Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18304-x

Real-time diabetic foot ulcer classification based on deep learning & parallel hardware computational tools

Mohammed A. Fadhel[1] · Laith Alzubaidi[2,3,4] · Yuantong Gu[2,3] · Jose Santamaría[5] · Ye Duan[6]

Received: 16 June 2022 / Revised: 8 October 2023 / Accepted: 19 January 2024

Abstract
Meeting the rising global demand for healthcare diagnostic tools is crucial, especially with a shortage of medical professionals. This issue has increased interest in utilizing deep learning (DL) and telemedicine technologies. DL, a branch of artificial intelligence, has progressed due to advancements in digital technology and data availability and has proven to be effective in solving previously challenging learning problems. Convolutional neural networks (CNNs) show potential in image detection and recognition, particularly in healthcare applications. However, due to their resource-intensiveness, they surpass the capabilities of general-purpose CPUs. Therefore, hardware accelerators such as application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), and graphics processing units (GPUs) have been developed. With their parallelism efficiency and energy-saving capabilities, FPGAs have gained popularity for DL networks. This research aims to automate the classification of normal and abnormal (specifically Diabetic Foot Ulcer—DFU) classes using various parallel hardware accelerators. The study introduces two CNN models, namely DFU_FNet and DFU_TFNet. DFU_FNet is a simple model that extracts features used to train classifiers like SVM and KNN. On the other hand, DFU_TFNet is a deeper model that employs transfer learning to test hardware efficiency on both shallow and deep models. DFU_TFNet has outperformed AlexNet, VGG16, and GoogleNet benchmarks with an accuracy 99.81%, precision 99.38% and F1-Score 99.25%. In addition, the study evaluated two high-performance computing platforms, GPUs and FPGAs, for real-time system requirements. The comparison of processing time and power consumption revealed that while GPUs outpace FPGAs in processing speed, FPGAs exhibit significantly lower power consumption than GPUs.

Keywords Deep learning · Double transfer learning · FPGA · GPU · DFU · Real-time · Parallel hardware · Medical imaging

Extended author information available on the last page of the article

1 Introduction

Machine learning (ML) and artificial intelligence (AI) have become an integral part of our daily lives and have transformed various domains, such as image processing and speech recognition [1–3]. A subset of ML called deep learning (DL) has been instrumental in this revolution by enabling automatic feature extraction from large datasets, particularly in domains like natural language processing, speech recognition, and computer vision [4].

The increasing global demand for healthcare diagnostic technologies and a shortage of medical personnel have led to a growing interest in using Deep Learning (DL) and telemedicine systems. However, DL's dependence on large annotated datasets presents a significant challenge in medical image analysis. Transfer Learning (TL) has emerged as a potential solution to tackle this challenge, allowing models to leverage pre-training on reference datasets before fine-tuning specific tasks [3].

Despite TL's success, researchers and application specialists are constrained by the need for accelerated hardware to scale DL algorithms beyond current capacities. Graphics Processing Units (GPUs) have become a dominant hardware accelerator for DL due to their superior parallel computation capabilities [5, 6]. However, the paper contends that there is a need to explore alternative hardware platforms, specifically Field-Programmable Gate Arrays (FPGAs), which exhibit advantages such as adaptable hardware configurations and power-saving performance for subprograms crucial to DL [7].

Although FPGAs offer attractive features, their adoption can be hindered by the requirement of specialized hardware knowledge. However, recent advancements in FPGA tools, especially those that involve OpenCL, have made them more accessible to a broader audience, including application scientists and hardware researchers. For deep learning researchers, FPGAs provide a compelling option due to their high parallelism, reconfigurability, and user-friendly development tools [8].

This paper aims to argue that FPGAs are the best hardware acceleration platforms for deep learning (DL) among the current options available. It aims to provide an overview of recent FPGA support for DL, highlight their associated limitations, and suggest future trends in parallel hardware computational tools. The paper focuses explicitly on meeting real-time requirements in learning strategies, particularly emphasizing the Diabetic Foot Ulcer (DFU) classification.

The paper has three primary objectives. First, it aims to demonstrate that FPGAs are superior to other contemporary hardware acceleration platforms. Second, it highlights the recent support for DL on FPGAs while also identifying their limitations. Lastly, it aims to provide insights and recommendations for future trends in parallel hardware computational tools, specifically Convolutional Neural Networks (CNNs). The paper presents case studies of DFU classification on two high-performance computing platforms, GPUs and FPGAs, to accelerate the classification process and potentially prevent amputations in individuals with DFU disease.

The unique contributions of this work include the introduction of two DFU classification models (DFU_TFNet and DFU_FNet) employing a novel TL strategy, evaluation on two parallel hardware platforms (FPGA and GPU), comparison with traditional classifiers (SVM and KNN), training and testing of various CNN models (AlexNet, VGG16, and GoogleNet) on the same dataset, calculation of power consumption and processing time values, and the demonstration that FPGA can be a viable choice for portable embedded devices, with the DFU_TFNet model achieving a remarkable accuracy 99.81%, precision 99.38% and F1-Score 99.25%.

The rest of this paper is organized as follows: preliminaries and definitions are described in Section 2. A brief discussion of related work is presented in Section 3. The methodology for the proposed models and the hardware setup by FPGA for real-time requirements are illustrated in Section 4. An evaluation of proposed models in terms of accuracy, precision, F1-Score, processing time, and power consumption is listed in Section 5. Lastly, the paper ends with a conclusion, which is Section 6.

2 Brief overview of CNNs with FPGA

In the implementation of DL methods using FPGAs, refer to Appendix 1. Notably, the primary impediment in achieving the requisite hardware lies in the design size, posing a significant challenge in this context. The exchange between density and the ability of design reconfiguration means that the FPGA circuits are generally considered less dense than hardware replacements. Thus, it is only sometimes possible to implement large neural networks. Conversely, deep networks become applied on single FPGA schemes because the current FPGA incorporates strengthened computational units and the common FPGA fabric and continues developing reduced feature sizes for enhancing density. Figure 1 illustrates a summarized year-sequence of significant events in deep-learning research-based FPGAs.

In the early 1990s, Cox et al. were the first group of researchers to implement neural networks using FPGAs [9]. A few years later, Cloutier et al. recorded the first implementation of CNN using FPGAs [10]. These studies were restricted to utilizing low-precision arithmetic due to FPGA size limitations. Moreover, density-strengthened multiply-accumulate units were yet to be available in FPGAs. Thus, arithmetic was an extremely slow and expensive resource. The FPGA technology was then significantly modified, increasing the strengthened computation units available in FPGAs, more inspired by reducing transistor (feature) size and increasing the density of FPGAs fabric. The current CNN implementations using FPGAs have benefited from these design developments.

To the best of our knowledge, a team at Microsoft recently achieved forward propagation of CNN using FPGAs. Using the 1 K dataset on the ImageNet network, Ovtcharov et al. [11] reported that the processing amount of 134 images/second was achieved when operated on a Stratix V D5 at 25 W. This processing amount is approximately three times the processing amount of their closest competitor. However, it is predicted that an improved performance of up to approximately 233 images/second on an Arria 10 GX1150 with the

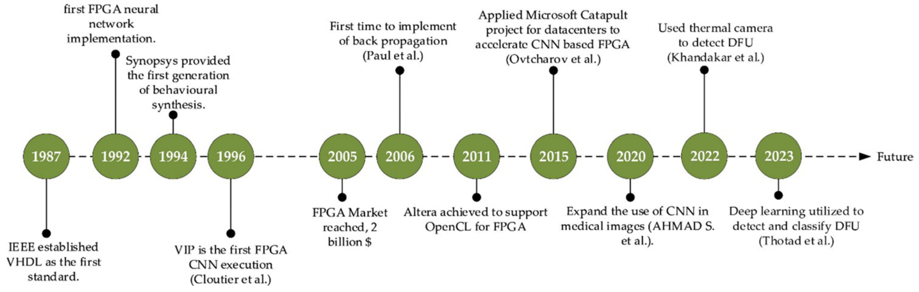
GANGILON is the first FPGA neural network implementation.
Synopsys provided the first generation of behavioural synthesis.
First time to implement of back propagation (Paul et al.)
Applied Microsoft Catapult project for datacenters to accelerate CNN based FPGA (Ovtcharov et al.)
Used thermal camera to detect DFU (Khandakar et al.)
1987
1992
1994
1996
2005
2006
2011
2015
2020
2022
2023
Future
IEEE established VHDL as the first standard.
VIP is the first FPGA CNN execution (Cloutier et al.)
FPGA Market reached, 2 billion $
Altera achieved to support OpenCL for FPGA
Expand the use of CNN in medical images (AHMAD S. et al.).
Deep learning utilized to detect and classify DFU (Thotad et al.)

Fig. 1 Key events in the history of FPGA DL research

same power consumption can be achieved by utilizing the latest FPGAs. In contrast, the high-performing GPU systems (Caffe + cuDNN) achieved 500–824 images/second with 235 W power consumption. This performance was attained utilizing FPGA servers and boards designed by Microsoft as part of an investigational project where FPGAs integrate inside the data centre applications. This project also increased the performance of large-scale search engines (twice) to show the capacity of such FPGA applications.

Zhang et al. are the closest competitor to realize another significant achievement: processing 46 images/second on Virtex 7485 T without reducing the power consumption [12]. These results are better than several significant works presented by their competitors [13–15]. These examples have a similar architectural design, including several parallel processing units applied on FPGA fabric (generally employed for convolution), buffered output and input, the ability to configure software layers, and usually utilizing off-chip memory access. However, there are also significant differences in using FPGAs, such as utilizing various operation frequencies, look-up table types, soft-cores, data-transfer mechanisms, memory subsystems, and completely diverse FPGAs. Therefore, more research is required to identify optimal architecture decisions [16].

Transfer learning is often utilized to build medical imaging models with little training data. One of the initial ideas for employing transfer learning [17] was to use pre-trained ImageNet models instead of training from scratch. As the pre-trained CNN is effective in computation and ease of algorithms, the main benefit of including FPGAs is accelerating the forward propagation of such systems and informing the attained processing amount. This issue is very significant for application engineers as they want to utilize viable pre-trained networks for processing sizeable volumes of data effectively and rapidly. Conversely, accelerating rearward propagation is another aspect to consider in CNN design using FPGA. The first to use parallelism in the learning phase on Virtex E FPGA was Paul et al. in 2006 [18], who focused on accelerating the classification process inside CNN and boosted this by using different software or hardware platforms to take advantage of parallelism techniques.

In the realm of early detection and prognosis for diabetic foot ulcers, Thotad et al. paved the way by introducing the use of the EfficientNet—a robust deep neural network model [19]. Building upon this foundation, various end-to-end CNN-based deep learning architectures, including AlexNet, VGG16/19, GoogLeNet, ResNet50.101, MobileNet, SqueezeNet, and DenseNet, have been explored for infection and ischemia categorization. This exploration was carried out using the DFU2020 benchmark dataset [20]. Applying machine learning to infrared images offers promising avenues for the early diagnosis of diabetic foot complications. Researchers delved into classical machine learning algorithms incorporating feature engineering, convolutional neural networks (CNN), and image enhancement techniques. These investigations aimed to pinpoint the most effective network for classifying thermograms [21]. In a different approach, [22] tackled the initial dataset's disparity by leveraging the synthetic minority oversampling strategy. Through a univariable analysis, nine key variables—random blood glucose, years with diabetes, cardiovascular diseases, peripheral arterial diseases, DFU history, smoking history, albumin, creatinine, and C-reactive protein—were identified. Subsequently, risk prediction models were independently developed using five machine learning algorithms: decision tree, random forest, logistic regression, support vector machine, and extreme gradient boosting (XGBoost). This multifaceted exploration underscores the diverse strategies employed to enhance the accuracy and effectiveness of diabetic foot ulcer prediction models. A comprehensive examination yielded Table 1, which provides an insightful overview of advancements,

Table 1 Overview of Advancements, Methodologies, and Research Gaps in Relevant Studies

Reference	Key Advancements	Methodologies	Research Gaps
[9]	- Implemented neural networks using FPGAs in the early 1990s	- Used low-precision arithmetic due to FPGA size limitations	- Restricted by slow and expensive resources due to FPGA size and arithmetic constraints
[10]	- Recorded the first implementation of CNN using FPGAs	- Limited to low-precision arithmetic due to FPGA size limitations	- Faced challenges due to the absence of density-strengthened multiply-accumulate units in FPGAs at that time
[11]	- Achieved forward propagation of CNN using FPGAs at high processing speed	- Utilized the Stratix V D5 FPGA, achieving 134 images/second on the 1K dataset	- Predicted improved performance of up to 233 images/second on an Arria 10 GX1150 with the latest FPGAs
[12]	- Realized a processing amount of 46 images/second on Virtex 7485T	- Demonstrated competitive results without reducing power consumption	- Need for more research to identify optimal FPGA architecture decisions
[18]	- Introduced parallelism in the learning phase on Virtex E FPGA	- Focused on accelerating the classification process inside CNN	- Explored different software or hardware platforms to leverage parallelism techniques
[19]	- Introduced the use of EfficientNet for diabetic foot ulcer detection and prognosis	- Explored various end-to-end CNN-based architectures for infection and ischemia categorization	- Potential gaps in understanding the optimal architecture for specific applications within diabetic foot ulcer detection
[20]	- Explored CNN-based deep learning architectures for infection and ischemia categorization	- Investigate and select the best CNN model for DFU classification	- Early detection and treatment may increase survival and decrease death
[21]	- Investigated classical machine learning algorithms and CNN with image enhancement for thermogram classification	- Explored feature engineering and convolutional neural networks for infrared image analysis	- Increase machine learning performance for diabetic foot ulcer identification
[22]	- Create a machine learning-based prediction algorithm to identify newly admitted DFU patients who need minor amputation swiftly	- Implemented decision tree, random forest, logistic regression, support vector machine, and extreme gradient boosting	- No clinical prediction methods for DFU minor amputations

methodologies, and identified research gaps in the relevant studies. This table serves as a valuable reference, encapsulating the current knowledge landscape and highlighting areas where further research is warranted.

3 Methodology

This section is organized into two distinct parts. The first part centres around Diabetic Foot Ulcer Classification Models, delving into the software-driven aspects of these models. The discussion in this segment revolves around the intricacies of developing and refining classification models for diabetic foot ulcers.

In the second part, the focus shifts to Hardware Implementation on GPUs and FPGAs. This section explores the practical implementation of the aforementioned software models on Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs). It elucidates the hardware-related considerations and optimizations essential for effectively deploying and executing these models on specialized computing architectures. Together, these two parts provide a comprehensive view of the software and hardware dimensions of diabetic foot ulcer classification systems.

3.1 Diabetic foot ulcer classification models

The Diabetic Foot Ulcer Classification Models section encompasses the analysis of a specific dataset, details on image pre-processing techniques, and a focused exploration of proposed models tailored for diabetic foot ulcer identification. It provides a concise yet comprehensive view of the key elements contributing to effective classification in this context.

3.1.1 Data set

Our team collected the dataset from patients in the Diabetic Center Department at Nasiriyah Hospital in Thi-Qar, Iraq, and some samples are shown in Fig. 2. The dataset is public now and available at the following link (https://www.kaggle.com/laithjj/diabetic-foot-ulcer-dfu). The dataset comprises 754 images of the feet of healthy and DFU patients. The images were taken using a Samsung Galaxy Note 8 and iPad, and different angles and lighting conditions were used to capture the images adequately. The images are color, standardized for training the DFU_TFNet, DFU_FNet model, and pre-training well-known models, i.e., VGG-16, GoogleNet, and AlexNet.

3.1.2 Image pre-processing

Some pre-processing tasks were needed before using the dataset for the proposed and pre-trained models. First, the images were cropped to a size of 224 × 224 pixels. The resulting images show patches, so-called Regions of Interest (ROI), since each contains either the ulcer and its surrounding tissues or healthy skin. The dataset's total number of skin patches was 1,609, including 542 healthy skin and 1,067 DFU patches. Next, these patches were categorized by the medical expert into two types: healthy (normal) and DFU (abnormal). The data augmentation techniques were used to increase the dataset and avoid the unbalanced issue. Finally, the labelled patches are all used for training. There are 200 samples

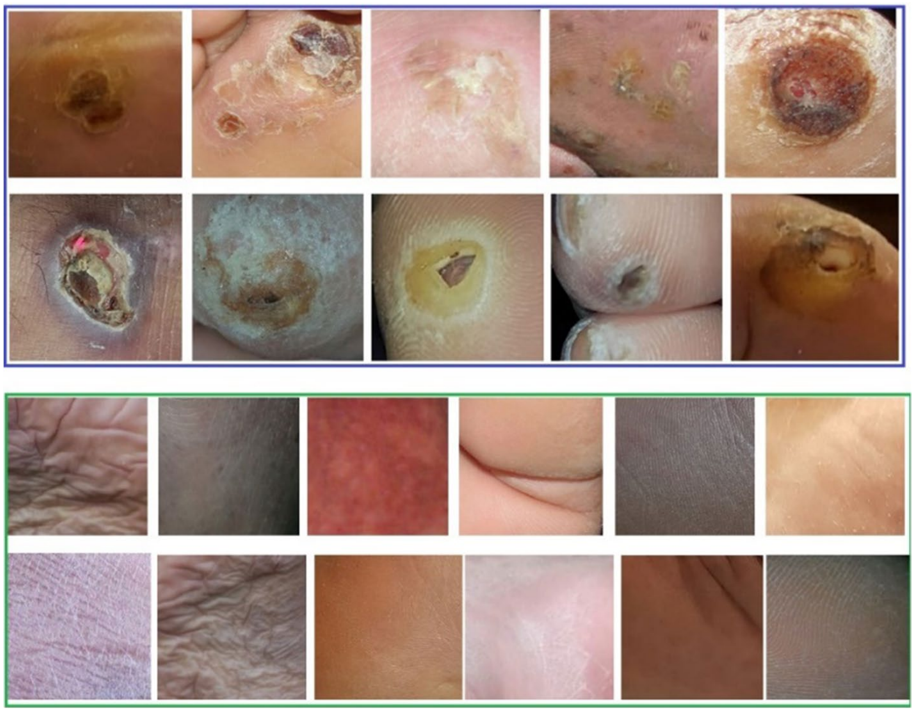

Fig. 2 Samples of our datasets. The blue box samples are abnormal, and the green box samples are normal

that were collected for testing. These samples will be public once the ethical approval is finished. The data augmentation techniques were applied only to the training set. Samples of the initial dataset (before cropping) are illustrated in Fig. 3.

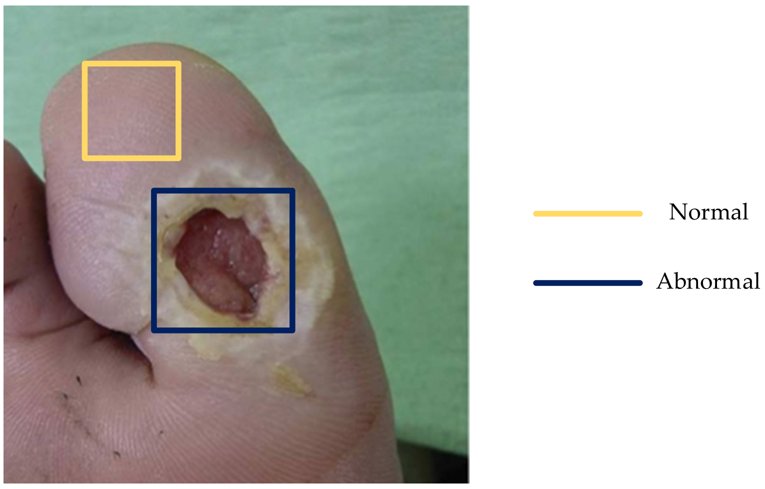

Fig. 3 Normal vs. Abnormal skin patches on a patient’s foot

3.1.3 DFU proposed models

Two CNN models, namely DFU_FNet and DFU_TFNet, have been introduced. DFU_FNet, characterized by its simplicity, extracts features utilized for training classifiers such as SVM and KNN. On the other hand, DFU_TFNet, a deeper model leveraging transfer learning, assesses hardware efficiency across both shallow and deep models. The evaluation parameters employed in setting up these proposed models include a learning rate of 0.001, a batch size of 32, 100 epochs, and using the SGD optimizer.

DFU_FNet model This CNN model is introduced in the proposed system to improve the extraction of the critical features required for DFU classification. The concept of Directed Acyclic Graph (DAG) is the basis of the DFU_FNet model [23]. Two key challenges should be considered when using these types of networks. The first challenge involves enhancing the model accuracy using additional convolution layers compared to the traditional network. Unfortunately, adding layers can decrease the performance of the model. The second challenge is that discriminating between normal and abnormal DFU types requires extracting additional vital features. Thus, a more complicated structure is required. In this research, the width of the DFU_FNet model was increased, which can increase the comparative computing cost.

The structure of the DFU_FNet model, see Fig. 4, was instrumental in accelerating the possible learning details and enhancing its accuracy. The structure included eight layers.

i. Input layer: Three channels with 224×224 pixels each. The final patches were entered through these channels to train the model.
ii. Convolution layer: The output of the input layer is convolved via a set of learnable filters [24]. As the weights identify these filters, two-dimensional filter activation maps are generated, and all filters are slipped across the input volume, height, and width. It should be noted that all filters had the exact depth of the input. Three hyperparameters manipulated the output size: zero-padding (to preserve its size, zeros are padded around the input borders); stride (number of skipped pixels when the filter slides through the image); and depth (number of operated filters identifying structures like a blob, corner, or edge over the image). This work had 17 convolution layers, and all filters had a size of 3×3 pixels. Two types of layers, batch normalization and the rectified linear unit, followed each convolution layer.
iii. Batch normalization layer: A mini-batch was used to normalize each input channel, diminish the sensitivity of the network initialization, and speed up the training process of the CNNs [4]. This was located after the convolutional layer and consisted of 17 layers. Subtracting the mini-batch average from each channel's activations and dividing by the mini-batch standard deviation was the first step of the layer mechanism (i.e., normalizing these activations). Next, a learnable offset β and a learnable scale factor γ were added and scaled, respectively.
iv. Rectified linear unit (ReLU): Data filtering was the main objective of this layer, and the function max (0, x) [4] was used to achieve this goal (note that x is the neuron input).
v. Addition layer: The inputs of two or more neural network layers were added to this layer. To use this layer correctly, these inputs should have similar dimensions.

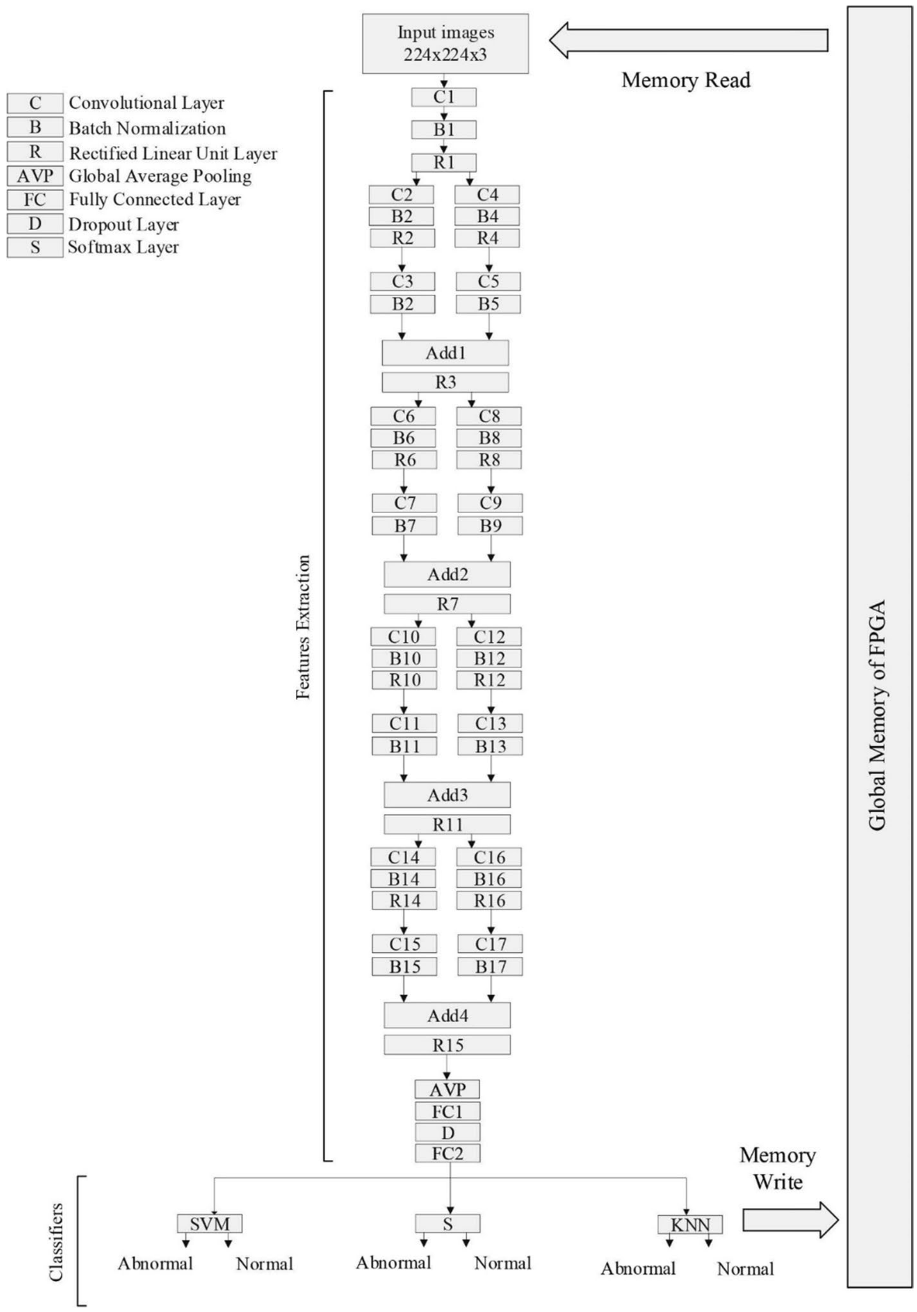

Fig. 4 DFU_FNet architecture on FPGA

vi. Average pooling layer: The input of this layer was partitioned into smaller pooling regions of various dimensions like 3×3, 2×2, etc., to reduce the input size. The average of each small spatial block, which may have vital and a reduced amount of vital pixel information to signify improving features, was then calculated to get normalized feature information [25]. It is important to note that in traditional CNNs, the max-pooling layer is next to the convolution layer and could lose valuable features. In this study, the average pooling was applied at the last partition of the network.

vii. Dropout layer: This is utilized to enhance model performance by preventing the occurrence of overfitting [26]. More specifically, neurons were arbitrarily turned off and on in all layers to avoid overfitting. This model used one dropout layer with a probability of p = 0.5 between the fully connected layers.

viii. Fully connected layer: All previous layer neurons were connected to this layer, thereby mixing the DFU patch classification features. This network used only two fully connected layers.

The overall number of layers in the model was 58. The output layer was placed over the second fully connected layer. The softmax function, located in the output layer, was utilized for classification. The output extracted features of the model were employed to train the classifiers of Support vector machines (SVM) (DFU_FNet + SVM) and k-nearest neighbors (KNN) (DFU_FNet + KNN). The SVM classifier was margin-based. The concept for the SVM algorithm was to determine the optimum partition line between two classes, aiming that objects would have the largest distance from that line. The SVM utilized kernels such as polynomial, linear, Sigmund, and radial basis functions.

In comparison, the KNN classifier of the object depended on the nearest training samples inside the feature space. Several discriminative features were available in each convolution layer. In contrast, skin abnormalities (ulcers) produced higher activations. A public dataset was used to train the proposed and pre-trained models for 100 epochs pending the learning termination.

As a final consideration, the pre-trained models (i.e., VGG-16 [27], AlexNet [28], and GoogleNet [29]) were trained with our dataset using the same training parameters that were utilized to train both the DFU_FNet. The pre-trained models have been fine-tuned for the DUF task by transferring the knowledge of these models learned from the ImageNet dataset.

DFU_TFNet model With this model, we provide a new TL technique for addressing the issue of the small dataset of DFU. This TL type helps to overcome the issue of transfer learning from the pre-trained models of ImageNet to medical imaging applications where ImageNet images are different from medical images, which could not help with small datasets. At the same time, the proposed TL helps to learn the relevant features. It also helps to reduce the time of the annotation process of medical images. Because there has been significant growth in the amount of unannotated medical images, the recommended technique was based on training the DFU_TFNet model on a large number of unannotated images that look similar to DFU images. The collected images for TL include different datasets of skin cancer and wounds. The total number of images is 100 thousand images. The DFU_TFNet model is then fine-tuned and trained on the DFU dataset.

In order to improve the feature extraction as well as address the gradient-vanishing and overfitting concerns, we designed the DFU_TFNet model with the following components that make it robust against the aforementioned concern:

1. Typical convolutional layers at the model's beginning minimize the size of input images.
2. Parallel convolutional layers with varied filter sizes extract diverse data, ensuring the model learns small and large features.

3. For enhanced extracting features, residue and deep interconnectivity are used. Additionally, these connections alleviate the gradient vanishing issue.
4. In order to speed up the training process, batch normalization was used.
5. The vanishing gradient problem is less of an issue because a rectified linear unit (ReLU) does not compress the input value.
6. Dropout to prevent the problem of overfitting.
7. Global average pooling reduces complexity to one dimension. This layer helps reduce overfitting.

Figure 5 provides detailed explanations of the DFU_TFNet model. The model begins with two standard convolutional layers that are applied in succession. The first convolution has a filter size of 3×3, whereas the second has a filter size of 5×5. Following both convolutional layers are the BN and ReLU layers. We avoided using tiny filters, such as 1×1, at the start of the model to avoid losing little features, which would restrict the results. Following the typical convolutional layers are six blocks of parallel convolutional layers. Each block is made up of four parallel convolutional layers with four different filter sizes (1×1, 3×3, 5×5, and 7×7). The output of these four levels is combined at the concatenation layer before proceeding to the next block. Following all convolutional layers in all six blocks are the BN and ReLU layers. The blocks are linked by 10 links. Some are short, others are long, and all have a single convolutional layer. These links maintain the model's capacity to have multiple degrees of features for improved feature representation. Because gradient propagation may occur from several channels, parallel convolutions and connections are necessary. Two connected layers with one dropout layer between them are employed. Our structure combines 34 convolutional layers.

The training process was achieved by three different cycles:

1. Cycle#1: Training the DFU_TFNet only with the DFU dataset.
2. Cycle#2: Training the DFU_TFNet with the DFU dataset plus augmented data.
3. Cycle#3: In the first step, training the DFU_TFNet on a large number of look-like images to DFU, such as the DermNet collection [30]. Then, the DFU_TFNet only with the DFU dataset. Figure 6 depicts the general concept of the transfer learning technique.

Obtaining a large number of labeled images for some medical imaging applications, such as DFU, is challenging. To the authors' knowledge, there are only two public DFU datasets [23] and [30]. Therefore, the proposed TL can solve the issue of the small dataset and help the model to generalize very well. We have used the proposed TL with the DFU_TFNet due to its deep architecture, which requires a large amount of data to perform well. Moreover, the proposed TL may be easily adapted to any medical imaging application utilizing the same domain transfer learning. The scarcity of annotated medical data drove the decision to use same-domain transfer learning. This enabled the model to harness features acquired from ImageNet, expediting training, tailoring generic features to medical imaging tasks, and improving performance through effective generalization. Figure 7 shows some learned features from the first convolutional layer of the DFU_TFNet model.

Another tool used to visualize, Grad-CAM, or Gradient-weighted Class Activation Mapping, stands out as a potent method in interpretability for deep learning models. Its significance lies in being a valuable resource for comprehending and illustrating how neural networks arrive at decisions, especially in tasks related to image classification. The fundamental concept behind Grad-CAM involves emphasizing the sections of an input image

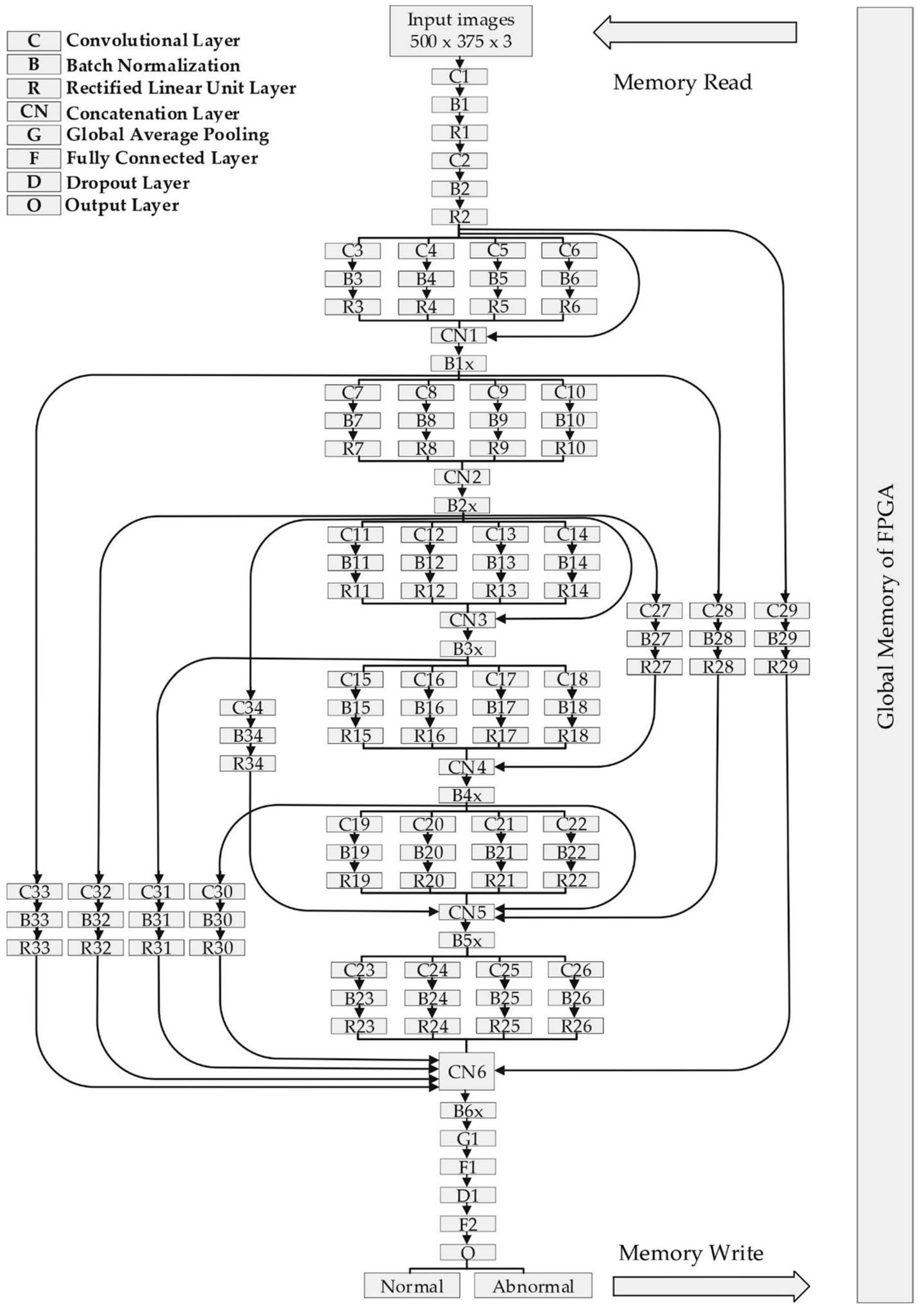

Fig. 5 The DFU_TFNet model structure

that contribute significantly to the model's predictions. This is achieved by utilizing gradient information from the final convolutional layer. The outcome is a heatmap that visually depicts noteworthy areas, shedding light on the features and patterns the network considers

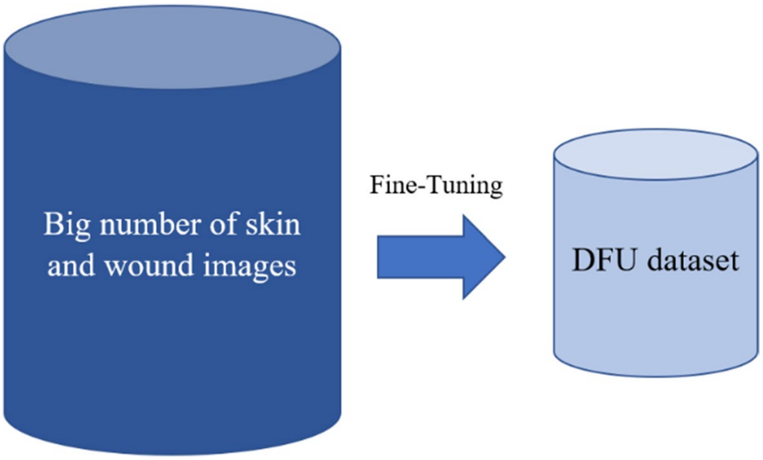

Fig. 6 The transfer learning approach

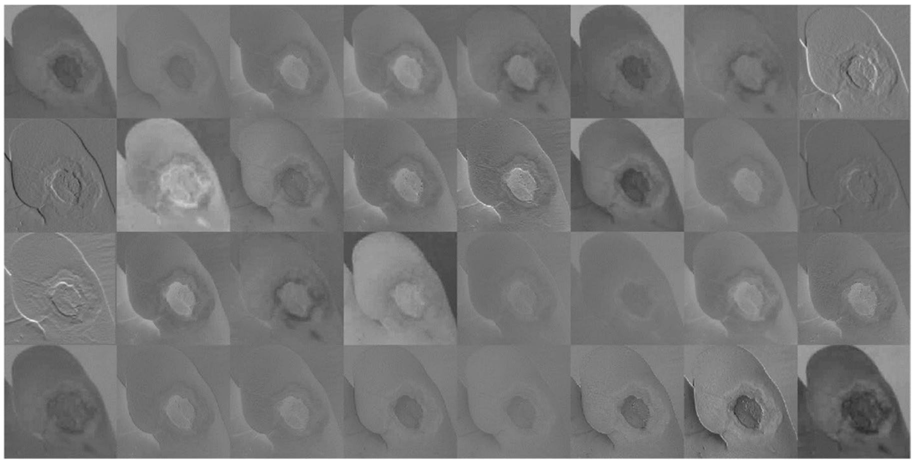

Fig. 7 Some learned features from the first convolutional layer of the DFU_TFNet model

during decision-making. Grad-CAM plays a crucial role in improving model transparency as an indispensable tool for researchers and practitioners aiming to demystify the opaque nature of deep learning models [31].

3.2 Hardware Implementation on GPUs and FPGAs

This section is divided into two parts. The first part explores model implementation on Graphics Processing Units (GPUs), scrutinizing optimizations for this hardware. The second part focuses on Field-Programmable Gate Arrays (FPGAs), elucidating the intricacies of adapting models for efficient execution on these platforms.

3.2.1 GPU

The experimental work used the combination of the Intel i7-9750H processor, RTX 3070 Ti GPU with 8 GB of VRAM, and 16 GB of RAM, providing a robust setup for experimental work. The high clock speed of 3.3 GHz on the i7-9750H is beneficial for CPU-intensive tasks, while the RTX 3070 Ti GPU brings substantial parallel processing power, especially with its 8 GB of VRAM, making it well-suited for deep learning tasks.

Having a powerful GPU like the RTX 3070 Ti significantly accelerates computations, especially in scenarios involving machine learning and deep learning where parallel processing is crucial [32]. The ample 16 GB of RAM ensures the system has enough memory to handle large datasets and complex computations without bottlenecks. This hardware configuration seems well-matched for the experimental work described in the paper, particularly in training and testing various models for diabetic foot ulcer classification. Combining a high-performance CPU and GPU is essential for achieving optimal results in tasks that demand significant computational power.

3.2.2 FPGA

The potential of ML in serving people is growing rapidly, and there is an increasing requirement for ML to operate in real-time. The hardware accelerator-based FPGA is similar to the motherboard CPU in a general-purpose computer. Specifically, the FPGA system (board) can be divided into three primary partitions: FPGA (parallel processing array), HPS (control unit), and the memory partition (software storage), as displayed in Fig. 8.

Usually, HPS is mainly composed of a microprocessor unit (MPU) subsystem with single or dual processors, synchronous DRAM (SDRAM), flash memory controllers, support and interface peripherals, on-chip memories, debug capabilities, and phase-locked loops (PLLs). However, the fabric of FPGA includes a CB (control block), PLLs, and high-speed

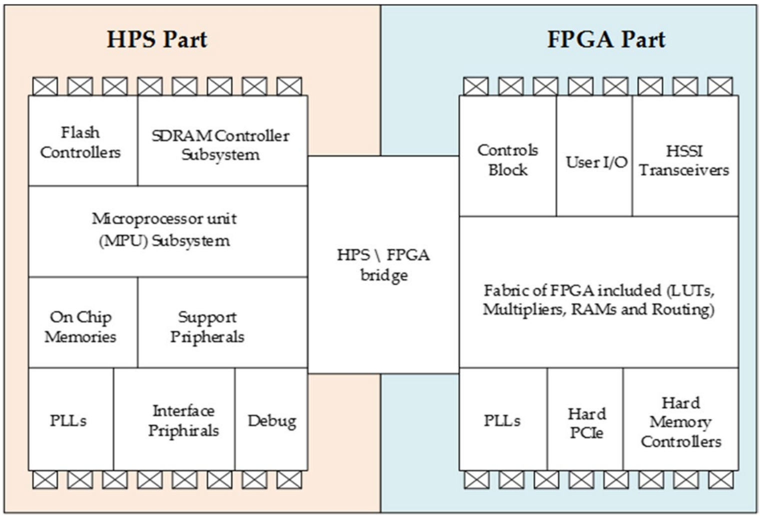

Fig. 8 Block diagram of SoC FPGA

serial interface (HSSI), depending on the device version. Additionally, it can incorporate HSSI transceivers, hard PCI Express (PCIe) controllers, RAM, and multipliers [33, 34].

The HPS and FPGA elements are separated, as shown in Fig. 8. From one of many sources, it booted for HPS and deployed the FPGAs via the HPS or any external device to switch them between.

Please note: FPGA refers to the whole system (the board), and FPGA Part refers to the computational partition of the board.

More specifically, the FPGA performs all calculations and computations similar to the CPU but in parallel with the HPS, interpreting the system and user commands and the memory partition for storing data, system, and user programs. The HPS handles the commands and the rest of the layer computations, including ReLU and max-pooling. Due to memory limitations on the board, loading input and filters to registers is performed line by line in a split manner. In this research, the Altera DE1-SoC board is the type of FPGA system selected [35], as shown in Fig. 9.

Working with FPGA first requires coding the user program using the unique programming language Verilog. The user program and its data are stored in the memory portion. The HPS interprets each line in the user program and generates suitable commands for execution. The user program has several functions and commands (system programs); one of the essential functions is Send_command, which generates a command for the FPGA to set the next state and the number of packets it is supposed to receive.

Initially, the HPS imports an image and decodes it. The model weights are also loaded and ready for processing at the FPGA input. The HPS sends a compute command to the FPGA to perform the required computations and returns the result to the HPS. The result is a feature of the input image sent to the monitor for display through the VGA port on the FPGA board. Note that the input images are pre-processed using MATLAB 2021a. The pre-processing functions include extracting the RGB values, calculating the mean values, and subtracting from the original data. Subtraction of the dataset is very helpful in centering the data, thus boosting the learning speed. Experimentally, when a 16 fixed-point format of 1:7:8 is adopted,

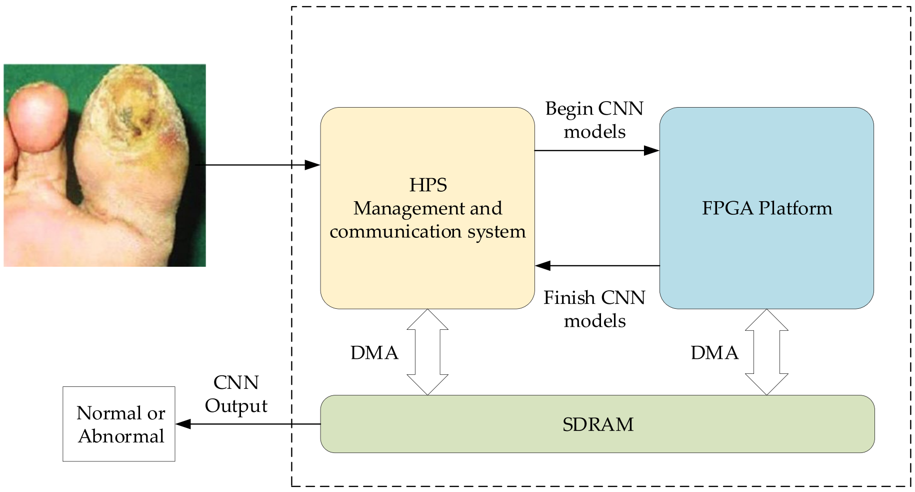

Fig. 9 Block diagram of DFU classification-based FPGA

the input data range is -127 to 127, and all the weights have relatively small values of between 0.03–0.3.

Due to the presence of three input lines, three registers are loaded to implement the pre-trained CNN models using the function load_mem. This function saves the input file pointer, numbering the required locations for reading and setting the padding options. It sends a line from the input file to the FPGA with two data in a packet each time. When padding is enabled, 0 data are added at the line's front and end. The whole line is sent if that line is the first or the last. All the data are sent out at the function end, waiting for the ACK signal and returning with the pointer to read the following line. Next, the function load_fil is applied to load the first 16 filters into the filter register inside the FPGA. This function saves the input file pointer and sends the content of the 16 filters to the FPGA, with each filter containing nine weights. At the end of the function, all the weights are sent out, waiting for the ACK signal and returned with the pointer for reading the next filter. When the input becomes ready, the function compute (which sends a command to the FPGA to compute and wait for the ACK signal when it finishes the computations) is applied to perform the convolutional computation for the first 16 filters as well as the first three lines of the input file. The next step is to read the result of the 16 filters and save it in the local files using the function to get the result. This step is repeated pending the whole filters are handled. The last step is loading a new line, and then the process is repeated. This step is also iterated, assuming all filters are multiplied by the whole lines in the input files.

4 Results and Discussion

DFU proposed a 64-computations array of 16-bit DSP on FPGA DE1-SoC accelerated other pre-training models. This acceleration process contained two elements: the software used for control, known as HPS, and the hardware responsible for convolutional calculations. Only 13 convolutional layers were used to increase efficiency while adjusting to the limitations of FPGA fabrics on DE1-Soc. Software completed The remaining calculations as they could be performed faster than hardware.

Each convolution layer (CONV) mainly comprises separate control logic and parallel adder. At the same time, the multipliers, which serve as the primary computational engines are linked throughout all layers, as seen in Fig. 10. The data input for the convolution is saved in the on-chip buffers, and the multiplier outputs are transferred to CONV for summing and accumulation. The results of CONV are routed to several different on-chip memory, which will be utilized for the next stage.

Accuracy assesses overall correctness, precision evaluates the accuracy of positive predictions, and recall measures the model's ability to capture all positive instances. Together, these metrics provide a nuanced understanding of a classification model's performance. Recall (R) and precision (P) are fundamental metrics for evaluating a suggested method. (see Eqs. 1, 2, 3, 4).

$$Accuracy = {}^{(TP+TN)}\big/_{(TP+TN+FP+FN)} \tag{1}$$

$$Precision\ (P) = {}^{TP}\big/_{TP+FP} \tag{2}$$

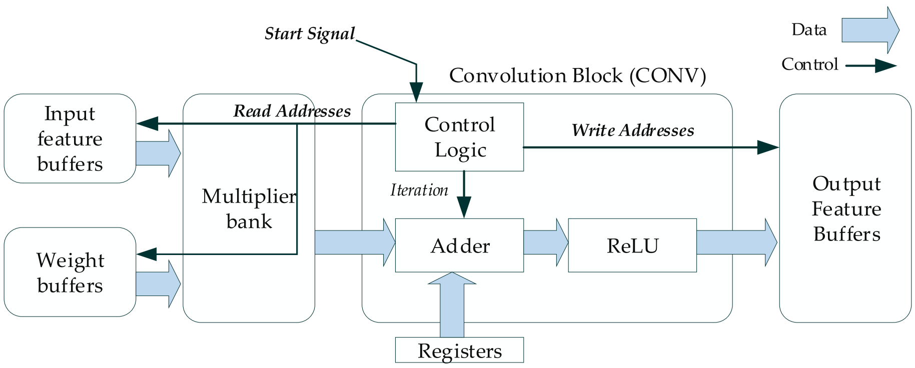

Fig. 10 Convolutional Block Diagram inside FPGA

$$Recall\ (R) = {}^{TP}/_{TP+FN} \tag{3}$$

$$F1 - Score = 2 \times ({}^{P\times R}/_{P+R}) \tag{4}$$

Here, TP (True Positive) denotes the number of relevant images properly identified by the network. A true negative is the number of images properly identified as irrelevant by the network as TN (True Negative). The number of images the network incorrectly classifies as relevant is denoted by the letter FP (False Positive). The number of relevant images the network fails to recognize is denoted by the FN (False Negative).

The evaluation of our models involved calculating key metrics such as accuracy, precision, and F1-Score, and comparing their performance across different scenarios. Table 2 presented a comprehensive comparison of various classifiers, each configured with different approaches, based on their time of processing, power consumption, and performance metrics. Notably, the DFU_TFNet series undergoes cycles, with processing times ranging from 102 to 310 ms. While DFU_TFNet (Cycle#1) achieves the lowest processing time and power consumption at 8.00 W, DFU_TFNet (Cycle#3) attains the highest accuracy, precision, and F1-Score at 99.81%, 99.38%, and 99.25%, respectively. On the other hand, DFU_FNet + SoftMax demonstrates lower processing time and power consumption,

Table 2 Comparison between DFU_FNet and DFU_TFNet

Classifier	Time of processing		Power Consumption		Accuracy	Precision	F1-Score
	FPGA	**GPU**	**FPGA**	**GPU**			
DFU_FNet + SoftMax	143 ms	109 ms	8.60 W	290 W	92.67%	93.21%	93.4%
DFU_FNet + KNN	155 ms	136 ms	8.88 W	290 W	92.85%	92.82%	93.2%
DFU_FNet + SVM	187 ms	119 ms	8.91 W	290 W	94.71%	93.95%	94.5%
DFU_TFNet (Cycle#1)	102 ms	97 ms	8.00 W	290 W	88.19%	86.71	86.00%
DFU_TFNet (Cycle#2)	227 ms	177 ms	9.02 W	290 W	96.34%	96.44%	96.25%
DFU_TFNet (Cycle#3), Ours	**310 ms**	**184 ms**	**9.16 W**	**290 W**	**99.81%**	**99.38%**	**99.25%**

making it an efficient alternative. Figure 11 shows the heatmap through virtualization using the DFU_TFNet model with Grad-CAM.

Moving to Table 3, shown a comparative analysis of various deep learning models, including AlexNet, VGG16, GoogleNet, DFU_FNet + SVM, and DFU_TFNet (Cycle#3), across key performance metrics. Notably, the processing times for these models vary, with FPGA consistently demonstrating lower processing times than GPU. Furthermore, the power consumption of FPGA is considerably lower than that of GPU across all models, underscoring its energy efficiency. In terms of accuracy, precision, and F1-Score, DFU_TFNet (Cycle#3) emerges as a standout performer, boasting an impressive accuracy of 99.81%, precision of 99.38%, and an F1-Score of 99.25%. These metrics reflect the model's robust performance. The trade-offs between FPGA and GPU are evident, with FPGA offering energy efficiency at the cost of slightly longer processing times. The FPGA evaluations in Table 4 considered essential resources like total logic elements, block memory, and logic registers within the DE1-Soc.

The processing time comparison between GPU and FPGA revealed that while GPU is faster, it demands significant power. Conversely, FPGA exhibits substantially lower power consumption, making it an attractive choice for smart devices with limited battery resources. With advancements in FPGA properties, processing times could become comparable to GPUs. As summarized in Table 5, the overall results guide us to conclude the preferred platforms based on the achieved metrics and performance benchmarks.

Table 6 presented a comparative analysis of various deep learning models, predominantly focused on DFU detection. The EfficientNet [19] achieved an impressive 98.97% accuracy, accompanied by high F1-score, recall, and precision on a GPU, with correspondingly high-power consumption. ResNet50 [20] demonstrated a notable 99.49% accuracy for Ischaemia and 84.76% for infection, also on a GPU with high power consumption. DFU_QUTNet [23] and DFUNet [26], both utilizing GPUs, exhibited a F1-score of 94.5% and an accuracy of 96.1%, respectively. The proposed model, DFU_TFNet (Cycle#3), stands out with remarkable accuracy, precision, recall, and F1-score of 99.81%, 99.38%, 99.76%, and 99.25%, respectively. Notably, DFU_TFNet utilizes both FPGA and GPU, potentially mitigating power consumption with a low setting on FPGA. This combination of high performance and potentially lower power usage makes DFU_TFNet an intriguing prospect for real-world applications in medical imaging and diagnostics.

5 Conclusions

This research proposes new diagnostic tools with real-time processing capabilities for DFU classification, which addresses a significant healthcare challenge. The key findings of this research include the effectiveness of pre-trained CNN models, namely, DFU_FNet and DFU_TFNet, in automatically categorizing DFU cases into normal and abnormal foot skin. These models were designed to overcome deep learning pitfalls, utilizing techniques such as domain-transfer learning. The results of this research indicate that when compared with various classifiers like SVM, KNN, and pre-trained CNN models like AlexNet, VGG16, and GoogleNet, DFU_FNet and DFU_TFNet exhibit superior performance. The models were trained and tested on different HPC parallel platforms, including GPUs and FPGAs, significantly reducing power consumption and execution time. Additionally, features extracted by DFU_FNet were leveraged to train SVM and KNN classifiers, further enhancing the overall classification process. The proposed framework, particularly DFU_TFNet

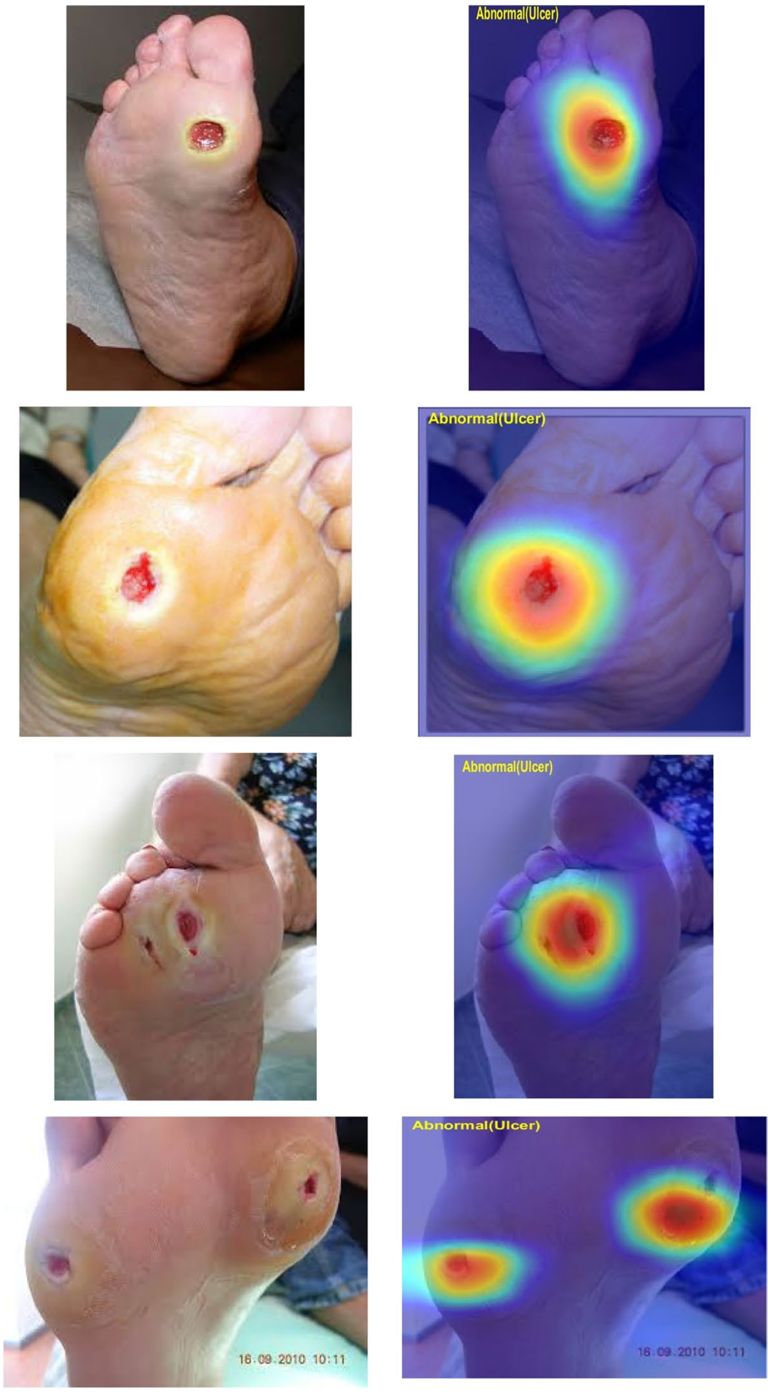

Fig. 11 Grad-CAM heatmap visualization for DFU_TFNet (Cycle#3) model

Table 3 Our proposal vs. pre-trained CNN models

Models	Time of processing		Power Consumption		Accuracy	Precision	F1-Score
	FPGA	**GPU**	**FPGA**	**GPU**			
AlexNet	153 ms	136 ms	2.1 W	290 W	89.11%	88.35%	88.1%
VGG16	4.32 s	8.6 s	5 W	290 W	90.37%	89.35%	90.9%
GoogleNet	7.38 s	14.3 s	23 W	290 W	91.93%	92.5%	92.9%
DFU_FNet+SVM, Ours	**187 ms**	**119 ms**	**8.98 W**	**290 W**	**94.71%**	**93.95%**	**94.5%**
DFU_TFNet (Cycle#3), Ours	**310 ms**	**184 ms**	**9.16 W**	**290 W**	**99.81%**	**99.38%**	**99.25%**

Table 4 Summary of resources for DE1- Soc

Models	Total logic elements out of 32,070	Total block memory out of 4,056,280 bits	Total logic registers
AlexNet	11,983	63,789	32,985
VGG16	21,617	81,920	44,773
GoogleNet	29,833	240,763	89,762
DFU_FNet+SVM	12,974	94,670	65,765
DFU_TFNet (Cycle#3), Ours	**18,018**	**121,471**	**89,111**

Table 5 Platforms are recommended based on hardware analysis

Feature	Evaluation	Winner
Training	GPU floating-point performance has improved	GPU
Analyzing large amounts of data	FPGAs are excellent for inline computation	FPGA
Power Consumption	Personalized designs might be preferable	FPGA
Time of processing	GPUs triumph due to their superior processing power	GPU
Interfaces	FPGAs may integrate a wide range of interfaces	FPGA
Changeability	GPUs make it simpler to make modifications to application capabilities	GPU
Customization	FPGAs enable more adaptability	FPGA
Size	FPGA's lower power consumption leads to smaller volume solutions	FPGA

(Cycle#3), achieved an impressive accuracy, precision, recall, and F1-score of 99.81%, 99.38%, 99.76%, and 99.25%, respectively, surpassing current methodologies. The FPGA implementation, utilizing DE1-SoC resources, exhibited a reasonable power consumption of 9.16W. Future directions involve transforming the model into a wearable smart application, enabling patients to monitor their condition anytime, anywhere, with prolonged battery life and swift processing. Training images will be securely stored on an online server to prioritize data privacy. During testing, the wearable device designated for testing purposes will reduce data exposure and align with security best practices. Furthermore, strong encryption will safeguard data transmission between the device and the server.

Table 6 Comparison between out proposed model and other studies

Ref	Name of model	Evaluation parameters	Hardware used	Power Consumption (High | Low)
[19]	EfficientNet	Accuracy, F1-score, recall, and precision of 98.97%, 98%, 98%, and 99%, respectively	GPU	High
[20]	ResNet50	Accuracy of 99.49% and 84.76% for Ischaemia and infection, respectively	GPU	High
[23]	DFU_QUTNet	F1-score 94.5%	GPU	High
[26]	DFUNet	Accuracy 96.1%	GPU	High
Our proposed	DFU_TFNet (Cycle#3)	Accuracy, precision, recall, F1-score of 99.81%, 99.38, 99.76%, 99.25% respectively	FPGA | GPU	Low | High

Appendix 1

Preliminaries and Definitions

Deep learning

A traditional technique used in artificial intelligence is to employ computations to solve problems analytically, and this requires specific knowledge of the field and the problem under consideration [1]. This approach was able to address simple problems since the programs were small and accessible in design. In addition, domain-specific knowledge could convert an ordinary volume of data into helpful representations for learning. The progress of artificial intelligence has enhanced interest in solving extra-complicated problems where relevant knowledge is hard to extract. However, professional knowledge which is related to problems like medical research, speech transcription, and face recognition is hard to express, and traditional techniques are less successful at processing implied information in the raw data. In addition, the great evolution in data storage and acquisition indicates that a significant capacity to employ implied information is more important than ever. Lately, different applications demonstrating innovative performance have been based on the recent technique of DL. In this technique, the implied information is extracted automatically via learning task-related features available in the raw data. Several reviews have recently developed due to the interest in this research field [2, 3].

The DL applications and models have several common characteristics, which are well matched for parallelization utilizing hardware accelerators, including [4, 5]:

a. Data parallelism – In pixel-based sensory input, the parallelism characteristic establishes itself in tasks by simultaneously applying it to the local areas or the whole pixels. In addition, nearly all common methods of training models are through processing "mini-batches" of normally hundreds/thousands of examples, and not by processing one example at a time.
b. Model parallelism – Such models include biologically inspired models. They consist of redundant processing units, i.e., they can be updated in parallel and allocated in hardware. One of the recent works on accelerating CNN utilizing multi-GPUs has employed leading-edge approaches for balancing model-based parallelism and data, in which dissimilar segments of the architecture are parallelized in diverse but optimum methods [6].
c. Pipeline parallelism – The computation in architectures that has a feed-ahead nature such as CNN (i.e., well-suited hardware for exploring pipeline parallelism like FPGA) can present an actual benefit. However, GPUs and GPPs depend on processing parallel themes on multi-core hardware, while FPGA can produce personalized hardware circuits, which are characterized as integrally multiple threads and intensely pipelined.

FPGA

Assessing the acceleration of various hardware platforms should take into account the transaction between performance and flexibility. Considering the spectrum of the hardware platforms, GPPs are at one end of the spectrum. They offer simplicity of use and extensive flexibility, but with comparatively inefficient performance. Such platforms are suitable for an extensive variety of uses/reuses, are made inexpensively, and tend to be easily accessible. At the other end of the spectrum are the ASICs (application-specific integrated circuits). These offer great performance and are cost-effective, but they are inflexible and time-consuming to produce as they are specialized for use with certain systems.

The development of customized CNN accelerators for each CNN model has been achieved before using register transfer level (RTL)—based FPGA approaches [7]. It is possible to make use of fine-grained hardware-level optimization to achieve excellent performance and energy efficiency in this manner. However, the customized RTL accelerator takes a significant amount of development time, such as several months, making it impossible to keep up with the constantly developing CNN algorithms and the wide range of applications that are available as displayed in Fig. 12.

However, FPGAs work cooperatively between the two ends of this spectrum. FPGAs belong to a common class of PLD (programmable logic devices) and are reconfigurable integrated circuits. This offers the reconfigurable flexibility of the general-purpose processors with the performance advantages of the integrated circuit. FPGAs can execute combinational logic via the utilization of look-up tables (LUT) and sequential logic via the use of flip-flops from a low-level point of view. Current FPGAs are composed of hardware components for dealing with frequent tasks like Block RAM, arithmetic cores, communication cores, and full processor cores. Current FPGAs also tend to have a system-on-chip (SoC) design method since FPGA and ARM co-processors are usually obtained on similar frameworks. Altera and Xilinx dominate the current FPGA market and together represent 85% of the total market [8]. For fixed-function logic, FPGAs are fast devices taking the place of application-specific integrated circuits and general-specific standard products. This is important in relation to DL, as FPGAs offer a clear capacity for achieving real-time results beyond the possible tasks able to be performed by conventional general-purpose CPU processors [9].

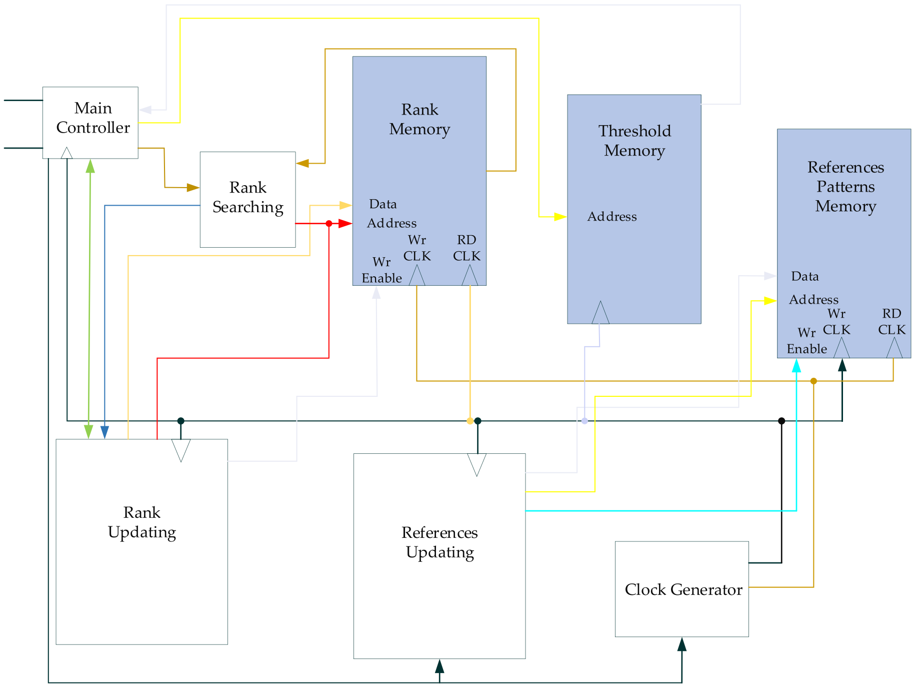

Fig. 12 Implementation of CNN learnable element in FPGA according to RTL block diagram

Author contributions Conceptualization, M.A.F., L.A., and Y.G.; methodology, M.A.F., L.A., and Y.G.; software, L.A., and M.A.F.; validation, M.A.F., J.S., L.A., and Y.G.; formal analysis, M.A.F., J.S., L.A., and Y.G.; investigation, M.A.F., J.S., Y.D., L.A., and Y.G.; resources, L.A., and M.A.F.; data curation, L.A. and M.A.F.; writing—original draft preparation, M.A.F., J.S., Y.D., L.A., and O.A.-S; writing—review and editing, M.A.F., J.S., Y.D., L.A., and Y.G.; visualization, L.A. and M.A.F.; supervision, L.A., J.S., and Y.D.; project administration, L.A., J.S., Y.G. and Y.D.; funding acquisition, L.A., Y.G.; All authors have read and agreed to the published version of the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions The authors would like to acknowledge the support received through the following funding schemes of Australian Government: ARC Industrial Transformation Training Centre (ITTC) for Joint Biomechanics under grant IC190100020. Laith Alzubaidi would like to acknowledge the support received through the QUT ECR SCHEME 2022, The Queensland University of Technology.

Data availability The collected dataset has been uploaded. ULR: https://www.kaggle.com/datasets/laithjj/diabetic-foot-ulcer-dfu

Declarations

Conflicts of interest The authors declare no conflict of interest.

Institutional review board Not applicable.

Informed consent Not applicable.

References

1. Alzubaidi L, Bai J, Al-Sabaawi A, Santamaría J, Albahri AS, Al-dabbagh BSN, Fadhel MA et al. (2023) A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. J Big Data 10, (1): 46
2. Albahri AS, Duhaim AM, Fadhel MA, Alnoor A, Baqer NS, Alzubaidi L, Albahri OS et al (2023) A systematic review of trustworthy and explainable artificial intelligence in healthcare: Assessment of quality, bias risk, and data fusion. Information Fusion
3. Nozawa T, Uchiyama M, Honda K, Nakano T, Miyake Y (2020) Speech Discrimination in Real-World Group Communication Using Audio-Motion Multimodal Sensing. Sensors 20(10):2948
4. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, (1): 1–74
5. Seritan S, Bannwarth C, Fales BS, Hohenstein EG, Isborn CM, Kokkila-Schumacher SIL, Li X et al. (2021) TeraChem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. Wiley Interdisciplinary Reviews: Computational Molecular Science 11, (2): e1494
6. Coates A, Huval B, Wang T, Wu D, Catanzaro B, Andrew N (2013) Deep learning with cots HPC systems. In Proceedings of the 30th International Conference on Machine Learning, 1337–1345
7. Fowers J, Brown G, Cooke P, Stitt G (2012) A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. In Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 47–56

8. Seng KP, Lee PJ, Ang LM (2021) Embedded Intelligence on FPGA: Survey, Applications and Challenges. Electronics 10, no. 8: 895
9. Cox CE, Ekkehard Blanz W (1992) GANGLION-a fast field-programmable gate array implementation of a connectionist classifier. IEEE J Solid-State Circuits 27, (3): 288–299
10. Cloutier J, Cosatto E, Pigeon S, Boyer FR, Simard PY (1996) Vip: An fpga-based processor for image processing and neural networks. In Proceedings of the Fifth International Conference on Microelectronics for Neural Networks, pp. 330–336. IEEE
11. Ovtcharov K, Ruwase O, Kim J-Y, Fowers J, Strauss K, Chung ES (2015) Accelerating deep convolutional neural networks using specialized hardware. Microsoft Research Whitepaper 2(11):1–4
12. Zhang C, Li P, Sun G, Guan Y, Xiao B, Cong J (2015) Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, pp. 161–170
13. Baba A, Bonny T (2023) FPGA-based parallel implementation to classify Hyperspectral images by using a Convolutional Neural Network. Integration 92:15–23
14. Parizotto R, Coelho BL, Nunes DC, Haque I, Schaeffer-Filho A (2023) Offloading Machine Learning to Programmable Data Planes: A Systematic Survey. ACM Computing Surveys
15. Almomany A, Ayyad WR, Jarrah A (2022) Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study. Journal of King Saud University-Computer and Information Sciences 34(6):3815–3827
16. Mohamed NA, Cavallaro JR (2023) A Unified Parallel CORDIC-based Hardware Architecture for LSTM Network Acceleration. IEEE Transactions on Computers
17. Heartlin MH, Kayalvizhi R, Malarvizhi S, Venkatraman R, Patil S, Senthil Kumar A (2023) Real-time deployment of BI-RADS breast cancer classifier using deep-learning and FPGA techniques. J Real-Time Image Process 20, no. 4: 80
18. Paul K and Rajopadhye S (2006) Back-propagation algorithm achieving 5 gops on the virtex-e. In FPGA Implementations of Neural Networks, pp. 137–165. Springer: Boston
19. Thotad PN, Bharamagoudar GR, Anami BS (2023) Diabetic foot ulcer detection using deep learning approaches. Sensors International 4:100210
20. Ahsan M, Naz S, Ahmad R, Ehsan H, Sikandar A (2023) A deep learning approach for diabetic foot ulcer classification and recognition. Information 14(1):36
21. Khandakar A, Chowdhury MEH, Reaz MBI, Md Ali SH, Abbas TO, Alam T, Ayari MA et al. (2022) Thermal change index-based diabetic foot thermogram image classification using machine learning techniques. Sensors 22, no. 5: 1793
22. Wang S, Wang J, Zhu MX, Tan Q (2022) Machine learning for the prediction of minor amputation in University of Texas grade 3 diabetic foot ulcers. Plos one 17, no. 12: e0278445
23. Alzubaidi L, Fadhel MA, Oleiwi SR, Al-Shamma O, Zhang J (2020) DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network. Multimed Tools Appl 79(21):15655–15677
24. Sarvamangala DR, Kulkarni RV (2022) Convolutional neural networks in medical image understanding: a survey. Evolutionary intelligence 15, no. 1: 1–22
25. Pattanayak S (2023) Pro Deep Learning with TensorFlow 2.0: A Mathematical Approach to Advanced Artificial Intelligence in Python. Apress
26. Goyal M, Reeves ND, Davison AK, Rajbhandari S, Spragg J, Yap MH (2020) DFUNet: Convolutional Neural Networks for Diabetic Foot Ulcer Classification." IEEE Transactions on Emerging Topics in Computational Intelligence 4, no. 5: 728–739
27. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
28. Iman M, Arabnia HR, Rasheed K (2023) A review of deep transfer learning and recent advancements. Technologies 11, no. 2: 40
29. Wang X, Chen G, Qian G, Gao P, Wei X-Y, Wang Y, Tian Y, Gao W (2023) Large-scale multimodal pre-trained models: A comprehensive survey. Machine Intelligence Research: 1–36
30. Dermnetnz Online Medical Resources | Home. Available online: https://www.dermnetnz.org/ (accessed on 5 March 2022)
31. Lim WX, Chen ZY, Ahmed A (2022) The adoption of deep learning interpretability techniques on diabetic retinopathy analysis: a review. Med Biol Eng Comput 60, no. 3: 633–642
32. Kazim M, Hong JG, Kim M-G, Kim K-KK (2023) Recent Advances in Path Integral Control for Trajectory Optimization: An Overview in Theoretical and Algorithmic Perspectives. arXiv preprint arXiv:2309.12566
33. Manual, DE1-SoC User. "Terasic Inc." Hsinchu, Taiwan, Feb (2014)
34. Akesson B, Nasri M, Nelissen G, Altmeyer S, Davis RI (2022) A comprehensive survey of industry practice in real-time systems. Real-Time Systems 58(3):358–398

35. Kashani S, Beuchat R (2020) Soc-fpga design guide de1-soc edition

Authors and Affiliations

Mohammed A. Fadhel[1] · Laith Alzubaidi[2,3,4] · Yuantong Gu[2,3] · Jose Santamaría[5] · Ye Duan[6]

✉ Laith Alzubaidi
l.alzubaidi@qut.edu.au

[1] College of Computer Science and Information Technology, University of Sumer, 64005, Thi Qar, Iraq

[2] School of Mechanical, Medical and Process Engineering, Queensland University of Technology, Brisbane, QLD 4000, Australia

[3] ARC Industrial Transformation Training Centre—Joint Biomechanics, Queensland University of Technology, Brisbane, QLD 4000, Australia

[4] Akunah Medical Technology Pty Ltd Company, Brisbane, QLD 4120, Australia

[5] Department of Computer Science, University of Jaén, 23071 Jaén, Spain

[6] School of Computing, Clemson University, Clemson, SC 29631, USA