


## Article

# Computer Vision-Based Kidney's (HK-2) Damaged Cells Classification with Reconfigurable Hardware Accelerator (FPGA)

Arfan Ghani <sup>1,\*</sup>, Rawad Hodeify <sup>2</sup>, Chan H. See <sup>3</sup>, Simeon Keates <sup>4</sup>, Dah-Jye Lee <sup>5</sup> and Ahmed Bouridane <sup>6</sup>

<sup>1</sup> Department of Computer Science and Engineering, School of Engineering, American University of Ras Al Khaimah, Ras Al Khaimah 10021, United Arab Emirates

<sup>2</sup> Department of Biotechnology, American University of Ras Al Khaimah, Ras Al Khaimah 10021, United Arab Emirates

<sup>3</sup> School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, Scotland, UK

<sup>4</sup> Deputy Vice-Chancellor (Research), University of Chichester, College Lane, Chichester PO19 6PE, West Sussex, UK

<sup>5</sup> Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84602, USA

<sup>6</sup> Cybersecurity and Data Analytics Center, University of Sharjah, Sharjah 27272, United Arab Emirates

\* Correspondence: arfan.ghani@aurak.ac.ae or ghani\_786@yahoo.com

**Abstract:** In medical and health sciences, the detection of cell injury plays an important role in diagnosis, personal treatment and disease prevention. Despite recent advancements in tools and methods for image classification, it is challenging to classify cell images with higher precision and accuracy. Cell classification based on computer vision offers significant benefits in biomedicine and healthcare. There have been studies reported where cell classification techniques have been complemented by Artificial Intelligence-based classifiers such as Convolutional Neural Networks. These classifiers suffer from the drawback of the scale of computational resources required for training and hence do not offer real-time classification capabilities for an embedded system platform. Field Programmable Gate Arrays (FPGAs) offer the flexibility of hardware reconfiguration and have emerged as a viable platform for algorithm acceleration. Given that the logic resources and on-chip memory available on a single device are still limited, hardware/software co-design is proposed where image pre-processing and network training were performed in software, and trained architectures were mapped onto an FPGA device (Nexys4DDR) for real-time cell classification. This paper demonstrates that the embedded hardware-based cell classifier performs with almost 100% accuracy in detecting different types of damaged kidney cells.

**Keywords:** artificial neural networks; cell classification; FPGAs; hardware accelerators; human kidney-damaged cells



**Citation:** Ghani, A.; Hodeify, R.; See, C.H.; Keates, S.; Lee, D.-J.; Bouridane, A. Computer Vision-Based Kidney's (HK-2) Damaged Cells Classification with Reconfigurable Hardware Accelerator (FPGA). *Electronics* **2022**, *11*, 4234. <https://doi.org/10.3390/electronics11244234>

Academic Editor: Chunjie Zhang

Received: 3 November 2022

Accepted: 15 December 2022

Published: 19 December 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

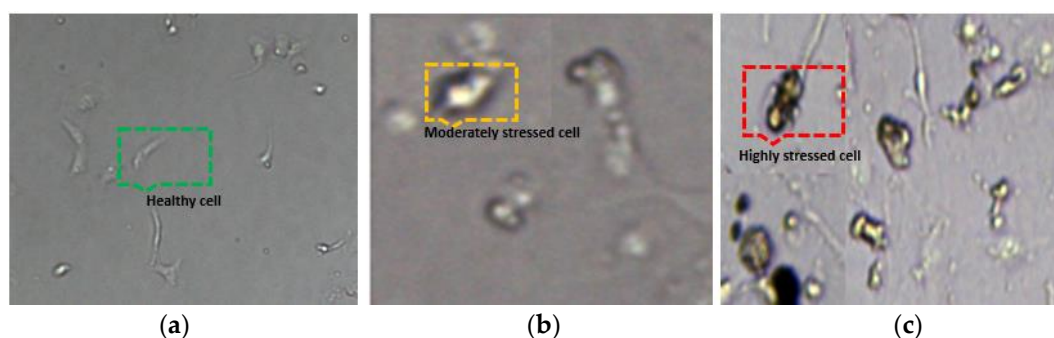
In the digital healthcare revolution, real-time data processing is a requirement for prompt diagnosis and treatment. However, in existing computer vision-based classification methods, solutions often have limited performance capabilities and fail to offer an integrated embedded hardware/software paradigm that can produce output in a digital format with real-time continuous update capabilities. This paper addresses this technological gap by offering a hardware/software-based co-design that could help overcome some of the limitations of existing AI-enabled classifiers by extracting kidney cell microscopic images and testing the classifier in real time on a reconfigurable FPGA device [1–8].

As microscopy imaging technology has improved, there have been several studies where microscopy image data are being generated in biomedicine [9–14]. Calcium ions [Ca<sup>2+</sup>] play a crucial role in several cellular functions where the concentration of Ca<sup>2+</sup> ions

regulates metabolic equilibrium maintained by several complex biological mechanisms that operate via the autonomic nervous system to offset disrupting changes [15,16]. Furthermore,  $\text{Ca}^{2+}$  is a critical signalling molecule that plays an active role in kidney development and kidney cellular functions and is an important biomarker in kidney diseases [15]. It has been demonstrated in recent studies that disruption of  $\text{Ca}^{2+}$  signalling can lead to kidney disease [15,16].

The authors of this paper have conducted a study where the effect of extracellular  $[\text{Ca}^{2+}]$  on human proximal cells (HK-2) was observed with varying calcium levels [17]. A deep neural network-based model was developed to predict injury to HK-2 cells. Morphological changes were detected using light microscopy, and the dataset was collected as healthy, moderate and highly stressed cells. Data augmentation was performed on the collected samples and used for training and testing a CNN base classifier. Further details are reported in [17].

This work was conducted in collaboration with the Department of Biotechnology at the American University of Ras al Khaimah, UAE, where HK-2 cells were cultured in varying levels of calcium in retrospect to healthy cells. Light microscopy imaging was used with an inverted OPTIKA XDS-2 microscope. Normal cells as well as cells cultured with different calcium concentrations, were observed and classified as healthy, moderate and highly stressed cells. Figure 1 shows a collection of such cells where plot A shows a healthy cell, plot B is a moderately stressed one, and plot C shows a highly stressed cell.



**Figure 1.** This figure shows the microscopy image (zoomed) of healthy (a), moderate (b) and highly stressed (c) HK-2 cells. The difference in shape demonstrates the cell condition.

Blood cell classification techniques have been reported in the literature where red and white blood cells were classified to diagnose conditions such as inflammation and the response of the immune system [18,19]. Existing methods require experience and expertise to analyse microscope cell image scans which include costly imaging equipment and input from medical experts; however, such methods are time-consuming as they include many stages of processing [20,21]. For example, one typical stage of image processing is feature extraction and filter-based techniques, such as Gabor wavelets, have been widely used in the literature [22,23], and this is not a quick process.

As reported in [24], multilayer feed-forward neural networks have been used for image classification in many instances, but newer deep learning-based convolutional neural networks are showing great promise for image classification [25]. As reported in [17], a CNN-based model could successfully be developed for multi-class HK-2 cell classification where feature extraction techniques could be avoided, thus saving processing time. Recent studies in CNN-based medical imaging have been reported in [26–28]. CNN models can be successfully applied in cell image classification, but there are a number of drawbacks, such as the need for a comparatively much larger dataset for training. Secondly, it takes much longer to train a CNN on general-purpose computer processors (CPUs); hence, specialised dedicated hardware, such as GPUs (graphics processor units), is required to speed up the training process. The CNN model offers an end-to-end solution that does not necessarily require manual feature extraction. However, mapping such architectures is impractical for

an embedded hardware platform such as FPGA because of the limited number of logic available on any single device. To overcome this limitation, it is essential to include manual feature extraction that facilitates implementing a rather shallow classifier on-chip. In this paper, a novel method of an enhanced Canny edge detector is proposed for feature selection and HK-2 cell classification with a rather simpler feed-forward neural network.

Implementing CNN-based architectures on a reconfigurable embedded platform is particularly challenging due to the enormous number of parameters and operations required to be implemented on a limited resource device. One of the challenges in implementing real-time neural classifiers on an FPGA device is the limited number of DSP hardware operators available. Secondly, the total amount of logic available on any device is still limited.

The main contributions of this paper can be summarised as follows:

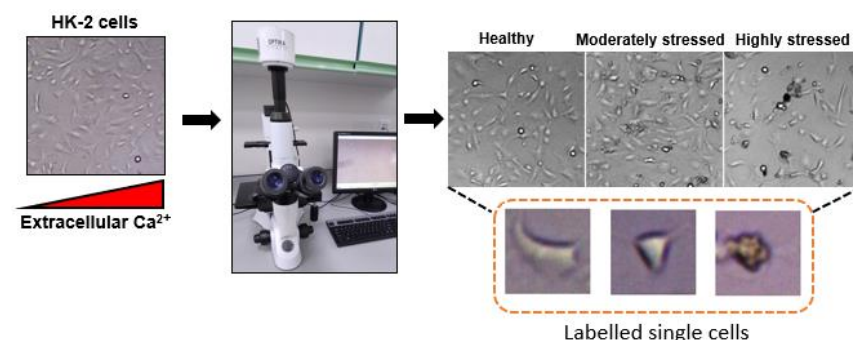
- (1) Bespoke data collected from cell cultures in the biotechnology lab and cell labelling for HK-2 kidney cells (healthy, moderate and highly stressed).
- (2) Developing an integrated enhanced Canny edge detector to calculate pixel area for backend neural classifier implementation and retrospective comparison with histogram-based technique. This technique offers a simpler shallow network that is easier to implement on both software and reconfigurable hardware platforms with almost 100% accuracy.
- (3) A small number of training samples are required, which is demonstrated as a technological solution for the real-time classification of HK-2 damaged kidney cells.

## 2. Materials and Methods

### 2.1. Cell Preparation and Data Collection

Data was collected from an immortalised proximal tubule epithelial cell line (HK-2) from an adult human kidney using an inverted OPTIKA XDS-2 microscope (Optika, Ponteranica, Italy) with a 10× objective. Under normal conditions, HK2 cells were grown in serum-supplemented Dulbecco's Modified Eagle Medium, containing high glucose, 2 mM L-glutamine, and 1% penicillin-streptomycin at 37 °C in a 5% CO<sub>2</sub> incubator. The standard culture medium contains 1.8 mM calcium, as per the company datasheet. The human proximal kidney (HK-2) cell line was ordered from Applied Biological Materials Inc., Richmond, BC, Canada.

To induce incremental degrees of cytotoxicity, cells were cultured in media with increasing levels of calcium adjusted using calcium chloride. Cells were plated overnight in the standard medium before it was replaced with a medium containing elevated levels of calcium and grown for an additional 18–20 h before imaging. Between 15 and 25 images with a resolution of 2592 × 1944 pixels were taken across each condition and stored in JPEG format. The imaged cells were divided into three categories based on our previous cytotoxicity studies [17]. Cells maintained in the standard medium were labelled as (healthy); cells at 16 mM calcium were referred to as (moderately stressed), and cells cultured at 32 mM calcium were labelled as (highly stressed). To extract individual cells from the images in the dataset, we used Marquee Tool in Photoshop to extract an 88 × 88 pixel crop around the identified cells. The overall setup for HK-2 cell collection is shown in Figure 2.

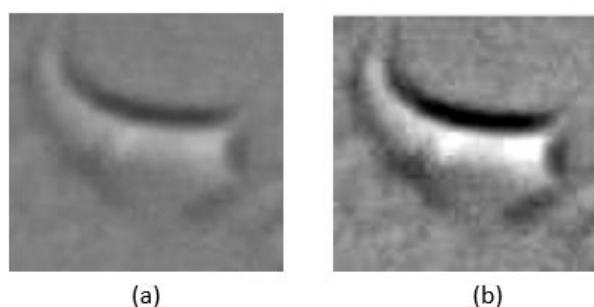


**Figure 2.** This figure shows the setup for HK-2 cell data collection.

## 2.2. Image Pre-Processing

Edge detection is an important task for cell classification. Several edge detection algorithms have been reported in the literature, such as Sobel, Roberts, Laplace and LoG [29–33]. The Canny edge detector was selected as it is widely used in computer vision tasks and performs well [28,29]. One of the objectives of edge detection is to find boundaries that are distinct within an image. In this case, we intended to localise and later classify cell images as healthy, moderately stressed or highly stressed. The major requirements for an edge detector are to minimise detecting wrong edges, improve accuracy by detecting edges as closely as possible to the real edges in an image and, lastly, be capable of detecting a true edge point. The overall aim of this study is applied in nature, where a real-life application is investigated, developed, implemented and prototyped. A retrospective analysis is provided in a wider context of the literature. Hence, this study investigates a system-level approach that takes a raw input cell image, pre-processes that image and selects features for simplified neural network-based classification, and finally exploits the reconfigurability of an FPGA device to map the software neural network architecture for real-time classification.

As lighting conditions during the biological sample collection in the lab environment could vary from one sample to another, it was important to adjust the image intensity values after converting an RGB cell image into a grayscale image. The image adjustment process improves the contrast of an image by saturating the top and bottom 1% of all pixel values and mapping them onto new values. Please see Figure 3a, where a healthy cell sample image was adjusted by enhancing the contrast, and Figure 3b.



**Figure 3.** (a) Input (grayscale image)—Healthy cell; (b) Enhanced image.

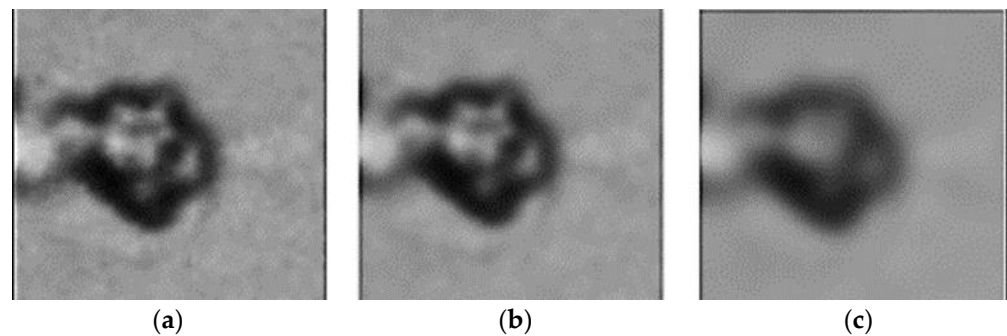
Low-light conditions and short exposure times raise major challenges as they significantly degrade image quality. It is predominantly a Gaussian noise which is a common problem in microscopic imaging. As the Canny algorithm [29] suffers from the drawback of being vulnerable to noise, it may detect false edges as well as missing the fine details of true edges in an image. Therefore, once the image is enhanced, it is important to remove noise and improve the overall quality of the image, as without removing this noise, it would not be possible to obtain image gradients for edge detection. In the Gaussian smoothing function, an image is convolved with a Gaussian filter as expressed by Equation (1).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

The Gaussian filter performs smoothing by changing the image structure. As shown in Equation (1), the standard deviation  $\sigma$  plays an important role because the image structure changes according to the value of  $\sigma$ . In equation 1,  $x$  and  $y$  are the spatial coordinates on the Gaussian filter and  $\sigma$  is the pre-defined standard deviation. To select the appropriate value, different sigma values were applied to an input image. For a retrospective comparison, please see below an input image in correspondence to different sigma values. In this work, the Gaussian kernel size is set to be  $3 \times 3$  pixels, and the standard deviation in the  $x$ -direction and  $y$ -direction is set as 2.

As shown in Figure 4, for a smaller value of sigma, it was possible to detect fine details and edges, whereas, with a higher value of sigma, it was possible to detect large-scale

edges. Once the finer edges with the smaller value of sigma were detected, the canny edge detector was used for further image processing.



**Figure 4.** Input image (a), image smoothing with sigma value 2 (b) and sigma value 3 (c).

As the gradient intensity changes, it impacts edges and introduces a change in the maximal intensity along a particular orientation. Therefore, the derivative of the gradient of every pixel in an image is used to find edges [29,34]. The Canny edge detection method [29] involves image smoothing through a Gaussian filter before the edge detection, which helps reduce the false detection of edges. Once the Gaussian filter is applied, the image gradient (partial derivatives) for each pixel is computed by taking the  $x$  and  $y$  derivatives of the Gaussian filter. Once the derivatives of each pixel concerning  $x$  and  $y$  are calculated, the magnitude and orientation maps are built. The gradient vector of an image ( $\nabla k$ ) is represented by Equation (2).

$$\nabla k = \left[ \frac{\partial k}{\partial x} \quad \frac{\partial k}{\partial y} \right] \quad (2)$$

The derivative function is approximated and applied in the horizontal and vertical directions of the image for  $x$  and  $y$ . The gradient direction and magnitude of each pixel of the image are calculated by Equations (3) and (4), respectively.

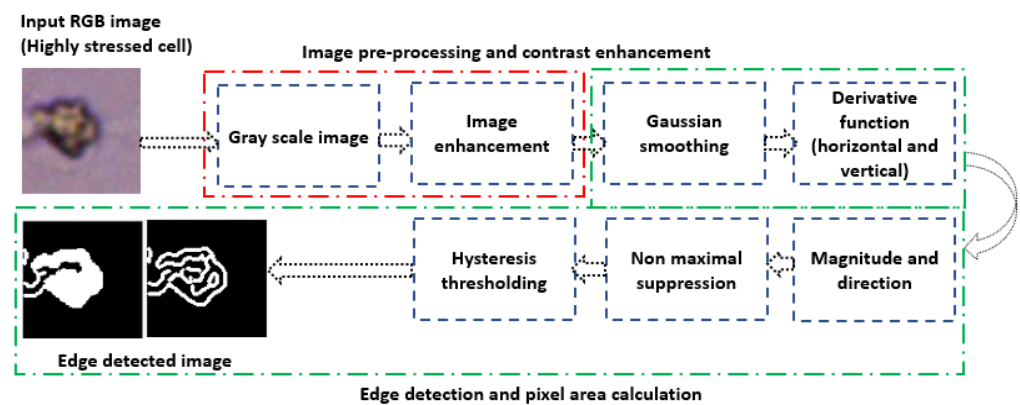
$$\text{Direction } (\theta) = \arctan\left(\frac{\frac{\partial k}{\partial y}}{\frac{\partial k}{\partial x}}\right) \quad (3)$$

The edge strength is calculated by the gradient magnitude as shown below:

$$\text{Magnitude} = \sqrt{\left(\frac{\partial k}{\partial x}\right)^2 + \left(\frac{\partial k}{\partial y}\right)^2} \quad (4)$$

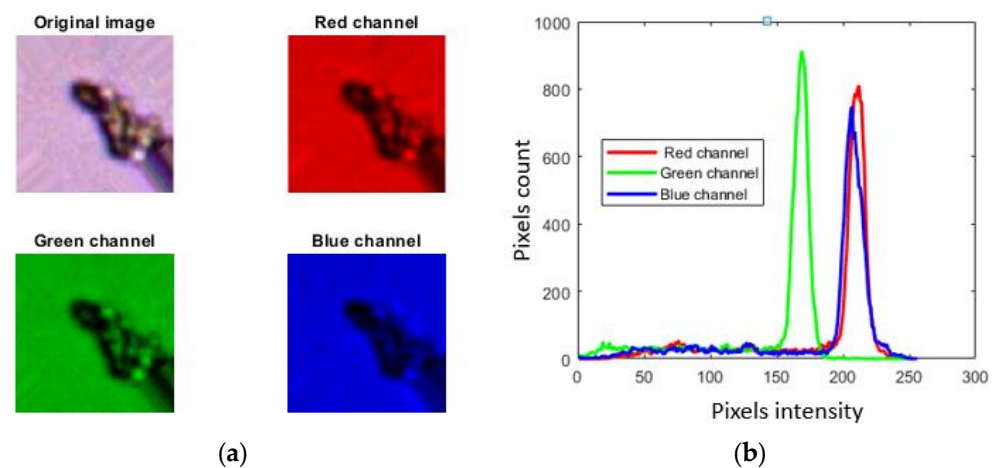
Further steps include non-maximal suppression, which makes the detected edge regions thinner. Finally, hysteresis thresholding was used for both high and low thresholds. Considering the wide range of hysteresis threshold values, it was necessary to manually set the threshold values to detect certain edges of healthy, moderate and highly stressed cells that were smoothed by the Gaussian smoothing filter. The threshold values of 0.1 and 0.4 were used, which means that the edge pixels above the upper limit of 0.4 were considered, and edge pixels below the threshold value of 0.1 were discarded. An overall flowchart of image pre-processing and edge detection is shown in Figure 5.





**Figure 5.** This figure shows an overall flow chart of enhanced edge detection by using Canny operators.

The Canny edge detection method integrated with image pre-processing/enhancement yielded good results; however, this method is time-consuming and requires much effort. As one of the shortcomings of the Canny edge detector is that it does not properly capture the edges due to the Gaussian smoothing function, a histogram-based method was also investigated for image pre-processing where the green channel of the cell image was extracted. The green channel offers the highest local contrast in comparison to the red and blue channels, and the intensity difference between the cell boundary and background image is best observed, which appears to be the most appropriate in terms of under-lit and over-saturated regions (please see Figure 6). The steps involved in this approach included: converting the RGB cell image into the green channel and locating the cell boundary by finding the maximum intensity pixels. The segmented image was binarised, and the exact area was located to create the dataset for backend ANN training, testing and mapping onto an FPGA device.



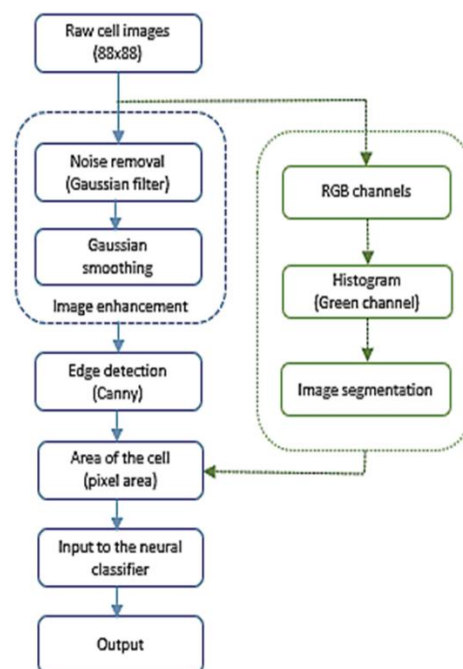
**Figure 6.** (a) Microscopy zoomed image of a highly stressed HK-2 cell and its corresponding channels; (b) histogram of the image.

In this paper, a modified Canny's edge detection method was chosen because modifying its parameters allowed greater flexibility and made it suitable for this specific application.

### 2.3. Backend Classification Using Artificial Neural Network (ANN)

The data extracted through the image pre-processing method in the previous section were used to classify cell images between healthy and moderately stressed cells and between moderately stressed and highly stressed cells. The ANN was implemented in software for simulations, and later the architecture was mapped onto an FPGA device, as

elaborated in the following section. The trained neural classifier was tested with the test samples with ten test images each for healthy, moderately stressed and highly stressed kidney cells. ANN-based classification for health-related dataset have been reported in the literature and details are provided in [35,36]. Several lightweight feed-forward neural architectures were explored in this study to classify healthy, moderately stressed and highly stressed images. An overall simulation flowchart is shown in Figure 7, and the neural architecture is shown in Figure 8. As one of the major challenges in implementing neural network architecture on FPGAs is the limited number of multipliers available on any device; therefore, it is important that a network is mapped with a limited number of neurons and hidden layers. The proposed architecture strikes a good balance between accuracy and the required numbers of neurons and hidden layers that could effectively be mapped on a resource-limited FPGA device. In order to train the network, the function “trainbr” was used. This function is particularly useful because it performs Bayesian regularisation backpropagation that disables validation stops by default. As validation is usually used as a form of regularisation, “trainbr” has its own form of validation built into the algorithm. The training goal was set as 0.001, where the error was calculated by using the ‘mse’ function. The neural architecture shown in Figure 8 offered the best results in terms of test image classification accuracy with a minimum number of neurons and hidden layers. This was particularly important in the context of the hardware implementation for real-time classification.



**Figure 7.** This figure shows the steps involved in raw image processing and NN training and testing.

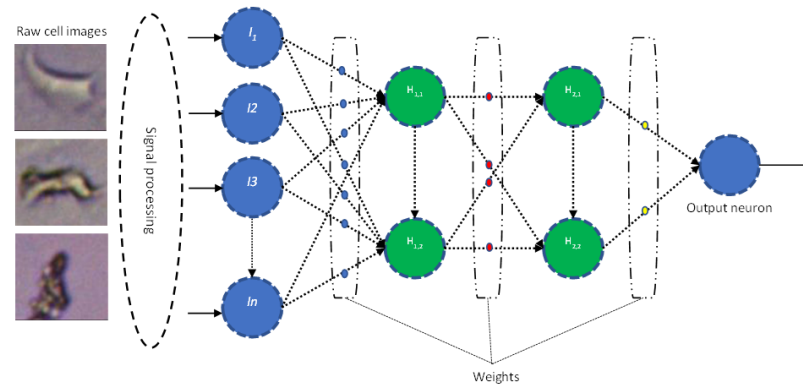


Figure 8. Neural network architecture for healthy, moderate and highly stressed cell classification.

### 3. Results

#### 3.1. Software

The dataset was formulated by using the previously mentioned pre-processing techniques, where test samples were classified as healthy, moderately stressed and highly stressed cells. The best results were achieved when image enhancement and Gaussian noise removal were used in combination with the Canny edge detector. In total, 15 samples, each from healthy, moderate and highly stressed cells, were used for training, where the calculated cell area was used as an input to the network. The network was tested with ten samples from each class. The classification accuracy between healthy and moderately stressed cells and moderately stressed and highly stressed cells in terms of regression is shown in Figures 9a and 10a. The convergence graphs for training between healthy and moderately stressed and moderately stressed and highly stressed cells are shown in Figures 9b and 10b, respectively. The confusion matrices for both scenarios are shown in Figure 11a,b, respectively.

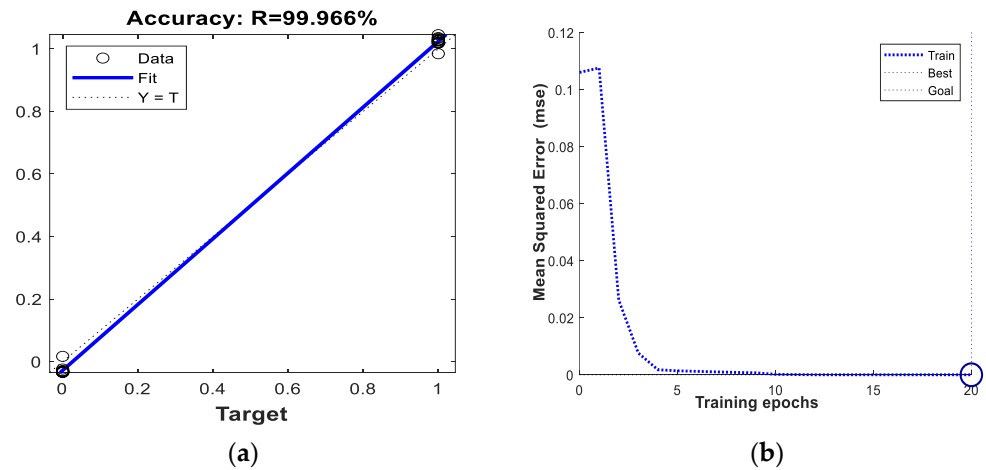


Figure 9. Training accuracy between healthy and moderately stressed cells (a) and corresponding convergence graph (b).



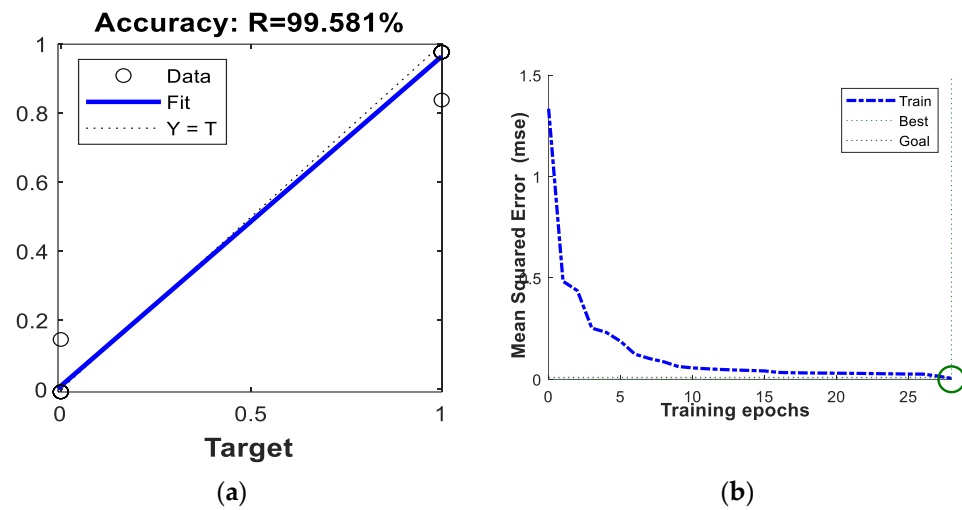


Figure 10. Training accuracy between moderate and highly stressed cells (a) and corresponding convergence graph (b).

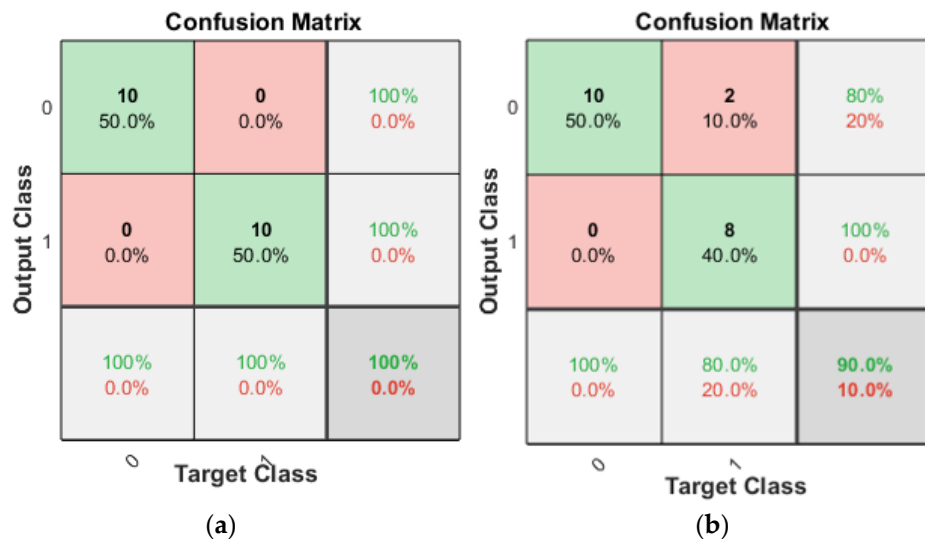


Figure 11. This figure shows the test accuracies in terms of confusion matrices for healthy and moderately stressed cells (a) and moderately and highly stressed cells (b).

As seen in Figure 9, the network for healthy and moderately stressed cells converged in a total of 20 epochs, whereas 28 epochs were used for the network to converge for moderately stressed and highly stressed cells. In total, almost 100% training accuracy was achieved with both classifiers with the feed-forward network. The neural network performed well with a relatively small number of training examples as the training loss decreased to 0, which is what was needed for these experiments. As in each backpropagation training session, different weights and biases were initialised; thus, several networks were trained to ensure that an appropriate network with good generalisation was found. When the training performance goal was achieved, further training was stopped, which also helped to improve the generalisation of the network. Once the training was stopped, the weights and biases were returned. Given that a relatively small dataset and a shallow network were used for training the network to improve generalisation and address the network overfitting, a regularisation technique was adopted. This technique involved modifying the performance function, and instead of using the sum of square performance function to calculate network error, another term was added to the performance function that consisted of the mean of the sum of the square of network weights and biases, as shown in Equation (5), where  $\gamma$  is the performance ratio,  $mresg$  is the mean square error

regularised and  $mstw$  (mean square weights) is represented by Equation (6). Hence, by having smaller weights and biases, the network response could be forced to become smoother, which is less likely to overfit; however, the drawback of this function is that it does not use the validation set [37,38].

$$msereg = \gamma * mstw + (1 - \gamma) * mse \quad (5)$$

$$mstw = \frac{1}{n} \sum_{j=1}^n (w_j)^2 \quad (6)$$

The mean square error is calculated as shown in Equation (7).

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (7)$$

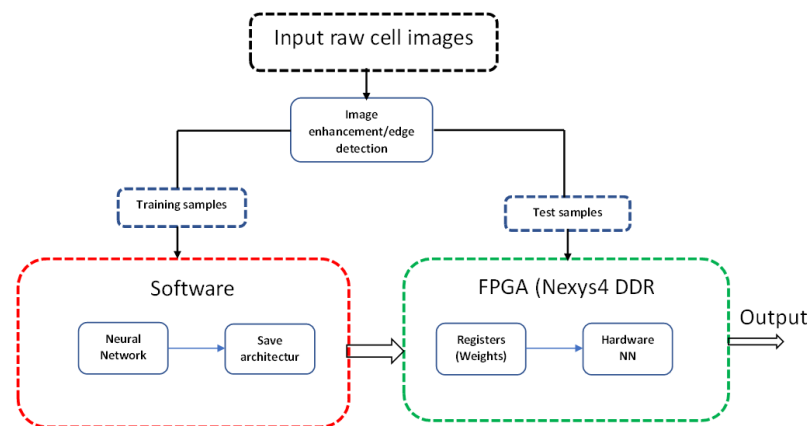
As shown in Equation (7), MSE is the mean square error,  $n$  the number of data points,  $Y_i$  the observed values and  $\hat{Y}_i$  the predicted values.

The classification accuracy in terms of regression for healthy and moderately stressed cells was recorded as 99.96%, as shown in Figure 9a. The classification accuracy for moderately stressed and highly stressed cells was recorded as 99.58%, as shown in Figure 10a.

The total dataset was divided into training and testing where completely independent test samples were used to test the trained network. In total, ten samples, each from healthy, moderately stressed and highly stressed cells, were taken for classifier testing, and relevant confusion matrices are shown in Figure 11. As shown in Figure 11a,b, the correctly classified samples were identified by diagonal green squares where 100% accuracy was achieved for test datasets between healthy and moderately stressed and 90% between moderately and highly stressed cells. As shown in Figure 11b, two highly stressed samples were misclassified. Therefore, all the moderately stressed cells were classified as 100%, and the overall accuracy for highly stressed cells is calculated as 80%. As shown in Figure 11a that the first two diagonal cells show the number and percentage of correct classifications by the trained network. In total, 10 normal cell images were correctly classified. This corresponds to 50% of all images. Similarly, 10 samples were correctly classified as moderately stressed cells which correspond to 50% of all images. None of the images was misclassified, which corresponds to 100% overall accuracy. In Figure 11b, all 10 images were correctly classified as moderately stressed cells, whereas 8 images were correctly classified as highly stressed cells which are 40% of all images, and 2 images were misclassified, which is 20% of highly stressed cells. Hence, total classification accuracy corresponds to 90%.

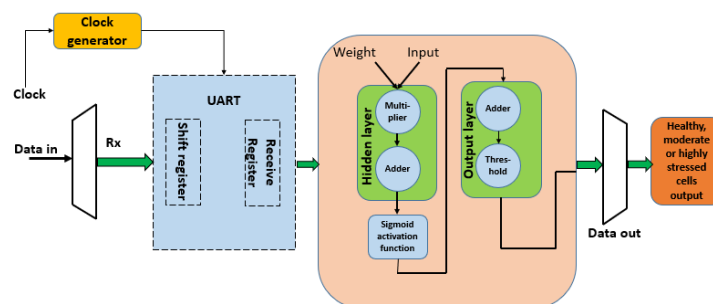
### 3.2. Hardware Accelerator (FPGA)

As biological neurons are inherently parallel, to fully exploit the inherent parallelism of artificial neurons, heterogeneous embedded platforms are needed to accelerate computational tasks inspired by the architecture of the human brain. To overcome the limitations of homogenous CPU-based platforms, embedded hardware platforms have been investigated, such as GPUs, FPGAs and ASIC (Application Specific Integrated Circuits) [1–8]. FPGA implementations are particularly promising because of the reconfigurability, cost and accelerated computing speed they offer. In this paper, the authors demonstrated an FPGA prototype where software-based neural architecture explored in the previous section of this paper was mapped onto an FPGA Artix-7 chip, and results are reported. The hardware implementation was performed on the Nexys4FPGA board (Nexys4DDR) to minimise the classification time between healthy, moderately stressed and highly stressed kidney cells. The software architecture investigated in this paper is implemented onto an FPGA device by using Hardware Description Language (VHDL). The hardware-software design flow that includes *MATLAB* and Xilinx Vivado FPGA implementation is shown in Figure 12, where the architecture was mapped onto an Artix7 FPGA chip.



**Figure 12.** This figure shows the block diagram of neural network architecture implementation on the Nexys4 DDR FPGA device.

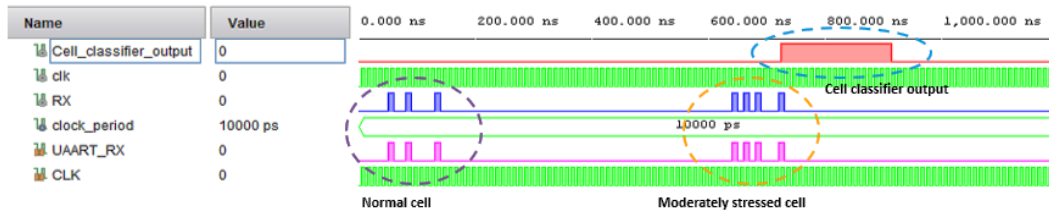
As shown in Figure 12, the raw cell images were used as input to the software (*MATLAB*) environment where image pre-processing was performed, and once the input images were pre-processed, the dataset was split into training and test samples. The training samples were used to train the neural network, and the test images were used to evaluate the classification accuracy. Neural network architecture parameters were extracted from the previously saved software-based design and mapped onto the hardware architecture. The hardware implementation involved simulation and synthesis of design at the gate level. The input to the FPGA was interfaced through the UART (Universal Asynchronous Receiver and Transmitter), where a 16-bit input in terms of the area of the segmented cell was fed into the architecture as a test sample. For the serial interface with the *MATLAB* environment, UART was modelled and interfaced with the number of hardware blocks, as shown in Figure 13. Data transmission protocol was established by sending data bit by bit with a start and stop bit to differentiate between different samples. The sixteen-bit data was loaded into the shift register and transmitted through the UART for the neural hardware classifier.



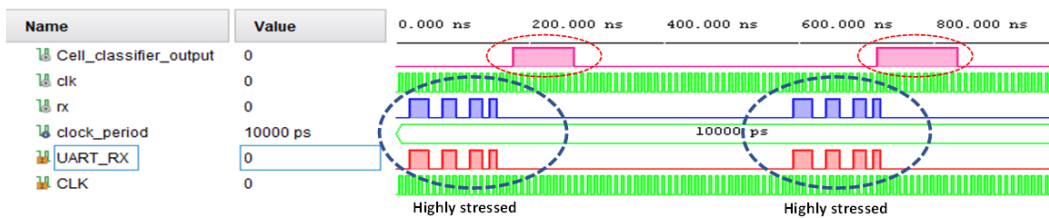
**Figure 13.** Hardware neural network architecture interfaced through UART.

The hardware design was simulated and synthesised by using the Xilinx Vivado toolchain. The hardware classifier implementation results between healthy and moderately stressed cells are shown in Figure 14, and the simulation results for highly stressed cells consecutively presented to the network are shown in Figure 15. As shown in Figure 14, the cell classifier output stays '0' and only becomes '1' when a moderately stressed cell sample is presented to the network. To rigorously test the architecture, Figure 15 shows two consecutive samples of highly stressed cells presented to the network and were correctly classified with '1' at the output. The software was implemented in *MATLAB* running on Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz, 8 GB RAM, *MATLAB*, 9.10.0.1739362 (R2021a), USA. The software test samples took 0.22 s per sample to classify, whereas hardware-based implementation on FPGA took 400 ns to compute one sample. In comparison to the software implementation, it is almost  $5.5 \times 10^5$  times faster on a dedicated hardware accelerator.

The total power consumption was calculated by using the Xilinx Vivado power estimation and analysis tool. The device’s static power is referred to as leakage, which represents the transistor leakage power when the device is powered on and not configured. The design dynamic power consumption represents the power consumption from the logic utilisation and switching activity. Total on-chip power was recorded as 19 W. A further breakdown of total power consumption is shown in Table 1.



**Figure 14.** This figure shows the classifier output between healthy normal cells and moderately stressed cells.



**Figure 15.** This figure shows the classifier output for consecutive presentation of two highly stressed cells.

**Table 1.** Total on-chip power consumption.

<b>Static Power Consumption</b>		<b>0.53 W (1%)</b>
Dynamic Power Consumption	18.47 W (99%)	Logic (14.5 W) Signals (3.79 W) I/O (0.067 W)

Tables 2 and 3 show the neural network response on both software and hardware platforms. Tables 4 and 5 show the total chip area utilisation of the implemented hardware design. Total on-chip power consumption shows that dynamic power significantly dominates static power. The FPGA test environment is shown in Figure 16 where hardware design was downloaded on the FPGA chip and interfaced with an external LED as well as an oscilloscope to verify the classified samples.

**Table 2.** This table shows the output of healthy and moderately stressed cells from both MATLAB and the FPGA device.

NN Output (MATLAB)	NN Output FPGA	NN Output (MATLAB)	NN Output FPGA
0.1425 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0058 (Healthy)	0 (Healthy)	0.9213 (Moderately stressed)	1 (Moderately stressed)
0.0418 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0056 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0086 (Healthy)	0 (Healthy)	0.9213 (Moderately stressed)	1 (Moderately stressed)
0.0425 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0058 (Healthy)	0 (Healthy)	0.9213 (Moderately stressed)	1 (Moderately stressed)
0.0418 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0056 (Healthy)	0 (Healthy)	0.942 (Moderately stressed)	1 (Moderately stressed)
0.0086 (Healthy)	0 (Healthy)	0.9213 (Moderately stressed)	1 (Moderately stressed)

**Table 3.** This table shows the output of moderately stressed cells and highly stressed cells from both MATLAB and the FPGA device.

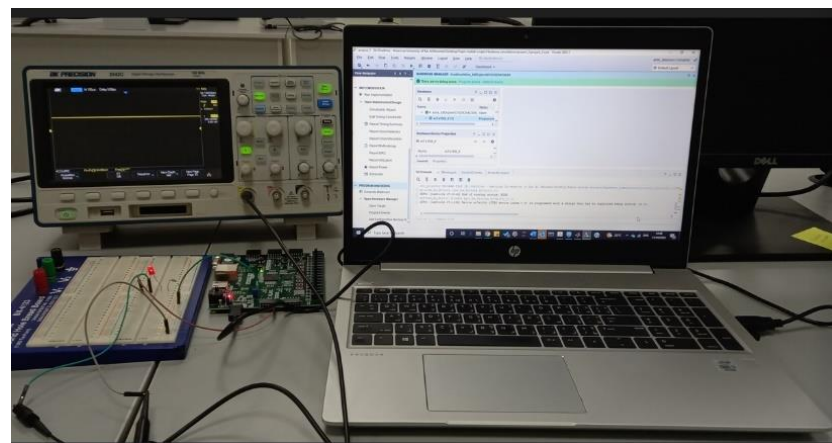
NN Output (MATLAB)	NN Output FPGA	NN Output (MATLAB)	NN Output FPGA
0.0016 (moderately stressed)	0 (moderately stressed)	1.0044 (highly stressed)	1 (highly stressed)
0.0015 (moderately stressed)	0 (moderately stressed)	1.0059 (highly stressed)	1 (highly stressed)
0.0017 (moderately stressed)	0 (moderately stressed)	1.0047 (highly stressed)	1 (highly stressed)
0.0014 (moderately stressed)	0 (moderately stressed)	0.9302 (highly stressed)	1 (highly stressed)
0.0015 (moderately stressed)	0 (moderately stressed)	1.0044 (highly stressed)	1 (highly stressed)
0.0314 (moderately stressed)	0 (moderately stressed)	0.0035 (moderately stressed)	0 (moderately stressed)
0.0215 (moderately stressed)	0 (moderately stressed)	1.0047 (highly stressed)	1 (highly stressed)
0.0051 (moderately stressed)	0 (moderately stressed)	0.9302 (highly stressed)	1 (highly stressed)
0.0745 (moderately stressed)	0 (moderately stressed)	1.0044 (highly stressed)	1 (highly stressed)
0.0562 (moderately stressed)	0 (moderately stressed)	0.0016 (moderately stressed)	0 (moderately stressed)

**Table 4.** Resource utilisation of FPGA device (Nexyx4DDR—Artix 7).

FPGA Resources	Available	Utilised
Slice (LUTs)	63,400	2408 (<4%)
Slice registers (FFs)	126,800	35 (<1%)
Bonded IO blocks	210	3 (<2%)
BUFGCTRL	31	1 (<4%)

**Table 5.** Neural hardware architecture utilisation.

FPGA Logic Blocks	Utilisation
Adder	10
Multiplier	490
Sigmoid	1890
UART	119



**Figure 16.** FPGA test environment.

#### 4. Discussion

Automated cell classification is a challenging task in computer vision. There are very few studies reported in the literature that specifically addresses the problem from a hardware-software co-design perspective. This paper demonstrates a promising solution based on a rather simple shallow neural network for the detection of HK-2 kidney cell injury. The cell injury is caused by high amounts of extracellular calcium. The HK-2 cells were cultured in our lab at the American University of Ras Al-Khaimah, UAE. The classification accuracy of our proposed ANN is reported to be 100% for detection between



healthy and moderately stressed cells and 90% between moderately stressed and highly stressed cells.

Deep learning-based techniques such as CNN have shown great promise for cell toxicity detection, as reported in [17]. However, such classifiers suffer from several aspects, including the need for a very large amount of raw data for network training, which retrospectively requires much longer training and processing time, and computational resources. This limitation makes it less practical for real-time cell detection and classification. Since image enhancement is an important aspect of cell analysis, this study elaborated on its use in computer vision in retrospect to the applied AI. As demonstrated in this study, that image enhancement, complemented by appropriate edge detection, significantly improves the overall network performance. As CNN-based approaches do not necessarily require the processing of input cell images, raw images could be used directly as input to the network. However, collecting a very large dataset for cell classification in the lab environment is not practical. It would almost always be necessary to use data augmentation techniques and create synthetic data to facilitate a sufficient number of examples to train the network with multiple layers [39,40]. As reported in [39], the authors investigated CNN for cell classification using microscope images, where CNN was built using VGG-16 architecture. The authors of the paper [39] reported having used 13 convolutional layers, 5 max pooling and 2 fully connected layers followed by three output neurons to classify three different categories of cells. Images were cropped from  $224 \times 224$  to  $64 \times 64$  sizes and were used as input, where the first 14 layers were fine-tuned with transfer learning. The training was performed on Nvidia GeForce GTX GPU, which took 1000 epochs to train the network. The network training required 6 h. The train and test accuracies are reported as 93%. Similarly, authors in [16] reported CNN-based cell classification for six different categories. Microscopy cell images with a resolution of  $2592 \times 1944$  were used, and data was augmented to produce a total of 4587 train images. A relatively small CNN was implemented with four convolutional layers, three layers each for max pooling and a dense layer followed by a softmax layer which contained six neurons to classify six different cell image categories. The network was trained on Intel(R) Core(TM) i7-10700T CPU @ 2.00 GHz. Both training and test accuracies were reported as 97% and 98%, respectively. It was reported that almost 13–18 h were required to train the network.

While CNN-based approaches could be applied for cell classification, it is obvious that high-end computational resources and much longer training time are required. Despite the use of hardware accelerators such as GPUs, the training time is impractical to meet the requirement of real-time classification. As cellular features of the HK-2 kidney cells cannot easily be detected by the human eye, we proposed and demonstrated a hardware-software co-design approach to train a shallow network and improve diagnostics in real-time. The superiority of the proposed co-design technique was demonstrated with 100% training and test accuracy for healthy and moderately stressed cells and 100% training and 90% test accuracy for moderately stressed and highly stressed cells. As the quality of images varies from sample to sample due to varying light intensities and the limitations of current light microscope imaging technology, the results achieved are promising. We also demonstrated that while image enhancement techniques do not play an important role in CNN-based classification, however, for real-time FPGA implementation, it is a requirement due to the limited number of computational resources available on a single FPGA device. Instead of relying on a very deep network to learn patterns in cellular image samples, we can alleviate this burden at the network level by rather adopting and enhancing techniques for image enhancement and including edge detection to precisely detect the boundaries of the region of interest. As can be seen from the confusion matrices provided for training and testing, we achieved 100% results, except for two test samples which were misclassified between moderately stressed and highly stressed cells. We assume that it was mainly due to the similarity between the cell structures, which were less distinct in comparison to the healthy and moderately stressed cells, which had a clear distinction in terms of cell boundaries and shapes.

Most of the existing studies in biomedicine over-rely on CNN-based classification techniques. This is perhaps due to the ease and availability of existing tools and libraries that help facilitate such exploration. Nonetheless, the authors argue that using CNN as a black box will restrict further advancement in the field unless the low-level dynamics are properly understood. As a limited number of datasets are currently available and most of the existing datasets need to be augmented for deep neural networks training, in this paper, we successfully demonstrated that a small dataset collected in our biotechnology lab can be used with appropriate image enhancement and detection techniques to achieve excellent performance in real-time. This method further enhances the clinical relevance of our study, where real-time information is needed for diagnostics. It is particularly important to address the limitations of long training times and the requirement for specialised hardware resources for CNN-based implementations. The hardware acceleration proposed in this study demonstrates that cell detection can be performed in real-time by mapping the trained network onto an FPGA device that paves the way for personalised digital healthcare. In the proposed study, image pre-processing (software) took 2–2.5 s for each sample to compute. For software, only image detection on a trained neural network took 0.22 s. However, once the trained neural network is mapped onto an FPGA device, the image detection only took 400 ns to compute one sample.

Despite the number of advantages this study offers, one of the limitations is the amount of time required to collect samples in the lab environment. Secondly, our study results cannot yet be directly applied in clinical setup because cell images could include several types, which require expert insight into the cell morphologies. We envisage that further enhancement in image acquisition quality will improve the hardware software classification of cell images in real time.

## 5. Conclusions

In this study, the authors of the paper demonstrated a unique solution based on hardware-software co-design. The proposed solution is user-friendly and could be used by researchers and clinicians. While clinical experts can provide input and detect any changes or injury reflected through the kidney HK-2 cells, they are prone to errors. Therefore, automating such techniques and developing a robust AI-enabled real-time platform is advantageous. This paper proposes a framework to detect cell toxicity in real time by mapping a shallow neural network on an FPGA device. The proposed holistic approach integrates the biological sample images collected from the lab environment, developed and deployed onto an AI-enabled hardware accelerator for real-time classification. The input samples were pre-processed and mapped onto the software and digital hardware platforms, which is critical to optimise personalised care and management. While deep learning-based techniques hold great promise, we demonstrated that such techniques require an enormous amount of time for training as well as specialised computational resources. This limitation can be overcome with the proposed framework, where input samples are carefully pre-processed, and a network is developed and mapped onto embedded hardware for real-time detection.

**Author Contributions:** Conceptualization, A.G.; methodology, A.G.; software, A.G.; validation, A.G., R.H. and C.H.S.; formal analysis, A.G.; investigation, A.G. and R.H.; resources, A.G., C.H.S., S.K., R.H. and A.B.; data curation, A.G. and R.H.; writing—original draft preparation, A.G.; writing—review and editing, A.G., R.H., C.H.S., D.-J.L., S.K. and A.B.; visualization, A.G. and R.H.; supervision, A.G.; project administration, A.G.; 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:** Not applicable.

**Acknowledgments:** We would like to thank the support of American University of Ras al Khaimah, UAE.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Himavathi, S.; Anitha, D.; Muthuramalingam, A. Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization. *IEEE Trans. Neural Netw.* **2007**, *18*, 880–888. [[CrossRef](#)] [[PubMed](#)]
2. Medus, L.D.; Iakymchuk, T.; Frances-Villora, J.V.; Bataller-Mompean, M.; Rosado-Munoz, A. A Novel Systolic Parallel Hardware Architecture for the FPGA Acceleration of Feedforward Neural Networks. *IEEE Access* **2019**, *7*, 76084–76103. [[CrossRef](#)]
3. Zhang, C.; Wu, D.; Sun, J.; Sun, G.; Luo, G.; Cong, J. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster. In Proceedings of the ISLPED'16: International Symposium on Low Power Electronics and Design, San Francisco, CA, USA, 8–10 August 2016; pp. 326–331.
4. Ghani, A.; McGinnity, T.M.; Maguire, L.P.; Harkin, J. Area Efficient Architecture for Large Scale Implementation of Biologically Plausible Spiking Neural Networks on Reconfigurable Hardware. In Proceedings of the 2006 International Conference on Field Programmable Logic and Applications, Madrid, Spain, 28–30 August 2006; pp. 1–2. [[CrossRef](#)]
5. Bataller-Mompeán, M.; Martínez-Villena, J.M.; Rosado-Muñoz, A.; Francés-Villora, J.V.; Guerrero-Martínez, J.F.; Wegrzyn, M.; Adamski, M. Support tool for the combined software/hardware design of on-chip ELM training for SLFF neural networks. *IEEE Trans. Ind. Informat.* **2016**, *12*, 1114–1123. [[CrossRef](#)]
6. Nikitakis, A.; Makantasis, K.; Tampouratzis, N.; Papaefstathiou, I. A Unified Novel Neural Network Approach and a Prototype Hardware Implementation for Ultra-Low Power EEG Classification. *IEEE Trans. Biomed. Circuits Syst.* **2019**, *13*, 670–681. [[CrossRef](#)] [[PubMed](#)]
7. Ghani, A.; Aina, A.; See, C.H.; Yu, H.; Keates, S. Accelerated Diagnosis of Novel Coronavirus (COVID-19)—Computer Vision with Convolutional Neural Networks (CNNs). *Electronics* **2022**, *11*, 1148. [[CrossRef](#)]
8. Ghani, A.; See, C.H.; Sudhakaran, V.; Ahmad, J.; Abd-Alhameed, R. Accelerating Retinal Fundus Image Classification Using Artificial Neural Networks (ANNs) and Reconfigurable Hardware (FPGA). *Electronics* **2019**, *8*, 1522. [[CrossRef](#)]
9. Chen, X.; Zhou, X.; Wong, S.T.C. Automated Segmentation, Classification, and Tracking of Cancer Cell Nuclei in Time-Lapse Microscopy. *IEEE Trans. Biomed. Eng.* **2006**, *53*, 762–766. [[CrossRef](#)]
10. Moen, E.; Bannon, D.; Kudo, T.; Graf, W.; Covert, M.; Van Valen, D. Deep learning for cellular image analysis. *Nat. Methods* **2019**, *16*, 1233–1246. [[CrossRef](#)]
11. Grimm, J.B.; Muthusamy, A.K.; Liang, Y.; Brown, T.A.; Lemon, W.C.; Patel, R.; Lu, R.; Macklin, J.J.; Keller, P.J.; Ji, N.; et al. A general method to fine-tune fluorophores for live-cell and in vivo imaging. *Nat. Methods* **2017**, *14*, 987–994. [[CrossRef](#)]
12. Megason, S.G. In toto imaging of embryogenesis with confocal time-lapse microscopy. *Methods Mol. Biol.* **2009**, *546*, 317–332.
13. Chen, B.-C.; Legant, W.R.; Wang, K.; Shao, L.; Milkie, D.E.; Davidson, M.W.; Janetopoulos, C.; Wu, X.S.; Hammer, J.A., 3rd; Liu, Z.; et al. Lattice light-sheet microscopy: Imaging molecules to embryos at high spatiotemporal resolution. *Science* **2014**, *346*, 1257998. [[CrossRef](#)] [[PubMed](#)]
14. Smith, K.; Piccinini, F.; Balassa, T.; Koos, K.; Danka, T.; Azizpour, H.; Horvath, P. Phenotypic Image Analysis Software Tools for Exploring and Understanding Big Image Data from Cell-Based Assays. *Cell Syst.* **2018**, *6*, 636–653. [[CrossRef](#)] [[PubMed](#)]
15. Zhou, Y.; Greka, A. Calcium-permeable ion channels in the kidney. *Am. J. Physiol. Renal. Physiol.* **2016**, *310*, F1157–F1167. [[CrossRef](#)] [[PubMed](#)]
16. Rouse, D.; Suki, W.N. Renal control of extracellular calcium. *Kidney Int.* **1990**, *38*, 700–708. [[CrossRef](#)] [[PubMed](#)]
17. 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](#)]
18. Huang, Q.; Li, W.; Zhang, B.; Li, Q.; Tao, R.; Lovell, N.H. Blood Cell Classification Based on Hyperspectral Imaging With Modulated Gabor and CNN. *IEEE J. Biomed. Health Inform.* **2019**, *24*, 160–170. [[CrossRef](#)]
19. Saraswat, M.; Arya, K. Automated microscopic image analysis for leukocytes identification: A survey. *Micron* **2014**, *65*, 20–33. [[CrossRef](#)]
20. Wang, Q.; Chang, L.; Zhou, M.; Li, Q.; Liu, H.; Guo, F. A spectral and morphologic method for white blood cell classification. *Opt. Laser Technol.* **2016**, *84*, 144–148. [[CrossRef](#)]
21. Froom, P.; Havis, R.; Barak, M. The rate of manual peripheral blood smear reviews in outpatients. *Clin. Chem. Lab. Med.* **2009**, *47*, 1401–1405. [[CrossRef](#)]
22. Li, W.; Du, Q. Gabor-filtering based nearest regularized subspace for hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 1012–1022. [[CrossRef](#)]
23. Kang, X.; Li, C.; Li, S.; Lin, H. Classification of hyperspectral images by Gabor filtering based deep network. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *11*, 1166–1178. [[CrossRef](#)]
24. Hinton, G.E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [[CrossRef](#)]
25. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information, Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
26. Kumar, A.; Kim, J.; Lyndon, D.; Fulham, M.; Feng, D. An ensemble of fine-tuned convolutional neural networks for medical image classification. *IEEE J. Biomed. Health Inform.* **2017**, *21*, 31–40. [[CrossRef](#)]

27. Gao, Z.; Wang, L.; Zhou, L.; Zhang, J. Hep-2 cell image classification with deep convolutional neural networks. *IEEE J. Biomed. Health Inform.* **2017**, *21*, 416–428. [[CrossRef](#)]
28. Wang, Q.; Zheng, Y.; Yang, G.; Jin, W.; Chen, X.; Yin, Y. Multiscale rotation-invariant convolutional neural networks for lung texture classification. *IEEE J. Biomed. Health Inform.* **2018**, *22*, 184–195. [[CrossRef](#)]
29. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [[CrossRef](#)]
30. Marr, D.; Hildreth, E. Theory of edge detection. *Proc. R. Soc. London. Ser. B. Biol. Sci.* **1980**, *207*, 187–217.
31. Torre, V.; Poggio, T.A. On edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 147–163. [[CrossRef](#)]
32. Gunn, S.R. On the discrete representation of the Laplacian of Gaussian. *Pattern Recognit.* **1999**, *32*, 1463–1472. [[CrossRef](#)]
33. Mlsna, P.A.; Rodriguez, J.J. Gradient and Laplacian edge detection. In *The Essential Guide to Image Processing*; Academic Press: Cambridge, MA, USA, 2009; pp. 495–524.
34. Parker, J.R. *Algorithms for Image Processing and Computer Vision*; John Wiley & Sons, Inc.: New York, NY, USA, 1997; pp. 23–29.
35. Sadanandan, S.K.; Ranefall, P.; Le Guyader, S.; Wählby, C. Automated training of deep convolutional neural networks for cell segmentation. *Sci. Rep.* **2017**, *7*, 7860. [[CrossRef](#)]
36. Stringer, C.; Wang, T.; Michaelos, M.; Pachitariu, M. Cellpose: A generalist algorithm for cellular segmentation. *Nat. Methods* **2020**, *18*, 100–106. [[CrossRef](#)] [[PubMed](#)]
37. Krogh, A.; Hertz, K. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems 4*; Morgan Kaufmann: San Francisco, CA, USA, 1992.
38. MacKay, D.J.C. Bayesian interpolation. *Neural Comput.* **1992**, *4*, 415–447. [[CrossRef](#)]
39. Oei, R.W.; Hou, G.; Liu, F.; Zhong, J.; Zhang, J.; An, Z.; Xu, L.; Yang, Y.-H. Convolutional neural network for cell classification using microscope images of intracellular actin networks. *PLoS ONE* **2019**, *14*, e0213626. [[CrossRef](#)]
40. Shin, H.C.; Roth, H.R.; Gao, M.; Lu, L.; Xu, Z.; Nogues, I.; Yao, J.; Mollura, D.; Summers, R.M. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Trans. Med. Imaging* **2016**, *35*, 1285. [[CrossRef](#)]