IoT

MDPI

Article

An Optimised CNN Hardware Accelerator Applicable to IoT End Nodes for Disruptive Healthcare

Arfan Ghani [1,*], Akinyemi Aina [2] and Chan Hwang See [3]

[1] Department of Computer Science and Engineering, School of Engineering and Computing, American University of Ras al Khaimah, Ras al Khaimah 72603, United Arab Emirates
[2] SMC European Technical Centre, Vincent Avenue, Crownhill, Milton Keynes MK8 0AN, UK; aaina@smceu.com
[3] School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK; c.see@napier.ac.uk
* Correspondence: arfan.ghani@aurak.ac.ae

Abstract: In the evolving landscape of computer vision, the integration of machine learning algorithms with cutting-edge hardware platforms is increasingly pivotal, especially in the context of disruptive healthcare systems. This study introduces an optimized implementation of a Convolutional Neural Network (CNN) on the Basys3 FPGA, designed specifically for accelerating the classification of cytotoxicity in human kidney cells. Addressing the challenges posed by constrained dataset sizes, compute-intensive AI algorithms, and hardware limitations, the approach presented in this paper leverages efficient image augmentation and pre-processing techniques to enhance both prediction accuracy and the training efficiency. The CNN, quantized to 8-bit precision and tailored for the FPGA's resource constraints, significantly accelerates training by a factor of three while consuming only 1.33% of the power compared to a traditional software-based CNN running on an NVIDIA K80 GPU. The network architecture, composed of seven layers with excessive hyperparameters, processes downscale grayscale images, achieving notable gains in speed and energy efficiency. A cornerstone of our methodology is the emphasis on parallel processing, data type optimization, and reduced logic space usage through 8-bit integer operations. We conducted extensive image pre-processing, including histogram equalization and artefact removal, to maximize feature extraction from the augmented dataset. Achieving an accuracy of approximately 91% on unseen images, this FPGA-implemented CNN demonstrates the potential for rapid, low-power medical diagnostics within a broader IoT ecosystem where data could be assessed online. This work underscores the feasibility of deploying resource-efficient AI models in environments where traditional high-performance computing resources are unavailable, typically in healthcare settings, paving the way for and contributing to advanced computer vision techniques in embedded systems.

Keywords: computer vision; applied artificial intelligence; IoT for healthcare; CNN; integer-based architecture

Citation: Ghani, A.; Aina, A.; Hwang See, C. An Optimised CNN Hardware Accelerator Applicable to IoT End Nodes for Disruptive Healthcare. *IoT* **2024**, *5*, 901–921. https://doi.org/10.3390/iot5040041

Academic Editor: Amiya Nayak

Received: 16 September 2024
Revised: 2 December 2024
Accepted: 3 December 2024
Published: 6 December 2024

1. Introduction

The recent advancements in machine learning (ML) techniques and their integration into computer vision have become increasingly prominent, offering transformative benefits for medical research and treatment. In particular, CNNs have emerged as a highly effective approach for various image-based tasks, including medical image classification and diagnosis. CNNs excel at identifying patterns and features in image data, making them invaluable for medical applications where accurate image analysis is crucial [1]. The advent of specialized hardware, such as Field-Programmable Gate Arrays (FPGAs), has further enhanced the capabilities of CNNs by providing a platform for accelerated computation and improved energy efficiency [2]. Deploying CNNs on FPGAs marks a significant advancement in the field of medical image analysis. Traditional software implementations, while effective,

are often limited by the processing speed and power consumption. In contrast, FPGAs offer a flexible hardware solution that can be optimized for specific tasks, such as image classification, resulting in faster processing times and reduced energy usage [3]. This paper presents an optimized CNN implementation on a Basys3 FPGA designed specifically for the classification of extracellular Ca^{2+} mutations in human proximal kidney cells (HK-2s). The Basys3 FPGA device has a small form factor and is optimized for power efficiency. As the CNN model proposed in this work is not too deep, it provides sufficient performance and the model can fit well within the constraints of the Artix-7 architecture, offering fast and parallelized computation at a low cost. Hence, the device used was selected primarily for its balance between availability, affordability, and sufficient computational resources to demonstrate the feasibility of an edge-compatible solution for medical diagnostics [4].

One of the main challenges addressed in this study is the utilization of a limited dataset of light microscopy samples. Effective data augmentation and pre-processing are critical in this context to maximize the utility of the available data and improve the performance of the CNN. Techniques such as rotation, cropping, and histogram equalization were employed to enhance the dataset's quality and ensure that the CNN could achieve high prediction accuracy. These pre-processing steps are essential for mitigating artefacts and ensuring that the model can generalize well across different image conditions [5]. The use of image augmentation techniques also helps in addressing the issue of overfitting, a common problem when training deep learning models on small datasets [6]. The quantization of the CNN to an 8-bit format was a key optimization strategy used to adapt the model for the limited resources of the Basys3 FPGA. Quantization involves reducing the precision of the model's weights and activations, which can lead to significant reductions in memory usage and computational requirements [7].

Despite the potential trade-off in accuracy, the careful design and optimization of the quantized model can mitigate these effects and allow for efficient FPGA implementation. Significant model pruning and data reallocation were performed to fit the CNN, with its seven-layer architecture and 13,464 hyperparameters, into the FPGA's block RAM (BRAM) constraint. The performance of the FPGA-implemented CNN was evaluated against a software-based CNN running on an NVIDIA K80 GPU [8]. The FPGA implementation demonstrated a threefold increase in the initial training speed while consuming only 1.33% of the power, approximately 4 watts. This substantial improvement in both speed and power efficiency highlights the advantages of FPGA-based implementations, particularly in scenarios where computational resources and power availability are limited for broader IoT-based solutions. The low-power and high-speed capabilities of the FPGA make it an attractive option for real-time medical diagnostics, where rapid and efficient analysis is crucial [9]. This paper also discusses the broader implications of these advancements for medical image classification. The ability to perform training and classification tasks on unseen images with minimal hardware resources and power consumption is particularly relevant in clinical settings where access to a high-performance computing infrastructure is limited [10]. By optimizing the CNN architecture for FPGA deployment, this work demonstrates a viable approach for achieving both speed and energy efficiency in low-power medical diagnostics [11,12]. Integrating deep learning with FPGA technology significantly advances medical image classification. The optimized CNN implementation on the Basys3 FPGA not only enhances the training efficiency and power consumption but also contributes to the broader understanding of hardware-specific adaptations in machine learning. This approach has the potential to advance medical diagnostics by providing a cost-effective and efficient IoT-based solution where imaging datasets could be hosted on cloud platforms for analysis and diagnostics [13].

The primary objectives and significant contributions of this paper are as follows:

1. Quantizing the CNN to 8-bit precision and leveraging FPGA-specific optimization for toxicity classification.
2. Demonstrating the use of FPGA technology for healthcare engineering and its potential in clinical settings.

3. Offering a viable solution for real-time, low-cost medical diagnostics.

The rest of this paper is organized as follows: Section 2 presents the software design and image pre-processing techniques. Section 3 discusses the hardware design, the forward pass algorithm, caching, and backpropagation for training an integer-based CNN. Section 4 presents the hardware and software results, followed by the discussion, future work and limitations, and conclusion in Sections 5–7, respectively.

2. Materials and Methods

When implementing the hardware-based Convolutional Neural Network (CNN), several critical factors were considered to optimize both performance and efficiency. One of the primary considerations was parallelism, a crucial aspect for accelerating CNN operations on an FPGA. The FPGA's architecture inherently supports parallel processing, which is ideal for the parallelizable nature of CNN computations, including convolutions and pooling operations [14]. By leveraging this parallelism, the design aimed to significantly enhance the processing speed and overall efficiency. Another significant factor was the optimization of data types. To address the constraints of FPGA resources, the CNN implementation utilized 8-bit integers for data representation. This quantization strategy was instrumental in reducing the memory footprint and power consumption of the FPGA implementation. Prior research has demonstrated that binary and fixed-point precision algorithms can achieve high accuracy while optimizing hardware efficiency [15,16]. The use of 8-bit quantization adopted in this research allowed for a more compact model that fits within the FPGA's limited resources without a substantial loss of accuracy. This approach enables the CNN to be effectively deployed on resource-constrained FPGA hardware. The speed and throughput were also paramount in the design of the FPGA-based CNN. The implementation sought to accelerate both the training and inference processes by employing techniques such as pipelining and parallel processing. These methods maximize throughput by processing multiple image segments simultaneously, thus reducing the overall computation time [17]. The FPGA-based CNN demonstrated a threefold increase in the training speed compared to a software implementation running on a personal laptop HP ZBook containing NVIDIA K80 GPU [8]. This reduction in the training time was achieved through efficient hardware utilization and optimized computation [18]. Additionally, the reduction in the training time was a critical objective. The FPGA implementation provides a significant advantage over traditional software-based methods by minimizing the time required for model training. The efficient use of FPGA resources, coupled with optimized computation techniques, facilitates faster training processes compared to conventional software implementations [19]. In the data pre-processing stage, several techniques were applied to enhance feature extraction and improve classification performance. For multiclass classification tasks, effective image processing is essential for segmenting and highlighting relevant features within the images. The pre-processing involved histogram equalization, enhancing contrast and improving the features' visibility. Then, median filtering was applied to reduce noise and preserve important edges [20,21]. Afterwards, grayscale conversion was performed to simplify the image data by focusing on intensity values, which reduces the computational complexity [20]. These pre-processing steps not only enhance the quality of the input images but also contribute to faster training times. By simplifying the data and reducing the need for extensive computations, the pre-processing techniques act as a form of quantization, optimizing the use of available FPGA resources [22]. The combination of these methods ensures that the CNN can process high-quality data efficiently, leading to improved model performance and a reduced training time.

To increase the diversity of the images, data augmentation was performed using Keras's "ImageDataGenerator" class. As shown in Figure 1, the image was randomly rotated within a range of ± 40 degrees, introducing rotational variation. It was horizontally and vertically shifted by 20% of its width and height to incorporate slight repositioning. Furthermore, shear transformations were applied to distort the image by zooming randomly by 20%. The image was flipped horizontally to create a mirrored version of the

original image, and the nearest pixel value was used to fill any new areas created by these transformations. The results of the augmented image processing can be observed in Figure 1. In this figure, only one sample of the highly stressed images is shown. However, the authors' previously published work [8] provides a detailed description of the data acquisition technique, as well as the image samples for normal, moderately stressed, and highly stressed conditions. The original images were rotated by 45 degrees in both clockwise and counter-clockwise directions. This rotation introduced artefacts around the edges of the augmented images as illustrated in Figure 2. Furthermore, it was noted that the pixel distributions of the non-cropped augmented images were too similar in range and depth, which could hinder the efficiency of training with the custom CNN. To address this, a histogram analysis of the pixel distribution was performed, highlighting the differences between the cropped and non-cropped images as shown in Figure 2. The goal was to ensure a more diverse and informative feature set for training, thus enhancing the CNN's ability to generalize from the data. Given the memory and computational constraints of our low-power, cost-effective FPGA, it was crucial to extract as many meaningful features from the dataset as possible. The dataset images, being of low resolution, necessitated the application of specific image processing techniques. To improve the feature extraction process, high-contrast and band-pass filters were employed to increase the pixel distribution levels. These techniques were essential for creating a more distinct feature set and improving the model's performance. Additionally, methods such as segmentation and masking, as detailed in previous works [23–25], were utilized to further refine the feature extraction process. These techniques helped in isolating relevant features from the dataset images, which is vital for effective training on a resource-constrained FPGA platform. The aim was to optimize the image pre-processing pipeline to balance feature richness with the FPGA's limited computational resources, ultimately achieving a better classification performance.

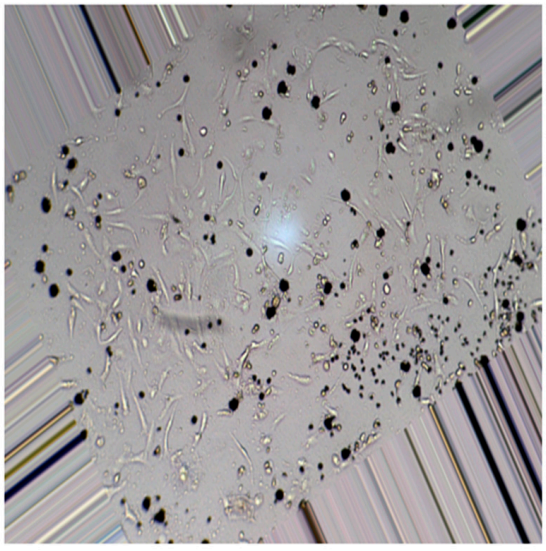

Figure 1. Augmented highly stressed image.

Figure 2 illustrates the original augmented image rotated by 45 degrees. The rotation introduces a stretching effect at the corners of the image, which can negatively impact both the accuracy and speed of the training process. This distortion creates artefacts that may interfere with the CNN's ability to learn effectively. To address this issue, a histogram analysis of the pixel values was performed. The histogram demonstrates the distribution and separation of pixel values. By enhancing this separation, the goal was to improve the contextual segmentation of the pixels, which helped in refining feature extraction and boosting the overall performance of the CNN.

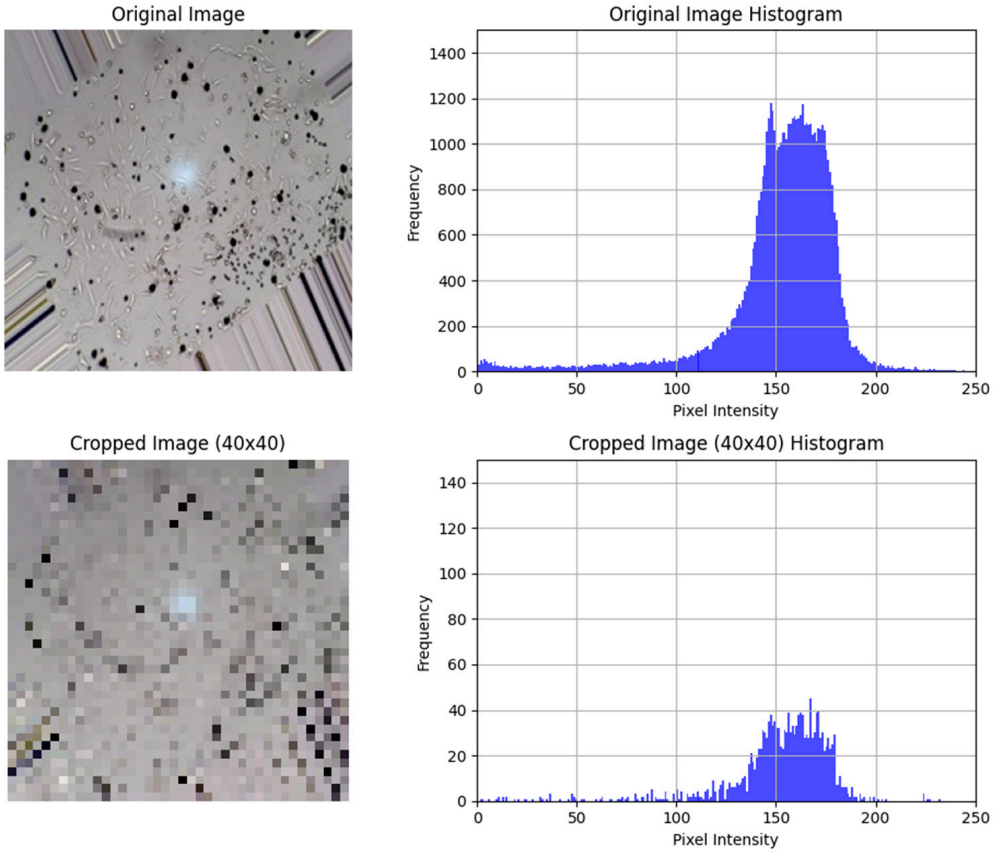

Figure 2. Adjusted augmented image with histogram.

Figure 3 illustrates the overall pipeline for the CNN training process, detailing the flow from the user's PC to the FPGA. All pre-processing tasks were performed on the client's PC using custom software, which applies various filters and augmentation procedures to a large dataset. This software converts the grayscale images into 8-bit, 40 × 40 binary images. The converted images are then saved to an SD card, which is inserted into the FPGA. The performance evaluation of the SD card read operation is measured to be approximately 62.5 µs per image. While this does add latency, the impact on the overall throughput is mitigated by pipelining the data transfer process with the FPGA-based CNN inference, ensuring minimal bottlenecks. This additional latency does not impact the overall performance of the inference time needed for classification or training. The FPGA is also connected to the PC and recognized by the custom software. The binary files are organized sequentially, with the class labels aligned with the corresponding images. These files are randomized and fed into the FPGA-based CNN for training. The training process continues until a stopping condition is met, which is determined based on the loss rate during backpropagation. Training halts when the loss rate does not decrease below a specified threshold over five passes of the dataset. The proposed implementation of the FPGA-based CNN focuses on local image processing for medical diagnostics. Future work will expand its integration with the IoT ecosystem. Currently, medical image data are transferred to the FPGA using an SD card, allowing for offline operation. To fully leverage the benefits of the IoT, future work aims to extend the system to include cloud connectivity, enabling seamless real-time data transmission. Cloud-based storage and processing would facilitate remote monitoring and enable the aggregation of medical data, addressing key challenges such as network connectivity, power consumption, and data scalability in IoT environments. This future integration would enhance the scalability, accessibility, and real-time capabilities of the proposed solution, making it more suitable for deployment in a variety of healthcare settings.

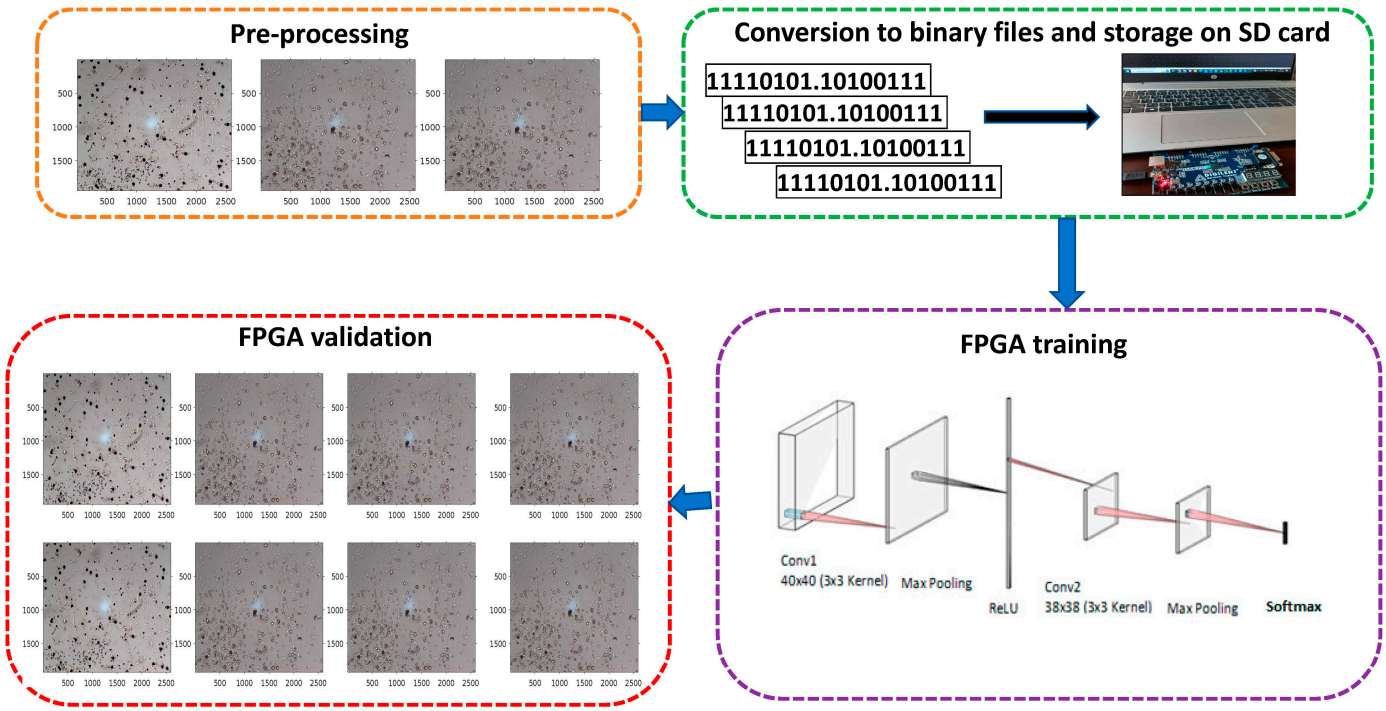

Figure 3. Pre-processing and FPGA training pipeline.

2.1. Model of the Custom CNN

The proposed custom CNN drew inspiration from the modified LeNet architecture, as described in previous work [26]. This CNN was substantially scaled down to fit within the memory constraints of the Basys3 FPGA. The input to the FPGA consists of 40 × 40 pre-processed binary images. These images are processed in banks of 3 × 3 matrices, which slide across the image to perform convolution operations, effectively emulating the convolutional layer (Conv1) depicted in Figure 4. Following the Conv1 layer, a max pooling layer is employed to reduce the number of lower value activations. This pooling operation shrinks the output matrix, preparing it for further convolutional processing in the Conv2 layer, which consists of 38 × 38 matrices to reduce the computational load and memory usage. It was observed that reducing the size further leads to a loss of spatial details and important features. This negatively impacts the model's ability to distinguish between different classes or detect fine-grained patterns in the image. If the matrix becomes too small, the network might not have enough capacity to learn complex features, which could lead to overfitting if the model becomes too tailored to the training data. The activations and pixel locations are stored in weight matrices, which are updated with each pass of the dataset. A second max pooling layer is applied to further decrease the number of activations, leading to the final classification stage, which utilizes a softmax layer to determine the output categories. The final classification results are categorized into three classes: normal, moderately stressed, and highly stressed. The CNN model was first compiled and tested on a CPU and GPU using MATLAB version 9.12 (R2022a) (https://www.mathworks.com/products/matlab.html). These initial tests provided valuable performance benchmarks, which were subsequently used to guide the HDL synthesis for FPGA implementation.

2.2. Flow Chart

Figure 5 illustrates the flowchart of the Basys3 CNN training process. The workflow is divided into three distinct subprocesses: pre-processing, FPGA training, and monitoring and validation, which occur sequentially. Feedback regarding the status of each subprocess is provided through the console window. Commands to control the process are issued via UART, while the images are loaded onto an SD card, which is inserted into the FPGA,

as shown in Figure 6. The Microblaze processor on the FPGA reads the images from the SD card and passes them to the hardware CNN block for processing. The FPGA system provides feedback through interrupt lines configured as active-low, signalling when a sample has been processed and a valid result has been produced. This setup ensures efficient communication and monitoring throughout the training process.

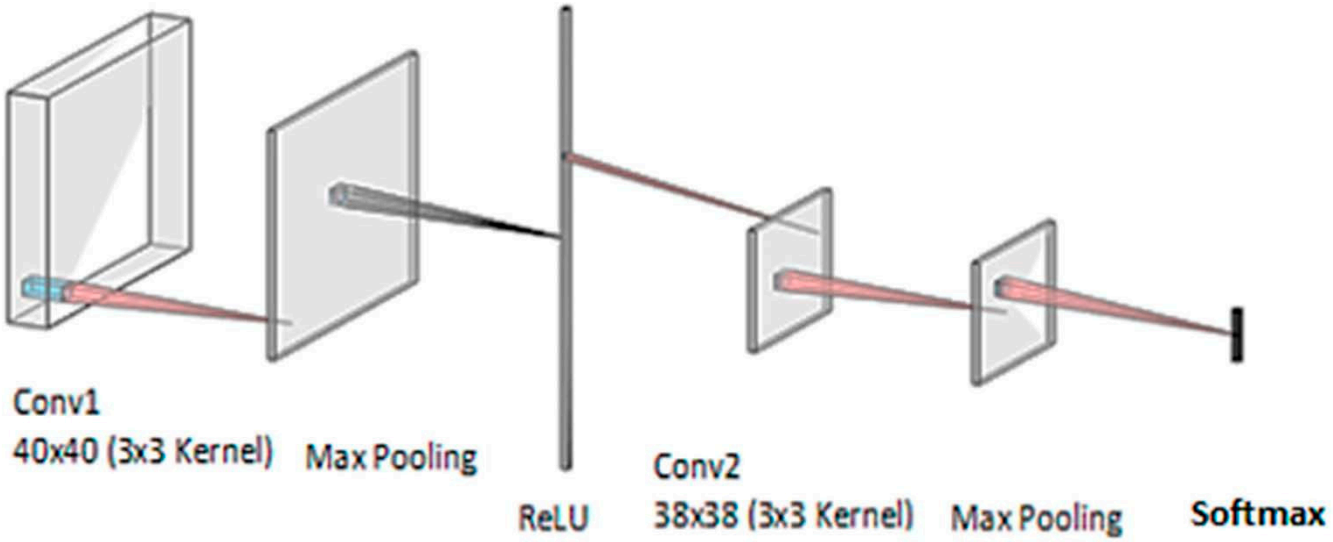

Figure 4. Custom CNN layers.

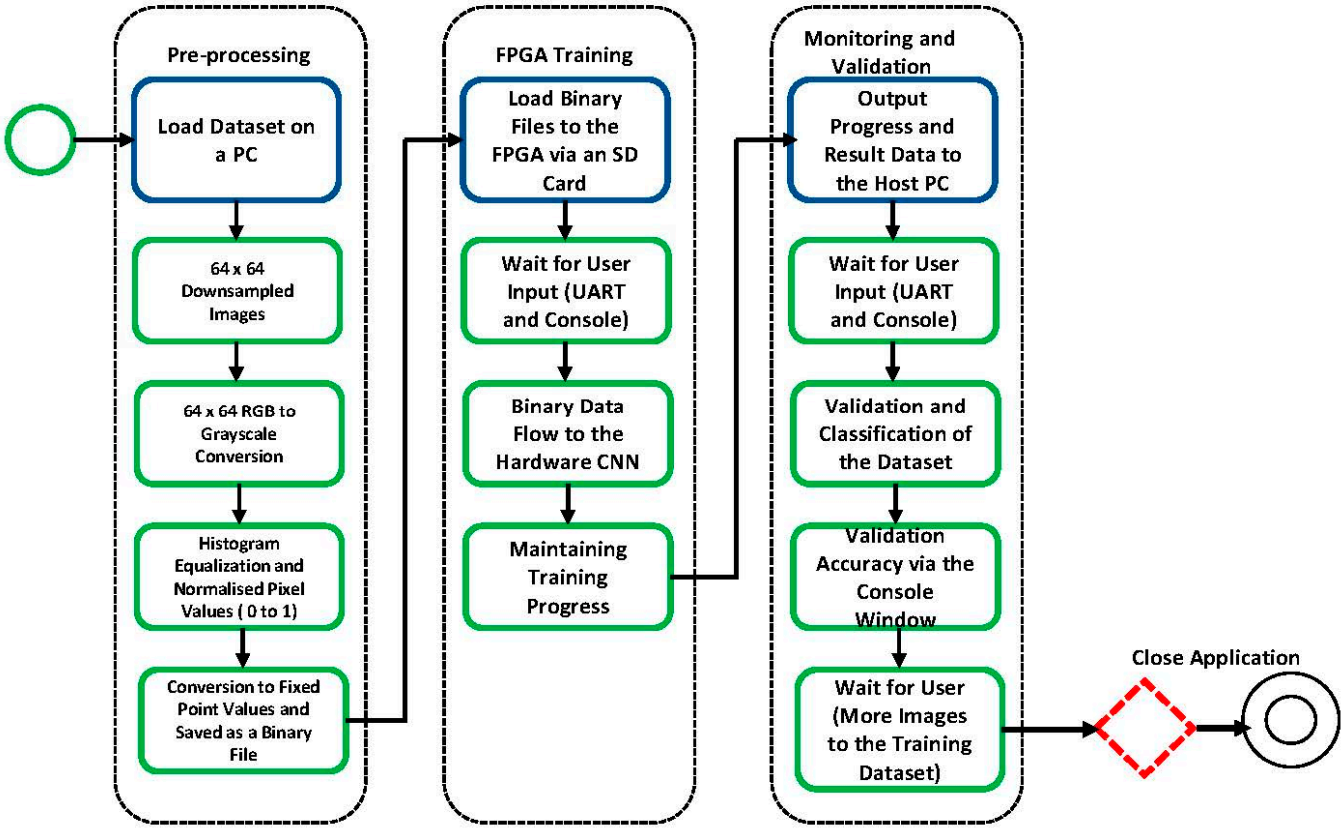

Figure 5. PC to FPGA training flowchart.

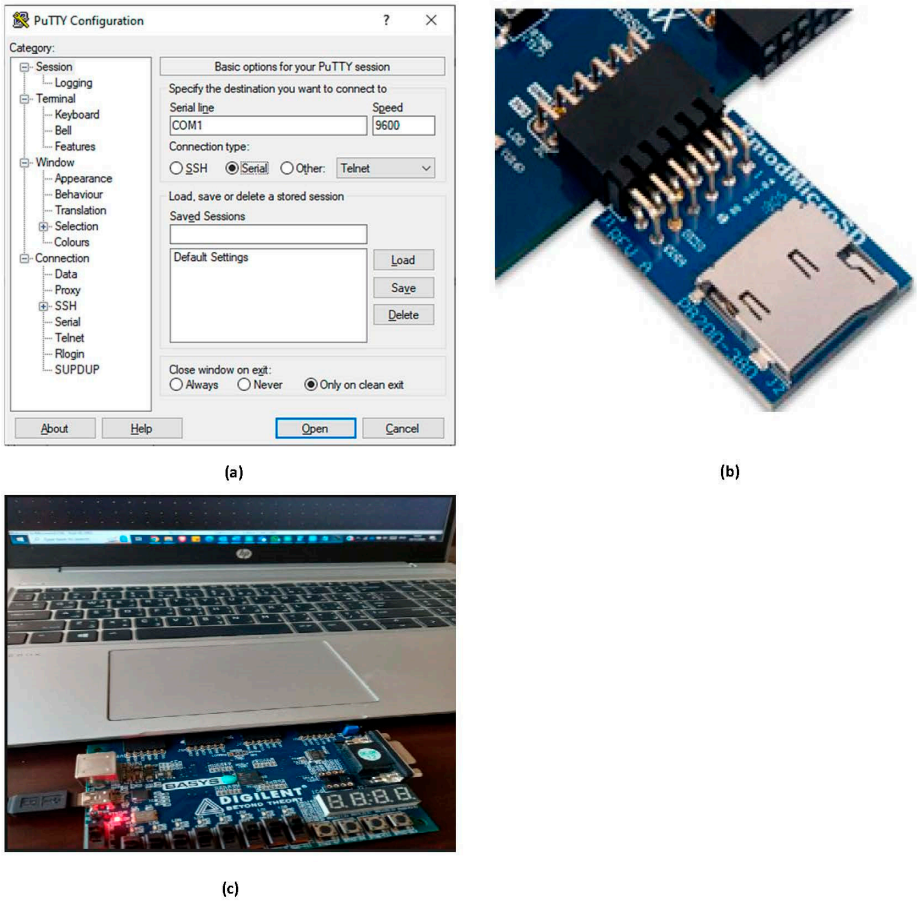

Figure 6. (**a**) Control software on the PC connected to the FPGA. (**b**) PMOD SD card reader connected to the Basys3 FPGA. (**c**) PC connected to the Basys3 FPGA via a micro-USB.

2.3. Simulation Results

The simulations were conducted using Xilinx Vivado 2021.1, specifically employing its built-in simulation suite to verify functional correctness and evaluate timing metrics. The timing figures were obtained under typical conditions configured in Vivado using post-implementation timing analysis. The simulation results indicate that the average time to classify a single image is 163.795 microseconds, as shown in Figure 7. The output, Output 1, as shown in Figure 7, displays the number of activations computed, which were subsequently processed by the softmax layer to produce the final classification result. This specific output corresponds to a healthy sample. The input arrays consisted of values fed into the FPGA from binary files, which were converted from pre-processed grayscale images. Backpropagation methods were evaluated by testing the CNN with 1000 samples from each category: highly stressed, moderately stressed, and normal. In total, there were three output classes (normal, moderately stressed, and highly stressed) where backpropagation was used to help manage the complexities of the learning process by systematically updating weights throughout the network, ensuring that each layer contributed effectively to the learning process. Figures 8–10 illustrate the simulation results, showing both correct and incorrect classifications of the dataset images. As shown in Figure 7, "output1" is the fully connected layer's final value, "prediction" is the predicted class, and "clk" represents the base clock. Figure 8 shows the correct counts value, which represents the number of correctly predicted samples. This test shows the accuracy of the highly stressed image classification samples, where 97.5% accuracy was achieved. Figure 9 shows the classification performance of the moderately stressed samples. The correct counts were measured at 985, which equates to 98.5% accuracy, while Figure 10 shows the classification process of normal samples. The correct counts' waveforms changed consistently during the classification process,

incrementing after each successful prediction. As shown in Figure 10, an 86.4% accuracy was achieved, which translates to 864 out of 1000 normal samples classified correctly.

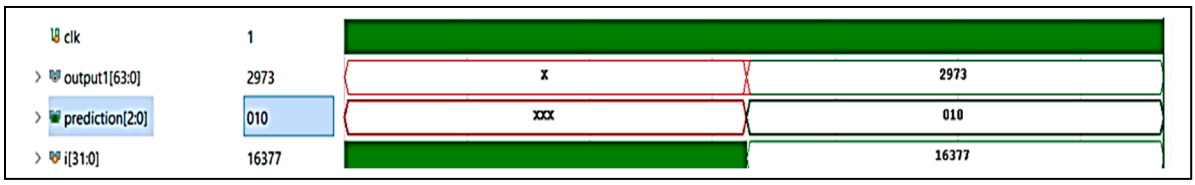

Figure 7. Highly stressed image classified correctly by the FPGA CNN.

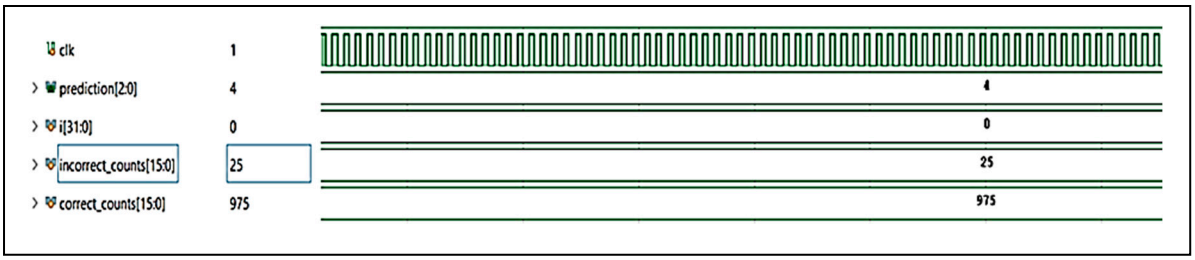

Figure 8. Highly stressed sample classification performance.

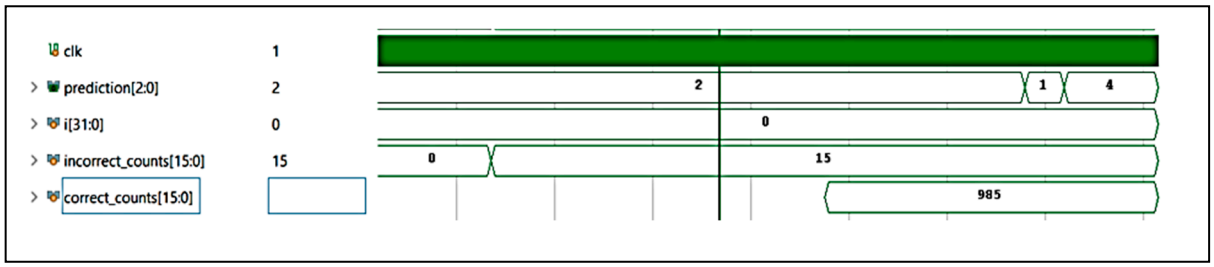

Figure 9. Moderately stressed sample classification performance.

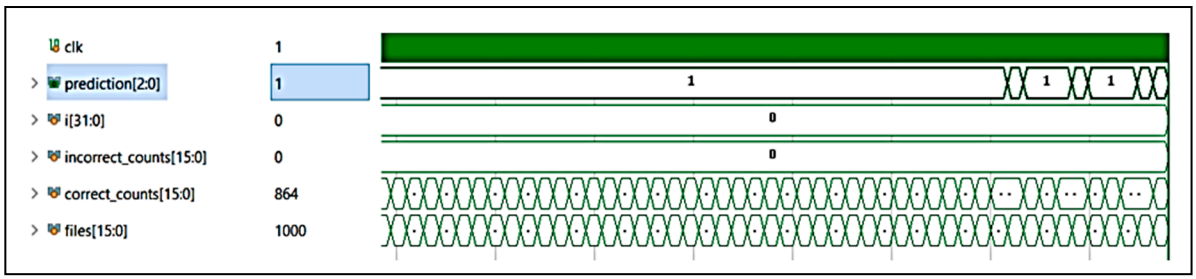

Figure 10. Normal sample classification performance.

Figures 8–10 demonstrate that the backpropagation algorithm achieves an average accuracy of approximately 95% during training. The time required to classify 1000 samples of each image type averages 163.5 milliseconds, representing a threefold improvement over the software-based CNN solution.

3. Hardware Design

The high-level block diagram shown in Figure 11 illustrates the general layout and operation of the hardware CNN. The FPGA is fed three 3×1 input values from the binary images, which are used as inputs for the parallelized convolution process. The kernel weights are stored in separate BRAMs and are updated after a single pass from the MAC operations to the softmax classification result. The hardware synthesis schematic diagrams shown in Figures 12 and 13 demonstrate how the high-level diagram was translated to

produce a functional solution, while Figure 11 shows that the values passed into the FPGA via SPI are fed into three kernel BRAMS. These contain the weights of the filters used in the convolution layer. They are multiplied by the SPI input and the kernel weight values. The results are sent to the adder/activation map generator. The block performs the calculation of the batch normalization of values after the map is fully populated to 38 × 38 matrices as explained in Section 3.

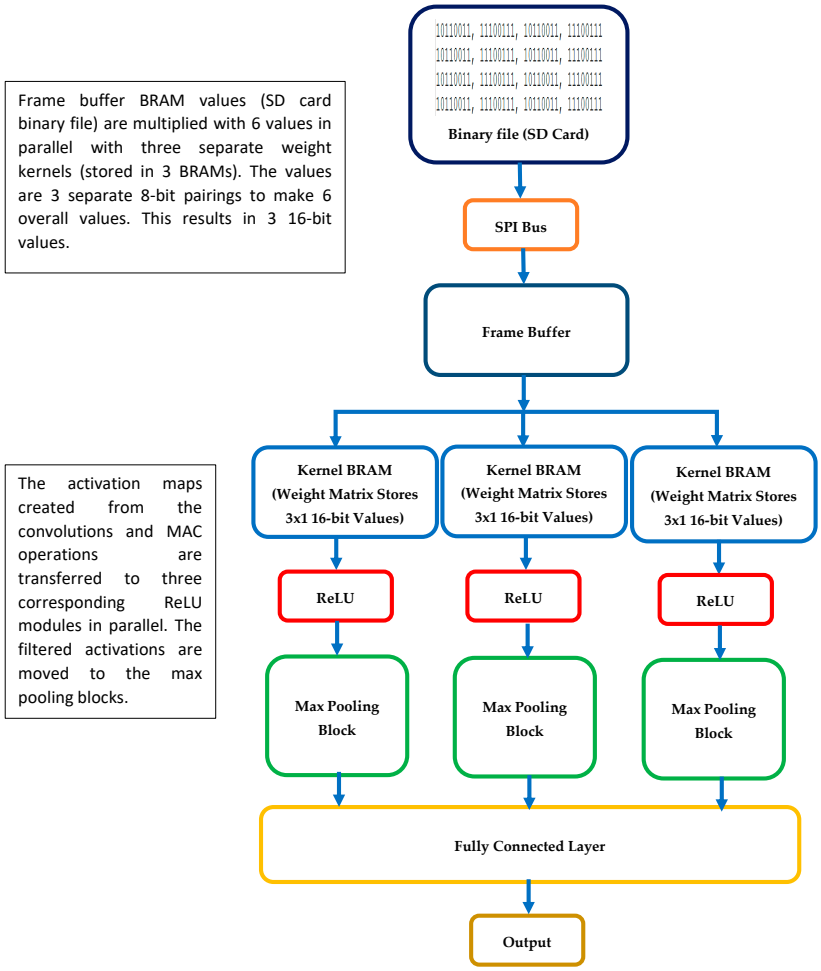

Figure 11. High-level parallelization block diagram.

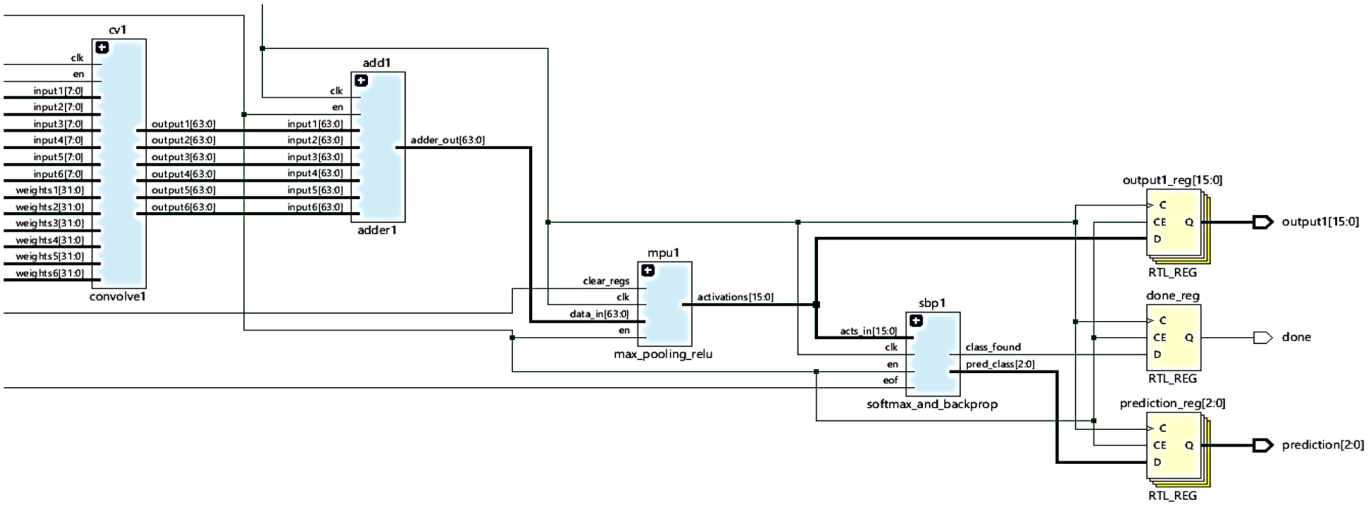

Figure 12. HDL output logic block diagram.

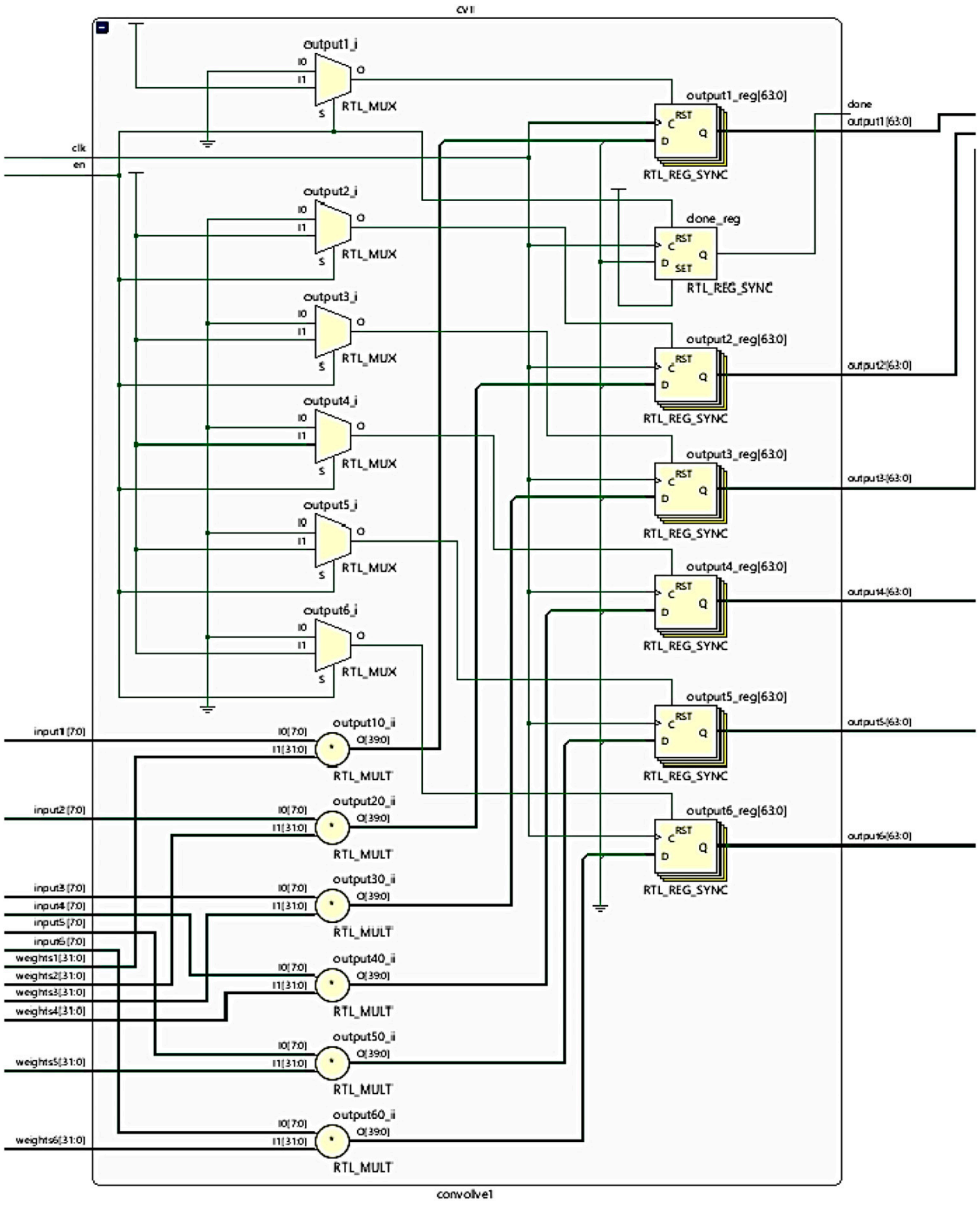

Figure 13. HDL output logic block diagram with extended detail.

As shown in Figure 11, the parallelization design approach involves the custom CNN's implementation on the FPGA device. It involves the input of binary data from the SD card into the FPGA CNN classifier. The binary file data are transferred via a 3×1 matrix of 16-bit values split into six 8-bit values (as input data) and are multiplied by the contents of the kernel BRAM weights. The binary files are reconstructed to match the behaviour of a convolution shift after each forward pass. This was accomplished to reduce the number of logic gates used where extra resources would be needed to create a separate module to perform convolutions. Data are shifted sequentially, six values simultaneously at a time, into the classifier module's input. These values are multiplied by the weights in the kernel BRAMs. The outputs from these MAC operations are transferred to the respective ReLU modules for each activation map generated. The filtered activations are passed on to the max pooling modules to further reduce most activations into smaller resolution tensor maps. These are finally passed to the fully connected layer, which determines the final classification output class. The FPGA logic diagrams are shown in Figures 12 and 13, respectively.

With this custom CNN implementation, minimizing latency was paramount. The layout of the MAC modules, BRAMs, and logic gates was designed to ensure low-latency

signal connections, preventing the disruption of the critical path. Operating at 100 MHz, this configuration achieves optimal power efficiency and the lowest overall latency and resource utilization as shown in Tables 1 and 2, respectively.

Table 1. Timing characteristics.

Setup	2.837 ns
Worst Hold Stack	0.136 ns
Pulse Width	4.5 ns

Table 2. Utilization metrics.

LUTs	202 (20,800 available)
FFs	260 (41,600 available)
DSP	12 (90 available)
BRAMs	8

The dynamic power could be calculated based on the expression stated in Equation (1), where the capacitive load, C_{load}, FPGA Voltage, V, and clock frequency, f, were taken into account by the Xilinx Power Estimator (XPE), which is an integrated part of the Xilinx Vivado design suite.

$$P_d = C_{load} \times V^2 \times f \quad (1)$$

The dynamic power was calculated to be 0.074 W, and the static power was calculated to be 0.039 W. The combined power for the CNN modules was estimated at 0.113 W. The entire system, which includes the external peripherals of the Basys3, Microblaze softcore, and the PMOD SD card, was estimated at a total of 4 W.

The combined total on-chip power was measured to be 0.113 W, whereas the setup, worst hold stack, and pulse width characteristics met the timing requirements for this configuration at 100 MHz and a 1.8 V supply voltage. This solution significantly reduces power during use compared to CPU- and GPU-based solutions. For example, training the network on an Intel Core i7-10700T CPU at 2.00 GHz took approximately 13–18 h, and when training on an NVIDIA GeForce GTX GPU, the network required 1000 epochs and took around 6 h to complete [8].

3.1. Forward Pass Algorithm and Caching

During the forward pass stage of the training phase, the inputs, outputs, and weights are saved to an addressable memory block (using block RAMs) and are updated for each backpropagation stage. The convolutional layers and the softmax weights are updated with the max pooling and ReLU inputs being saved during forward passes. Equation (2) shows the activation algorithm for the MAC operations, where i (image pixel) and w (weight) are multiplied together and added to b (bias). The final accumulated sum of activations is transferred to the ReLU activation sequence shown in Equation (3).

$$(a)^c = \sum_{k=0}^{n} i_x i_y w_x w_y + b \quad (2)$$

Dot matrix multiplications are performed where i is the pixel of the image, w is the weights, and b is the bias. This is performed concurrently in a 3×3 kernel with the last three elements excluded to reduce the footprint of the MAC logic blocks. Equation (3) states the MAC result. If the value is above the threshold, the pixel is considered to be activated; otherwise, no activation is recorded. Equation (4) calculates the overall activations for the 40×40 image. The summation of all activations and the current activation is realized by Equations (4) and (5).

$$A_{xy} = \begin{cases} 1, & a_{xy} > T_a \\ 0, & a_{xy} \leq T_a \end{cases} \quad (3)$$

$$r_t = \sum_{x=0}^{n} \sum_{y=0}^{m} A_{xy} + r_x \tag{4}$$

$$m_z = \max_{0 \le x \le n} r_t \tag{5}$$

Max pooling is applied to the 40 × 40 convolution image to extract the most relevant activations with a 2 × 2 kernel.

3.2. Backpropagation for Training an Integer-Based CNN

The softmax layer utilizes a backpropagation algorithm based on cached totals of the weights, biases, and inputs. Since softmax relies on percentage values to calculate the model accuracy, forward passes use signed 16-bit integers. A signed 1-bit base with a 14-bit mantissa fixed-point representation is employed for updating the weights. The weights in the convolutional layers are then converted back to signed integers.

$$L = -ln(P_c) \tag{6}$$

Equation (6) shows the loss equation used in the softmax to determine the class of the sample, where L is the loss and P is the probability. In hardware, the exponential function requires a large amount of logic space to be implemented. An approximation of the exponential function is used to represent e using the Taylor series in Equation (7).

$$e^x = \sum_{n=0}^{\infty} x^n = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \dots \tag{7}$$

$$T_a = \frac{\sum_{i=0}^{s} a(n)}{S} \tag{8}$$

The calculation of the new threshold for the softmax is performed using Equation (8), where $a(n)$ is the summation of activations cached in the forward pass and S is the total number of samples in the forward pass.

For the custom CNN hardware implementation, a bounded ReLU activation system is used. The count of activations is cached in memory during a forward pass. When a certain number of samples are passed, the gradient of the activations is calculated. A comparison between the actual class and the predicted classes is performed by measuring the gradients of the accuracies during classification. The weights of the convolutional layers and the softmax layers are updated with each iteration of forward passes during the training phase. This is represented in Equation (9), where *Ta* refers to the total activations and S is the number of samples.

$$w(T) = \frac{e^{Ta}\left(S - e^{Ta}\right)}{S^2} \tag{9}$$

As shown in Figure 14, each coloured region represents a boundary within which a sample is classified into a specific class. These boundaries are flexible and can change in size based on their thresholds. The thresholds are calculated using the gradients of the totals of a given dataset against the current thresholds. Over a given set of data samples, the regions will shift to reduce the accuracy loss percentage. Since the dataset images are 40 × 40, a bit width of 16 bits is used to cover every pixel in each sample. The total number of pixels in a 40 × 40 image is 1600. The total precision of signed 16-bit values ranges from −32,768 to 32,768. For the activation algorithm used in the softmax layer to function with the given number of pixels, 16-bit precision is required. This modified softmax reduces the memory space requirements typically found in larger CNNs designed for GPU or CPU platforms, achieving a more compact method of classification with performance comparable to larger CNN models. The equations for adjusting and calculating new weights during training are shown in this section, with Equation (9) detailing the custom cross-entropy loss calculation method. It is a first-order approximation of the standard cross-entropy model, designed for this specific 16-bit fixed point.

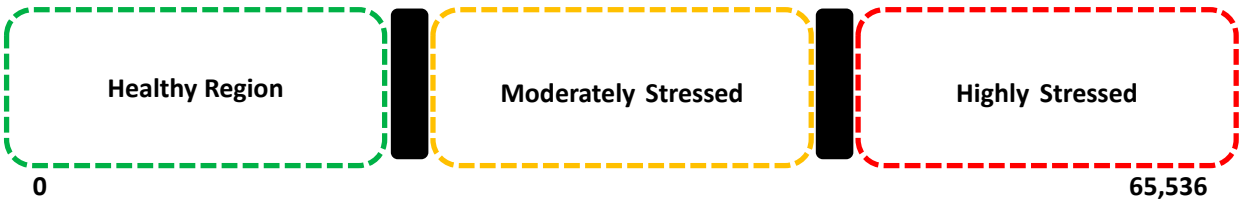

Figure 14. Visual representation of the bounded ReLu activation-based softmax.

4. MATLAB and FPGA CNN Performance

After finalizing the most effective configuration to match the FPGA and MATLAB CNN models, both were evaluated and compared in terms of classification and training performance.

4.1. MATLAB Custom CNN Performance

The custom CNN model was trained on 4587 images, with 1378 images used for testing and 937 for validation in MATLAB. The parameters set for the training process were as follows: a maximum of 30 epochs, 70 iterations per epoch, and a maximum of 2100 iterations. The learning rate was set to 0.001, and the GPU used was an NVIDIA K80. The training accuracy progress is shown in Figure 15, and the final validation accuracy of 94.07% is presented in Figure 16.

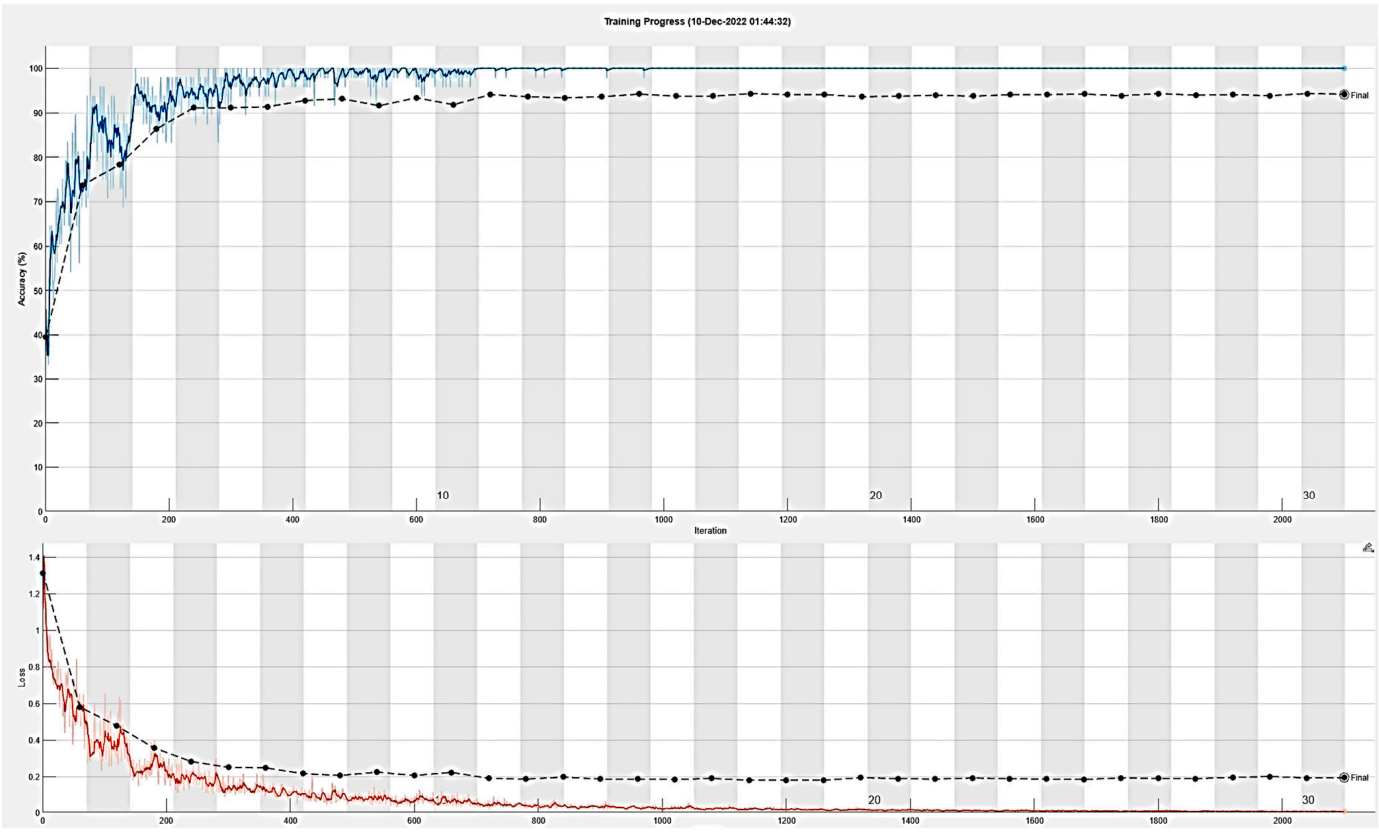

Figure 15. MATLAB custom CNN training accuracy and loss.

Confusion matrices were generated for the MATLAB implementation, including all three classes. Figure 17 presents the results from a smaller unseen test dataset consisting of 18 normal images, 19 moderately stressed images, and 7 highly stressed images. The MATLAB model achieved an overall accuracy of 94.07%, with its weakest classification performance observed in distinguishing between moderately stressed and normal images, as well as highly stressed images, as shown in Figure 17.

Results	
Validation accuracy:	94.07%
Training finished:	Max epochs completed
Training Time	
Start time:	10-Dec-2022 01:44:32
Elapsed time:	3 min 6 sec
Training Cycle	
Epoch:	30 of 30
Iteration:	2100 of 2100
Iterations per epoch:	70
Maximum iterations:	2100
Validation	
Frequency:	60 iterations
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.001

Figure 16. MATLAB validation accuracy and associated parameters.

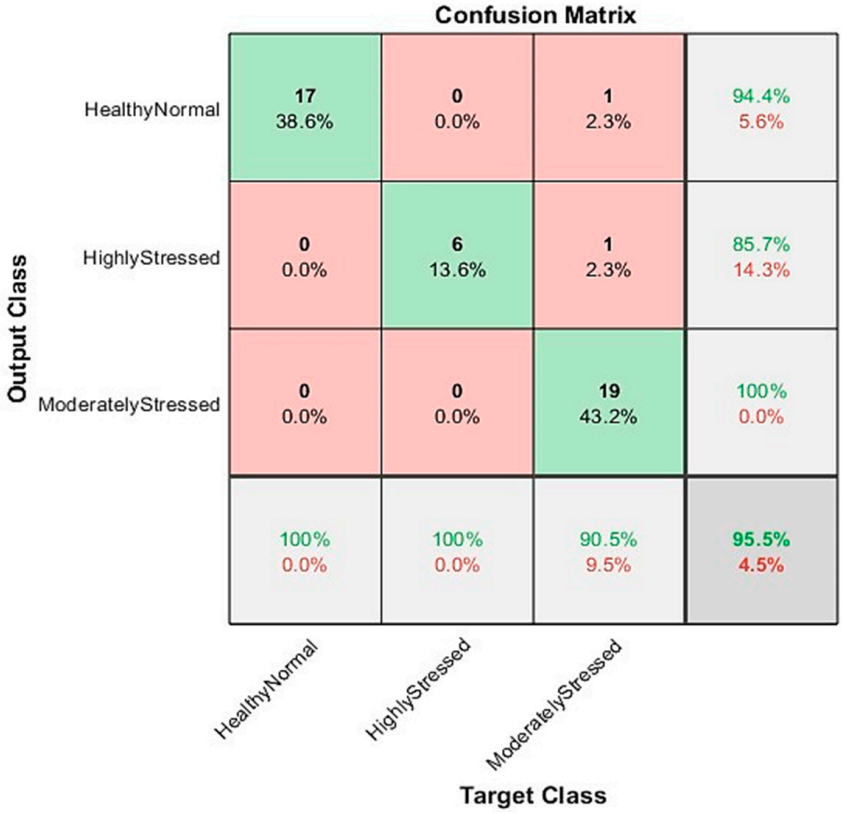

Figure 17. Multiclass confusion matrix in MATLAB.

4.2. Custom CNN FPGA Performance

Figure 18 shows a confusion matrix generated from the results of the FPGA's custom CNN. These matrices were compiled based on classifications between two categories at a time. Darker sections of the matrices represent higher percentage values between the two categories, while lighter sections indicate lower percentage values. The FPGA CNN model demonstrates a stronger capability to differentiate between normal samples

and both moderately and highly stressed samples. However, it is less effective at distinguishing between moderately stressed and highly stressed samples. When combining all confusion matrices, the classification accuracy for unseen images is approximately 91%. Each test case was evaluated with the same unseen test dataset used for the MATLAB CNN implementation.

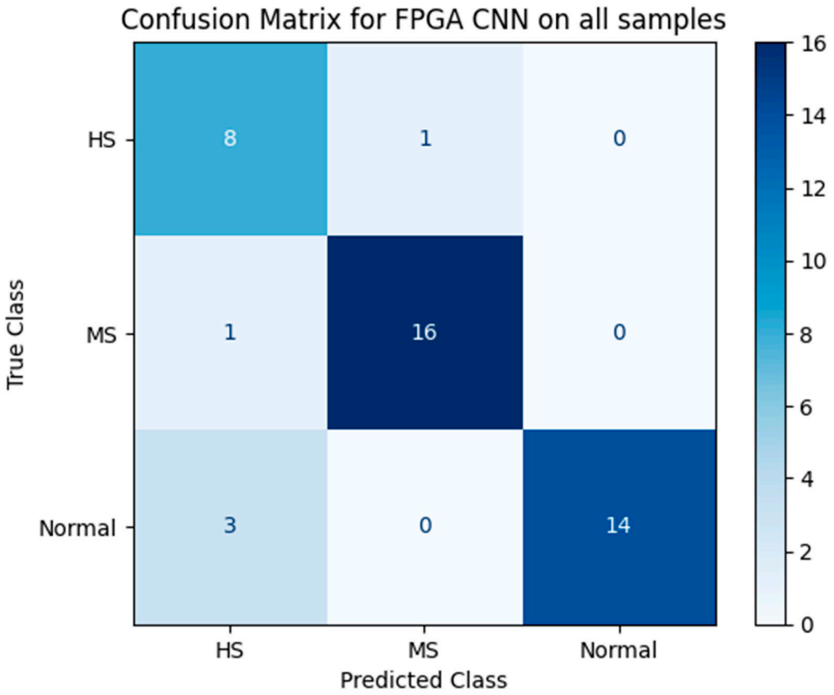

Figure 18. Multiclass classification performance for the FPGA CNN model on unseen images.

The next metric measured was the CNN's training capability. The FPGA CNN was trained on 4587 images, with 1378 images used for testing and 937 for validation. The plots shown in Figures 19 and 20 display the accuracy percentage over time and the model's loss rate, respectively. Figure 19 illustrates an optimum progression in accuracy, confirming that the backpropagation algorithms operate as intended, even with applied approximations and quantization. After epoch 15, the model reaches saturation, and the loss no longer decreases, leading to the cessation of training. Each epoch takes 1.45 s to complete in the FPGA, resulting in a total training time of 1 min and 46 s to achieve 91.3% accuracy.

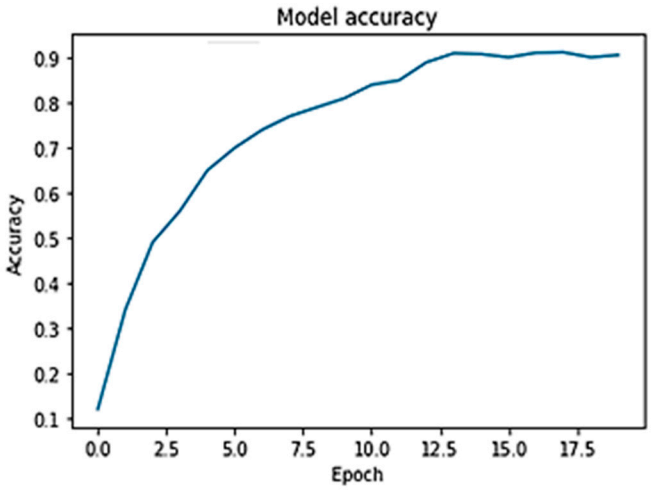

Figure 19. Accuracy of highly stressed samples against moderately stressed samples.

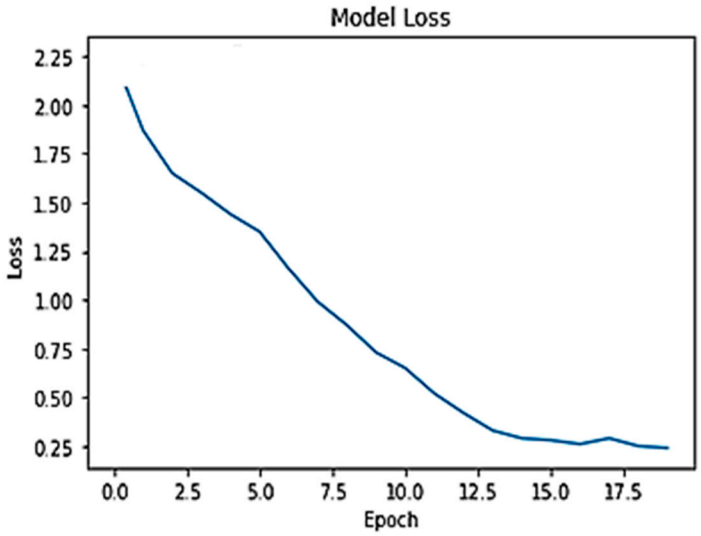

Figure 20. Loss rate of highly stressed samples against moderately stressed samples.

The overall resource utilization is 4% of the FPGA, and the scalability to add additional modules for different use cases or image datasets is feasible. The modular design of each module allows for the reconfigurability of the model, as well as scalability for larger and deeper CNN designs. The dataset used in this study was augmented to create both training and test samples. This augmentation process was essential in overcoming the challenge posed by the limited size of the original dataset available in the biotechnology lab due to the manual data imaging technique used. While data augmentation did help improve the generalization of the CNN model, the accuracies between highly and moderately stressed classes differ due to the nature of the data augmentation and the inherent characteristics of the classes themselves. In hardware-based classification, where the quantization of the CNN model to 8-bit precision provided speed improvements, it did result in a minor trade-off in classification accuracy. Nonetheless, it was demonstrated that the optimization for hardware did not substantially impact the classification accuracy, which was the main focus of this research.

5. Discussion

The proposed FPGA-accelerated CNN offers significant advantages in medical image classification, particularly in scenarios where high-performance computing resources are unavailable, one of the requirements for low-power IoT-based healthcare systems. The low-power, high-efficiency nature of FPGAs makes them an ideal choice for real-time diagnostics in resource-constrained environments, such as remote or underserved healthcare facilities. In particular, this study demonstrates how the parallel processing capabilities of FPGAs can be leveraged to accelerate CNN operations, leading to a threefold increase in the training speed and a significant reduction in power consumption compared to traditional CPU- and GPU-based approaches. One of the primary advantages of FPGA-based implementations is their ability to operate with significantly lower power consumption, making them suitable for deployment in IoT-based environments where energy efficiency is paramount. FPGAs are inherently more energy-efficient than GPUs and CPUs, particularly when tailored for specific tasks like image classification. The findings of this study indicate that the entire FPGA implementation, including all external modules in the power profile, was estimated to draw approximately 4 W overall. In comparison, the software-based CNN running on an NVIDIA K80 GPU consumes 300 W [27]. This represents a significant improvement in power efficiency, with the FPGA-based implementation using only about 1.33% of the power consumed by the GPU-based implementation. The ability to maintain high classification accuracy while operating under such low-power conditions underscores the potential of FPGAs in medical diagnostics, particularly in scenarios requiring continuous,

real-time analysis without the support of an HPC infrastructure. Another merit of the proposed FPGA solution is its adaptability to edge computing paradigms in healthcare. Edge computing, which involves processing data closer to the source rather than relying on centralized HPC systems, is increasingly being recognized for its potential to revolutionize medical diagnostics. Edge intelligence enables faster decision-making and reduces the latency associated with transmitting large datasets to distant servers for processing. The FPGA-based CNN demonstrated in this study exemplifies this trend, providing a low-cost, offline solution for the automated diagnosis of cytotoxicity in calcium cells. By interfacing with a simple software program via UART, the FPGA can operate independently, offering real-time classification and data collection capabilities, which are essential for timely medical interventions.

However, while the FPGA's low-power operation and efficiency are clear advantages, the technology does come with certain limitations. FPGAs are less flexible than GPUs and CPUs in handling a wide range of tasks, which could pose challenges when adapting the FPGA to new diagnostic applications without significant reconfiguration. While FPGAs excel in specific, highly optimized tasks, their re-programmability for different workloads is more complex and time-consuming compared to software-based implementations. This limitation means that while the FPGA-based CNN performs exceptionally well in its designated task, expanding its use to other diagnostic areas would require substantial redesign and optimization efforts. In terms of hardware resource utilization, the quantization of the CNN to 8-bit precision and its optimization for the FPGA's limited resources allowed the model to fit within the constraints of the Basys3 FPGA without a significant loss of accuracy. Quantization techniques enable more compact models that can be deployed on resource-constrained devices without compromising performance. This study further supports this view by demonstrating that the careful design and optimization of the quantized model can mitigate potential trade-offs in accuracy, allowing for efficient FPGA implementation even in medical applications that demand high precision. While the FPGA-based CNN offers numerous advantages, particularly in terms of energy efficiency and speed, it is also important to consider the broader implications of this approach for medical diagnostics. The ability to perform training and classification tasks on large datasets with minimal hardware resources opens up new possibilities for deploying advanced diagnostic tools in environments where traditional HPC resources are unavailable. This capability is particularly relevant for remote and low-resource settings, where rapid and accurate diagnostics are crucial yet access to a sophisticated computing infrastructure is limited.

Hence, the proposed FPGA-accelerated CNN represents a significant step forward in applying AI-driven techniques to healthcare. By combining the energy efficiency and speed of FPGAs with the precision of CNNs, this study offers a viable solution for real-time, low-cost medical diagnostics. As healthcare systems continue to evolve and incorporate AI-driven tools, the role of specialized hardware is likely to expand, providing new opportunities for rapid, efficient, and accessible medical diagnostics [28].

6. Future Work and Limitations

The study presented in this paper demonstrates the viability of accelerating CNN models on reconfigurable hardware for medical diagnostics; there are several future research avenues and associated limitations which require further research. To enhance the speed and efficiency of the CNN model, further quantization and optimization techniques could be explored. As accuracy is one of the key factors in medical imaging diagnostics, the dataset could be extended to include a more diverse range of cytotoxicity conditions in generalized clinical applications. The adaptability of FPGA devices is still a major challenge for diagnostic tasks; therefore, future efforts require more flexible architectures that can be easily reconfigured for different applications, reducing the complexity associated with redesigning new use cases. To address scalability, investigating the use of multiple FPGAs could enhance processing capabilities and accommodate larger datasets. Real-time

processing in clinical settings with a user interface could facilitate broader adoption by developing custom-designed hardware for healthcare diagnostics [29–31].

7. Conclusions

In this study, we have successfully developed and optimized a CNN implementation on the Basys3 FPGA for the classification of cytotoxicity in human kidney cells (HK-2s). By addressing the constraints of limited dataset sizes and hardware resources, we achieved significant improvements in the training speed and energy efficiency. Our approach incorporated efficient image augmentation and pre-processing techniques, which played a critical role in enhancing prediction accuracy and model performance. The FPGA implementation demonstrated a threefold increase in the training speed compared to a software-based CNN on an NVIDIA K80 GPU, while consuming only 1.33% of the power. The optimized CNN architecture, with its seven layers and 13,464 hyperparameters, processed 40×40 grayscale images efficiently, achieving an accuracy of approximately 91% for unseen images. This work underscores the potential of FPGA-based implementations for rapid and resource-efficient medical diagnostics, especially in the context of IoT and edge computing. The integration of deep learning with FPGA technology enables computing at the edge, where traditional HPC resources may be unavailable or impractical. By quantizing the CNN to 8-bit precision and leveraging FPGA-specific optimizations, we have demonstrated a viable approach for deploying AI models in scenarios where traditional high-performance computing resources are unavailable. The successful integration of deep learning with FPGA technology presents a significant advancement in medical image classification, paving the way for more accessible and efficient diagnostic solutions. In scenarios where the automated diagnosis of large samples is required without access to an HPC, a low-power FPGA (or array of FPGAs) can be used for local classification. It provides an offline low-cost solution for fast and accurate automated diagnosis. The proposed solution requires minimal hardware to operate. A software program that interfaces with the FPGA via UART can read and write data, allowing for the collection of valuable information. It is envisaged that the proposed FPGA-based CNN could be used for other medical imaging diagnostics such as histopathology to broaden its utility in clinical settings. Furthermore, integrating real-time data processing and cloud connectivity could enable the CNN to learn more efficiently and robustly about new patient data, further improving diagnostic accuracy.

Author Contributions: Conceptualization, A.G.; methodology, A.G. and A.A.; software, A.G. and A.A.; validation, A.G., C.H.S. and A.A.; formal analysis, A.G., C.H.S. and A.A.; investigation, A.G. and A.A.; resources, A.G.; data curation, A.G. and A.A.; writing—original draft preparation, A.A. and A.G.; writing—review and editing, A.G., A.A. and C.H.S.; visualization, A.G., A.A. and C.H.S.; supervision, A.G.; project administration, A.G. and C.H.S.; funding acquisition, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data was created in this research; however, the first author of the paper would be happy to share further information upon request with interested researchers.

Acknowledgments: The author would like to thank the technical staff at Intel UK for their insightful discussion. The author acknowledges the use of Grammarly 14.1202.0 in the process of translating and improving the clarity and quality of the English language in this manuscript. All ideas, including software and hardware design, hardware simulation and synthesis, FPGA characterization, and system-level integration and testing, are the intellectual property of the authors of this paper.

Conflicts of Interest: The author declares that at the time the paper is published they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
2. Shawahna, A.; Sait, S.M.; El-Maleh, A. FPGA-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access* **2019**, *7*, 7823–7859. [CrossRef]
3. Yang, X.; Zhuang, C.; Feng, W.; Yang, Z.; Wang, Q. FPGA Implementation of a Deep Learning Acceleration Core Architecture for Image Target Detection. *Appl. Sci.* **2023**, *13*, 4144. [CrossRef]
4. Syed, R.T.; Andjelkovic, M.; Ulbricht, M.; Krstic, M. Towards Reconfigurable CNN Accelerator for FPGA Implementation. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 1249–1253. [CrossRef]
5. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
6. Xiao, M.; Wu, Y.; Zuo, G.; Fan, S.; Yu, H.; Shaikh, Z.A.; Wen, Z. Addressing Overfitting Problem in Deep Learning-Based Solutions for Next Generation Data-Driven Networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 8493795. [CrossRef]
7. Finotti, V.; Albertini, B. Simulating quantized inference on convolutional neural networks. *Comput. Electr. Eng.* **2021**, *95*, 107446. [CrossRef]
8. Hodeify, R.; Ghani, A.; Matar, R.; Vazhappilly, C.G.; Merheb, M.; Zouabi, H.A.; Marton, J. Adenosine Triphosphate Protects from Elevated Extracellular Calcium-Induced Damage in Human Proximal Kidney Cells: Using Deep Learning to Predict Cytotoxicity. *Cell Physiol. Biochem.* **2022**, *56*, 484–499. [CrossRef] [PubMed]
9. Wang, C.; Luo, Z. A Review of the Optimal Design of Neural Networks Based on FPGA. *Appl. Sci.* **2022**, *12*, 10771. [CrossRef]
10. Thiyyakat, M.; Kalambur, S.; Sitaram, D. Constraint-Aware Federated Scheduling for Data Center Workloads. *IoT* **2023**, *4*, 534–557. [CrossRef]
11. Restrepo-Parra, E.; Ariza-Colpas, P.P.; Torres-Bonilla, L.V.; Piñeres-Melo, M.A.; Urina-Triana, M.A.; Butt-Aziz, S. Home Monitoring Tools to Support Tracking Patients with Cardio–Cerebrovascular Diseases: Scientometric Review. *IoT* **2024**, *5*, 524–559. [CrossRef]
12. Tang, K.; Kumar, A.; Nadeem, M.; Maaz, I. CNN-Based Smart Sleep Posture Recognition System. *IoT* **2021**, *2*, 119–139. [CrossRef]
13. Ravindran, A.A. Internet-of-Things Edge Computing Systems for Streaming Video Analytics: Trails Behind and the Paths Ahead. *IoT* **2023**, *4*, 486–513. [CrossRef]
14. Véstias, M.P.; Duarte, R.P.; De Sousa, J.T.; Neto, H.C. A configurable architecture for running hybrid convolutional neural networks in low-density FPGAs. *IEEE Access* **2020**, *8*, 107229–107243. [CrossRef]
15. Cho, J.; Jung, Y.; Lee, S.; Jung, Y. Reconfigurable binary neural network accelerator with adaptive parallelism scheme. *Electronics* **2021**, *10*, 230. [CrossRef]
16. Vita, A.D.; Russo, A.; Pau, D.; Benedetto, L.D.; Rubino, A.; Licciardo, G.D. A Partially Binarized Hybrid Neural Network System for Low-Power and Resource Constrained Human Activity Recognition. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 3893–3904. [CrossRef]
17. Jiang, W.; Yu, H.; Ha, Y. A high-throughput full-dataflow mobilenetv2 accelerator on edge fpga. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 1532–1545. [CrossRef]
18. Ding, W.; Huang, Z.; Huang, Z.; Tian, L.; Wang, H.; Feng, S. Designing efficient accelerator of depthwise separable convolutional neural network on FPGA. *J. Syst. Archit.* **2019**, *97*, 278–286. [CrossRef]
19. Guo, K.; Sui, L.; Qiu, J.; Yu, J.; Wang, J.; Yao, S.; Han, S.; Wang, Y.; Yang, H. Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 35–47. [CrossRef]
20. Salvi, M.; Acharya, U.R.; Molinari, F.; Meiburger, K.M. The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis. *Comput. Biol. Med.* **2021**, *128*, 104129. [CrossRef]
21. Wu, H.; Phan, J.H.; Bhatia, A.K.; Cundiff, C.A.; Shehata, B.M.; Wang, M.D. Detection of blur artefacts in histopathological whole-slide images of endomyocardial biopsies. In Proceedings of the 2015 37th Annual International Conference of the IEEE Engineering in Medicine and biology society (EMBC), Milan, Italy, 25–29 August 2015; pp. 727–730.
22. Liang, Y.; Lu, L.; Xiao, Q.; Yan, S. Evaluating fast algorithms for convolutional neural networks on FPGAs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 857–870. [CrossRef]
23. Zhang, K.; Wang, H.; Cheng, Y.; Liu, H.; Gong, Q.; Zeng, Q.; Zhang, T.; Wei, G.; Wei, Z.; Chen, D. Early gastric cancer detection and lesion segmentation based on deep learning and gastroscopic images. *Sci. Rep.* **2024**, *14*, 7847. [CrossRef]
24. Shibata, T.; Teramoto, A.; Yamada, H.; Ohmiya, N.; Fujita, H. Automated detection and segmentation of early gastric cancer from endoscopic images using mask R-CNN. *Appl. Sci.* **2020**, *10*, 3842. [CrossRef]
25. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern. Anal.* **2020**, *42*, 386–397. [CrossRef]
26. Hou, Y.; Li, Z.; Wang, P.; Zhang, W. Skeleton optical spectra-based action recognition convolutional neural networks. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 807–811. [CrossRef]
27. Board Specification TESLA K80 GPU ACCELERATOR. 2015. Available online: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/Tesla-K80-BoardSpec-07317-001-v05.pdf (accessed on 15 November 2024).
28. Ghani, A.; Dowrick, T.; McDaid, L.J. OSPEN: An open-source platform for emulating neuromorphic hardware. *Int. J. Reconfigurable Embed. Syst.* **2023**, *12*, 1–8. [CrossRef]
29. Erbas, I.; Amarnath, A.; Pandey, V.; Swaminathan, K.; Wang, N.; Intes, X. Unlocking Real-Time Fluorescence Lifetime Imaging: Multi-Pixel Parallelism for FPGA-Accelerated Processing. *arXiv* **2024**, arXiv:2410.07364. [CrossRef]

30. Attarmoghaddam, N.; Li, K.F. An Area-Efficient FPGA Implementation of a Real-Time Multi-Class Classifier for Binary Images. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 2306–2310. [CrossRef]
31. Nechi, A.; Groth, L.; Mulhem, S.; Merchant, F.; Buchty, R.; Berekovic, M. FPGA-based Deep Learning Inference Accelerators: Where Are We Standing? *ACM Trans. Reconfigurable Technol. Syst.* **2023**, *16*, 60. [CrossRef]