware-compatible adaptation for **FPGA** deployment. The research examines trade-offs in performance, accuracy, and energy efficiency, demonstrating the feasibility of **FPGA**-based **PET** denoising for low-power, edge-computing applications.

1.4 Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary background on PET technology, imaging and reconstruction techniques. It follows with an exploration of FCN architectures, FPGA development tools, numerical precision formatting, and model quantization. Chapter 3 reviews current literature on FPGA acceleration in medical imaging, deep learning for PET denoising, and identifies key research gaps, concluding with a problem statement. Chapter 4 details the software implementation, comprising of dataset selection, preprocessing, proposed model architecture, training environment, and evaluation metrics. Chapter 5 discusses FPGA deployment using Intel's OpenVINO and FPGA AI Suite development toolset. Chapter 6 demonstrates the visual results of the experiment, comparing performance, latency, power consumption, and resource utilization across CPU, GPU, and FPGA platforms. Finally, Chapter 7 presents conclusions and potential future research directions.

1.5 Publications

E.-K. Dao, K. Zukotynski, S. E. Black, and V. Gaudet, "Hardware-Compatible U-Net for Low-Dose PET Reconstruction," in Proc. 55th Int. Symp. Multiple-Valued Logic (ISMVL), Montreal, Canada, June 2025, pp. *in press*.

Chapter 2

Background & Technical Foundation

This chapter provides the foundational background and technical concepts necessary to contextualize this thesis. It presents an overview of the [PET](#) imaging process, introduces [FCN](#) architectures in machine learning, and explores Intel's [FPGA](#) development toolset. Additionally, it discusses numerical formats for computing and the fundamental concepts used in model quantization. These topics collectively establish the theoretical and practical foundation required to understand the methodologies and contributions of this thesis.

2.1 Positron Emission Tomography

2.1.1 Physical Principles

Radioactive isotopes of elements spontaneously decay, often producing energy in the form of charged particles or photons. Radioactive molecules, being naturally unstable, decay to a more stable configuration, often releasing excess energy in the process. The rate of decay is characterized by a half-life, which defines the time required for half of a radioactive sample to decay. Depending on the type of emitted radiation, radioactive decay can be classified into one of five categories:

1. **Alpha (α) decay** – An unstable nucleus releases a particle consisting of two protons and two neutrons, leading to a reduction in both atomic and mass numbers.

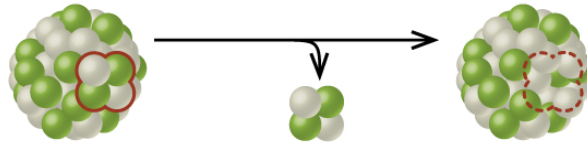


Figure 2.1: Alpha Decay, adapted from [6]

2. **Beta (β) decay** – A neutron is converted into a proton, accompanied by the emission of an electron to maintain charge balance within the isotope.

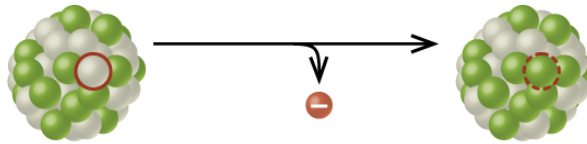


Figure 2.2: Beta Decay, adapted from [6]

3. **Gamma (γ) decay** – A nuclide releases excess energy in the form of high-energy photons called gamma rays. This process only releases energy and therefore does not change the atomic number or mass number serving only as a mechanism to stabilize the nucleus.

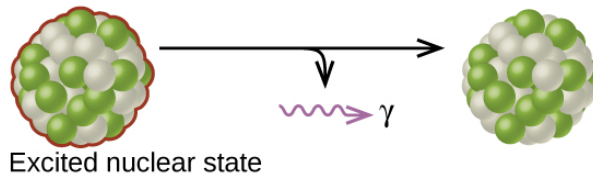


Figure 2.3: Gamma Emission, adapted from [6]

4. **Electron Capture (EC)** – An inner orbital electron is absorbed by the nucleus, where it combines with a proton to form a neutron. This process also decreases the atomic number by one and results in the emission of characteristic X-rays.

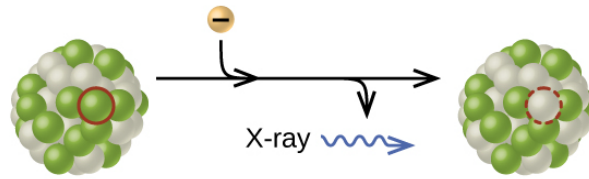


Figure 2.4: Electron Capture, adapted from [6]

5. **Positron Emission (β^+ decay)** – A special form of Beta decay, where a proton within the nucleus is transformed into a neutron while emitting a positron (e^+) and a neutrino (ν_e). This results in a decrease in atomic number by one.

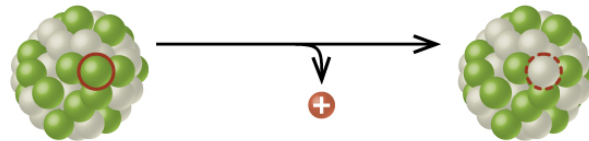
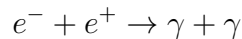


Figure 2.5: Positron Emission, adapted from [6]

If the emitted positron comes into contact with an electron, with a specific orientation, they annihilate each other, converting their mass to pure energy via two gamma photons:



These gamma photons are released with 511 keV of energy, which are emitted 180 degrees away from each other, following the principle of conservation of energy and momentum. Since these gamma photon emissions coincide at 180 degrees, it is possible to capture the pair of photons, and localize them to a certain point in space using time-of-flight imaging.

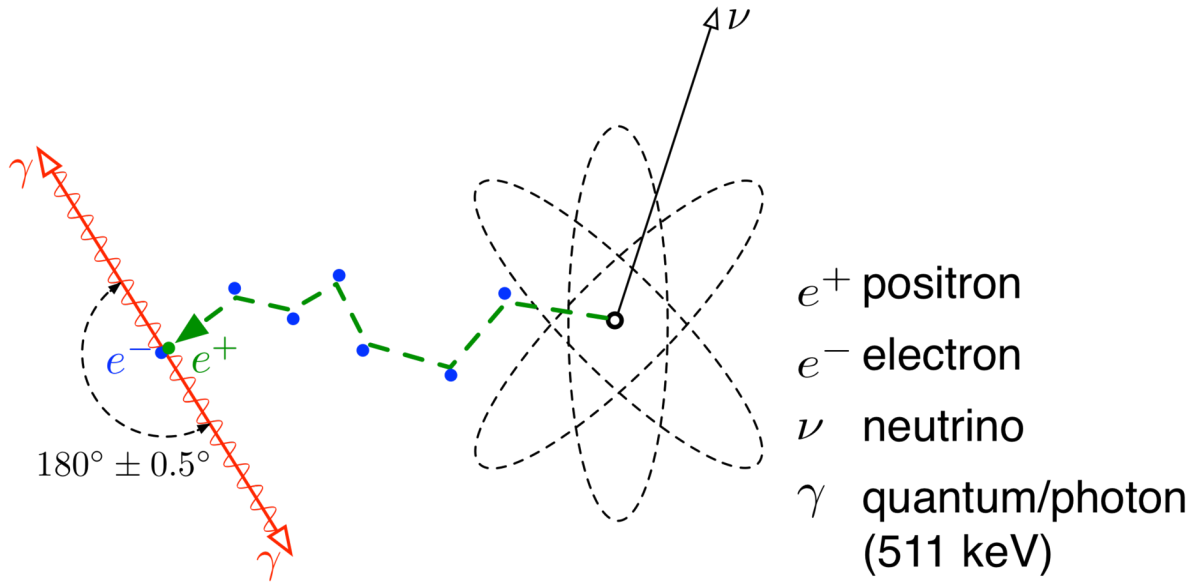


Figure 2.6: Positron Electron Annihilation, adapted from [23]

2.1.2 Radiotracers for PET Imaging

Radiotracers are a group of pharmaceutical drugs that contain radioactive isotopes, serving as a fundamental component in PET imaging. They enable the visualization of metabolic and physiological processes within organisms, which allow for assessment of bodily function, and diagnoses of disease. Specifically within PET imaging, radiotracers are often synthesized using a device called a cyclotron. The cyclotron uses controlled electromagnetic fields to propel charged protons in a circular motion. With each revolution, the protons undergo extreme acceleration and are eventually expelled from the cyclotron, aimed at a target material. The fast-moving protons contain enough energy to replace existing atoms when bombarded into the target material. This process causes the target material to become radioactive, with its specifications (half-life, potency, etc) resulting from a combination of the cyclotron parameters, particle speed, and target material.

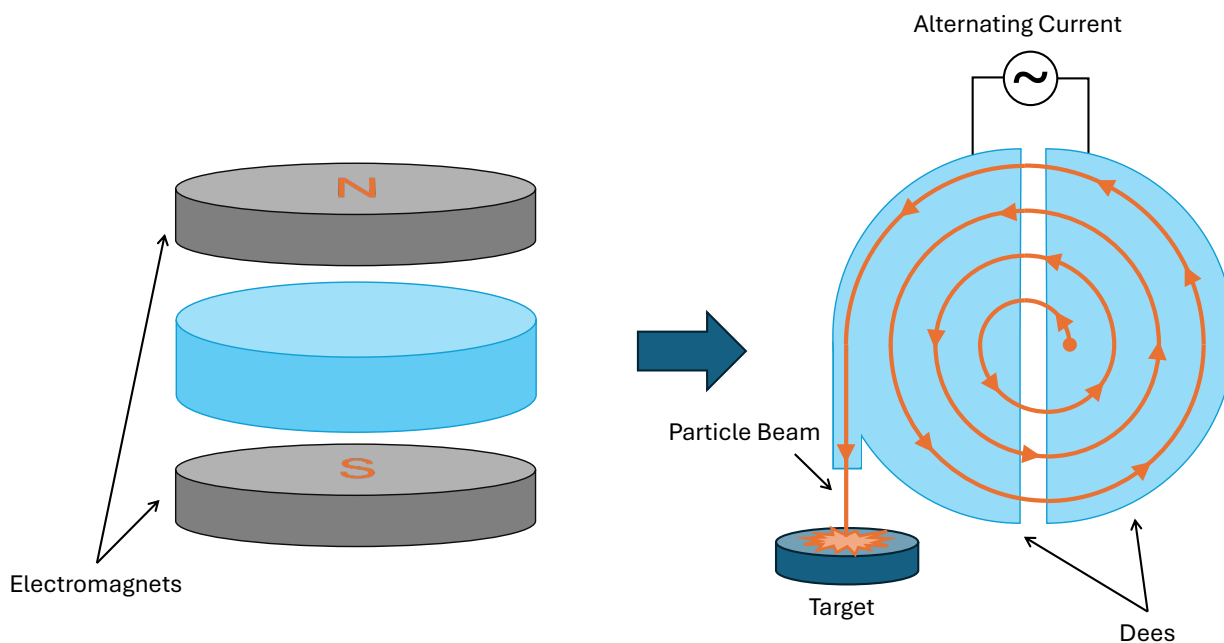


Figure 2.7: Diagrammatic Representation of a Cyclotron

Fluorodeoxyglucose (FDG), the most commonly used radiotracer in PET imaging for cancer diagnosis, is synthesized using the cyclotron. The beam of charged protons bombard oxygen-18 rich water [$H_2^{18}O$]. Upon collision, high-speed protons replace neutrons within the oxygen-18 atoms, creating a radioactive isotope of fluorine known as fluorine-18 [^{18}F]. ^{18}F ions are suspended in solution, which are extracted and purified for use in radiotracer synthesis.

FDG can then be synthesized by combining the extracted ^{18}F ions with a glucose-like molecule. The resulting molecule resembles glucose, except with one of its hydroxyl groups replaced by the fluorine-18 atom. It is shortly thereafter injected intravenously into the patient, as FDG's half-life is approximately only 110 minutes long. The body breaks down FDG in a similar fashion to regular glucose - resulting in the body's glucose transport chain moving the radiotracer to areas in the body with high metabolic activity, such as tumour sites, the brain, heart, or areas with infected tissue. Due to the presence of the ^{18}F , the body can no longer break FDG via glycolysis, leading to metabolic trapping of the FDG, where an accumulation of radiotracer in the target tissue is present.

2.1.3 Imaging Technology

Upon the decay of **FDG** within the body, it undergoes positron emission, as described in Section 2.1.1. The emitted positron subsequently annihilates with an electron, producing two gamma photons that travel in opposite directions, 180 degrees apart. These photons are then detected by the scanner and processed for diagnostic purposes.

The core technology in modern **PET** scanners consists of a ring of scintillator crystals that emit visible light when exposed to ionizing radiation. During a **PET** scan, the patient is positioned inside this ring, where gamma photons interact with the crystals at opposite ends. The emitted light is transmitted to photomultipliers, which convert it into electrical signals for analog or digital processing by a computer. When an annihilation coincidence event occurs, the scintillator crystals detect the photons at slightly different times, assuming the event is not perfectly centred within the ring. By analyzing the time difference between photon arrivals, the precise location of the coincidence event can be determined. Depending on factors such as the type of radiotracer used, scan duration, scanner size, and the region of the body being imaged, hundreds of millions to billions of coincidence events are recorded. Computational algorithms then process these counts to reconstruct a detailed, human-interpretable image.

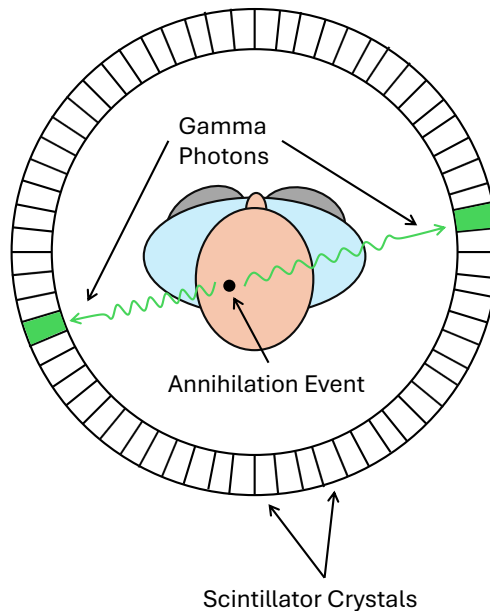


Figure 2.8: Axial Diagrammatic View of PET Scanner

2.1.4 Reconstruction

After acquiring all event counts during a PET scan, the raw data and their corresponding locations are processed by a reconstruction algorithm to generate an image that can be analyzed by clinicians. Reconstruction algorithms generally fall into two categories: analytic and iterative.

Analytic reconstruction methods attempt to construct the entire image at once using the raw event counts. After reconstruction, the image undergoes filtering to reduce unwanted noise. A common example of this approach is [Filtered-Back Projection \(FBP\)](#). However, a major drawback of analytic methods is that they do not inherently account for common distortions in PET imaging, such as attenuation and scatter. These methods assume that corrections for these distortions have already been applied, increasing the preprocessing burden and setup time for a PET scan [12].

- **Attenuation** occurs when emitted gamma photons are absorbed or blocked by tissue or bone, causing the detectors to register less metabolic activity than what is present. [12]
- **Scatter** occurs when gamma photons deviate from a perfect 180-degree alignment due to interactions with tissue or bone, leading to an incorrect [Line of Response \(LOR\)](#) and reduced image accuracy [12].

In contrast, iterative reconstruction methods, such as [Line-of-Response Row-Action Maximum-Likelihood Algorithm \(LOR-RAMLA\)](#), refine an initial estimate of the image by continuously comparing it to the measured data. This process is repeated using a cost function and optimization algorithm, gradually improving the accuracy of the reconstructed image. One of the key advantages of iterative methods is their built-in correction for attenuation and scatter, which are incorporated directly into the mathematical model governing the reconstruction process [12].

Additionally, [LOR-RAMLA](#) operates row by row, reducing computational complexity compared to analytic methods, which attempt to reconstruct the entire image simultaneously. This makes iterative techniques more adaptable and effective in producing high-quality PET images, particularly in cases where noise and distortions are significant [12]. Given these challenges, there is growing interest in leveraging neural networks for image reconstruction. In the next section, the progression of convolutional neural networks will be explored to understand their potential in this context.

2.2 Fully Convolutional Network Architectures

Early deep learning models for medical imaging were primarily based on CNNs and were initially designed for image classification tasks. Architectures such as LeNet-5 [20], AlexNet [19], and VGGNet [33] demonstrated significant advancements in automated medical diagnostics by classifying entire medical images such as tumour vs. non-tumour or benign vs. malignant lesions. These networks followed a structured pipeline where the input images are processed through multiple convolutional layers that extract spatial features, followed by activation and pooling layers that introduce non-linearity and increase receptive field, respectively. The final classification is performed using fully connected layers, which map the extracted features to specific diagnostic labels [30].

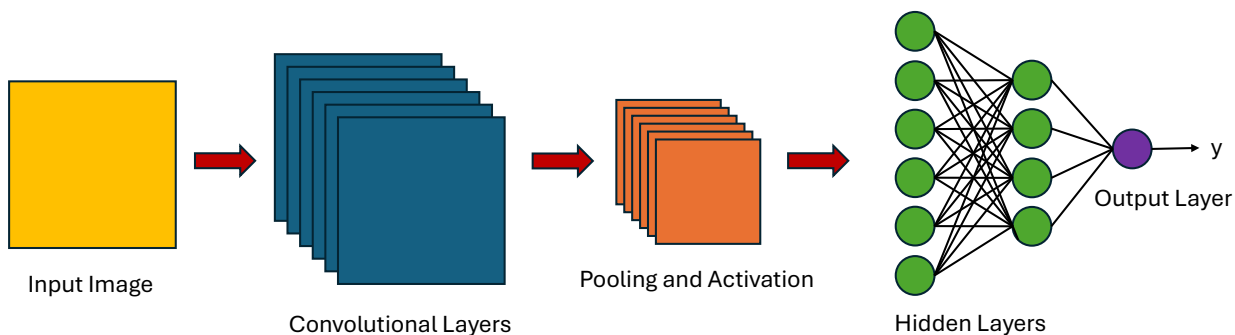


Figure 2.9: General Structure of a CNN Classification Network

The major drawback of classification models was that they provide only a single label per image, without any information on precise disease location. This lack of spatial awareness creates challenges for radiologists and doctors to precisely pinpoint abnormalities, limiting their usefulness for tasks that require detailed lesion localization and segmentation.

To address these shortcomings, advancements in CNN architectures led to the development of models capable of segmentation, enabling more detailed analysis of medical images. By integrating region-based processing techniques and pixel-wise classification approaches, researchers designed networks that could not only identify the presence of diseases but also localize their spatial extent.

2.2.1 Fully Convolutional Networks

To overcome the spatial limitations of classification networks, FCNs were introduced for pixel-wise classification, enabling semantic segmentation of medical images. FCNs, first proposed in 2015 by Long *et al.* [22], removed the fully connected layers from contemporary classification models, and replaced them with convolutional layers and transposed convolutions. This architecture's structure mimics that of the encoder-decoder architecture, as proposed by Sutskever, *et al.* [34]. This allowed FCN to output full segmentation maps instead of a single classification label, making them ideal for medical imaging tasks such as tumour/organ segmentation, and blood vessel localization.

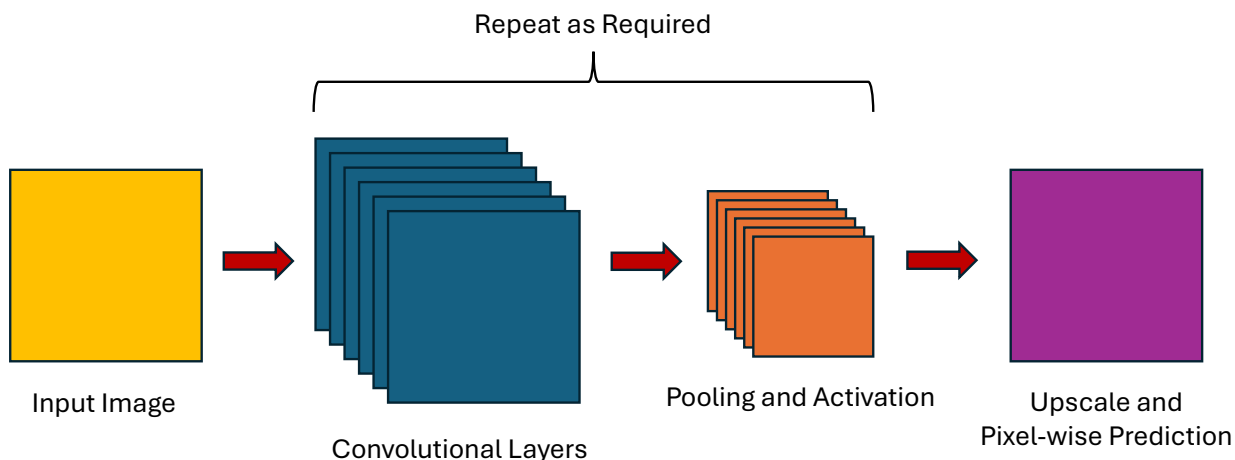


Figure 2.10: General Structure of a Fully Convolutional Network

One of the most influential FCN-based architectures in medical imaging is the U-Net, proposed by Ronneberger, *et al.*, which was specifically designed for biomedical image segmentation. U-Net introduced skip connections that helped retain fine-grained details from early convolutional layers while capturing high-level abstract features [31]. This improvement significantly enhanced segmentation accuracy in applications like cell segmentation in microscopy [31], organ segmentation in MRI/CT scans [18], and lung nodule detection in CT scans [41].

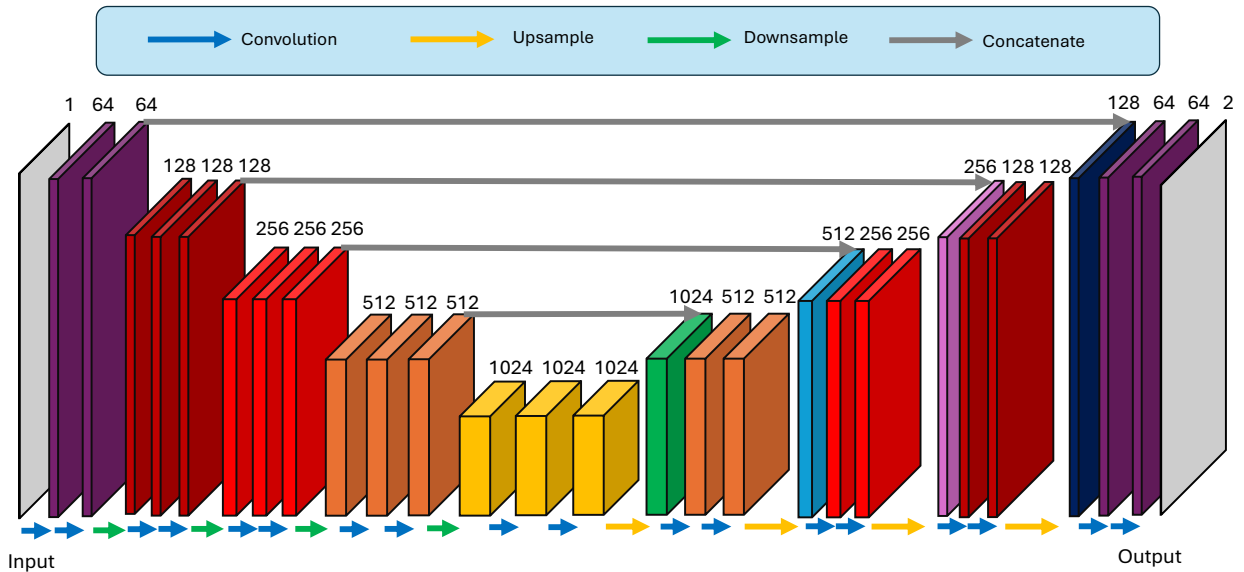


Figure 2.11: Original U-Net Architecture as Proposed by [31]

FCNs revolutionized medical imaging by providing detailed spatial localization, enabling automated segmentation tools that could assist radiologists in diagnosing and tracking diseases over time. However, traditional FCNs were only able to provide pixel-wise classification of images, which proved to be a limitation in tasks involving image denoising, reconstruction, and synthesis.

2.2.2 Fully Convolutional Residual Networks

Building on FCNs and the U-Net, Zhang, *et al.* introduced residual connections, leading to Fully Convolutional Residual Network (FCRN) [46], commonly known as ResUnet. Inspired by ResNet, these networks mitigated the degradation problem in deep architectures by incorporating skip connections that bypass multiple layers [17]. A key advantage of applying residual deep learning to FCN is its effectiveness in image denoising, as demonstrated by Zhang, *et al.* [45]. Their approach predicts the residual noise pattern rather than the restored image itself, allowing for more efficient noise removal. However, FCRN has immense computational demands of FCRN, with the vast majority of implementations relying on CPU and GPU for acceleration. In the next section, the focus will shift to introducing tooling and methodology used in leveraging FPGA as a deep neural network accelerator.

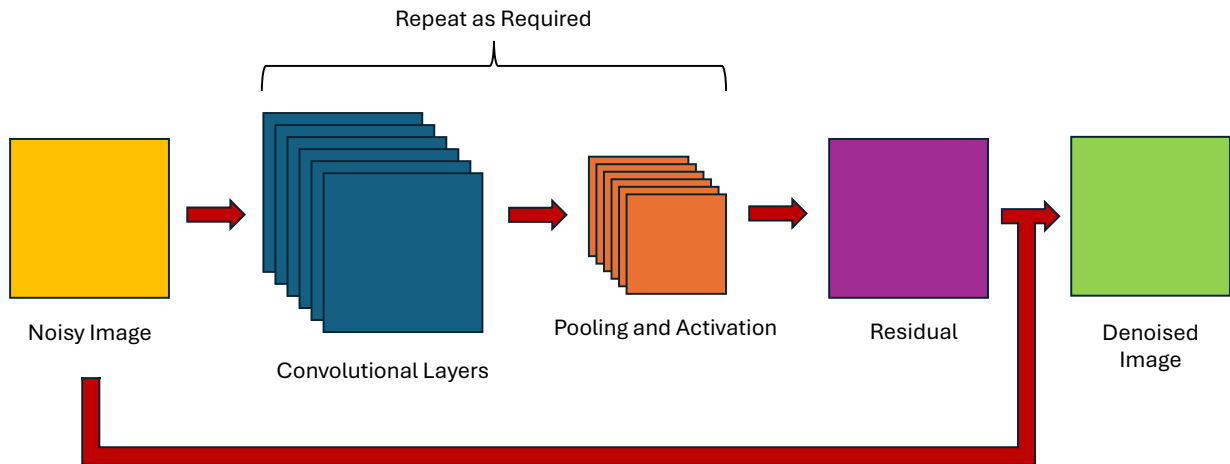


Figure 2.12: Original DNCNN Architecture as Proposed by [45]

2.3 Intel OpenVINO and FPGA AI Suite

The [Intel FPGA AI Suite \(IFAS\)](#) is a comprehensive toolset designed to accelerate machine learning workloads on Intel [FPGAs](#), specializing in embedded and edge computing environments. It has supported integration with the Intel OpenVINO Toolkit, a widely used framework for optimizing and deploying deep learning models across heterogeneous computing platforms. The OpenVINO Toolkit provides an [Intermediate Representation \(IR\)](#) of deep learning models, enabling seamless model conversion, quantization, and optimization for [FPGA](#) deployment. This integration allows development on deep learning frameworks such as Meta’s PyTorch and Google’s TensorFlow as contrasted with traditional methods such as [High Level Synthesis \(HLS\)](#)[8].

[IFAS](#) includes support for numerous architectures within the Intel [FPGA](#) product lines, such as Stratix, Agilex, and Arria families. It also incorporates support for [Digital Signal Processor \(DSP\)](#) blocks optimized for tensor-based calculations, fundamental operations underlying deep learning. Additionally, it provides a streamlined flow for deep learning model deployment, allowing developers to specify device resources, [DSP](#) utilization, and throughput requirements, thereby generating an optimized [FPGA](#) implementation tailored to the target hardware and end-use case [8]. However, efficiently mapping deep learning models onto [FPGA](#) requires consideration of numerical precision and formatting. In the next section, machine learning quantization techniques and their impact on [FPGA](#)-based deep learning acceleration will be discussed.

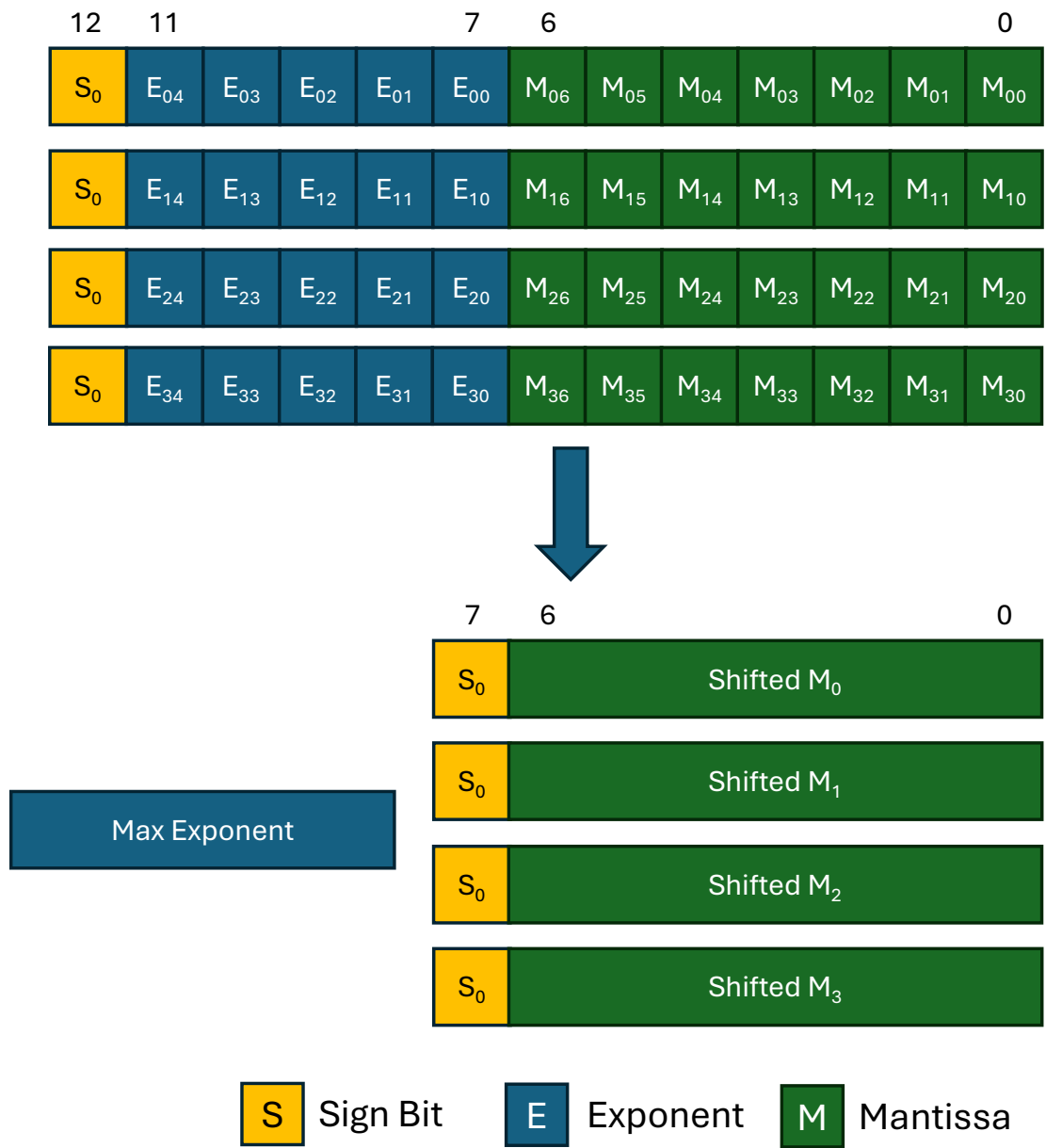


Figure 2.16: Packing Structure of INT7BFP

2.4.2 Quantization Principles

The core operation in deep learning accelerators is [Multiply-and-Accumulate \(MAC\)](#) operations, defined as:

$$A = \sum Wx + b, \tag{2.1}$$

Where W is the weight, x is the input, and b is a bias term, which is all typically represented as FP32. This operation accounts for the vast majority of the computations within deep learning accelerators. Therefore, using lower-bit representations such as fixed-point substantially reduces the computational load, and henceforth lowers energy consumption. To achieve this, quantization attempts to map floating-point values to fixed-point values by approximating the floating-point value as the product of some quantized vector x_{quant} (typically an integer) and scaling factor s_x as follows:

$$\hat{x} = s_x \cdot x_{quant} \approx x, \tag{2.2}$$

The most common form of quantization is known as [Uniform Quantization \(UQ\)](#), which linearly maps floating-point values onto a grid of integers, each floating-point value equidistant from every other value. The simplest form of [UQ](#) is known as [Symmetric Uniform Quantization \(SUQ\)](#). The process involves defining three parameters: the scale factor s , zero-point z , and bit-width b . The scale factor is calculated as:

$$s = \frac{2^{b-1} - 1}{\beta} \tag{2.3}$$

where β represent the absolute maximum and minimum of input x , and b is the bit width. The zero-point in [SUQ](#) is assigned to 0, eliminating to account for zero-point offset during de-quantization. However, this technique reduces the efficiency of the mapping, and forces the user to consider the trade-off between a signed or unsigned mapping, which affects the relation that determines x_{quant} :

$$x_{quant} = \text{clamp}(\text{round}(\frac{x}{s}) : 0, 2^{b-1} + 1) \text{ for unsigned} \tag{2.4}$$

$$x_{quant} = \text{clamp}(\text{round}(\frac{x}{s}) : -2^{b-1} + 1, 2^{b-1} + 1) \text{ for signed} \tag{2.5}$$

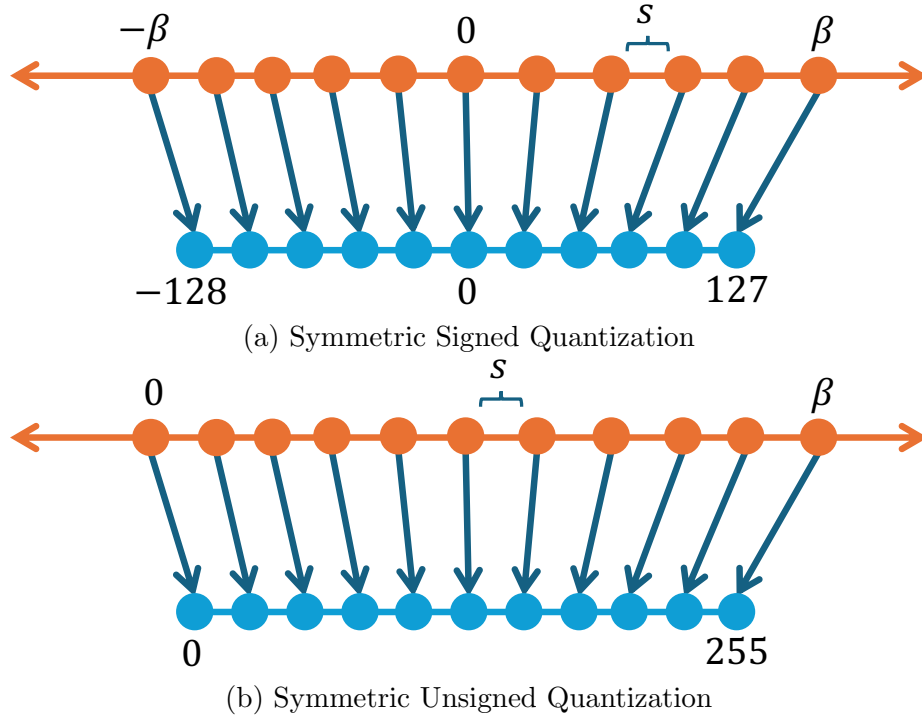


Figure 2.17: Visualization of symmetric quantization; floating-point values (orange line) are mapped onto the integer grid (blue line). s denotes scale factor, β denotes maximum of floating-point input.

Asymmetric Uniform Quantization (AUQ) quantization requires the same parameters as **SUQ**, with the scale and zero-point redefined. The scale is redefined as:

$$s = \frac{2^b - 1}{\alpha - \beta} \quad (2.6)$$

where α and β are the minimum and maximum of the input x , respectively. The zero point is defined as:

$$z = -\text{round}(\beta \cdot s) - 2^b - 1 \quad (2.7)$$

Due to the presence of a non-zero zero-point, the input mapping becomes:

$$x_{quant} = \text{clamp}(\text{round}(\frac{x}{s} + z) : -2^b - 1, 2^{b-1} - 1) \quad (2.8)$$

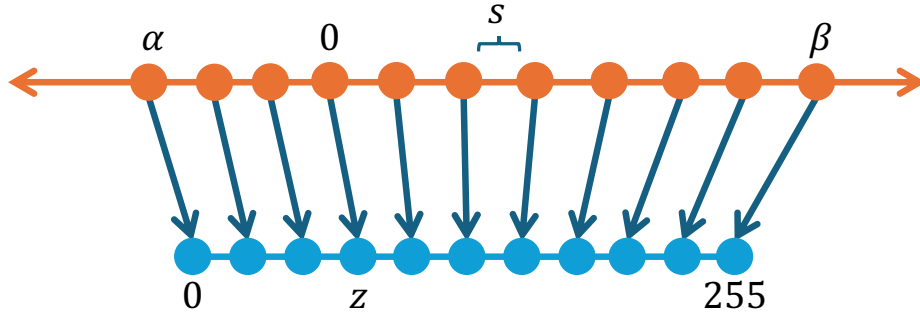


Figure 2.18: Visualization of asymmetric quantization; floating-point values (orange line) are mapped onto the integer grid (blue line). s denotes scale factor, z denotes the zero-point, $[\alpha, \beta]$ denotes minimum and maximum of floating-point input, respectively.

Following the quantization operation and performing inference, the results are de-quantized before exiting the deep learning model. Since the scale factor and zero-point are stored along with the quantized weights, de-quantization for [AUQ](#) simply subtracts the zero-point offset and inverts the scale:

$$x_{dequant} = \frac{x_{quant} - z}{s} \quad (2.9)$$

Similarity for [SUQ](#):

$$x_{dequant} = \frac{x_{quant}}{s} \quad (2.10)$$

2.5 Conclusion

In this chapter, we provided an overview of the fundamental concepts necessary to understand the scope of this thesis. We introduced the principles of [PET](#) imaging technology, discussed [FCN](#) architecture, and explored Intel’s OpenVINO and [FPGA](#) AI Suite toolset. Additionally, we highlighted the impact of model quantization on performance and efficiency. These topics establish the technical groundwork for the subsequent chapters, where the detailed implementation and evaluation of the proposed approach are presented.

Chapter 3

Literature Review

This chapter provides a comprehensive review of the existing literature on hardware acceleration for deep learning in medical imaging. It begins by examining the application of **FPGA** acceleration in various imaging modalities, including **Magnetic Resonance Imaging (MRI)** and **Computed Tomography (CT)**. This is followed by an analysis of research focused on deep learning techniques for enhancing low-dose **PET** imaging. Finally, the intersection of these two domains is explored, identifying the gap in **FPGA**-accelerated deep learning for low-dose **PET** denoising and presenting a formal problem statement to address this research challenge.

3.1 **FPGA Acceleration for Medical Image Processing**

Saglam, *et al.* [32] investigated the implementation of lightweight binary classification algorithms in **FPGA** for medical image analysis, focusing on malaria detection from blood cell images. The experiment explores hardware-optimized implementations of k-Nearest Neighbors (k-NN), **CNN**, and Decision Tree classifiers, evaluating their performance on a publicly available malaria dataset, and carried out on a Xilinx Zynq SoC **FPGA**. They concluded that **FPGA**-based classification significantly outperforms software simulations, achieving processing speeds thousands of times faster while reducing resource utilization. Compared to existing studies, the proposed designs require 73.9% fewer **Look-Up Tables (LUT)** for k-NN, 57% fewer for CNN, and 96.7% fewer for the Decision Tree classifier. The Decision Tree achieved the highest accuracy (99.33%), followed by CNN (97.67%) and k-NN (95.33%). The results highlight the feasibility of **FPGA**-based medical image classification when optimized.

Othman, *et al.* [5] proposed an FPGA-based MRI brain image classification for detection of abnormal brains. They used SVM with an RBF kernel, achieving a 65% accuracy. Wavelet-based feature extraction was applied to MRI scans from 32 patients (22 abnormal, 10 normal). FPGA implementation on the Altera Cyclone II with a NIOS II processor demonstrated advantages in parallel processing and real-time execution over CPUs. However, the experiment noted SVM’s sensitivity to dataset size, suggesting the use of techniques such as K-means or hierarchical clustering for improved classification.

Dillinger, *et al.* [13] presents an FPGA-based real-time 3D segmentation for MRI scans, using the 3D Grey-Value Structure Code (3D-GSC) algorithm. The hardware implementation was performed on a Xilinx Virtex-II Pro FPGA board equipped with external memory that reached speeds of 6.4 GB/s. The paper reports a 98.5% voxel classification accuracy in simulated MRI datasets. It was concluded that hardware implementations prove to have an estimated runtime latency reduction between 30-100x. Further proposed improvements included upgrading the FPGA platform to a Xilinx Virtex-4, which boasts a 2x increase in programmable memory and introduces specialized MAC DSP [3], a basic component used in hardware accelerators [29].

Xiong, *et al.* [42] investigated the implementation of an FPGA-based accelerator for 3D brain tumour segmentation, using a CNN FCN architecture called the U-Net [31]. The experiment evaluated the performance and power efficiency of the proposed FPGA implementation using the BraTS19/20 datasets and accelerated using the Xilinx Alveo FPGA platform. BraTS is a brain tumour segmentation competition, where proposed machine learning models attempt to achieve the highest score [24]. They concluded that FPGA-based segmentation significantly outperforms traditional CPU and GPU processing, achieving speedups of 5.21x and 44.47x, respectively, while improving energy efficiency by factors of 11.22x and 82.33x. The results highlight FPGA’s effectiveness in 3D brain tumour segmentation by reducing computational load and power consumption.

Neiso, *et al.* [25] investigated an optimized U-Net model for FPGA-based inference in 2D brain tumour segmentation using the BraTS20 dataset. The model was modified with reduced depth and filters to improve efficiency on FPGA hardware. It was implemented using HLS for Machine Learning (HLS4ML), an open-source framework used to convert software-based machine learning models to Hardware Description Language (HDL) [14], targeting the Xilinx Kintex Ultrascale FPGA. Optimizations included FIFO depth adjustment, leading to a 55% reduction in Block RAM (BRAM) usage, a 43% decrease in Flip-Flops (FF), and a 49% reduction in LUT. In C/Register-Transfer Level (RTL) co-simulation, their model achieved an Intersection over Union (IoU) score of 74% similarity between software and hardware models. The results highlight the feasibility of the optimized U-Net models for efficient 2D brain tumour segmentation on FPGA platforms.

3.2 Deep Learning Low-Dose PET denoising

Wang, *et al.* [38] proposed a 3D cGAN-based framework for estimating high-quality full-dose PET images from low-dose inputs. They make use of **Generative Adversarial Network (GAN)**, a machine learning framework proposed in 2014 by Goodfellow, *et al.* It involves two neural networks, named generator and discriminator. The generator network attempts to synthesize data, and the discriminator network attempts to distinguish synthesized data from real data, improving over repeated iterations [16]. A 3D U-Net-like architecture was used as the generator to combine hierarchical features, while a concatenated GAN-based progressive refinement scheme further enhanced image quality. To address small sample size limitations, they extracted 125 large 3D patches per subject, increasing training samples from 16 images to 2000 patches, and adopted a **Leave-One-Out (LOO)** strategy for model evaluation. Experimental results on brain PET scans demonstrated that their approach outperformed benchmark methods for both normal and **Mild Cognitive Impairment (MCI)** subjects, generating full-dose PET images with improved quantitative accuracy. The method effectively preserved **Standard Uptake Values (SUV)** in specific regions of interest, meeting the requirements for reducing radiotracer dosage in PET scans. Future work includes integrating multi-modal data from PET/CT and PET/MRI and extending the framework to other body regions and pediatric imaging.

Zhou, *et al.* [47] proposed a supervised deep learning model, CycleWGANs, to enhance low-dose PET image quality. The experiment was conducted on a low-dose dataset simulated from real PET scans of patients with biopsy-proven primary lung cancer or suspicious radiological abnormalities. Low-dose PET images were reconstructed by randomly discarding events in PET list-mode data to a count level of 1 million. The model was compared against traditional denoising methods (Non-Local Mean (NLM) and BM3D) and deep learning approaches (RED-CNN and 3D-cGAN). Results demonstrated that CycleWGANs outperformed other methods in estimating full-dose PET images, achieving a lower mean and max SUV for both lesions and normal tissue; $(-2.06 \pm 3.50\%, -0.84 \pm 6.94\%)$ and $(-0.45 \pm 5.59\%, N/A)$, respectively. However, RED-CNN achieved the highest scores in **Structural Similarity Index (SSIM)**, **Peak Signal-to-Noise Ratio (PSNR)**, and **Normalized Root Mean Squared Error (NRMSE)**. Further correlation and profile analyses suggested that CycleWGANs better-preserved edge details and SUV values compared to other methods, albeit with slightly higher noise.

Xiang, *et al.* [40] proposed a deep auto-context CNN architecture for estimating standard-dose PET images from low-dose PET and T1-weighted MRI data. Their method integrates multiple CNN modules, following an auto-context strategy where early-stage standard PET estimations are iteratively refined by subsequent CNN. Their model elim-

inates patch representation and non-linear mapping, enabling direct end-to-end learning without pre/post-processing. Validations on real human brain PET/PET data demonstrated that their approach provides competitive PET image estimation quality while significantly improving efficiency, generating standard PET images in approximately 2 seconds compared to 16 minutes with state-of-the-art methods, achieving a speedup of up to 500x.

Cui, *et al.* [11] proposed an unsupervised deep learning framework for PET image denoising. Unlike conventional supervised approaches, their method does not require paired training data. The network takes a PET image as input while using the noisy PET image as the training label, allowing it to learn the intrinsic structure and output a restored image. The model was validated through simulations using the BrainWeb phantom and clinical evaluations on PET/CT and PET/MRI datasets. Compared to other methods such as Gaussian, non-local mean (NLM), BM4D, and Deep Decoder methods, their approach achieved the highest Contrast-to-Noise (CNR) ratio improvements, with a 53.35% increase for PET/CT and 46.80% for PET/MRI. Results highlight the potential for unsupervised learning in denoising of PET images.

Liu, *et al.* [21] proposed a deep learning model to enhance the SNR ratio of PET images using MRI images, without requiring higher SNR PET images for training. The model consists of three modified U-Nets (3U-Net), with PET training data reconstructed using FBP and target images generated through Maximum Likelihood Expectation Maximization (MLEM). Digital brain phantoms from BrainWeb were used for evaluation, with Poisson noise added to simulate a 6-minute brain PET scan. Experiments demonstrated that training with both PET and MRI data reduced the Mean Squared Error (MSE) by 31.3% for 1U-Net and 34.0% for 3U-Net, which corresponds to a 2.5x and 2.9x increase in count levels, respectively. The lesion CNR ratio was also improved by 2.7x for 1U-Net and 1.4 times for 3U-Net compared to MLEM images. These results suggest that the proposed method effectively improves PET SNR without the need for higher SNR PET images.

Xu, *et al.* [43] proposed a deep learning approach to reconstruct standard-dose PET images from ultra-low-dose scans, achieving an unprecedented Dose Reduction Factor (DRF) of 200—the first study to demonstrate such a substantial reduction in radiotracer dosage. Their method is inspired by the U-Net, using a fully convolutional residual network with concatenated skip connections. It incorporates a multi-slice input strategy, allowing the network to leverage additional structural information and enhance robustness to noise. Evaluations demonstrate that the method outperforms state-of-the-art techniques, achieving results comparable to standard-dose PET images with only 0.5% of the original dose. The proposed network significantly reduces noise while preserving image resolution and structural details, marking a breakthrough in ultra-low-dose PET imaging.

3.3 Research Gaps and Problem Definition

Despite significant advancements in both **FPGA**-based medical imaging and deep learning for **PET** denoising, there has been extremely limited research at the intersection of these two fields. **FPGA** technology has been widely explored for accelerating medical imaging applications due to its high computational efficiency, low power consumption, and real-time processing capabilities. Concurrently, deep learning has demonstrated remarkable success in improving **PET** image quality, enabling lower-dose acquisitions while preserving diagnostic integrity. However, current deep learning approaches for **PET** imaging are predominantly implemented on conventional **GPU**, which, while powerful, often come with high energy consumption and hardware overhead. The potential of **FPGAs** to optimize deep learning-based **PET** image denoising remains largely unexplored.

This thesis aims to bridge this gap by investigating the integration of **FPGA** acceleration into deep learning models for **PET** image denoising. By leveraging **FPGA** hardware for **CNN** acceleration, this research explores the feasibility of achieving **PET** image denoising with adequate performance, latency and reduced power consumption when compared to conventional methods. This research represents an emerging area of research that has the potential to significantly impact the future of low-dose **PET** imaging by enabling resource-efficient, high-performance deep learning techniques.

3.4 Conclusion

In this chapter, a review of the existing literature on **FPGA**-based medical imaging and deep learning for **PET** denoising was presented. The role of **FPGAs** in medical imaging, emphasizing their computational efficiency and real-time processing capabilities was explored. Additionally, deep learning approaches for **PET** image enhancement, particularly in the context of low-dose acquisitions were discussed. A critical gap was identified at the intersection of these two domains, with limited research on **FPGA**-accelerated deep learning for low-dose **PET** denoising. Finally, the motivation for this thesis was outlined, which aims to bridge the gap between these two fields by integrating **FPGA** hardware into deep learning-based **PET** denoising. This proposal offers a novel approach to reducing power consumption and presents opportunities for edge-computing applications.

Chapter 4

Software Implementation of Proposed Model

This chapter discusses, in detail, the selection of the dataset, preprocessing, and the proposed machine learning model. Following, the software implementation of the deep learning model, training environment, hyperparameters, and evaluation metrics are presented.

4.1 Dataset Selection and Preprocessing

4.1.1 Data

The data used for this thesis was provided by the [ADNI](#). [ADNI](#) began in 2004 as an investigative study aiming to slow or stop the progression of [AD](#). Data is collected and tracked over time through MRI/PET scans. These scans coupled with cognitive or neurophysiological tests determine a patient's brain structure and functionality [2].

From the [ADNI](#) database, 10 patients diagnosed with [AD](#) were selected. The average age of this study was 77.5 years, with 7 male and 3 female. Each patient's scan was captured using Philips Medical Systems GEMINI TF TOF 16 scanner. Data acquisition was performed with attenuation correction (CTAC-SG), decay correction (START), and scatter correction (SS-SIMUL) applied. The radiotracer used for imaging was 18F-fluorodeoxyglucose (18F-FDG).

Each scan contains 90 slices, with a pixel spacing of 2.0 mm in both the X and Y directions. The images were reconstructed using [LOR-RAMLA](#), with a 128×128 pixel grid resolution. Additionally, each acquisition consists of six frames, and the slice thickness for the reconstructed images is 2.0 mm.

A reduced dataset provides a worse-case scenario in which sufficient data to train a machine learning model is not available, common to data science on medical imaging [36]. To reduce the variance in data quality, patients were selected by filtering by a specific study that used the same scanner radiotracer, and pixel grid resolution.

4.1.2 Preprocessing

[ADNI](#)'s database stores files via the [Digital Imaging and Communications in Medicine \(DICOM\)](#) format, an international standard used to store and transmit data [37]. Each patient's raw data was extracted using the [PyDICOM](#) library [7], an open-source Python library used to manipulate [DICOM](#) files. The additional 5 frames were removed from each acquisition, leaving 90 slices containing 128×128 2D images. Following the methodology from [43], it was concluded that multi-slice inputs substantially improved the performance of the model. Consequentially, the same technique was adopted for this experiment, organizing each patient into 88 2.5D images. Each 2.5D image contains 3 slices, each window overlapping by one slice.

The [FPGA AI Suite](#) discussed in chapter 5.1 accepts the following image format as input: BMP, PNG, PBM, PGM, PPM, JPEG, JPG, JPE. From these, [bitmap \(BMP\)](#) was selected, an image format developed by Microsoft. [BMP](#) images are uncompressed, have native support of three channels, and store pixel data in an 8-bit.

From the [ADNI PET](#) images, corresponding low-dose images were generated. This was achieved by sampling each slice at a 0.5% sampling ratio following a Poisson distribution, achieving a [DRF](#) of 200x. A Poisson distribution was selected as it most accurately mimics the noise distribution of [LOR-RAMLA](#), the reconstruction algorithm used in the [ADNI](#) database [35]. Following, the 2.5D images were standardized using the Min-Max scaler, which maps the entire range of data between two values. Each image was standardized to integer values between 0 and 255 to be compatible with the [BMP](#) format. All images were then separated into training and validation sets following an 80:20 split, respectively. [Figure 4.1](#) summarizes the preprocessing workflow.

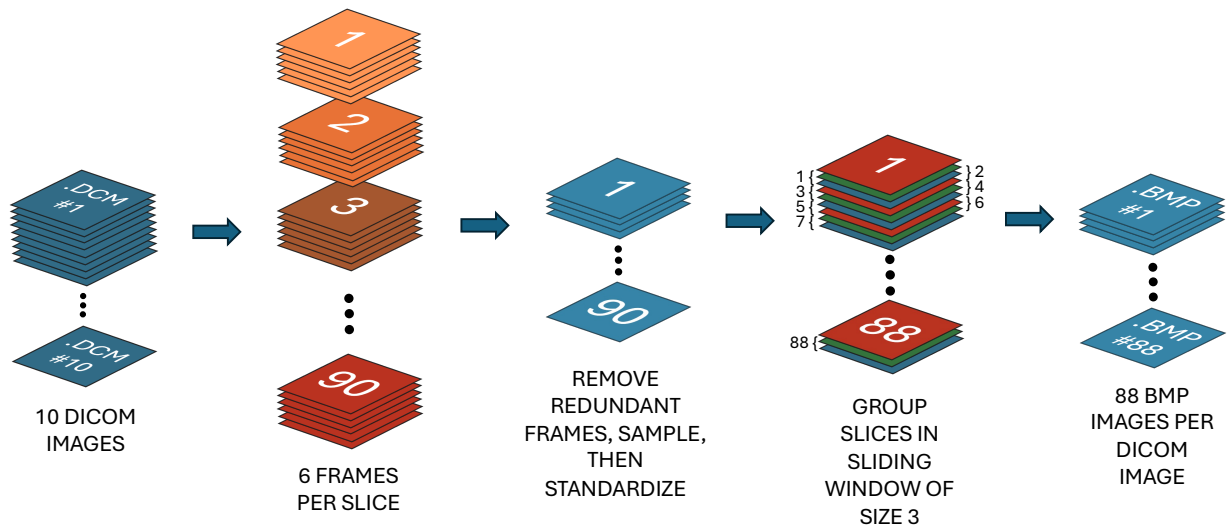


Figure 4.1: Preprocessing Workflow

4.2 Proposed Model Architecture

The proposed architecture is a modified version of the architecture proposed by [43], which heavily inspired from the U-Net, a FCN architecture introduced in 2015. The U-Net is a FCN, that is, a network which only makes use of convolutional layers. It was originally designed for biomedical imaging segmentation, making use of contracting (encoder) and expansive (decoder) data flow paths. It introduced the concept of skip connections, where sections of the decoder are concatenated with the corresponding section of the encoder. This proves to vastly increase the retention of fine-grained details [31].

Since the designated purpose of the original U-Net was for segmentation, modification of the U-Net is necessary for use in image denoising. [43] incorporated the use of Residual Networks (ResNet), a popular machine learning model introduced by Microsoft Research in 2015 [17]. ResNet makes use of residual connections, which are summing connections from an input layer to an output layer. This combination allows for image synthesis as presented in [43].

To adapt the model presented in [43] for the preprocessed ADNI data, the input image shape was modified from an input slice of 256×256 pixels to 128×128 pixels. Due to the decrease in input size, the first convolution layers were modified from 32 channels to 16 channels. Consequentially, the last layer before the residual was also modified from 32 channels to 16 channels. Figure 4.2 presents the proposed architecture.

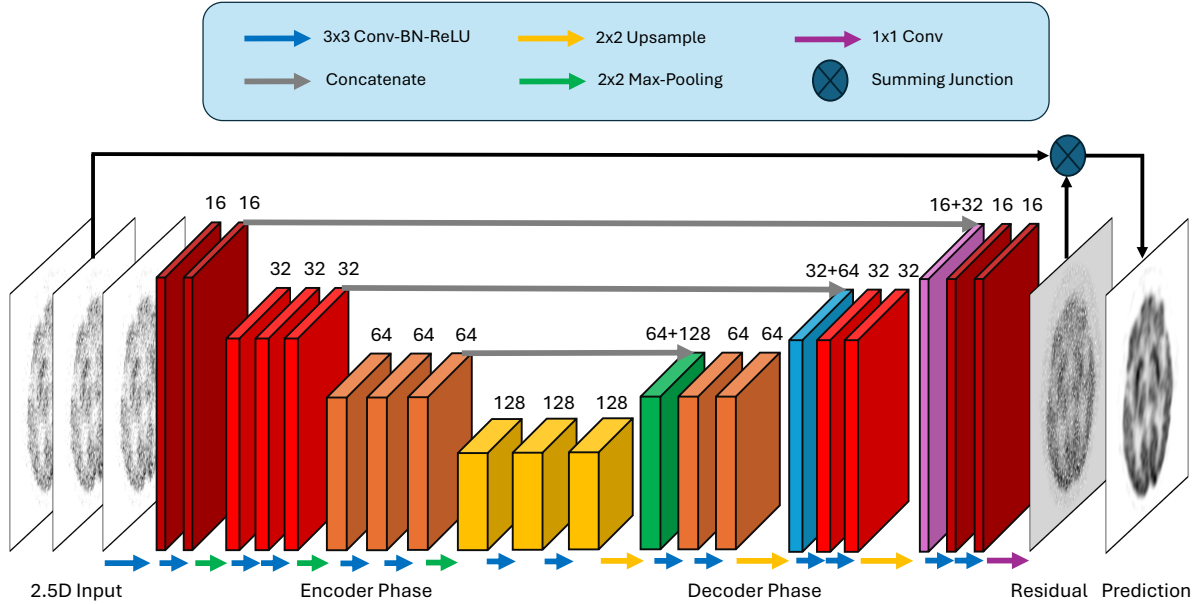


Figure 4.2: Proposed Model Architecture

The 2.5D input first passes through the encoder phase, where a 2×2 max-pool operation is performed after two convolution layers. Each convolution layer uses a 3×3 kernel, a stride of 1, and zero-padding to preserve image size. This process is executed a total of three times, transforming the input tensor from $128 \times 128 \times 3$ to $16 \times 16 \times 128$. The decoder phase follows a similar structure to the encoder phase, however, instead of a 2×2 max-pool, a 2×2 up-sample layer is executed in its place, using the nearest-neighbour heuristic. Figure 4.2 highlights the use of skip connections, where each successive double convolution layer in the encoder phase is concatenated with its corresponding decoder block. At the end of the decoder phase, the data passes through a 1×1 convolution layer, which produces a residual. This residual is summed with the second layer of the 2.5D input to produce the final prediction, with a tensor size of $128 \times 128 \times 1$.

4.3 Training Environment and Hyperparameters

The software model was implemented using the Pytorch framework, a popular machine learning framework developed by Meta [28]. Training was conducted on an Ubuntu 22.04 server, equipped with an Intel 9700F CPU, 16GB of memory, and a NVIDIA GTX 1060 GPU with 6GB of VRAM. The model was trained for 200 epochs, using the L1 loss function, and RMSprop optimizer. As presented in [43], a dynamic learning rate was adopted. The optimizer had an initial learning rate of 1.0×10^{-3} and linearly decayed to 2.5×10^{-4} .

4.4 Evaluation Metrics

To assess the image quality of model predictions, various evaluation metrics are used. The **NRMSE** quantifies the error between two images. To limit the effect of scale, Euclidean normalization is used. **NRMSE** is defined as:

$$\text{NRMSE}_{\text{euclidean}} = \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (I_{\text{true}}(i) - I_{\text{test}}(i))^2}}{\|I_{\text{true}}\|_2} \quad (4.1)$$

where $I_{\text{true}}(i)$ and $I_{\text{test}}(i)$ are the pixel values of the true and test images at pixel i , N is the total number of pixels, $\|I_{\text{true}}\|_2$ is the Frobenius norm of the true image, defined as:

$$\|I_{\text{true}}\|_2 = \sqrt{\sum_{i=1}^N (I_{\text{true}}(i))^2} \quad (4.2)$$

The **PSNR** compares the ratio between the maximum possible power of a signal and the power of unwanted noise. In images, the power of a signal is quantified by the value of each pixel, with higher values denoting higher power. **PSNR** is defined as:

$$\text{PSNR} = 20 \log_{10} \left(\frac{\max(I)}{\sqrt{\text{MSE}}} \right) \quad (4.3)$$

Where I is the intensity of the image, and MSE is the Mean Squared Error, given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (I(i) - \hat{I}(i))^2 \quad (4.4)$$

Finally, the [SSIM](#) is a metric designed to model human visual perception by analyzing three key aspects. Luminance assesses the similarity in overall brightness between images. Contrast evaluates differences in pixel intensity variations using standard deviation. Structure captures the spatial relationships between pixel clusters, reflecting how humans perceive patterns and textures in an image [\[39\]](#). [SSIM](#) It is computed as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (4.5)$$

where μ_x and μ_y are the mean values of the original and distorted images, σ_x^2 and σ_y^2 are the variances, and σ_{xy} is the covariance of the two images. The constants C_1 and C_2 are used to stabilize the division with weak denominators.

To compute the latency, Python’s time library was utilized to record the start and end times of the inference function. The difference between these two times was then calculated and divided by the total number of images, yielding an approximate value for the average inference latency per image, that takes into account the overhead of memory transport latencies.

Power consumption was measured using the pyRAPL library [\[27\]](#). Markers were placed before and after the inference function, which allowed the computation of the average CPU socket power using Intel’s [Running Average Power Limit \(RAPL\)](#). [RAPL](#) is a built-in feature in Intel [CPU](#) that tracks and reports the average power consumption at the [CPU](#) socket level.

4.5 Conclusion

In this chapter, the detailed development of the software implementation was introduced for low-dose [PET](#) scan denoising using deep learning. [ADNI](#) was introduced, outlining its objectives and the characteristics of the provided dataset. The process of dataset extraction and the associated preprocessing workflow were systematically described. Furthermore, the proposed [FCN](#) architecture was presented, serving as the foundation for this thesis, and the computational environment and selected hyperparameters were specified. Finally, the chapter concludes by defining the evaluation metrics used to assess the quality of the denoised images.

Chapter 5

FPGA Deployment with OpenVINO and FPGA AI Suite

This chapter details the workflow for deploying the proposed model in Chapter 4 using the IFAS toolset. It presents an adapted process based on [8] and outlines the selection process for the target hardware used in FPGA emulation. Additionally, it highlights the experimental setup and evaluates the proposed model across various FPGA architectures. Figure 5.1 summarizes the FPGA deployment workflow.

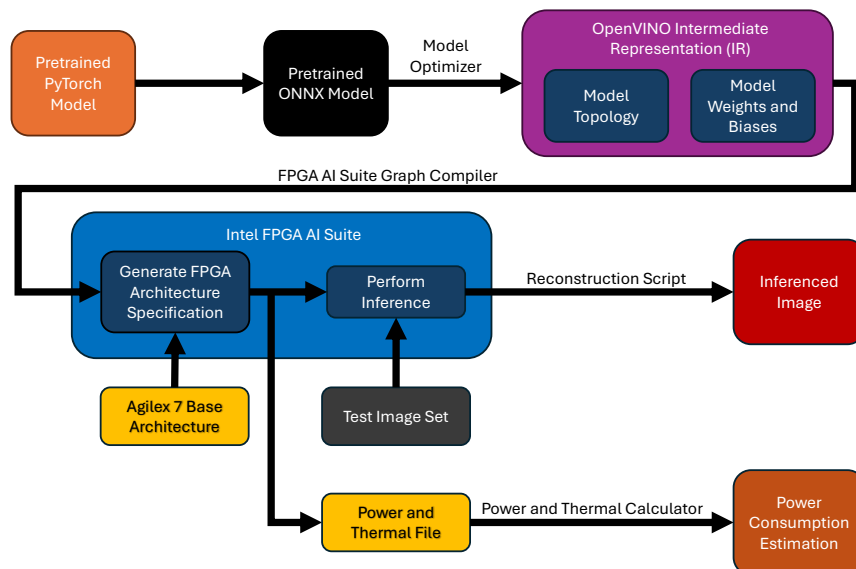


Figure 5.1: FPGA Deployment Workflow

5.1 Preprocessing the Trained Model

Following model training, the complete model architecture, including weights and biases, is saved to a file using PyTorch’s (.pt) format. To reduce the platform’s dependency on PyTorch, the model is converted to the [Open Neural Network Exchange \(ONNX\)](#) format. [ONNX](#) is an open-source framework designed to facilitate cross-platform compatibility and interoperability within deep learning research and development [4].

The pre-trained [ONNX](#) model is then converted into OpenVINO’s [IR](#) format, which provides a graph representation equivalent to the original [ONNX](#) architecture while ensuring compatibility with [IFAS](#). This conversion is performed using OpenVINO’s [Model Optimizer \(MO\)](#), which leverages the OpenVINO API to enable seamless transformation. The [MO](#) generates two output files: an XML file that defines the model topology and a BIN file that contains the corresponding trained weights and biases.

5.2 Hardware Architecture Generation

The [IFAS](#) framework provides support for various Intel-designed [FPGA](#) architectures, including the Arria 10, Agilex 5/7, and Stratix 10. The Agilex 7 was selected as the target hardware architecture due to its status as the most recent FPGA family, offering superior performance and energy efficiency compared to the Arria and Stratix series [10]. The Agilex 7 features an [Adaptive Logic Module \(ALM\)](#) count of 912,800, an [M20K Memory Block \(M20K\)](#) count of 13,272, and a [DSP](#) count of 8,528 [10]. Additionally, the architecture generation assumes that the emulated [FPGA](#) operates at its maximum physical frequency of 500 MHz.

To optimize model deployment, the generated OpenVINO [IR](#) was integrated with the Agilex 7 base architecture, creating a specialized configuration tailored for the proposed model. The architecture generation process was conducted for two distinct experiments. The first experiment prioritized performance, allowing the graph compiler to utilize up to 80% of the total Agilex 7 resources. The remaining 20% allowed for overhead in real-life deployment scenarios, where extra utilization would be allocated to perform timing optimization. The second experiment focused on power efficiency, where the graph compiler was iteratively executed with increasingly restrictive utilization limits until it was unable to produce a stable configuration, leaving only 2% of [FPGA](#) resource utilization.

5.3 Inference on Emulated FPGA Architectures

Inference was performed using the built-in tools provided by [IFAS](#), utilizing the same test dataset as used in the software-based implementation. The results were generated in a plain text file, where individual pixel values for each inference were stored in a row-order format. Latency per image was also provided in the results. To reconstruct the inference output into a coherent [PET](#) scan image, a Python script was developed to parse the text file, extract the pixel values, and rearrange them according to the original image dimensions. This approach allows the inference results obtained from the [FPGA](#)-based implementation to be directly compared to those from the software implementation, enabling a cohesive evaluation of performance, accuracy, and energy efficiency between the two methods.

5.4 Power Consumption Measurements

After converting the OpenVINO [IR](#) for hardware architecture generation, the [IFAS](#) Graph Compiler also produces a Power and Thermal Calculator (.ptc) file. This file can be analyzed using [Intel Power and Thermal Calculator \(IPTC\)](#), enabling precise estimation of the power consumption and thermal characteristics of the emulated [FPGA](#) architecture. Since the generated hardware architecture is specifically optimized for the given model, this approach provides an accurate representation of physical deployment conditions. The Extended -4 speed grade was selected for use in the [IPTC](#), tailored for extremely low-power applications. The calculations also assumed a maximum junction temperature of 100°C, and normal operation at [Standard Ambient Temperature and Pressure \(SATP\)](#) (25°C).

5.5 Conclusion

This chapter detailed the workflow for deploying the proposed deep learning model onto an emulated [FPGA](#) architecture using [IFAS](#). The PyTorch-based model was converted to OpenVINO's [IR](#) using the [ONNX](#) format as an intermediary step. The Agilix 7 [FPGA](#) architecture is selected, and specialized hardware architecture is generated using the OpenVINO [IR](#) and the base Agilix 7 [FPGA](#) configuration. Inference is then performed on the emulated [FPGA](#) architecture using [IFAS](#) tools, then reconstructed into a standard [PET](#) scan for qualitative comparison with software-based implementations. Finally, power consumption metrics are extracted for further post-analysis.

Chapter 6

Experimental Results and Evaluation

This chapter presents the results of the software and [FPGA](#) implementations discussed in Chapters [4](#) and [5](#). It includes visual representations of [PET](#) scan slices from all 10 patients, highlighting the low-dose input, software-based reconstruction, and both [FPGA](#) implementations optimized for performance and power, alongside the corresponding ground truth images. Additionally, summarized quantitative evaluations of inference accuracy, latency, power consumption, and [FPGA](#) resource utilization are provided to compare the effectiveness of the three different implementations.

6.1 Visual Representations of Experimental Results

To provide a visual representation of the inference results, random slices from each patient's [PET](#) scan were selected for analysis. Figures [6.1](#) through [6.10](#) illustrate the low-dose input, inference results from the software implementation, and both [FPGA](#) implementations (optimized for performance and power), alongside their corresponding ground truths. Additionally, a magnified view is included to allow pixel-wise comparisons between the different implementations.

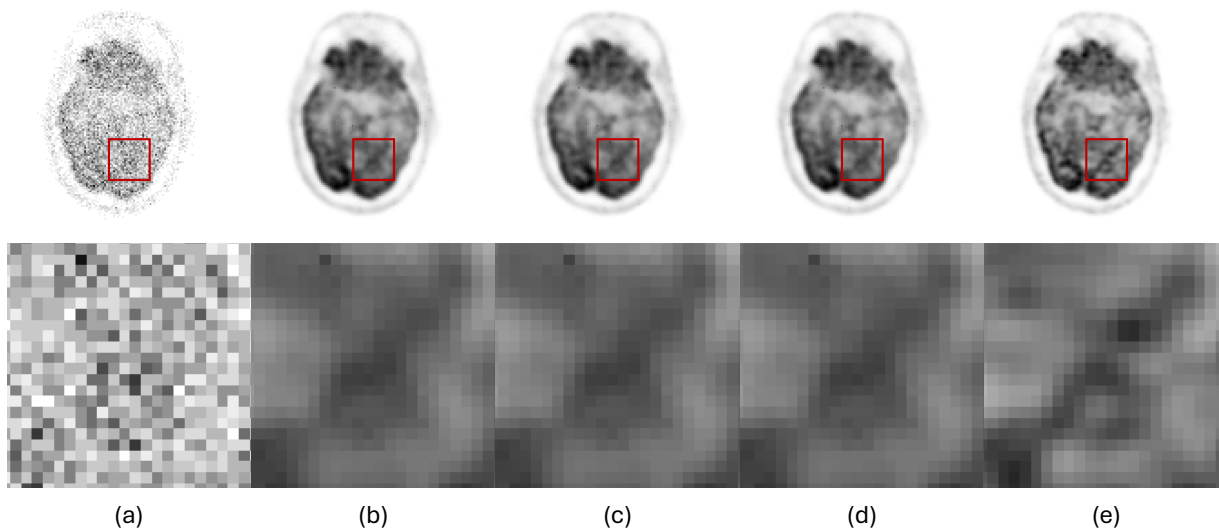


Figure 6.1: PET Scan of Patient 1. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

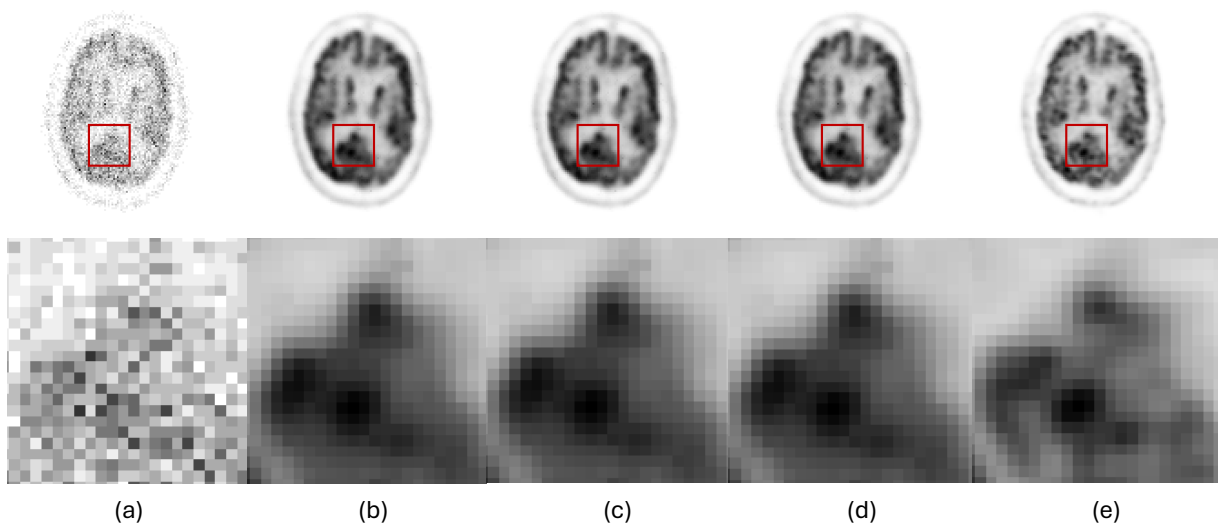


Figure 6.2: PET Scan of Patient 2. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

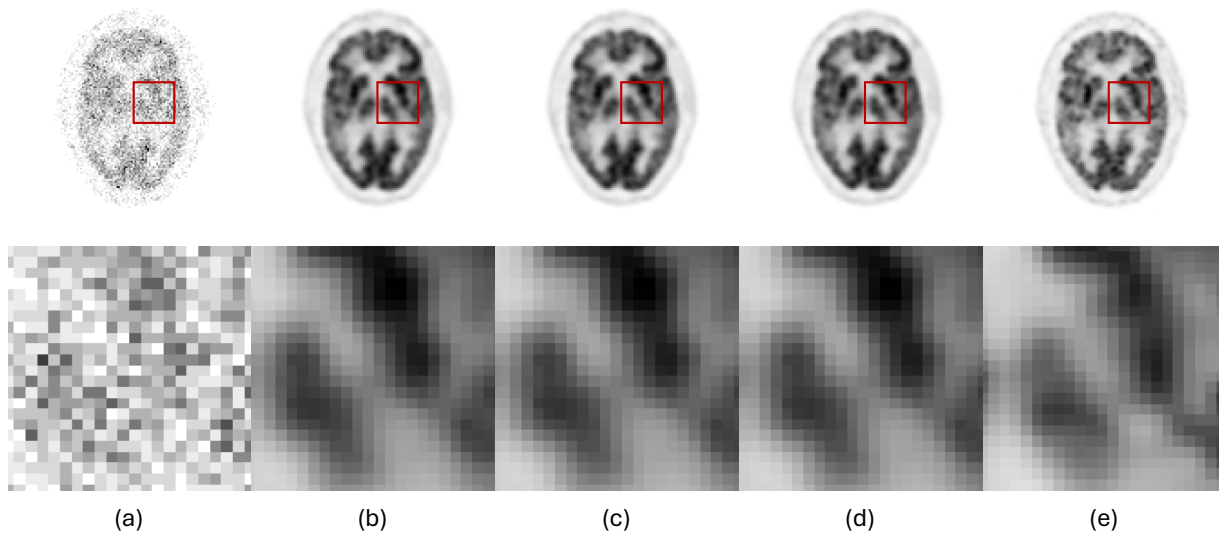


Figure 6.3: PET Scan of Patient 3. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

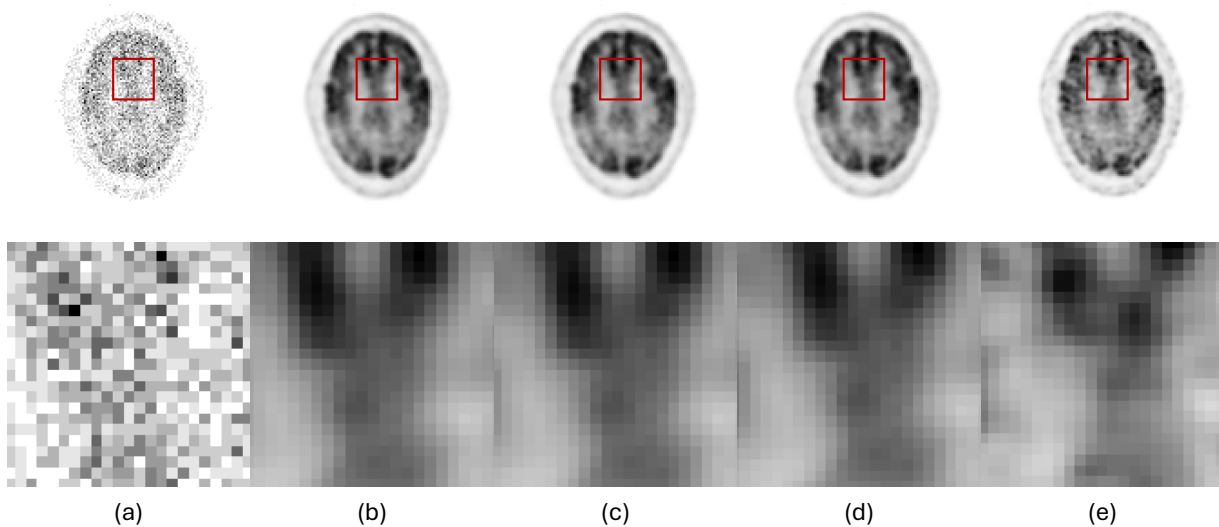


Figure 6.4: PET Scan of Patient 4. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

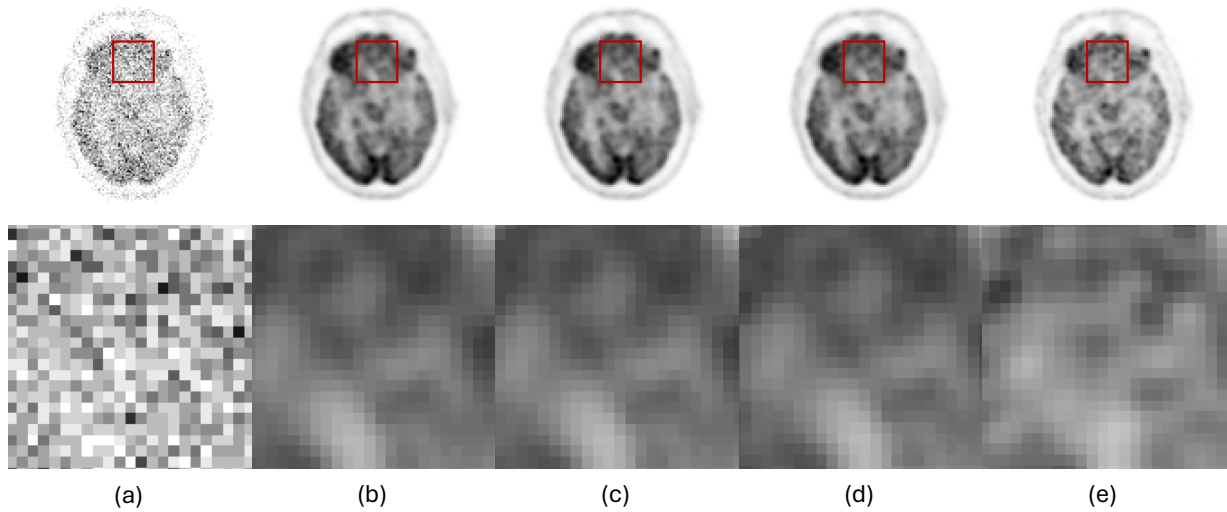


Figure 6.5: PET Scan of Patient 5. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

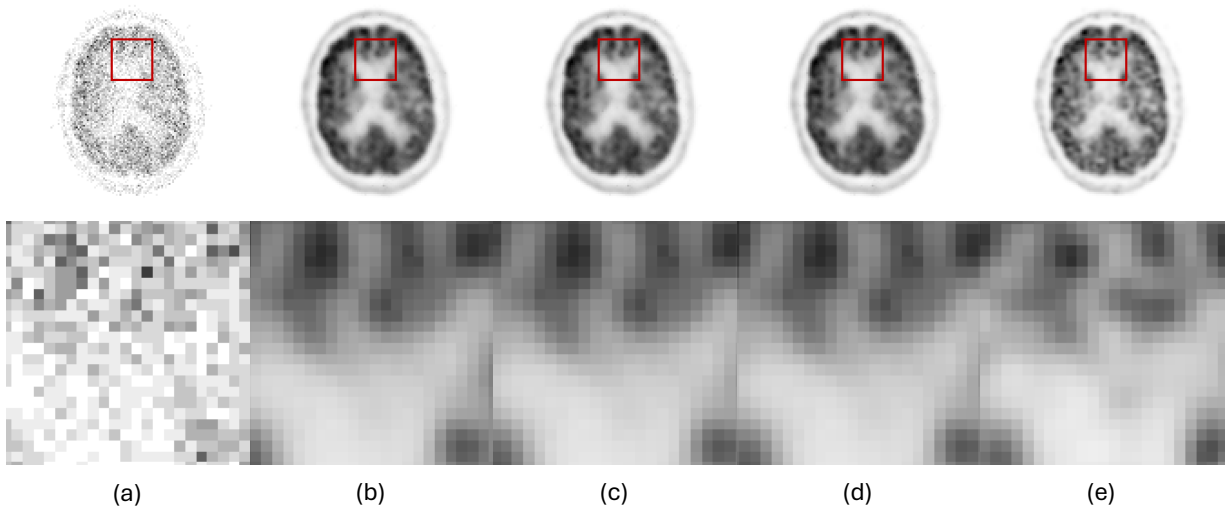


Figure 6.6: PET Scan of Patient 6. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

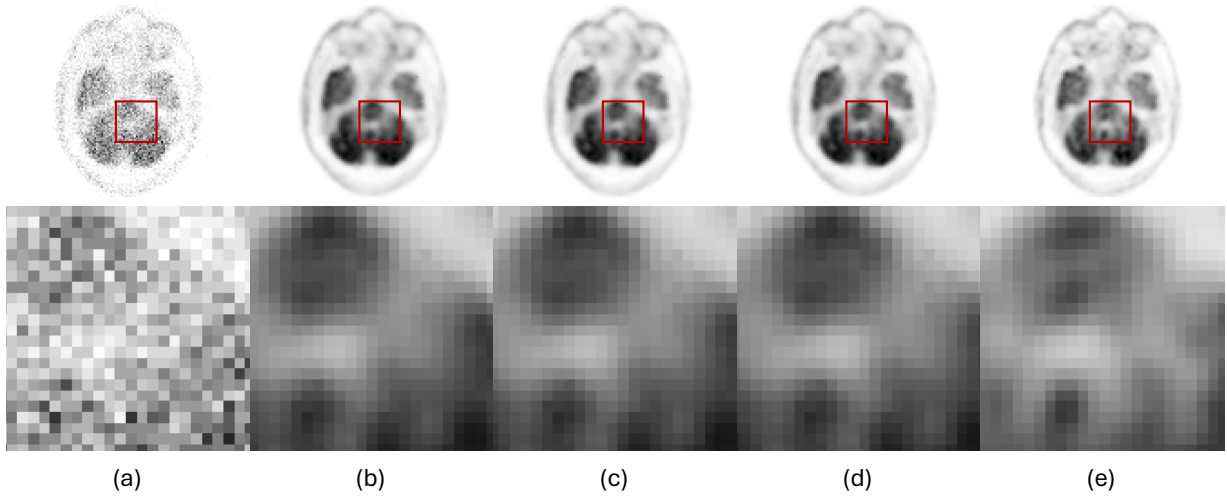


Figure 6.7: PET Scan of Patient 7. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

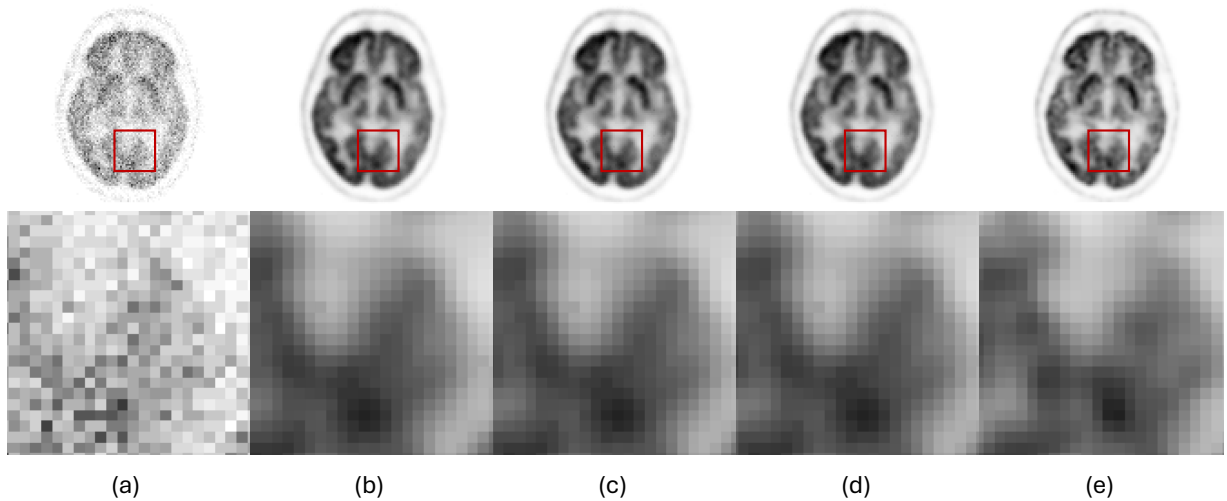


Figure 6.8: PET Scan of Patient 8. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

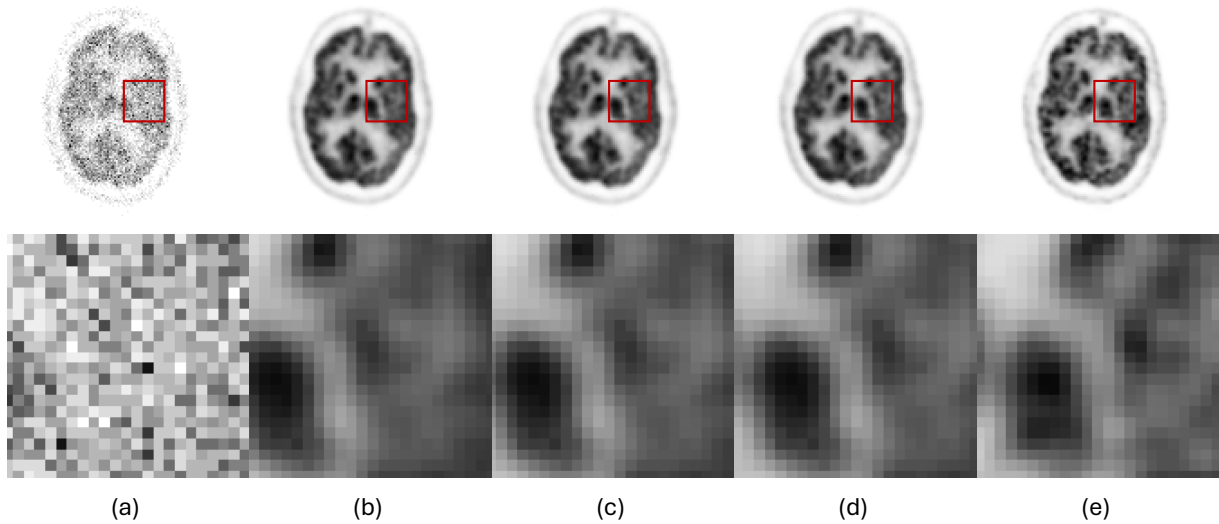


Figure 6.9: PET Scan of Patient 9. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

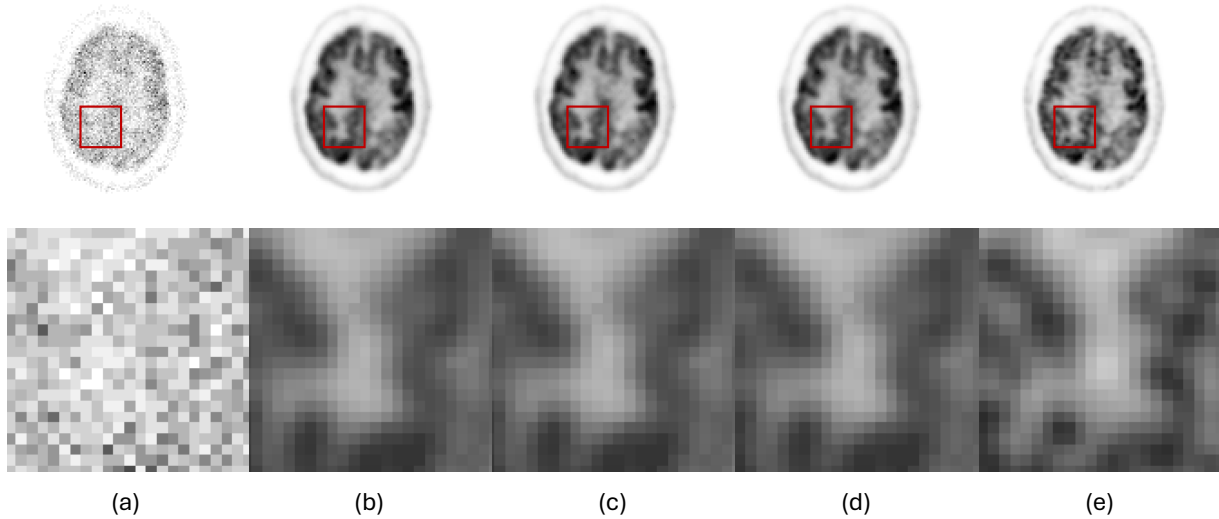


Figure 6.10: PET Scan of Patient 10. (a) Low Dose Input (b) Software Inference (c) High-Performance FPGA Inference (d) Low Power FPGA Inference (e) Ground Truth

6.2 Comparison of Inference Accuracy

Figure 6.11 and Table 6.1 provide a visual and quantitative representation of the average inference accuracy of the proposed model, respectively. Since the FPGA emulation model is bit-accurate, it delivers the same results as the software implementation.

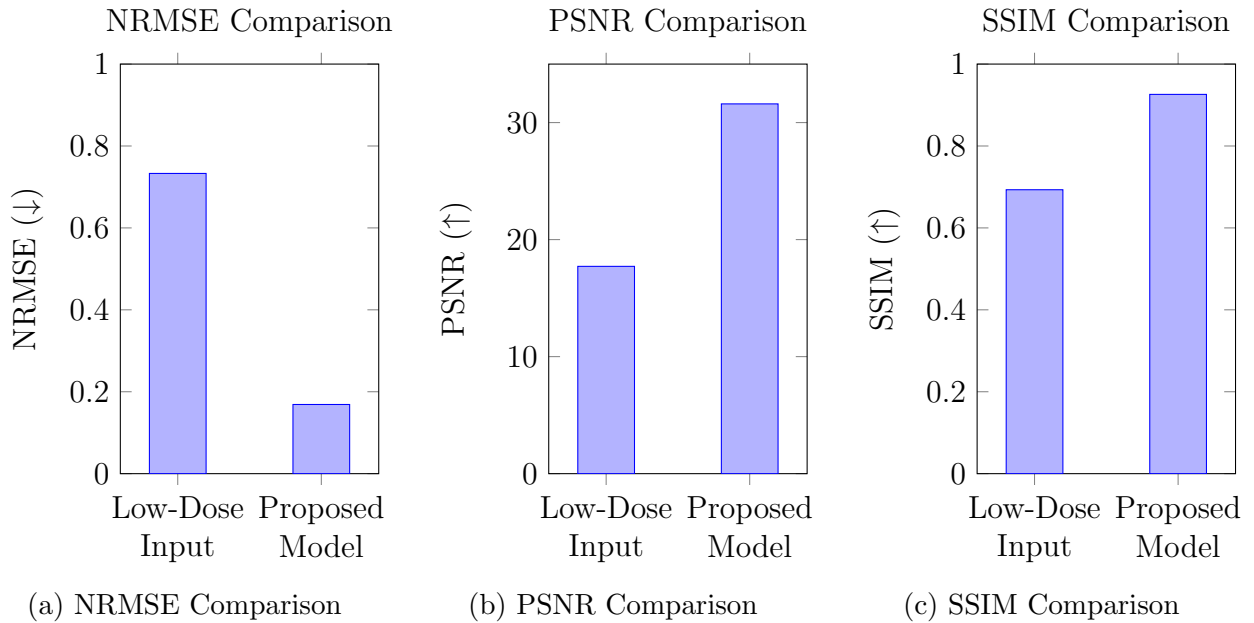


Figure 6.11: Comparison of Average Inference Accuracy Metrics

Table 6.1: Average Inference Accuracy

Method	NRMSE ↓	PSNR ↑	SSIM ↑
Low-Dose Input	0.7330	17.7156	0.6931
Proposed Model	0.1689	31.5982	0.9260

6.3 Comparison of Inference Latency

Figure 6.12 and Table 6.2 provide a visual and quantitative representation of the average inference latency per image of the proposed model, respectively.

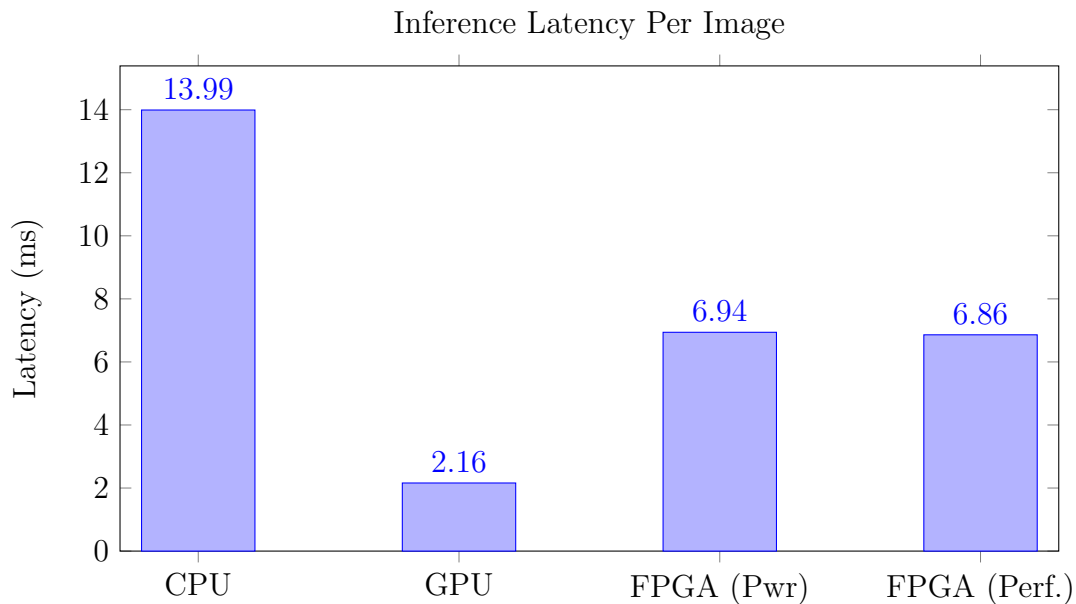


Figure 6.12: Inference Latency Comparison

Table 6.2: Inference Latency Per Image

Method	Latency (ms) ↓
CPU	13.99
GPU	2.16
FPGA (Low Power)	6.94
FPGA (High Performance)	6.86

6.4 Comparison of Power Consumption

Figure 6.13 and Table 6.3 provide a visual and quantitative representation of the inference accuracy of the proposed model, respectively.

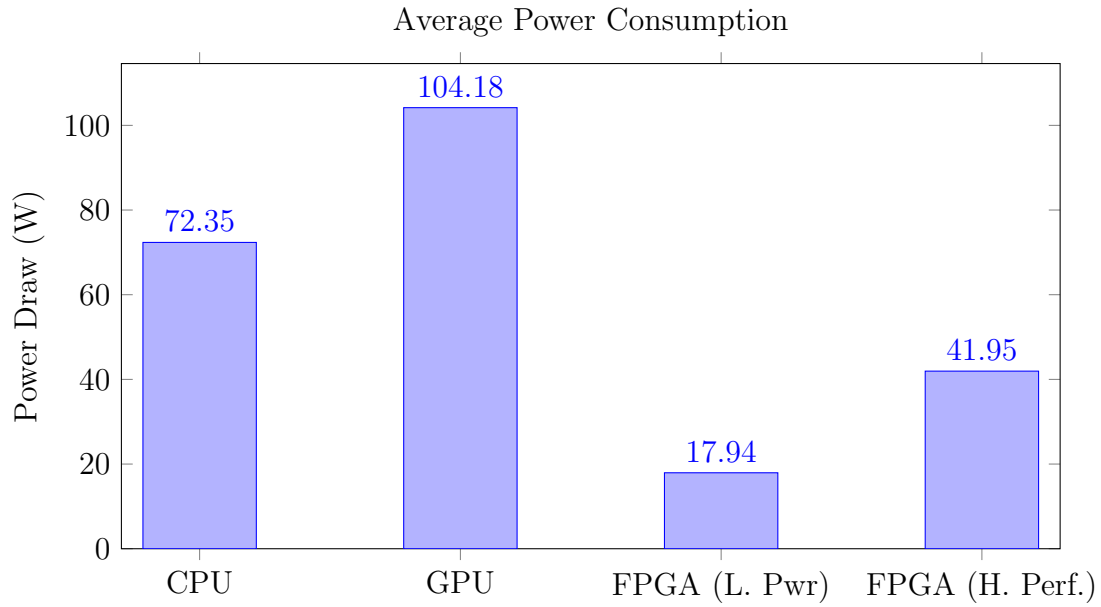


Figure 6.13: Average Power Consumption Comparison

Table 6.3: Average Power Consumption and Performance per Watt

Method	Power Draw (W) ↓	Performance per Watt (1/ms· W) ↑
CPU	72.35	0.00096
GPU	104.18	0.00452
FPGA (Low Power)	17.94	0.00799
FPGA (High Performance)	41.95	0.00346

6.5 FPGA Resource Utilization

Table 6.4 provides a quantitative representation of the resources utilized as a percentage for both the low power FPGA architecture, and the high performance FPGA architecture. The resources include: DSP, ALM, FF, M20K.

Table 6.4: Resource Utilization of FPGA Configurations

Metric	High Performance FPGA (%)	Low Power FPGA (%)
DSP	25.516%	3.377%
ALM	11.9211%	1.983%
FF	15.093%	1.551%
M20K	30.824%	2.923%

6.6 Conclusion

This chapter presented the experimental results of performing inference on the proposed model, comparing both software and FPGA implementations. The results were evaluated visually and qualitatively, with visual representations of the inference outputs provided, along with magnified views to assist pixel-wise comparisons. Additionally, performance metrics such as accuracy, latency, performance, and resource utilization were compared across all modes of acceleration, including CPU, GPU, low power FPGA, and high performance FPGA.

Chapter 7

Discussion and Conclusions

7.1 Evaluation of Results

The visual representations presented in Figures 6.1 through 6.10 provide visual insights into the performance of the proposed model. Across various image slices, the model accurately denoises structural details and hot spots, as evident in the magnified views. Notably, all inference results are pixel-wise identical, confirming the bit-true accuracy of the OpenVINO and IFAS conversion process. This accuracy is validated quantitatively in Figure 6.11 and Table 6.1, where significant improvements are observed in NRMSE (4.3x reduction), PSNR (24.4x increase), and SSIM. These results indicate that the proposed model, when compared to the low-dose input, surpasses the relative improvements reported in [43].

Between the software-based accelerators (CPU and GPU), the observed latency followed the expected results, with the GPU implementation achieving a 6.5x reduction in inference latency when compared to the CPU implementation. Both FPGA implementations demonstrated approximately 2x lower latency than the CPU implementation. Notably, the comparison between the low power and high performance FPGA architectures yielded only a marginal 1% reduction in latency in favour of the high-performance implementation. This result suggests that for this specific deep learning model architecture, the additional computational resources available in the high-performance implementation did not scale into a proportional reduction in latency, contrary to initial expectations.

When evaluating power consumption, the GPU implementation was found to consume the most power, followed by the CPU, high-performance FPGA, and low-power FPGA. Notably, the low power FPGA implementation outperformed the GPU by a factor of 5.8x, demonstrating excellent power efficiency while maintaining relatively low inference latency. Table 6.3 presents the performance-per-watt, which is calculated as the inverse of latency (in milliseconds) multiplied by power consumption (in watts). Since all four methods produced identical pixel-wise results, the performance per watt metric identifies the most energy-efficient approach for accelerating the proposed model. Table 6.3 reveals that the low-power FPGA implementation offers the highest performance-to-watt ratio of 0.00799, a 77% improvement over the GPU implementation.

7.2 Conclusions and Future Directions

PET imaging technology has revolutionized the field of medical imaging over the past half-century. As a non-invasive diagnostic tool, PET imaging offers critical insights into the metabolic activity of organs and tissues, enabling early detection and monitoring of various diseases. However, due to the fundamental nature of the technology, PET imaging relies on the use of radiotracers, which, when subjected to repeated exposure or used in immunodeficient patients, can cause adverse effects [26]. These effects, such as tissue damage or increased risk of cancer, demonstrate significant concerns for use in clinical practice.

To address these issues, reducing the amount of radiotracer used in PET scans presents a viable solution. However, the reduction in radiotracer directly results in a lower count of positron emissions, which produces lower-quality imagery. Increased noise will reduce the interpretability of the PET scan, making it more challenging for radiologists and clinicians to assess organ and tissue status. To remedy this concern, denoising techniques have been proven to mitigate these effects, restoring low-dose PET images to standard quality while using a reduced radiotracer dose [43].

The development of deep learning has introduced novel approaches to image denoising, offering techniques applicable for improving PET imaging denoising. CNN architectures such as the DNCNN [45], have shown demonstrable results in denoising images. However, traditional methods of accelerating the training and inference processes for these architectures have typically relied on GPU or CPU implementations. While these solutions are approachable and effective, they are limited by the requirement of additional hardware, higher computational overhead, and inefficiencies in power consumption when deployed in an edge-compute application.

This thesis proposed the use of **FPGA** hardware to accelerate the task of denoising low-dose **PET** scans. **FPGA** offer several advantages over conventional **GPU** and **CPU**, such as reduced power consumption and reconfigurability for specific end-user applications. By leveraging **FPGA**, the computational load of image denoising can be accelerated while increasing power efficiency as compared to contemporary methods.

The experimental results presented in this thesis demonstrate a clear advantage of using **FPGA** to accelerate deep learning models for **PET** image denoising. In particular, compared to the **GPU** implementation, the low power optimized **FPGA** implementation showed a 77% improvement in the performance-to-watt ratio. This validates the hypothesis that **FPGA** can effectively accelerate deep learning for medical image processing tasks while offering significant improvements in power efficiency. These results underscore the potential of **FPGA** as a viable solution for accelerating low-dose **PET** image denoising, providing an energy-efficient alternative to traditional **GPU** and **CPU** implementations.

While this thesis has demonstrated the feasibility of using **FPGA** for accelerating deep learning denoising of low-dose **PET** imaging, there are remaining avenues for future research. The next logical step is to deploy the proposed low power **FPGA** architecture onto real hardware. This thesis' experiments were performed in an emulated environment, which, while effective for preliminary validation of power efficiency, does not fully account for the potential overhead and other complexities present in real-world hardware. Deploying the model on physical hardware will allow for direct optimization to ensure results perfectly align with emulated results. Finally, incorporating feedback from radiologists and clinicians will prove to be an essential component to assess to ensure the model's performance.

References

- [1] IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.
- [2] Alzheimer’s Disease Neuroimaging Initiative. Alzheimer’s Disease Neuroimaging Initiative (ADNI) - About. <https://adni.loni.usc.edu/about/>, 2025. Accessed: January 31, 2025.
- [3] AMD. Virtex-4 Family Overview. Technical documentation, Advanced Micro Devices, Inc., 2025. Accessed: February 13, 2025.
- [4] Junjie Bai, Fang Lu, Ke Zhang, et al. ONNX: Open Neural Network Exchange. <https://github.com/onnx/onnx>, 2019.
- [5] Mohd Fauzi Bin Othman, Norarmalina Abdullah, and Nur Aizudin Bin Ahmad Rusli. An overview of MRI brain classification using FPGA implementation. In *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*, pages 623–628, 2010.
- [6] LibreTexts Chemistry. Radioactive Decay Diagram. Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).
- [7] PyDICOM Contributors. PyDICOM: A Python Package for DICOM Medical Imaging Data. <https://pydicom.github.io/>, 2025. Accessed: January 31, 2025.
- [8] Intel Corporation. Accelerating AI Workloads with FPGAs. Whitepaper, 2021. Accessed: 2025-02-24.
- [9] Intel Corporation. Low Precision Networks for Efficient Inference on FPGAs. Whitepaper, 2021. Accessed: 2025-02-24.
- [10] Intel Corporation. Agilix FPGA Portfolio Product Brief. Online, 2024. Accessed: 2025-03-09.

- [11] J. Cui, K. Gong, N. Guo, X. Li, C. Wu, X. Meng, K. Kim, K. Zheng, C. Catana, J. Qi, and Q. Li. PET image denoising using unsupervised deep learning. *European Journal of Nuclear Medicine and Molecular Imaging*, 46:2780–2789, 2019.
- [12] Michel Defrise, Paul E. Kinahan, and Claude J. Michel. Image Reconstruction Algorithms in PET. In D. L. Bailey, D. W. Townsend, P. E. Valk, and M. N. Maisey, editors, *Positron Emission Tomography*. Springer, London, 2005.
- [13] P. Dillinger, J.F. Vogelbruch, J. Leinen, S. Suslov, R. Patzak, H. Winkler, and K. Schwan. FPGA based real-time image segmentation for medical systems and data processing. In *14th IEEE-NPSS Real Time Conference, 2005.*, pages 5 pp.–, 2005.
- [14] FastML Team. [fastmachinelearning/hls4ml](https://github.com/fastmachinelearning/hls4ml), 2024.
- [15] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference, 2021.
- [16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015.
- [18] Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian Wirkert, and Klaus H. Maier-Hein. nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation, 2018.
- [19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.
- [20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [21] C. C. Liu and J. Qi. Higher SNR PET image prediction using a deep learning model and MRI image. *Physics in Medicine and Biology*, 64(11):115004, 2019.

- [43] Junshen Xu, Enhao Gong, John Pauly, and Greg Zaharchuk. 200x Low-dose PET Reconstruction using Deep Learning, 2017.
- [44] S. Yu. Review of F-FDG Synthesis and Quality Control. *Biomedical Imaging and Intervention Journal*, 2(4):e57, 2006.
- [45] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.
- [46] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, May 2018.
- [47] L. Zhou, J. D. Schaefferkoetter, I. W. K. Tham, G. Huang, and J. Yan. Supervised learning with CycleGAN for low-dose FDG PET image denoising. *Medical Image Analysis*, 65:101770, 2020.