Sensors and Materials, Vol. 37, No. 5 (2025) 2115–2133
MYU Tokyo

S & M 4046

Harnessing Deep Neural Networks for Rapid Knife Wound Identification in Forensic Science: A Proof-of-concept Study

Chun-Ta Wei,[1] Jeff Cheng-Lung Lee,[2] and Yu-Guang Chen[3*]

[1]School of Electronics and Communication Engineering, Quanzhou University of Information Engineering, No. 249, Bodong Road, Fengze District, Quanzhou, Fujian 362000 China
[2]Department of Criminal Investigation, Taiwan Police College, No. 153, Sec. 3, Xinglong Rd., Wenshan Dist., Taipei City 116078, Taiwan
[3]Department of Electrical Engineering, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 320317, Taiwan

(Received February 6, 2025; accepted April 28, 2025)

Keywords: crime scene investigations, forensic science, deep learning, field programmable gate array, knife mark

An automated knife-wound weapon-identification system, containing forensic science, image processing, and deep learning, is introduced in this study. Aimed at improving the analysis of tool marks, particularly knife-related injuries common in violent crimes, in this research, we address the limitations of traditional forensic methods, such as the lack of real wound images and the need for fast processing. To overcome these challenges, a simulated wound feature dataset was developed to train the LeNet-5 neural network model, which is deployed on a portable field-programmable gate-array-based system. This allows for the real-time recognition of tools from wound images in a camera sensor, achieving an average recognition accuracy of 98.8% on the test dataset and processing at least 600 operations per second, providing forensic experts with rapid and accurate insights. The importance of continuously refining the neural network and expanding the dataset to include real wound images is emphasized in this study, aiming for accuracy improvement. The potential of this system transforms crime scene investigations, offering quick, objective, and reliable analyses that can expedite weapon identification and aid in the swift resolution of criminal cases.

1. Introduction

Tool mark analysis is a vital forensic discipline that identifies and interprets marks left by tools at crime scenes.(1) These marks can link specific tools or weapons to crimes, playing a key role in investigations and prosecutions.(2) The Association of Firearms and Tool Mark Examiners defines a tool mark as an impression, scrape, or abrasion created when a tool interacts with a surface with sufficient force to leave a unique imprint.(3,4) Tool marks are categorized by their formation mechanisms: compression marks from pressing, friction marks from sliding, and cutting marks, such as those made by knives, which combine compression and friction. Knife

*Corresponding author: e-mail: andyygchen@ee.ncu.edu.tw
https://doi.org/10.18494/SAM5591

ISSN 0914-4935

marks, in particular, reveal both class and individual characteristics, including blade shape, size, and wear patterns. These analyses are affected by factors such as material hardness, applied pressure, motion, and surface irregularities.[5] Identifying knife marks is especially important in cases involving sharp force trauma, providing clues about the tools used and connecting them to suspects.[6] Advancements in tool mark identification have improved its accuracy and objectivity. Traditionally, examiners used comparison microscopes to visually match tool marks, relying on their expertise.[7] Recent innovations, such as mathematically objective similarity metrics developed by the National Institute of Standards and Technology, have enhanced the precision and reliability of analyses by directly measuring surface topography.[8,9] The study of knife marks is indispensable in forensic science, as it aids in identifying tools, reconstructing events, and solving crimes, ultimately ensuring justice.[10]

Current research on tool mark analysis in forensic science primarily focuses on traditional microscopic examination relying on feature-based methods to analyze tool marks, particularly on bones. Sanyavises and Vachirawongsakorn[11] examined how damaged knife blades affect the morphology of cut marks, whereas Shaw *et al.*[12] characterized chopping-induced marks on bone tissues using a gravity-accelerated chopping stage. Humphrey *et al.*[13] investigated incised bone wounds common in defensive injuries, and Maté-González *et al.*[14,15] employed metric and 3D techniques to differentiate cut marks made by metal, flint, and quartzite tools. McCardle and Stojanovski[16] identified distinguishing features of machete and katana-inflicted marks on various bone types, whereas Weber *et al.*[17] compared silicone casts of katana marks from crime scenes using light microscopy. Despite their forensic significance, these studies rely on handcrafted feature extraction and traditional analysis techniques, without incorporating modern artificial intelligence methods.

Meanwhile, machine learning and deep learning have been increasingly applied to wound recognition, although mostly in medical contexts such as wound care and recovery. Anisuzzaman *et al.*[17] reviewed rule-based, machine learning, and deep learning approaches for clinical wound assessment, whereas Li *et al.*[19] combined watershed segmentation with dynamic thresholding and deep learning for wound image analysis. However, most of these studies focus on chronic wounds rather than forensic applications. Wang *et al.*[20] developed a deep learning framework for segmenting chronic wounds, and Jiao *et al.*[21] employed mask region-based convolutional neural networks (CNNs) to detect burn wounds. In forensic science, deep learning applications remain limited. Oura *et al.*[22] used neural networks to assess gunshot wounds, Boymanov *et al.*[23] analyzed stab-cut wounds caused by kitchen knives, and Zimmermann *et al.*[24] classified skin wounds from forensic photographs. Despite these efforts, no group has comprehensively applied deep learning to analyze knife-inflicted wounds for forensic weapon identification. This research gap highlights the need for an automated system that leverages deep neural networks to enhance forensic tool mark analysis, providing rapid, objective, and accurate weapon identification.

In this study, we aim to develop a deep learning model for analyzing knife marks, addressing a largely unexplored area in forensic science, advancing the understanding of tool marks with novel technologies. By integrating forensic science, image processing, and deep learning, we developed a portable device capable of instantly recognizing tools associated with wounds from

camera-sensor-captured images. Traditional methods face challenges such as limited and unclassified wound image data, sparse datasets, and hardware constraints for portable devices. The constraints in data are overcome by a deep neural network trained on a simulated wound feature dataset and an optimized model. The hardware constraints are alleviated by deploying a portable field-programmable gate array (FPGA), enabling real-time tool recognition with efficient memory management. This development represents a significant leap in forensic technology, providing practical and innovative on-field solutions for crime scene investigations.[25]

2. Materials and Methods

2.1 Tool evaluation

In criminal investigations, accurately predicting and analyzing criminal behavior is crucial. One effective approach involves studying the types of weapon used at crime scenes. Knife-related violence remains significant globally. According to a global study on homicide, conducted by the United Nations Office in 2019,[26] South Africa reported 9424 knife-related deaths, Brazil 9885, and the Philippines 5491. In the U.S., knives are the second most common weapon used in homicides, with over 1500 fatalities annually.[27] In Scotland (2006–2011), 94% of sharp-object homicides involved kitchen knives,[28] and knives with single-edged blades were used in 97.8% of stabbing cases.[29] These statistics highlight the prevalence of knife violence in various regions.[30] On the basis of these findings, in our model, we assume that scissors, single-edged knives, and cleavers are the most commonly used weapons in crimes. Focusing on the features of these sharp tools improves the model's efficiency and generalizability while allowing for future expansion to include other weapon types. By analyzing the wound characteristics associated with these weapons, such as puncture wounds caused by scissors and single-edged knives or lacerations from cleavers, an effective forensic identification model is established.

As a cause and result, wound classification is as important as tool classification. Puncture wounds, typically deeper and wider, can be identified to be caused by single- or double-edged blades, whereas lacerations caused by cleavers are characterized by smooth surfaces, neat, everted edges, and greater length than depth. Understanding these patterns will enable law enforcement to identify suspects and predict crime trends, enhancing forensic accuracy and judicial fairness. Figure 1 illustrates the classification of scissors, single-edged knives, and cleavers, which form the foundation of our research.

2.2 Neural network model

Neural networks are a type of machine learning model, and deep neural networks are a more advanced form used in deep learning, which enables computers to handle highly complex and high-dimensional data. With large datasets and computational power, deep neural networks can outperform traditional machine learning models in tasks such as image recognition and speech

(a) (b) (c)

Fig. 1. (Color online) Tools: (a) scissor type, (b) single-edged type, and (c) chopping type.

processing. We adopted the LeNet-5 architecture,[31] a deep neural network proposed by Yann LeCun in 1998, originally designed for handwritten digit recognition, to construct an image recognition model to identify the tools by analyzing wound images. LeNet-5 was the first successful application of deep learning in this domain and remains relevant owing to its simplicity and efficiency. Its lightweight structure, with fewer parameters, makes it well suited for implementation on FPGA, where computational resources and memory are limited. This enables reduced computational load, rapid processing, and improved hardware performance. Despite being a classic model, LeNet-5 has undergone refinements, maintaining strong performance across diverse applications, making it feasible for this study.

The architecture of LeNet-5 (Fig. 2) consists of the following components. (1) The input layer handles 32 × 32 grayscale images, serving as the entry point for input data. (2) Convolution layers extract features by applying learnable filters to input images, followed by nonlinear activation functions such as Sigmoid or ReLU. (3) Pooling layers reduce the dimensionality of feature maps, decreasing computational requirements while improving feature extraction. Max pooling is commonly employed. (4) Fully connected layers flatten outputs from prior layers and apply linear transformations to generate the final classification results. (5) The output layer utilizes a softmax function for multiclass classification, providing the final predictions. LeNet-5 is a strong candidate for application to FPGA in forensic tool mark analysis because of its simplicity and effectiveness.

2.3 FPGA

Traditionally, powerful servers are necessary for computationally intensive tasks. However, the rise of edge computing necessitates deploying these models closer to data sources to reduce the latency and enhance data privacy. Deep learning in edge computing can reduce latency and bandwidth consumption, yet improve privacy and security, enabling real-time decision-making. This is especially important for applications such as autonomous driving, industrial inspection, and smart healthcare. In addition, through model compression and efficient hardware, computing and power consumption limitations can be overcome, allowing AI to run more efficiently on the terminals. Operating deep learning models on resource-constrained edge devices is essential, and FPGAs are an efficient solution. FPGAs are reprogrammable integrated circuits that enable hardware customization to meet specific application needs. Compared with central and graphics processing units, FPGA offers superior energy efficiency, computational performance, and flexibility, making them ideal for edge computing devices with power and space limitations. LeNet-5, a classic deep learning model for image recognition, is well suited for FPGA

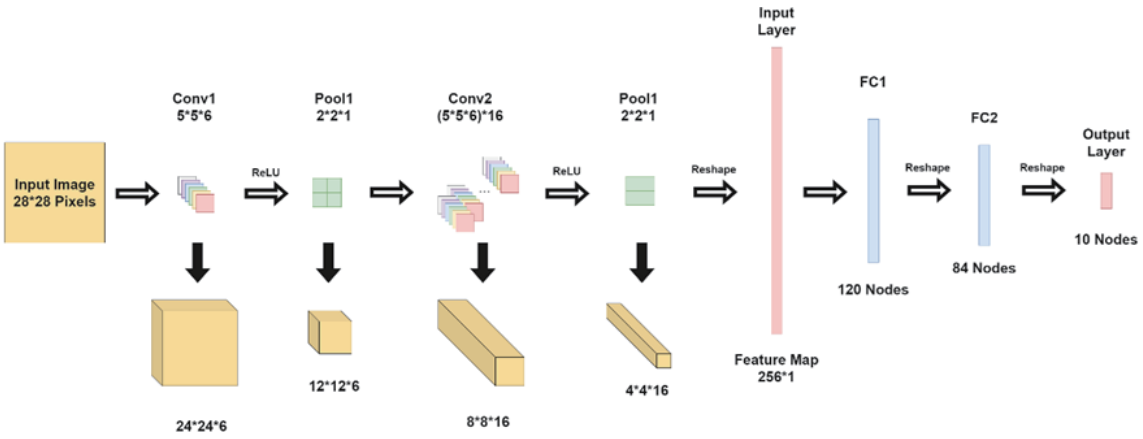


Fig. 2. (Color online) LeNet-5 architecture.

deployment but requires optimization to align with the FPGA architecture. Key challenges include adapting high-level model features to FPGA constraints and maximizing resource efficiency. To optimize recognition speed, LeNet-5 employs a hardware strategy where computationally intensive convolution operations are performed on the programmable logic (PL) side, leveraging parallel processing to boost speed. Fully connected layers, containing numerous parameters, are executed on the processing system (PS) side, conserving hardware resources. This task division capitalizes on FPGA capabilities, with PL handling parallelizable tasks and PS managing sequential processes, ensuring efficient image recognition.

For this project, we use the TUL PYNQ-Z2 development board paired with an OV5640 camera module. The PYNQ-Z2 board features a Xilinx Zynq system on a chip (SoC), combining a high-performance ARM Cortex-A9 processor with Xilinx 7-series FPGA logic. This enables the efficient handling of both high-level processing (via the ARM processor) and intensive PL operations (via the FPGA). The OV5640 camera module, with its 5-megapixel resolution and versatile functionalities, supports real-time image capture and processing, which are essential for neural network training and wound recognition tasks. Utilizing the PYNQ-Z2's FPGA logic enables the parallel processing of convolution operations in LeNet-5, making this setup ideal for real-time image processing with a balance of computational power and energy efficiency.

3. System Design

3.1 Overview

Figure 3 depicts a precise approach to developing a neural-network-based image PS, with an emphasis on iterative development, testing, and hardware integration, culminating in a robust design ready for deployment. (1) Initialization phase: This process begins with the neural network initialization, which lays the groundwork for the deep learning model. This is where the fundamental settings and hyperparameters of the neural network are defined. (2) Data preparation: The wound images are generated in the data end stage. These images are then sent

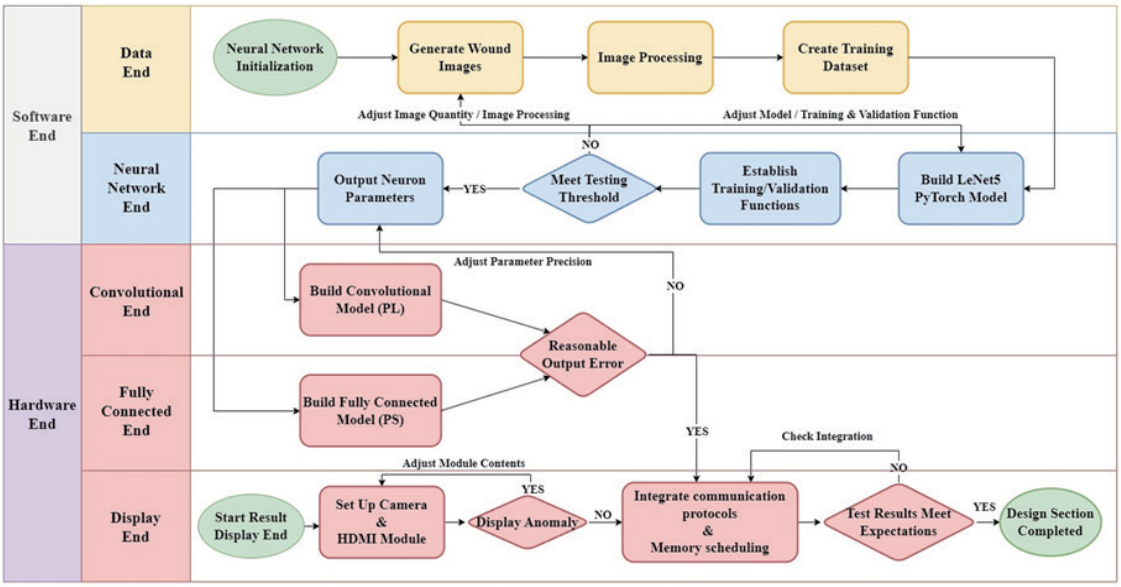


Fig. 3. (Color online) Wound identification system design framework.

through image processing, where they are likely enhanced, standardized, and transformed to be conducive to deep learning algorithms. (3) Model building and training: The processed images are then used to create the training dataset, which will serve as the input for the neural network training. The process then transitions to build the LeNet-5 PyTorch model, suggesting that a specific architecture (LeNet-5) is chosen for the CNN and implemented using the PyTorch framework. (4) Testing and refinement: The neural network end perhaps signifies the end of a phase, leading to iterative processes for model refinement. This is where the model goes through rigorous testing, checking if the output meets a certain testing threshold and if the output error is within reasonable limits. If not, it circles back to adjust the parameters, and this loop continues until the desired accuracy and performance are achieved. (5) Hardware integration: Simultaneously, the framework considers the hardware end of the system. The fully connected end and display end suggest that there are dedicated processes for managing the fully connected layers of the neural network and the display of results. A camera and a high-definition multimedia interface (HDMI) module are set up, hinting at the real-world application of the system where images are captured and results are displayed in real time. (6) Result handling and anomaly detection: The system is designed to display an anomaly when an issue is detected, indicating an advanced level of result interpretation. This suggests that the system not only classifies the images but also has a mechanism to flag and highlight unusual findings, which is crucial in medical imaging applications. (7) Communication and memory management: An essential part of the system is to integrate communication protocols and memory scheduling, ensuring that the system can communicate with other devices or networks and handle its memory resources efficiently. (8) Completion and validation: Finally, the process ends with the completion of the design section, marking the end of the system's design phase. This implies that

all system components are well integrated and the model meets the expectations as per the test results.

3.2 Datasets

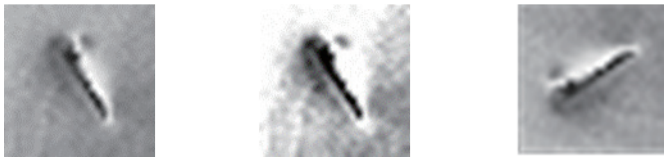
In light of the fragmented and unclassified nature of wound images, clay samples were collected to simulate the characteristics of real wounds. A dataset, established with extensive photography and precise categorization, is suitable for neural network training. The choice of clay was critical because of the variety of types available, each with different properties. To simulate real wounds as closely as possible, it was essential to select clay that closely resembles human skin. Following our experiments, resin clay was selected, which has several desirable properties: high toughness, allowing for repeated sampling; an oily composition, making it easy to preserve; a fiber tearing quality similar to skin; and moderate stickiness.

During sampling, we maintained the clay at a certain level of moisture to best mimic human skin. We established controlled variables and variable factors to collect a range of simulated wound features, as shown in Fig. 4. Controlled variables in our sampling included the moisture level of the clay, the smoothness of the clay surface, the angle and focus of the photography, and the wound location (centered). Variable factors included the force of the stabbing and withdrawal, the angle and depth of the stabbing and withdrawal, and the angle and intensity of lighting (to simulate different skin tones).

Before the images are fed into the neural network for training, they must first undergo enhancement and conversion into the appropriate input format. Initially, the unprocessed photos are randomly sorted and divided into a training dataset and a validation set in an 8:2 ratio. This division is done before image processing to prevent the validation set from containing any photos from the training dataset. Our dataset enhancement process includes adjusting the contrast, rotating the image by 90° four times, normalizing the image data, compressing it into a 28 × 28 × 1 grayscale image, and converting it into a tensor data structure, which includes dimensions, shape, and data type. Ultimately, the training dataset consists of 3272 enhanced photos of wounds corresponding to three types of blade, and the validation set contains 400 photos, as shown in Fig. 5.

(a) (b) (c)

Fig. 4. (Color online) Simulated wound: (a) yellow skin, scissors, (b) white skin, single-edged, and (c) black skin, chopping.



(a) (b) (c)

Fig. 5. (Color online) Simulated wound: (a) original, (b) contrast adjusted, and (c) rotated.

3.3 Neural network training

We chose the LeNet-5 architecture. The activation functions in its layers perform nonlinear transformations of the data, obtaining a sufficient combination of features. Since our study addresses a multiclass problem, we used the ReLU function.

$$f(x) = \max(0, x) \tag{1}$$

The purpose is to measure the discrepancy between the neural network's output and the actual output. During training, the neural network continually adjusts its weights and biases on the basis of the training data to minimize the loss function. For our multiclass problem, after empirical testing, we ultimately used the cross-entropy function.

$$C = -\frac{1}{n}\sum_{x} y \times \ln a + (1-y)\ln(1-a) \tag{2}$$

Here, n is the number of samples, x represents the types of label, y is a Boolean value, and a is the probability of successful prediction by this network. This function calculates the average probability of prediction failure for all wound photos at each training iteration.

The process begins with randomly initialized weights. Samples from the training dataset are input into the network, and the value of the loss function is calculated. Then, the backpropagation algorithm is used to calculate gradients, and the results are fed into an optimizer to update parameters. After empirical testing, the optimizer selected is the Adaptive Moment Estimation (Adam) algorithm,[32] which can automatically adjust the learning rate. In each training iteration, the performance of the neural network can be checked using a validation set. After training, adjustments can be made referring to the results from the validation, such as the composition of the wound training dataset, before starting a new round of training. After the training is complete, a test dataset is used to evaluate the performance of the neural network. Additionally, coordination with the hardware part is necessary for data comparison and adjustment, and will be detailed in the hardware section.

3.4 FPGA programming

Figure 6 depicts a hardware module schematic for the implementation of the LeNet-5 model on an FPGA hardware platform. The hardware is divided into two main sections: PL and PS.

The PL section includes the following: (1) Camera Driver: This receives image data from the camera sensor and preprocesses it, resizing to a 28 × 28 pixel resolution, which is the required input size for the LeNet-5 network. (2) LeNet-5: The convolution layers (conv) and pooling layers (pool) of the LeNet-5 model are implemented on the PL side. This exploits the FPGA's parallel processing capabilities to accelerate the most time-consuming operations. (3) video direct memory access (VDMA): This manages the data flow, ensuring efficient transfer of image data between different hardware modules. (4) HDMI Output: This outputs the processed image data to a display.

The PS section includes the following. (5) Fully connected layers (FC): These layers are implemented on the PS side and contain an enormous number of parameters and do not require parallel processing like the convolution layers. (6) Double data rate synchronous dynamic random access memory (DDR SDRAM or DDR): This is used for storing the neural network weights and other system data. DDR is a type of high-speed random access memory (RAM). (7) The arrows indicate the direction of data flow: from the camera to the convolutional and pooling layers of LeNet-5, then to the fully connected layers on the PS side, and finally, the output to the display through HDMI. Direct memory access (DMA) is used for efficient data transfer between PL and PS. The entire design is aimed at leveraging the high customizability and parallel processing power of FPGA to optimize the inference speed of the neural network.

From the LeNet-5 architecture shown in Fig. 2, we can identify several key layers and components: convolution layer 1, pooling layer, convolution layer 2, fully connected layer, as well as the camera and display systems. Next, we will introduce the design method of each component.

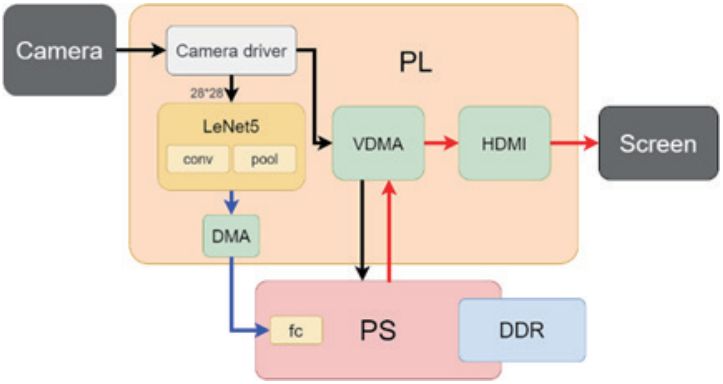
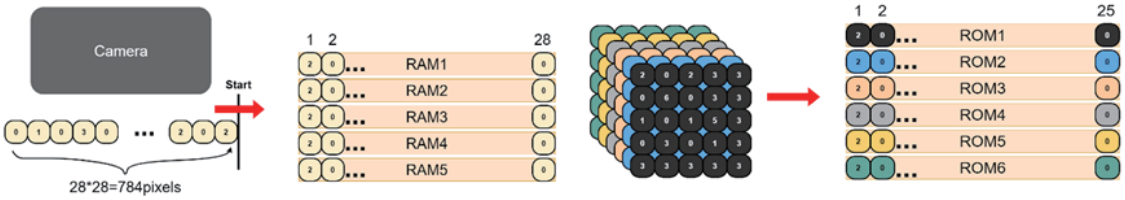


Fig. 6. (Color online) Hardware module connection diagram.

3.4.1 Convolution layer 1

Convolution layer 1 is designed to perform convolution operations with a 5 × 5 kernel size over six channels on the input image, which is 28 × 28 × 1 in dimension. The first step involves handling the input data. As illustrated in Fig. 7(a), we utilized five readable and writable RAMs, each storing 28 pixels of five rows of the input image. This setup allows for efficient access and manipulation of the image data required for the convolution operation. The next step involves the storage of kernel parameters. As shown in Fig. 7(b), we used six read-only memories (ROMs), each with 25 units, to store the parameters of the 6-channel kernel. These parameters are pretrained on the software side and are crucial for the convolution process. The ROMs provide quick and efficient access to these parameters during the convolution operations. In this configuration, the FPGA handles the computationally intensive task of convolving the input image with the kernels. The use of RAMs and ROMs ensures that both the image data and the kernel parameters are readily accessible, which is essential for the high-speed processing required in convolutional operations.

The single-channel convolution operation in convolution layer 1 is a critical part of the process. First, we extract five units from each of the five RAMs, obtaining a 5 × 5 matrix of data. This is demonstrated in Fig. 8(a). This matrix corresponds to a segment of the input image. The 5 × 5 data matrix is then convolved with the kernel parameters. This involves the element-wise multiplication of the data matrix with the kernel, followed by summing of all 25 values. The RAM data is then shifted to repeat this process for the next segment. One of the most challenging aspects is the shifting of data and the coordination of timing. As shown in Fig. 8(b), we designed an enable line to control data reading. After processing a row of 28 data points, the data in the RAMs are shifted: the second row moves to the first, the third to the second, and so on. The last row is then filled with the next line of input data. To speed up the process, we implemented six single-channel convolution units in hardware, as depicted in Fig. 8(c). This allows for parallel convolution operations across all six channels, significantly accelerating the process. Finally, to minimize the delay in data transmission between modules, we used registers at the output end of the convolution layer to implement bit concatenation, as illustrated in Fig. 8(d). This technique concatenates the 32-bit output from each of the six channels into a single 192-bit output data stream, which is then transmitted to the next layer. This architecture not only speeds up the convolution operation but also reduces the time taken for data transmission, enhancing the overall efficiency of the system. By parallelizing the



(a) (b)

Fig. 7. (Color online) Scrolling input buffer: (a) capture images into RAM and (b) parameters into ROM.

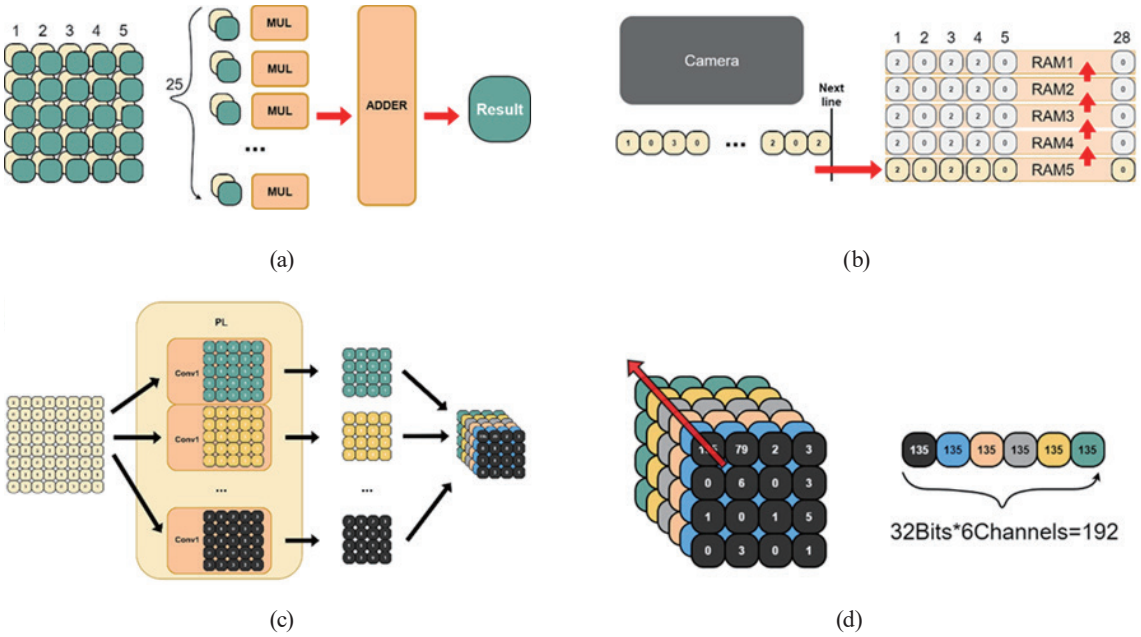
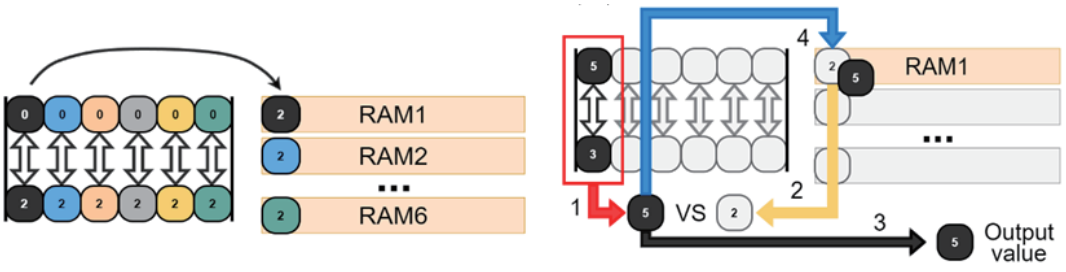


Fig. 8. (Color online) Parallel processing: (a) single-channel convolution operation, (b) data translation, (c) six-channel convolution operations processed in parallel, and (d) output data bit splicing.

convolution operations and optimizing data flow, we effectively managed the computational load and data handling, which are crucial for the real-time processing capabilities of our system.

3.4.2 Pooling layer

To implement a 2 × 2 max pooling operation in the pooling layer, we employed the concept of parallel processing, handling pooling operations on six channels simultaneously. The implementation of pooling also requires careful management of timing sequences. Given that the input data is a six-channel bit concatenation, to compare the maximum values in a 2 × 2 grid, we split the process into two parts. First, we compared two adjacent numbers in the same row, and then we compared the values between two rows. To compare two adjacent values in the same row, we temporarily stored the input signal and delayed it by one clock cycle. This approach ensures that two adjacent numbers are available simultaneously for pooling operations. The results were then stored in a readable and writable RAM. The comparison between the two rows involves aligning the row timings with an enable signal. We then retrieved the maximum value from the corresponding position of the previous row from RAM and compared it with the current large value in the pooling operation, as illustrated in Fig. 9(a), to obtain the maximum value of the 2 × 2 grid. In Fig. 9(b), we first compared two adjacent values, and then retrieved the maximum value from the corresponding position of the previous row from RAM. The maximum value obtained from this comparison becomes the desired output. This output was then stored in RAM, alongside the larger of the two adjacent values in the current row, optimizing space utilization. This method effectively accelerates the max pooling process while efficiently



(a) (b)

Fig. 9. (Color online) Max pooling operation: (a) two adjacent values in the same row and (b) pooling layer handling the 2 × 2 maximum.

handling data across multiple channels, contributing to the overall efficiency and speed of the neural network's operation in our system.

3.4.3 Convolution layer 2

Convolution layer 2 was designed to perform multichannel convolution operations. It processes a feature map of size 12 × 12 with six channels (output from convolution layer 1) using 16 sets of 5 × 5 size, six-channel convolution kernels. To compute the convolution for one output channel, it is necessary to multiply the values of all six input channels with the corresponding kernel parameters. After these multiplications, the results from each of the six channels (each providing 25 values) are summed to obtain the result of one convolution operation. To expedite the calculation, we designed five convolution processors that can handle six channels simultaneously. Each processor can perform a 5 × 1 × 6 column plane convolution operation. By using these five column plane convolution processors in parallel, we can obtain the result of a 5 × 5 × 6 convolution operation, as depicted in Fig. 10(a). To further save hardware space and optimize the process, we controlled the timing of the input feature map data. We delayed the operation by 16 cycles, allowing the reuse of the aforementioned convolution processors. This approach enables us to complete the convolution operations for all 16 layers of convolution kernels, as shown in Fig. 10(b). The implementation of convolution layer 2 is a critical part of the neural network, as it significantly increases the feature extraction capabilities. By optimizing the layer to handle multiple channels and utilizing parallel processing, we effectively managed the computational load. This design choice ensures that the convolution operations are not only accurate but also efficient, making the most of the FPGA's capabilities for real-time image processing tasks.

During the experimental phase, we encountered the challenge of limited hardware resources. We observed that the look-up table (LUT) usage reached 150% after synthesizing the circuit, exceeding the available resources of the FPGA and preventing system generation, as shown in Fig. 11. To address this, we analyzed the LUT usage of each layer and revised the original circuit design to achieve an optimal balance between board resources and computational efficiency. The final decision was to modify the structure of the neural network, specifically by reducing

(a) (b)

Fig. 10. (Color online) Multichannel convolution operation: (a) 5 × 5 × 6 convolution parallel processing and (b) 16-layer channel convolution operation.

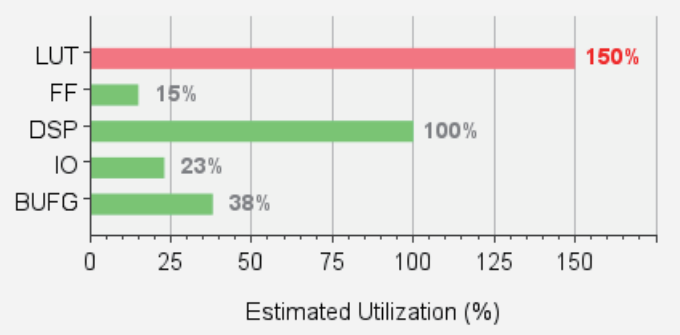


Fig. 11. (Color online) FPGA estimated utilization.

the size of the convolution kernels in convolution layer 2 from 5 × 5 to 3 × 3. This reduction was aimed at decreasing the scale of parallel processing in the convolution operations. However, the smaller kernel size meant that more time would be needed to compute the same size of data. To counteract this without significantly increasing the computation time, we reduced the number of output channels in convolution layer 2 from 16 to 3. As illustrated in Fig. 12, this adjustment successfully decreased hardware resource usage without adversely affecting performance. By optimizing the convolution layers in this manner, we were able to synthesize the circuit effectively, ensuring that the system could be implemented within the constraints of the FPGA's resources. This solution highlights the importance of flexible design and resource management in the hardware implementations of neural networks, particularly when working with limited hardware capabilities.

3.4.4 Fully connected layer

To minimize the use of hardware resources on the PL side, we designed the FC layers to be computed on the PS side. As illustrated in Fig. 13, to facilitate data transfer between the PL and PS sides, we used the AXI4-Stream communication protocol, known for its high-speed data streaming and capability of unlimited burst data transfer. Utilizing the dual-channel feature of Advanced eXtensible interface (AXI) DMA, we implemented memory map to stream (MM2S)

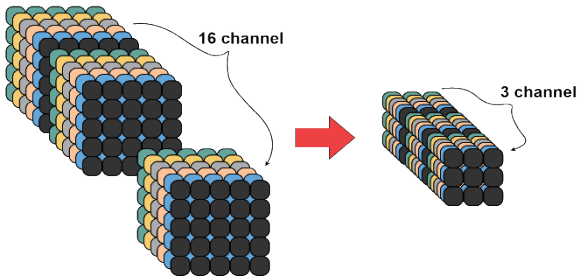


Fig. 12. (Color online) Improved design of convolution layer 2.

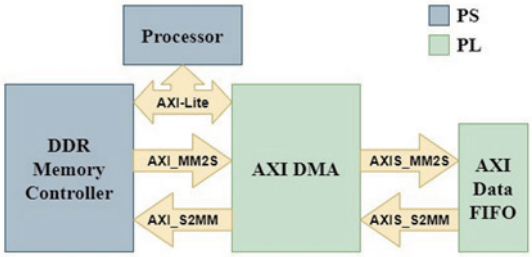


Fig. 13. (Color online) AXI DMA diagram.

to directly access addresses, allowing image data streams to be transferred from the PS side (master) to the PL side (slave) for executing the LeNet-5 model. Then, through the stream to memory map (S2MM) channel, the computed values from the PL side are written into memory, making them accessible to the PS side, thereby achieving PL–PS information interconnectivity. In the PL, the use of AXI DMA and AXI Data FIFO modules facilitates the storage of computed values from the LeNet-5 model into memory. The AXI Lite bus is used for configuring AXI DMA, while AXI_S2MM and AXI_MM2S are utilized for communication between the memory and the PS side. This approach efficiently manages the division of tasks between the PL and PS sides, with PL handling the computationally intensive parts and PS managing the data flow and final computations. This system architecture optimizes both the computational capabilities and resource utilization of the FPGA, making it ideal for the real-time processing demands of our portable wound recognition device.

3.4.5 Camera sensor and display system

We utilized the OV5640 camera module for image capture. After driving the camera with an open-source camera driver, the captured images are transmitted to DDR memory through VDMA and the video is transmitted to the AXI4-Stream IP. These images are then processed on the PS side, where specific image sections are extracted and provided to LeNet-5 for recognition. To display specific images and strings on the screen, we need to establish a display system on

the PL side of the FPGA and control the output strings from the PS side. First, we can use Xilinx's built-in IP, the Zynq7 PS, to set up the FPGA's clock and the connections between the PS and PL sides. Since we need to transfer data between the PS and PL sides, we can use the AXI protocol. One suitable IP for this purpose is the AXI VDMA, which is used for image transmission. Next, we used the AXI4-Stream to video out IP, coupled with a Video Timing Controller IP to control image resolution. This setup converts images from AXI signals to signals that the PL side can use. In this system, we fixed the image resolution at 640 × 480. Finally, as we need to transmit via HDMI, we connected the AXI4-Stream to video out signals to the DVI Transmitter IP. This setup enables us to display the output on an HDMI, effectively creating a comprehensive system for capturing, processing, and displaying images using the PYNQ-Z2 board and OV5640 camera module. This arrangement leverages the capabilities of FPGA for real-time image processing and display, which are essential for the efficient functioning of the wound recognition system.

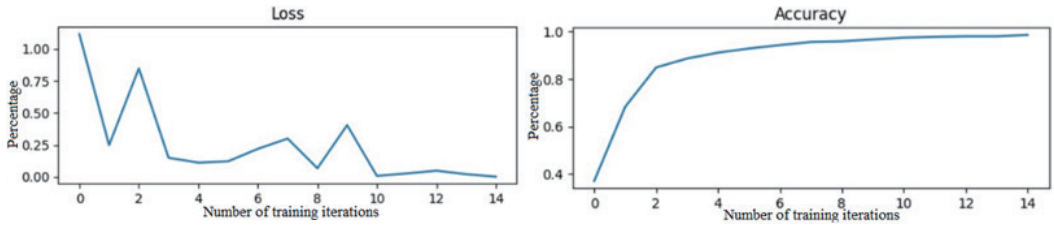
4. Results and Analysis

To evaluate the performance of our system, we first tested the accuracy of the system's software component. By adjusting the loss function to improve the accuracy of the dataset with Eq. (2), we assessed the system's recognition capability, as shown in Fig. 14(a). Our system, trained on a dataset of 3272 simulated wound images, analyzed 420 images in the test dataset, which included a mix of simulated and real wounds. We computed the accuracy rate by using the confusion matrix often used by deep neural networks with Eq. (3). As shown in Fig. 14(b), the system achieved an average recognition accuracy of 98.8%, indicating a high probability of correctly identifying the weapon from real wound images.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Our improved hardware design significantly optimized the resource utilization of the PYNQ-Z2 FPGA board, reducing LUT usage from 150% to 68%, as shown in Fig. 15. This figure outlines the estimated utilization of key FPGA resources as follows. (1) LUT: a substantial reduction in usage, critical for implementing logical functions. (2) LUTRAM: embedded memory within the FPGA, utilizing LUTs for storage. (3) Flip-flops (FF): basic storage elements used for creating memory registers and counters. (4) Digital signal processing slices (DSP): specialized blocks for high-speed arithmetic operations in signal processing tasks. (5) Input/output (I/O): blocks facilitating communication between the FPGA and external devices. (6) Global buffer (BUFG): buffers for routing high fan-out signals, such as clocks or resets.

Figure 15 demonstrates that resource utilization is appropriate within feasible limits (none exceeding 100%), confirming that our design efficiently fits the target hardware. This optimization enabled the successful deployment of the complex LeNet-5 model on the lightweight PYNQ-Z2 FPGA, leveraging hardware acceleration for improved performance.



(a) (b)

Fig. 14. (Color online) Change curve of LeNet-5. (a) Loss function change curve and (b) recognizing the accuracy of change curve.

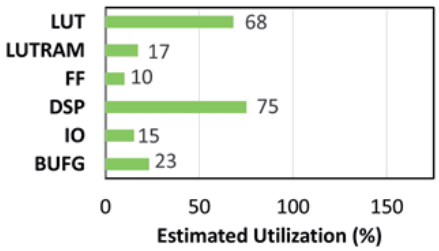


Fig. 15. (Color online) Optimized estimated utilization of various resources.

Table 1
Hardware latency.

Latency (cycles)		Latency (absolute)		Interval (cycles)	
Min	Max	Min	Max	Min	Max
163175	163175	1.632 ms	1.632 ms	163176	163176

Table 1 shows that the hardware can consistently complete an operation in 1.632 ms, and it is ready to start a new operation every 1.632 ms plus the time for one clock cycle, assuming the clock cycle time corresponds to the time difference between 163175 and 163176 cycles. At this speed, the system can perform approximately 613 operations per second (1000/1.632 ms per operation). Table 1 indicates that the hardware operation has a very stable performance with consistent latency and interval times. The ability to perform more than 600 operations per second is derived from the absolute latency. For many real-world applications, including interactive systems, video games, and most user interface operations, a latency under 10 ms is typically considered real time. For many applications, especially in user-facing systems, a response time of 1–2 ms is considered excellent and would be perceived as instantaneous by a human user. Hence, by general standards, a neural network with a 1.632 ms latency would meet the real-time standard.

5. Conclusions

We presented a portable recognition tool that accelerates the identification of tools that caused injuries in crime scene investigations by utilizing deep learning and hardware acceleration. The system integrates software training with hardware acceleration to accurately

recognize injury-inflicting tools on the basis of wound images. The software component employs the LeNet-5 model for wound recognition model training, utilizing a diverse and variable homemade dataset of simulated wounds and image processing techniques. The system achieved an impressive average recognition accuracy of 98.8% on the test dataset, demonstrating its efficacy in identifying wounds and the corresponding causative tools. In the hardware segment, the TUL PYNQ-Z2 development board combined with an OV5640 camera sensor facilitated efficient hardware acceleration. Through optimization of the LeNet-5 model's design and resource utilization, we successfully reduced hardware usage and achieved high-speed wound recognition capable of performing at least 600 operations per second, ensuring real-time and delay-free operation.

Nevertheless, we have identified areas for enhancement and encountered challenges. First, the LeNet-5 model's performance in accurately identifying a broad range of injury-inflicting tools necessitates further research to optimize and adjust its architecture. Second, owing to the limited number of real wound images available, our system primarily relies on a homemade dataset of simulated wounds for training. To enhance the system's universality and accuracy, expanding the dataset to include simulated wound images of different types of weapon and collecting more real wound images is imperative. Overall, the deep learning and hardware-accelerated wound recognition device developed in this study exhibits promising results and potential. The integrated and optimized system provides a reliable and efficient tool for the forensic field, that is capable of identifying the types of injury-inflicting tool from wound images and is expected to play a significant role in practical applications.

Acknowledgments

This work was supported by educational research project for young and middle-aged teachers from the Education Department of Fujian Province (grant number: JZ240083).

References

1 D. Q. Burd and R. S. Greene: J. Crim. L. & Criminology **39** (1948) 379.
2 D. Baldwin, J. Birkett, O. Facey, and G. Rabey: The Forensic Examination and Interpretation of Tool Marks (John Wiley & Sons, 2013).
3 A. Glossary: AFTE J. **30** (1998) 86.
4 B. Moran: AFTE J. **34** (2002) 227.
5 S. H. James and J. J. Nordby: Criminal Personality Profiling: In Forensic Science: An Introduction to Scientific and Investigative Techniques (CRC press, 2002) 1st ed., Chap. 20.
6 S. Hainsworth: Identification Marks–Knives and Other Implements: In Criminal Dismemberment (CRC Press, 2017) 1st ed., pp. 157–174.
7 P. Duez, T. Weller, M. Brubaker, R. E. Hockensmith, and R. Lilien: J. Forensic Sci. **63** (2018) 1069. https://doi.org/10.1111/1556-4029.13668
8 R. G. Nichols: J. Forensic Sci. **52** (2007) 586. https://doi.org/10.1111/j.1556-4029.2007.00422.x
9 E. Secula, Pml Study Supports Validity of Toolmark Identification in Forensics (2013).
10 H. C. Lee, T. Palmbach, and M. T. Miller: Henry Lee's Crime Scene Handbook (Academic Press, 2001) 1st ed.
11 K. Sanyavises and V. Vachirawongsakorn: Int. J. Legal Med. **138** (2023) 15. https://doi.org/10.1007/s00414-023-03102-1
12 K.-P. Shaw, J.-H. Chung, F.-C. Chung, B.-Y. Tseng, C.-H. Pan, K.-T. Yang, and C.-P. Yang: J. Forensic Sci. **56** (2011) 967. https://doi.org/10.1111/j.1556-4029.2011.01741.x
13 C. Humphrey, J. Kumaratilake, and M. Henneberg: J. Forensic Sci. **62** (2017) 1445. https://doi.org/10.1111/1556-4029.13467

14 M. A. M. González, J. Yravedra, D. González-Aguilera, J. F. Palomeque-González, and M. Domínguez-Rodrigo: J. Archaeol. Sci. **62** (2015) 128. https://doi.org/10.1016/j.jas.2015.08.006
15 M. Á. Maté-González, J. F. Palomeque-González, J. Yravedra, D. González-Aguilera, and M. Domínguez-Rodrigo: Archaeol. Anthropol. Sci. **10** (2018) 805. https://doi.org/10.1007/s12520-016-0401-5
16 P. McCardle and E. Stojanovski: J. Forensic Sci. **63** (2018) 1813. https://doi.org/10.1111/1556-4029.13754
17 M. Weber, S. Banaschak, and M. A. Rothschild: Int. J. Legal Med. **135** (2021) 313. https://doi.org/10.1007/s00414-020-02372-3
18 D. M. Anisuzzaman, C. B. Wang, B. Rostami, S. Gopalakrishnan, J. Niezgoda, and Z. Y. Yu: Adv. Wound Care **11** (2022) 687. https://doi.org/10.1089/wound.2021.0091
19 F. Li, C. Wang, X. Liu, Y. Peng, and S. Jin: Comput. Intell. Neurosci. **2018** (2018) 4149103. https://doi.org/10.1155/2018/4149103
20 C. Wang, D. Anisuzzaman, V. Williamson, M. K. Dhar, B. Rostami, J. Niezgoda, S. Gopalakrishnan, and Z. Yu: Sci. Rep. **10** (2020) 21897. https://doi.org/10.1038/s41598-020-78799-w
21 C. Jiao, K. Su, W. Xie, and Z. Ye: Burns & trauma **7** (2019). https://doi.org/10.1186/s41038-018-0137-9
22 P. Oura, A. Junno, and J. A. Junno: Int. J. Legal Med. **135** (2021) 2101. https://doi.org/10.1007/s00414-021-02566-3
23 F. K. Boymanov, A. M. Kushbakov, and F. F. Rashidov: Reports of Morphology **29** (2023) 32. https://doi.org/10.31393/morphology-journal-2023-29(2)-05
24 N. Zimmermann, T. Sieberth, and A. Dobay: Forensic Sci. Med. Pathol. **20** (2024) 443. https://doi.org/10.1007/s12024-023-00668-5
25 C. T. Wei, J. C. L. Lee, Y. G. Chen, W. H. Lin, H. T. Ting, and C. T. Pai: 2024 11th Int. Conf. Consumer Electronics-Taiwan, Icce-Taiwan **2024** (2024) 479. https://doi.org/10.1109/ICCE-Taiwan62264.2024.10674176
26 United Nations Office on Drugs Crime: Drugs and Crime, Global Study on Homicide 2019 (United Nations, 2019). https://doi.org/10.18356/9789210025713
27 United States. Federal Bureau of Investigation: Investigation, Uniform Crime Reporting Program Data: Supplementary Homicide Reports, United States, 2016 (Inter-university Consortium for Political and Social Research [distributor], 2018). https://doi.org/10.3886/ICPSR37064.v1
28 S. H. Kidd, N. S. Hughes, and J. H. Crichton: Med.Sci. Law **54** (2014) 167. https://doi.org/10.1177/0025802413496409
29 R. Jacques, S. Kogon, and M. Shkrum: Am. J. Forensic Med. Pathol. **35** (2014) 59. https://doi.org/10.1097/Paf.0000000000000078
30 J. Harms and M. Bush: J. Interpersonal Violence **37** (2022) NP17886. https://doi.org/10.1177/08862605211029620
31 Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Proc. IEEE **86** (1998) 2278. https://doi.org/10.1109/5.726791
32 D. P. Kingma and J. Ba: arXiv preprint arXiv:1412.6980 (2014). https://doi.org/10.48550/arXiv.1412.6980

About the Authors

Chun-Ta Wei received his B.S. degree in creative industries and digital film with a minor in information management (Information Technology Application Program) from Kainan University, Taoyuan, Taiwan, in 2016, and his M.S. degree in computer science and information engineering from Chung Cheng Institute of Technology, National Defense University (CCIT, NDU), Taoyuan, Taiwan, in 2019. From 2020 to 2023, he received a bonded scholarship from National Chung-Shan Institute of Science and Technology, Taoyuan, Taiwan, and he completed his Ph.D. degree in defense science (CSIE Program) from CCIT, NDU, in 2024. He is currently an associate professor with the School of Electronics and Communication Engineering, Quanzhou University of Information Engineering. His research interests include digital image processing, machine/deep learning, forensic science, remote sensing, and spectroscopy.

Jeff Cheng-Lung Lee holds a Ph.D. degree in biomedical engineering and environmental sciences from National Tsing Hua University, Taiwan, and B.S. and M.S. degrees in forensic science from Central Police University, Taiwan. From 2003 to 2005, he was a research scholar at the University of New Haven and a visiting scientist at the Connecticut Department of Public Safety, USA, supported by the National Science Council of Taiwan. He formerly served as Head of Forensic Science at the Hsinchu City Police Bureau. He also served as a joint associate professor at Qatar Police College from 2016 to 2019, helping to establish a state-of-the-art crime scene training center. Currently, he is a professor at the Department of Criminal Investigation, Taiwan Police College, and an adjunct professor at National Tsing Hua University and National Chung Cheng University. He has published over 100 academic papers and has delivered numerous international presentations. His research interests include forensic science, crime scene investigation, investigative technologies, and wrongful conviction studies.

Yu-Guang Chen received his B.S. and Ph.D. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2009 and 2016, respectively. He held the position of lecturer at Missouri University of Science and Technology, MO, USA, in 2015, and later served as a research fellow at the University of Notre Dame, IN, USA, in 2016. Following this, he worked as a project assistant on the ICT project at St. Kitts and Nevis with ICDF Taiwan from 2016 to 2017. From 2017 to 2019, he was part of the Department of Computer Science and Engineering at Yuan Ze University, Taoyuan, Taiwan. Currently, he is an associate professor in the Department of Electrical Engineering at National Central University, Taoyuan, Taiwan. Additionally, he has been an adjunct associate professor in the Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan. His research focuses on reliable circuit and system design, computing-in-memory (CIM) architecture design, AI for physical design, and hardware security.