Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.sciencedirect.com/journal/computer-methods-and-programs-in-biomedicine

Towards high-performance deep learning architecture and hardware accelerator design for robust analysis in diffuse correlation spectroscopy

Zhenya Zang, Quan Wang, Mingliang Pan, Yuanzhe Zhang, Xi Chen, Xingda Li, David Day Uei Li *

Department of Biomedical Engineering, University of Strathclyde, Glasgow, United Kingdom

A R T I C L E I N F O

Keywords:
Diffuse correlation spectroscope
Blood flow index
Deep neural networks
Deep-learning hardware accelerator

A B S T R A C T

This study proposes a compact deep learning (DL) architecture and a highly parallelized computing hardware platform to reconstruct the blood flow index (BFi) in diffuse correlation spectroscopy (DCS). We leveraged a rigorous analytical model to generate autocorrelation functions (ACFs) to train the DL network. We assessed the accuracy of the proposed DL using simulated and milk phantom data. Compared to convolutional neural networks (CNN), our lightweight DL architecture achieves 66.7% and 18.5% improvement in MSE for BFi and the coherence factor β, using synthetic data evaluation. The accuracy of rBFi over different algorithms was also investigated. We further simplified the DL computing primitives using subtraction for feature extraction, considering further hardware implementation. We extensively explored computing parallelism and fixed-point quantization within the DL architecture. With the DL model's compact size, we employed unrolling and pipelining optimizations for computation-intensive for-loops in the DL model while storing all learned parameters in on-chip BRAMs. We also achieved pixel-wise parallelism, enabling simultaneous, real-time processing of 10 and 15 autocorrelation functions on Zynq-7000 and Zynq-UltraScale+ field programmable gate array (FPGA), respectively. Unlike existing FPGA accelerators that produce BFi and the β from autocorrelation functions on standalone hardware, our approach is an encapsulated, end-to-end on-chip conversion process from intensity photon data to the temporal intensity ACF and subsequently reconstructing BFi and β. This hardware platform achieves an on-chip solution to replace post-processing and miniaturize modern DCS systems that use single-photon cameras. We also comprehensively compared the computational efficiency of our FPGA accelerator to CPU and GPU solutions.

1. Introduction

Blood flow is a critical bio-indicator to investigate the consumption and supplement of oxygen and glucose in the brain and muscles. Existing blood flow sensing techniques have been summarized in previous studies [1,2], among which diffuse correlation spectroscopy (DCS) is making a remarkable stride in monitoring cerebral [3,4] and muscular [5,6] blood flow variations in non-invasive, continuous manners. In essence, DCS measures how fast coherent light loses coherence because of the movement of red blood cells. DCS uses a near-infrared laser and a photon-sensitive detector (such as single-photon avalanche detectors (SPAD) [4,7–10], avalanche photodiodes [1,11], or photomultiplier tubes (PMT) [12]) placed near the laser with a source-detector distance (SDD). After the laser illuminates the tissue, the detector collects scattered photons. The detected intensity fluctuations are then fed into correlator hardware to compute the intensity auto-correlation function (ACF), i.e., $g_2(\tau)$. $g_2(\tau)$ is related to the electric field autocorrelation function, $g_1(\tau)$, defined by the Siegert relation [13]. The optical parameters (μ_a and μ_s'), blood flow index (BFi) and the coherence factor β can be retrieved by fitting measured $g_2(\tau)$ via analytical models. Fitting algorithms [14,9] are suitable for single-point detectors in terms of accuracy and speed. However, advanced CMOS SPAD arrays are emerging DCS detectors, as parallelized acquisition generates a higher signal-to-noise ratio (SNR) compared with single-point detectors. Massively parallelized acquisition increases data throughput and requires efficient algorithms and hardware architectures to interpret DCS data. Despite the efficacy of conventional fitting [14–17] and deep neural networks (DNNs) [18–21] for either single-point detectors or SPAD arrays, their speeds are not applicable for array detectors due to the high throughput. Furthermore, a highly integrated hardware

* Corresponding author.
E-mail address: david.li@strath.ac.uk (D.D.U. Li).

https://doi.org/10.1016/j.cmpb.2024.108471
Received 8 April 2024; Received in revised form 18 September 2024; Accepted 20 October 2024
Available online 28 October 2024

computing architecture is necessary for the miniaturization of DCS systems. We target the challenges mentioned and propose strategies in four aspects.

1) We propose an adder-based convolutional neural network (ACNN) tailored for hardware implementation, focusing on reconstructing BFi and β from intensity ACFs. The ACNN architecture utilizes multiplication-free convolutions to alleviate computational complexity, enabling higher parallelism and lower hardware utilization. Notably, addition operations save half of the latency compared with multiplication operations according to micro-instructions of various CPU operations [22].
2) To assess ACNN's performance in characterizing BFi and rBFi, we quantitatively compare speed and accuracy with a conventional CNN that performs the same network topology. This evaluation is based on a semi-infinite analytical model and *in-silico* Monte Carlo (MC) simulations of the single-layer model of milk with known diffuse parameters. Our results demonstrate that ACNN achieves accurate reconstructed BFi in the inference phase. We used diluted milk as the liquid phantom to evaluate ACNN, CNN, and nonlinear square fitting (NLSF).
3) We present a heterogeneous computing platform implemented on Zynq-7000 and UltraScale+ MPSoC field programmable gate arrays (FPGA). The ACNN accelerator is implemented on FPGAs. The ACNN accelerator fully explores the parallelism of the ACNN model, achieving different levels of parallelism ranging from nested for-loops unrolling to pixel-wise parallelism. Alongside FPGA fabric, single instruction multiple data (SIMD) on the embedded CPU are enabled to accelerate ACF generation.
4) To further miniaturize the accelerator, we employ various quantization strategies with different bit widths. Herein, we report the corresponding hardware utilization and speeds on cost-effective and high-performance FPGAs. This analysis examines the trade-off between reconstruction accuracy and hardware efficiency, facilitating choosing application-specific configurations aiding in selecting appropriate configurations for specific application needs.

The following sections are organized: Section 2 provides a comprehensive review of relevant literature, highlighting advancements in reconstruction algorithms and on-chip processing methods. Prospective enhancements in these areas are also presented. Section 3 illustrates the DCS theory for generating ACNN's training datasets. A canonical MC simulation was used as the reference to validate the consistency with the analytical model. Section 4 presents a detailed description of our ACNN and quantitatively compares it with conventional CNN. Section 5 delves into the details of the hardware implementation on the FPGA. Section 6 summarizes this study and indicates future work.

2. Prior work

This section reviews existing algorithms for reconstructing BFi and β from ACFs. Besides, state-of-the-art on-chip processing strategies are also examined. We illustrate potential improvements in the two aspects of our work targets.

2.1. Algorithms review

Existing BFi reconstruction algorithms can be categorized into two streams: optimization fitting algorithms and deep-learning approaches. Fitting measured ACF with analytical models to extract BFi and β is an ill-posed regression problem. The MATLAB NLSF (Mathwork, Inc., USA) functions, for example, *lsqnonlin(·)* using the interior-reflective Newton method [16], *fminsearch(·)* the Nelder-Mead simplex algorithm [20], and *optimset(·)* the Levenberg Marquardt method [23], have been adopted to reconstruct BFi and to assess errors resulting from uncertainties in optical properties and tissue thicknesses. Given that the fitting methods are constrained optimization problems, they involve numerous iterations and therefore time-consuming. A fast Nth-order model [24] was proposed using an Nth-order Taylor Polynomial to speed up the reconstruction of BFi compared to conventional fitting methods. This rigorous Nth-order approximation method is bespoke for continuous-wave (CW) illumination, making it arduous for other analytical models and experimental platforms, such as frequency-domain and time-domain techniques.

The time consumption and non-transferrable restrictions from prior knowledge can be remarkably mitigated using data driven DNNs. DNNs have catalysed improvements in enhancing reconstruction and accuracy. For example, intensity ACFs were first converted into 2-D images and fed into a 2-D CNN for BFi and β reconstruction [19], achieving a 23-fold speedup compared to a nonlinear fitting method. Another study proposed a long short-term memory (LSTM) for BFi reconstruction and relative blood flow analysis due to its superior capability for extracting features from sequences of data [20]. Similarly, gated recurrent units were embedded with a 1-D CNN to enhance information extraction, thereby retrieving relative BFi [21]. An LSTM variant [18] was proposed to denoise ACFs and extract BFi. Despite existing DNNs' high accuracy for DCS, redundant trainable parameters and complex topologies impede on-chip, real-time processing. The motivation to design a compact DNN is that most modern DCS systems [4,25] use SPAD arrays for data acquisition, where FPGAs are essential for controlling clocks and decoding data. We are inspired to embed the analysis on-chip to achieve end-to-end processing, taking the frame-based intensity as input and generating BFi and β.

2.2. Hardware processor review

Researchers have successfully implemented on-FPGA autocorrelators and on-FPGA BFi reconstruction. Buchholz *et al.* implemented a multi-channel autocorrelator for a 32 × 32 SPAD array, but the normalization of pixel-wise ACFs was not implemented [26]. To alleviate the computational burden, Rocca *et al.* [7] proposed an on-chip, scalable column-wise autocorrelator that can simultaneously compute up to 128 columns for a SPAD array with 192 × 64 enabled pixels. Another study [8] employed two FPGAs to accumulating detected photons and compute ACF for a 500 × 500 SPAD array, implementing element-wise matrix multiplications. But computationally expensive divisions and square operations for FPGAs are implemented on PCs.

Besides embedded autocorrelators, an iterative nonlinear curve-fitting algorithm was implemented on-FPGA using LabVIEW [17]. Although it achieves real-time BFi reconstruction, the high-level LabVIEW implementation exhibits a coarse control over allocating logic and data paths, leading to a redundant hardware overhead. Also, iterative operations significantly hinder on-FPGA data pipeline, thereby deteriorating the throughput. Overall, existing FPGA platforms merely integrate ACF generation and BFi reconstruction in a monolithic fashion. In this study, we take full advantage of reconfigurable heterogonous System-on-Chip (SoC) platforms embedding CPUs and programmable logic (PL), to encapsulate all computing pipelines on-chip, including ACF generation and DL-accelerators for BFi and β reconstruction. The proposed approach demonstrates superior efficiency compared to a common CPU and GPU. Extended from our previous work [27,28] for an FPGA-embedded DL processor for fluorescence lifetime imaging, we proposed a more concise, multiplication-free, CNN for estimating BFi and β.

In summary, from an algorithmic perspective, unlike existing DNNs for DCS, we proposed an end-to-end data-driven method that includes a synthetic data generation pipeline and a compact DL architecture design that eliminates the need for any multiplication. We also incorporated transfer learning functionality, allowing the model to be easily adapted to other experimental platforms with different optical and tissue properties by requiring only a few additional training epochs instead of complete re-training. On the hardware side, as existing implementations focus either on $g_2(\tau)$ generated from photon intensity data or on BFi reconstruction using iteration-based optimization algorithms, we propose an FPGA that integrates both a $g_2(\tau)$ generator and DNN-based BFi reconstruction.

3. Mathematical model of diffusion theory

This section introduces a generic DCS analytical model, adopted for synthetic dataset generation hereafter. Besides, we adopted MC simulations to simulate a semi-infinite phantom to validate the consistency between analytical models and MC simulations.

3.1. Analytical model

For typical DCS systems, intensity ACFs can be obtained using the photon intensity function of time

$$g_2(r,\tau) = \frac{\langle I(r,t)\cdot I(r,t+\tau)\rangle}{\langle I(r,t)\rangle^2}. \tag{1}$$

The electric field ACF $G_1(r,\tau) = \langle E(r,\ t)\cdot E^*(r,\ t+\tau)\rangle$ satisfied the correlation diffusion equation (CDE) in a scattering tissue [29]

$$\left(D\nabla^2 - v\mu_a - 1/3v\mu_s' k^2\alpha\langle\Delta r^2(\tau)\rangle\right)G_1(r,\tau) = -vS(r). \tag{2}$$

In Eq. (2), $D = v/3(\mu_a+\mu_s')$ denotes the photon diffusion coefficient, μ_a and μ_s' the absorption and reduced scattering coefficients, v the light speed in the medium, k is the wavenumber in the medium, and α is the ratio between dynamic scatters and all scatters. As the Brownian motion model has been widely adopted for specific biological tissues in DCS research [14,29,30], this work focuses only on Brownian motions as most previously reported studies advised [11,31]. With the Brownian motion model, $\langle\Delta r^2(\tau)\rangle = 6D_B\tau$ describes the mean-square displacement, and $S(r)$ is the CW isotropic light source.

The tissue can be modelled as a semi-infinite medium bounded by the tissue surface for biomedical tissues with a high-scattering property. And the solution [11,14] $G_1(r,\tau)$ in Eq. (1) can be represented as

$$G_1(\tau) = \frac{3\mu_s'}{4\pi}\left(\frac{\exp(-Rr_1)}{r_1} - \frac{\exp(-Rr_2)}{r_2}\right), \tag{3}$$

where $R^2 = 3\mu_s'\mu_a + \alpha\mu_s'^2k_0^2\langle\Delta r^2(\tau)\rangle$ and $r_1 = \sqrt{\rho^2+z_0^2}$ and $r_2 = \sqrt{\rho^2+(z_0+2z_b)^2}$ are shown in Fig. 1. Besides, $k_0 = (2\pi/\lambda)$, where λ is the wavelength in the medium. $R_n = n_i/n_o$, where n_i and n_o are the refractive indices inside and outside of the tissue. $z_0 = 1/\mu_s'$ means the distance between the virtual isotropic point source and the tissue surface. $z_b = 2(1+R_{eff})/3\mu_s'(1-R_{eff})$ means the distance between the extrapolated boundary and the tissue surface, where the effective reflection coefficient $R_{eff} = -1.440R_n^{-2} + 0.710R_n^{-1} + 0.668 +$

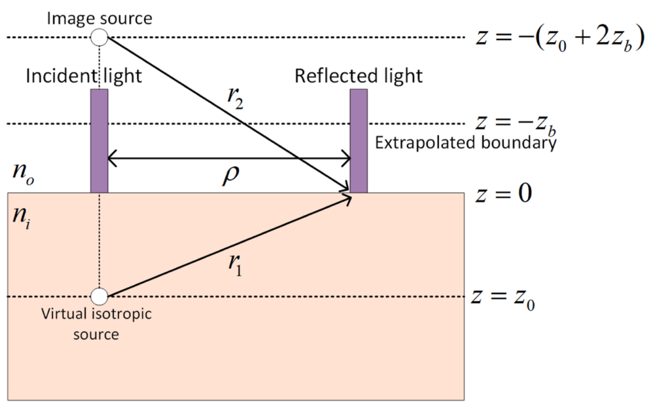

Fig. 1. The concept of spatial diffuse reflectometry in a semi-infinite geometry.

$0.0636R_n$ indicates the internal reflection coefficient between two media. As we mentioned, $\langle\Delta r^2(\tau)\rangle$ can be approximated to $6D_B\tau$ for diffusive motion [32]. Therefore, R^2 can be derived as $3\mu_s'\mu_a + \alpha\mu_s'^2k_0^2 6D_B\tau$. The details about spatial diffuse reflectometry have been explained in [33].

Recalling the Siegert relation [34] $g_2(\tau) = 1 + \beta|g_1(\tau)|^2$, $g_1(\tau) = G_1(\tau)/G_1(0)$. Here β is the coherence factor, mainly determined by the system setup. It has been proven that αD_B can represent BFi [3,35,36]. Also, for liquid phantoms, $\alpha \sim 1$ [35].

Alongside generating $g_2(\tau)$ with the analytical model, the noise model of $g_2(\tau)$ is also crucial for simulating noise in real media. Existing studies [11,14,37,38] state that the noise $\sigma(\tau)$ (standard deviation) of measured $(g_2(\tau)-1)$ can be approximated as

$$\begin{aligned}\sigma(\tau) \ &= \ \sqrt{\frac{T}{t}}\Bigg[\beta^2\frac{(1+e^{-2\Gamma T})(1+e^{-2\Gamma\tau}) + 2m(1-e^{-2\Gamma T})e^{-2\Gamma\tau}}{(1-e^{-2\Gamma T})} \\ &\quad + 2\langle n\rangle^{-1}\beta(1+e^{-2\Gamma\tau}) + \langle n\rangle^{-2}(1+\beta e^{-\Gamma\tau})\Bigg]^{\frac{1}{2}}.\end{aligned} \tag{4}$$

in a homogenous medium with an infinite geometry. Here, T is the time intervals between two adjacent τ, m is the bin index, t is the total averaging time. $\langle n\rangle = I\cdot T$ is the average number of photons, where I is the photon count rate. Γ is the decay rate of a single exponential function that approximates $g_1(\tau) = \exp(-\tau/\tau_c)$ [37]. Since τ_c is unknow, we can apply the fitting method *fminsearch(·)* in MATLAB to retrieve it after we obtain $g_2(\tau)$ from the Siegert relation. We investigated the sensitivity in terms of the averaging time and photon intensity. We set the other optical parameters as constant, namely, we fixed μ_a and μ_s' to 0.1 mm^{-1} and 2.0 mm^{-1}, SDD = 10 mm, BFi = 5×10^{-7} mm^2/s, $\beta = 0.5$, and λ to 785 nm. As shown in Fig. 2 (a), the noise is related to t, and $g_2(\tau)$ curves become noiser when t decreases. Similiarly, the amplitute of noise is negatively propotional to I.

3.2. Monte Carlo simulations

We chose the single-layer model of milk (μ_a= 0.0027 mm^{-1}, μ_s'=1.6 mm^{-1} at λ=785 nm, R_n=1.33 [39]) in the Monte Carlo eXtreme (MCX) photon propagation simulations [15,40]. We assumed that 10^7 photons were emitted from a light source. The radius of the detector is 1 mm. ρ was configured to be 10 mm. The volume size of the phantom is 60 mm^3. MCX employs *fminsearch(·)* as the non-linear fitting algorithm by default. The fitting method can obtain accurate results because the simulated curves are noise-free. The lag time is non-linearly sampled from [10^{-7}, 10^{-1}] s. Once we obtained parsed $g_1(\tau)$ using MCX studio, we used Siegert relation to calculate the corresponding $g_2(\tau)$, depicted by red lines in Fig. 3 (a) and Fig. 3 (b), respectively. Also, by using analytical model, we obtained the fitted $g_1(\tau)$ and $g_2(\tau)$ with reconstructed αD_B (equivalent to BFi) and β, shown by black lines in Fig. 3 (a) and (b). The Euclidian distances shown in Fig. 3 (c) and (d) indicates small errors. Therefore, the reconstructed BFi and β can be the reference for evaluating our algorithms. Also, the analytical model achieves nearly consistent $g_1(\tau)$ and $g_2(\tau)$ curves compared to MC simulations in MCX, meaning that we can quickly generate training datasets for our DNN model by constructing and automating the analytical model in MATLAB.

4. Deep learning architecture

Inspired by the previous 1-D CNN FPGA implementation [27,28,41], we proposed a similar but more compact DL network that does not involve multiplications apart from batch normalizations (BNs). The model features two unified adder-based convolutional (UAC) layers for primary feature extraction. It includes two branched pathways, each with three UAC layers, for reconstructing BFi and β. After feature extraction in the temporal dimension in the main branch, the tensor with a reduced temporal dimension is reshaped into a channel-wise tensor for processing by the subsequent branches. In each training iteration,

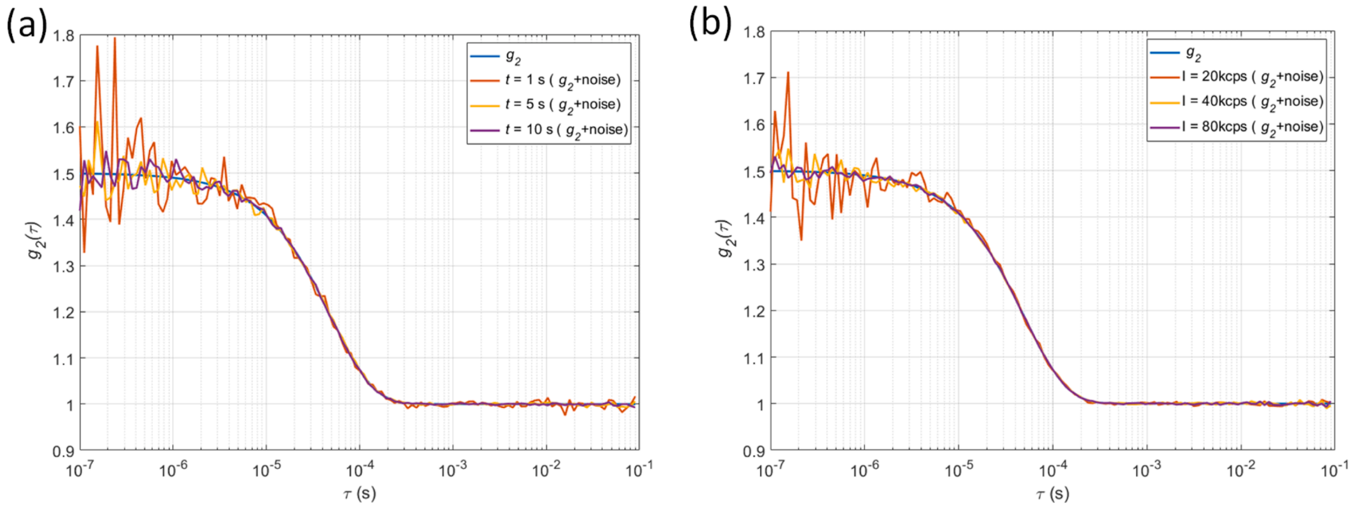

Fig. 2. $g_2(\tau)$ curves with fixed optical parameters but with (a) different averaging time (t = 1 s, 5 s, and 10 s) and (b) with different photon intensities (I = 20 kcps, 40 kcps, and 80 kcps).

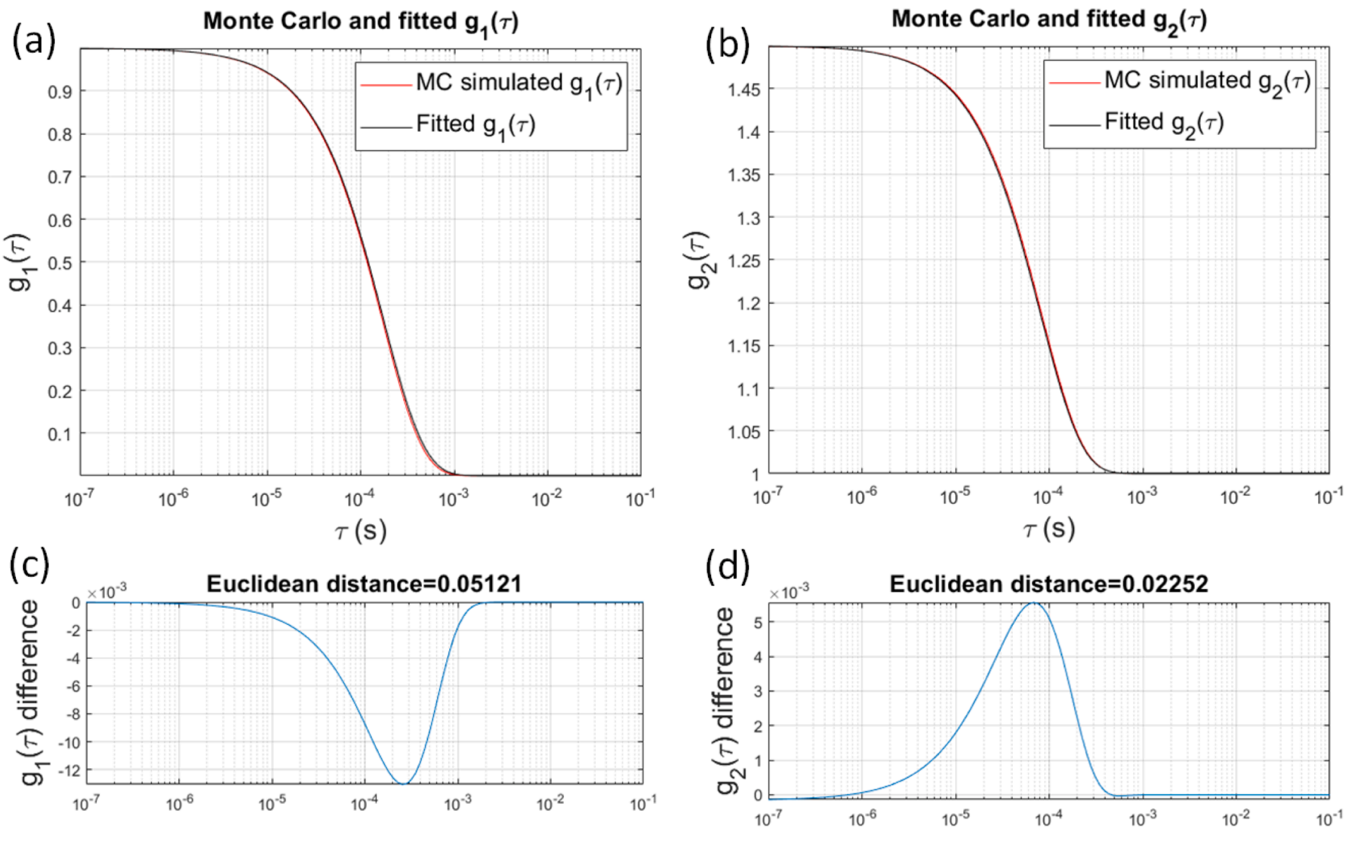

Fig. 3. Comparison between MC simulation and analytic model for $g_1(\tau)$ and $g_2(\tau)$ of milk. (a) and (b) generated and fitted curves using MCX and analytical models. (c) and (d) Euclidian distance between simulated and fitted curves.

estimated batches of BFi and β are sent to individual and same loss functions. The individual loss values are added to compute the gradient during the backpropagation. Fig. 4 illustrates the training and inference data pipelines and the dimensions of intermediate feature maps, whereas Table 1 details the layer configurations.

As the model is implemented in FPGA with constrained hardware resources and a high-speed demand, we used big receptive fields captured by each kernel and a small number of output channels to minimize the model size while maintaining accuracy. The configurations in Table 1 achieve a balanced trade-off. We do not use ResNet blocks [42] compared with the previous work [27] as its skip connections introduce data dependency that impedes data pipelining and for-loop unrolling on FPGAs. We have proven that our network can converge without a ResNet blocks. The quantitative results, including training time, model size (# parameters), # FLOPs, and averaged accuracy over 1,000 individual test curves, are presented in Table 2. The ACNN with a ResNet block has 1.0173 times more #FLOPs and 1.0547 times more parameters than the version without a ResNet block. Regarding accuracy, the ACNN with a ResNet block exhibits slightly higher accuracy in β but lower accuracy in BFi than the version without

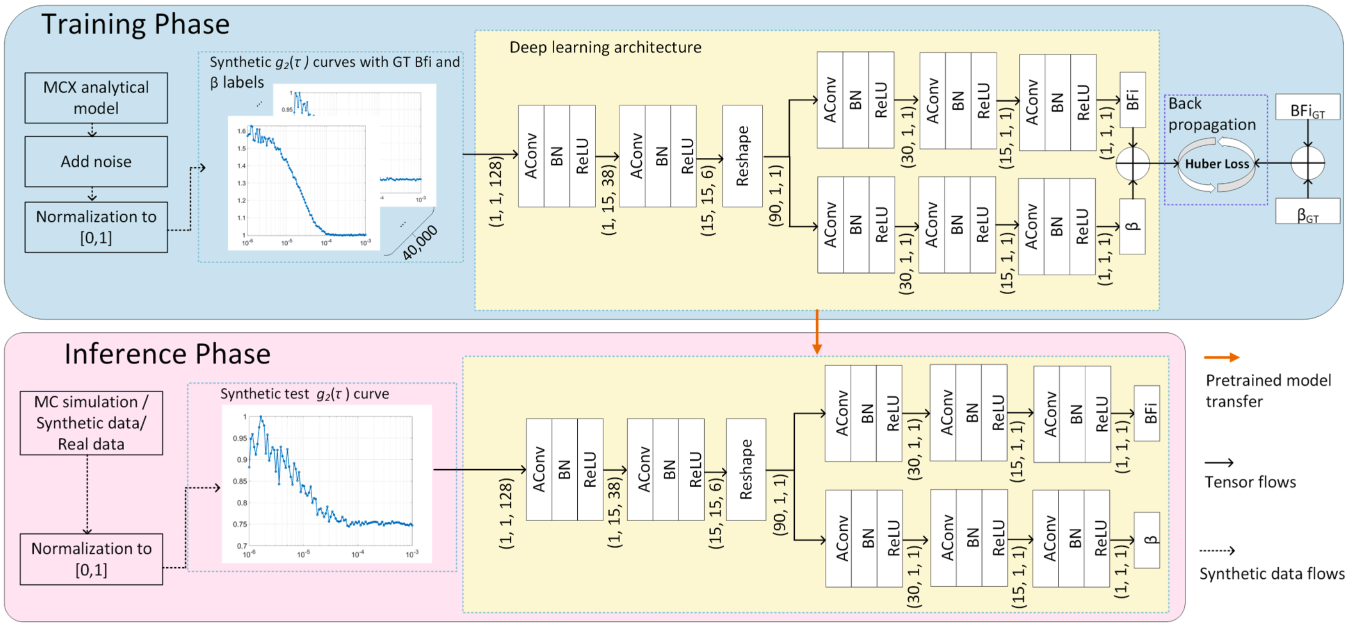

Fig. 4. ACNN architecture in training and inference phases.

Table 1
Configurations of each layer in the network.

Layers	Specifications
UAC1	Output Channel 15, Filters (1, 17), Stride (1, 3)
UAC2	Output Channel 15, Filters (1, 13), Stride (1, 5)
UAC 3_1	Output Channel 30, Filters (1, 1), Stride (1, 1)
UAC 3_2	Output Channel 15, Filters (1, 1), Stride (1, 1)
UAC 3_3	Output Channel 1, Filters (1, 1), Stride (1, 1)

Table 2
Quantitative comparisons of ACNN w/ and w/o a ResNet Block at main feature extraction branch.

Model	Training time	#FLOPs	#parameters	Accuracy from Test (MSE)	
				β	BFi (mm^2 /s)
ACNN w/ ResNet block	481.08 s	37.21 K ADDs	10.42 k	0.00015	0.024×10^{-7}
ACNN w/o ResNet block	370.93 s	36.58 K ADDs	9.88 k	0.00021	0.012×10^{-7}

a ResNet block. Although models with a ResNet block can converge faster during training, the eventual accuracy is similar. Besides, the version without a ResNet block is more efficient for hardware implementation.

As shown in Fig. 4. The $g_2(\tau)$ curves synthesized from the analytical model were applied to training and validation as the model is based on rigorous deductions and assumptions. On the other hand, using the curves generated from MC simulations for the test datasets provides a realistic evaluation of the model's performance in real-world scenarios.

We generated 40,000 $g_2(\tau)$ curves for training, using the analytical model (Eq. (3)) and the noise generator (Eq. (4)). The dataset is configured with $\beta \in (0, 1]$, BFi $\in [10^{-8}, 10^{-5}]$ mm^2 /s, $\rho = 20$ and 30 mm, and $\lambda = 785$ nm to emulate realistic experiments. We up-scaled BFi in the datasets by 10^5 to [0, 1], making its scale that same with β to achieve accurate training by using the same branch in Fig. 4. The noise rate varies from 20 kcps to 80 kcps. Averaging time is assigned to 1 s, 5 s, and 10 s. 10% (4,000) curves of the training dataset were used for validation during training. The optimizer is RMSprop. The Huber loss

$$L_\delta(a) = \begin{cases} \dfrac{1}{2a^2}\ if\ |a| \le \delta \\ \delta\left(|a| - \dfrac{1}{2\delta}\right) if\ |a| > \delta \end{cases}, \quad (5)$$

was adopted as the loss function, where a is the residual of predicted and GT values. The lag time is nonlinearly assigned between $[10^{-6}, 10^{-3}]$ s, divided by 100 data points, aligning with our data acquisition system. The learning rate is 0.005 with, decreasing by 0.5 factor every 30 epochs. It took 19 min for the NVIDIA RTX A1000 GPU to train the model. 35 patience epochs were used to avoid over-fitting. The total number of training epochs is 300. All $g_2(\tau)$ datasets included the noise model and were normalized to emulate realistic $g_2(\tau)$. Fig. 5 shows training and validation loss curves, where ACNN exhibits comparable convergence versus a CNN model with the same architecture but slightly different training strategy. ACNN and CNN use different feature extraction methods, thereby differentiating the backpropagation. The initial learning rate (0.005) of ACNN is not applicable as we noticed that the training terminates within only 30 epochs due to the high learning rate. Therefore, we use 0.001 as the initial learning rate for CNN. The number of parameters and floating-point operations per second (FLOPs) for each layer was summarized in Table 3. The compact model size and low FLOPs pave the way for high-parallel hardware implementation afterwards.

We also investigated the interpretability of our ACNN model using saliency maps [43], which are broadly adopted to visualize the gradient of the loss function for input pixels in image classification. We randomly selected four $g_2(\tau)$ curves from our test datasets to visualize the gradients most influencing the model's output. We normailzed the saliency maps in [0, 1] for better visualization. The red curves in Fig. 6 represent the gradients of the output for the input $g_2(\tau)$, indicating how sensitive the output is to the input variation. Since changes in BFi result in horizontal shifts on $g_2(\tau)$, the position of decay in each curve is a crucial region. Therefore, as indicated in Fig. 6, our ACNN model can identify the critical regions for extracting significant features from the $g_2(\tau)$ curves.

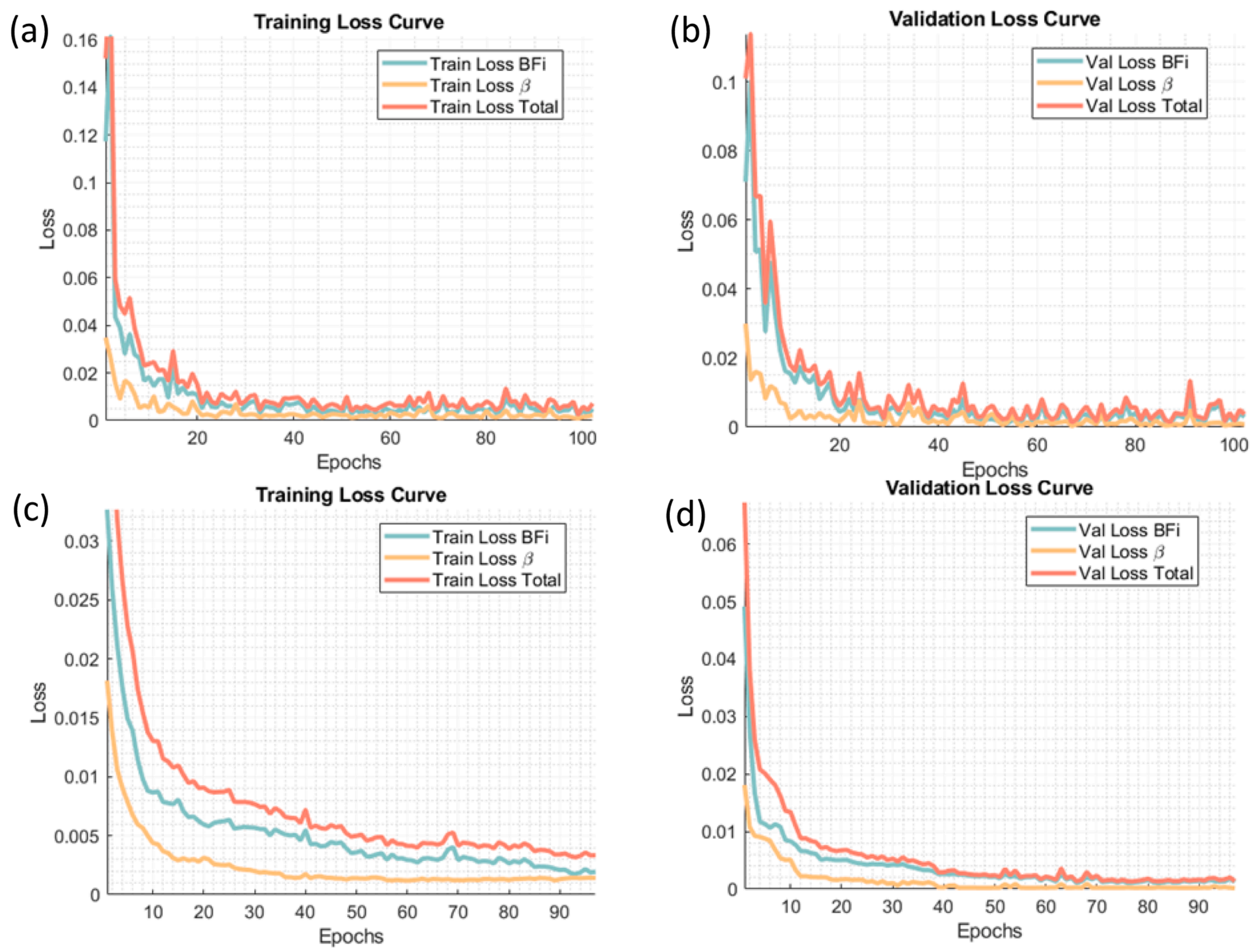

Fig. 5. Training and validation loss curves of ACNN and CNN. (a) and (b), ACNN training and validation loss curves in 109 epochs. (c) and (d), CNN training and validation loss curves in 96 epochs.

Table 3
Detailed computational information of each layer.

	UAC1	BN1	UAC2	BN2	UAC3_1	BN3_1	UAC3_2	BN3_2	UAC3_3	BN3_3	Total
#parameters	270	30	2,940	30	2,730×2	60×2	465×2	30×2	16×2	2×2	9,880
#FLOPs	10,260	1,140	17,640	180	2,730×2	60×2	465×2	30×2	16×2	2×2	35,830

Note: Some parameters and FLOPs multiply two because of the branched structure.

5. Accuracy evaluation

This section evaluates the accuracy and robustness of reconstructed BFi and rBFi from the ACNN model in different noise levels, comparing them with CNN and NLSF. Sections use both synthetic and real phantom datasets.

5.1. Evaluation on synthetic datasets

As our ACNN uses the l_1 norm to measure cross-correlation between weights and feature maps, we should ensure that ACNN's weights follow Laplace distributions according to [44]. As shown in Fig. 7(a) and (b), distribution weights from CNN and our ACNN exhibit Gaussian and Laplace distributions, in good agreement with the AdderNet theory [44]. To evaluate estimated BFi and β, in Fig. 8, we used R-squared and the mean square error (MSE) to evaluate the fitting goodness, in comparison with the ground truth (GT) BFi and β. Both ACNN and the conventional CNN offer nearly perfect R-squared results, However, for β, a few outliners from CNN (at small β) are shown. As for BFi's reconstruction, ACNN also attains better R-squared and MSE. We observed that increasing the BFi leads to higher reconstruction errors, as shown in Fig. 8. Large values in the saliency maps indicate which parts of the data significant impact the model's predictions or errors. To investigate this, we used saliency maps to create Fig. 9. Fig. 9(a) shows the saliency maps with increasing BFi with β being constant to visualize the changes. Fig. 9 (b)–(f) present the saliency maps for different BFi values. Our findings reveal that as BFi increases, the saliency map values also increase, leading to a more significant impact on the model's predictions. Intuitively, $g_2(\tau)$ with a bigger BFi shown in Fig. 9(f) exhibits fewer features (regarding the amplitude and shape) than other $g_2(\tau)$ curves shown in Fig. 9(b–e). However, the saliency map in Fig. 9(f) is higher than others, meaning that the model focuses on the less informative region, amplifying noise, and distorting the accuracy. We noticed the heteroscedasticity of BFi reconstruction also occurs in [19] (Fig. 4(c) in that

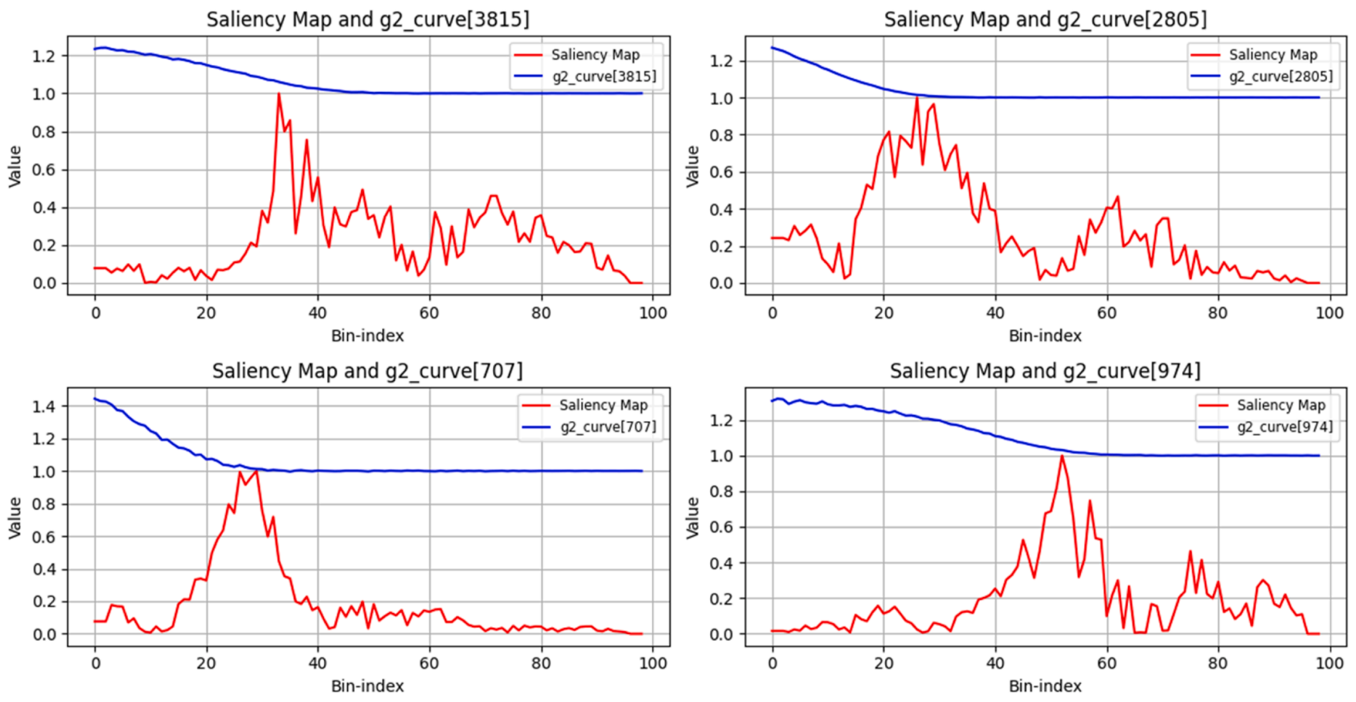

Fig. 6. Four randomly sampled $g_2(\tau)$ and corresponding computed saliency maps from test datasets.

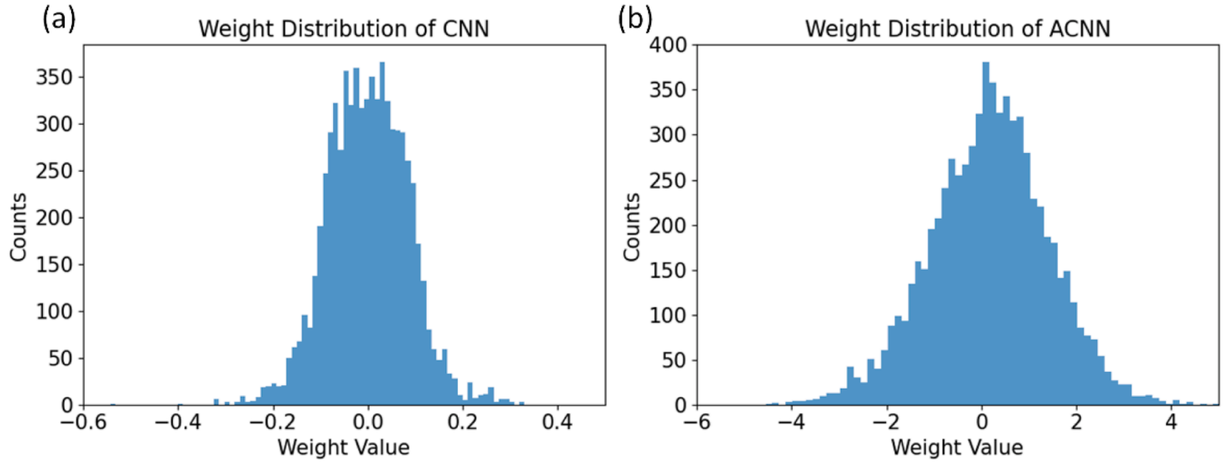

Fig. 7. Histograms depict the weights distribution of CNN and ACNN. (a) and (b) represent Gaussian and Laplace distributions of CNN's and ACNN's weights.

paper) when BFi increases. Therefore, we managed to use saliency maps to explain the phenomenon and demonstrate the interpretability of the model.

We also assessed the impact of the photon rate in the noise model on the reconstruction accuracy. We compared our ACNN with CNN at various photon intensity levels, ranging from 1,000 to 35,000 counts per second (cps). This range was divided into 34 groups, each containing 100 noisy ACF curves. The GT β and BFi values are 0.5 and 0.5 $\times 10^{-5}$ mm^2/s, respectively, and t (the averaging time) = 5 s. As shown in Fig. 10(a), the accuracy of β (ACNN and CNN) is compromised when the photon rate is low, leading to inaccuracy in the mean (*M*) and the standard deviation (*Std*). However, as the photon intensity increases, both architectures provide more accurate reconstructions. ACNN tends to offer better accuracy for most photon intensities, whereas CNN exhibits slightly better accuracy (*M*) and precision (*Std*) when photon counts are lower than 9,000. Regarding BFi, ACNN outperforms CNN in terms of accuracy and achieves similar precision. Both architectures produce accurate BFi at high photon rates, whereas CNN is more robust for low photon rates (before 5,000 cps).

Since rBFi is commonly used in physiological and clinical applications, we leveraged simulated datasets to evaluate our model's applicability in estimating rBFi instead of only absolute BFi using our model. Like absolute BFi evaluation, CNN and NLSF were adopted for comparison with ACNN. rBFi was computed from baseline BFi$_0$ and reconstructed BFi over time i.e. $rBFi = BFi/BFi_0$. In line with our previously proven rBFi evaluation [45], we assigned $BFi(w) = [1 + 0.05 \times (w - 1)] \times 10^{-6}$ mm^2/s, where w=1, 2, …, 20. BFi$_0$ was assigned when w=1. ρ was fixed as 20 mm. We selected four noise levels from Fig. 10, i.e., 2, 000, 6,000, 12,000, and 25,000 cps to investigate accuracy. The performance of the three algorithms is shown in Fig. 11. Black dots connected by red lines indicate the GT rBFi. The x-ticks represent increasing BFi over time. For lower photon intensities, shown in Fig. 11(a) and (b), ACNN and CNN are more robust than NLSF, as their reconstructed BFi distributions are closer to the GT values across different BFis. In contrast, for higher photon intensities, shown in Fig. 11(c) and (d), all three algorithms display similarly accurate distributions.

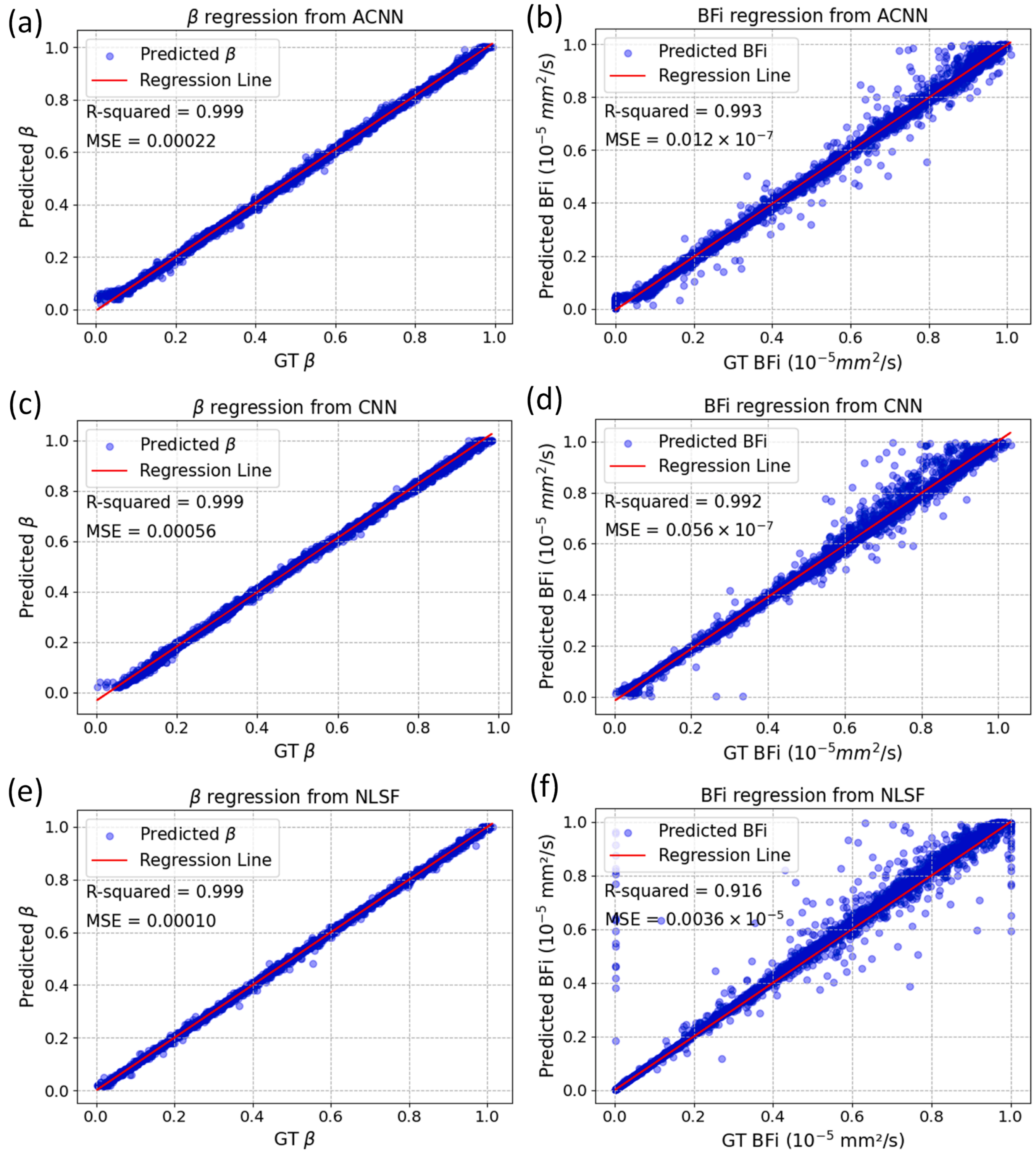

Fig. 8. R-square and MSE between ACNN and CNN. (a) and (b) R-squared of BFi and β from ACNN. (c) and (d) R-square of BFi and β from CNN.

5.2. Evaluation on real liquid phantom

As milk has similar scattering properties as living tissue [39], we used a milk solution with a water-to-milk dilution ratio of 2:1 as the phantom to investigate the performance of three algorithms. $g_2(\tau)$ curves were computed from a hardware correlator from raw photon intensity data following Eq. (1). We used a 785 nm CW laser (CrystalLaser, DL785-120-S), directed into the milk phantom through a multi-mode fiber. The scattered light from the phantom was captured through a single-mode fiber with $\rho = 20$ mm. Intensity data was detected by an APD (Hamamatsu, C13366-1350GD) and processed by a commercial correlator board to generate $g_2(\tau)$. The specification of the experiment platform is summarized in Table 4. Two groups of datasets (20 curves/group) were measured with $T = 1$ s and 10 s, under the same environment and the dilution ratio. Given the GT αD_B of the milk is unknown, the results fitted from the datasets ($T = 10$ s) were employed as a reference. We proved that the NLSF algorithm could achieve high accuracy for BFi and rBFi at a high SNR, as shown in Figs. 10 and 11. The

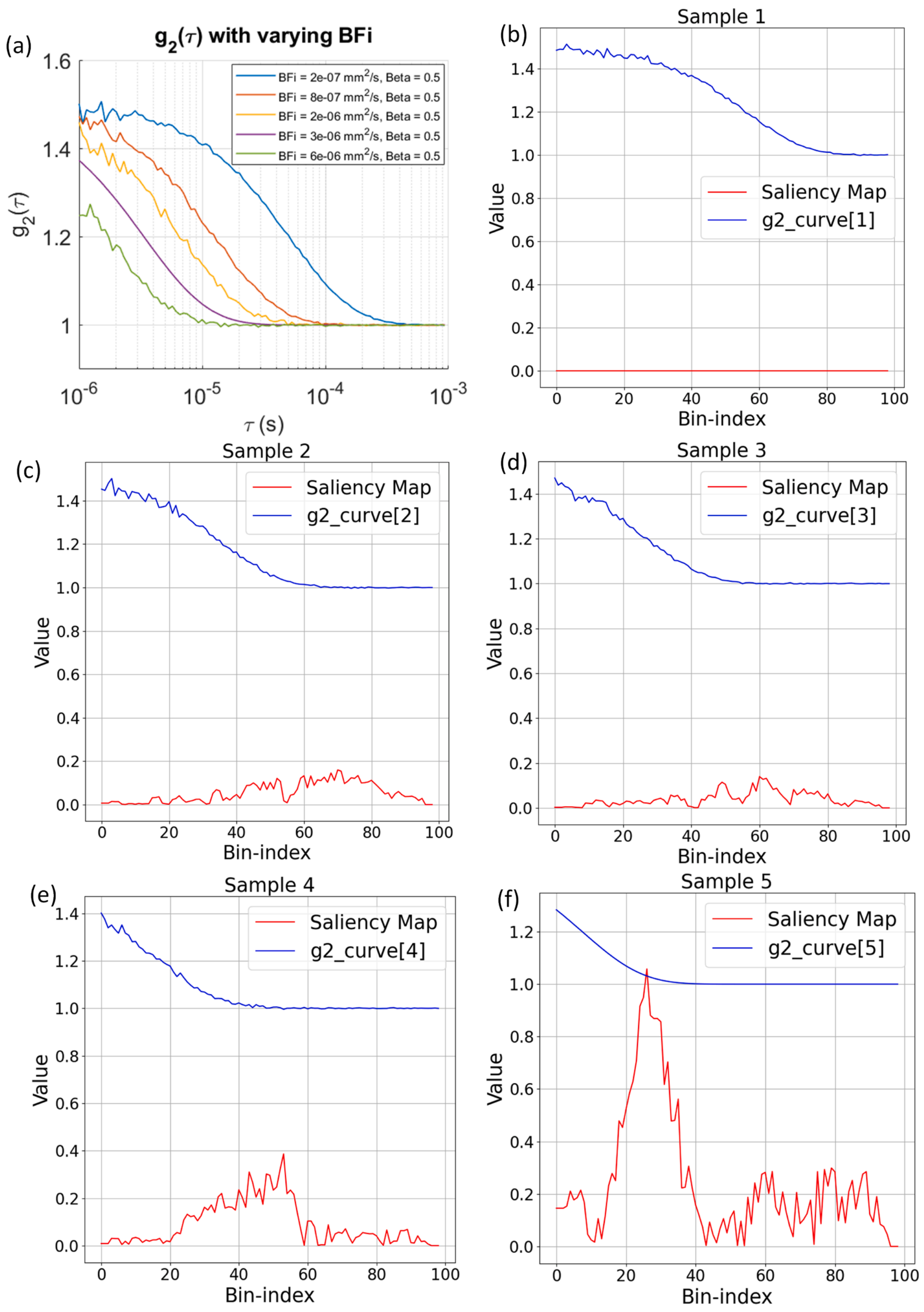

Fig. 9. (a) $g_2(\tau)$ curves with five increasing BFi (2×10^{-7}, 8×10^{-7}, 2×10^{-6}, 3×10^{-6}, and 6×10^{-6} mm^2/s) and constant β =0.5. (b)-(f), indivudual $g_2(\tau)$ in (a) with increasing BFi and corepcsonding seliance maps.

pseudo reference of $\alpha D_B = 5.19\times10^{-7}$ mm^2/s and $\beta = 0.419$ are reconstructed from the ensemble $g_2(\tau)$ using NLSF. The measured $g_2(\tau)$ is shown in Fig. 12, the curves with $T = 1$ s exhibited a higher standard deviation. The datasets with $T = 1$ s were used for evaluating the algorithms. Referring to the study [46], we determined our μ_a and μ_s' are 0.0027 and 1 mm^{-1}, respectively, according to our dilution ratio.

We developed a transfer learning mechanism in the training scripts for both ACNN and CNN models, enabling them to be transferable and adjustable for different optical or tissue parameters. In our case, we re-generated 40,000 curves in the training datasets, using μ_a and μ_s' are 0.0027 and 1 mm^{-1}. ACNN and CNN models required 60 epochs (5 min) and 92 epochs (8 min), respectively, to complete the training based on a previously saved pre-trained model with the old experimental parameters.

Fig. 13 presents the error distributions of reconstructed β and αD_B when $\rho = 10$, 20, and 30 mm. ACNN and CNN were re-trained using

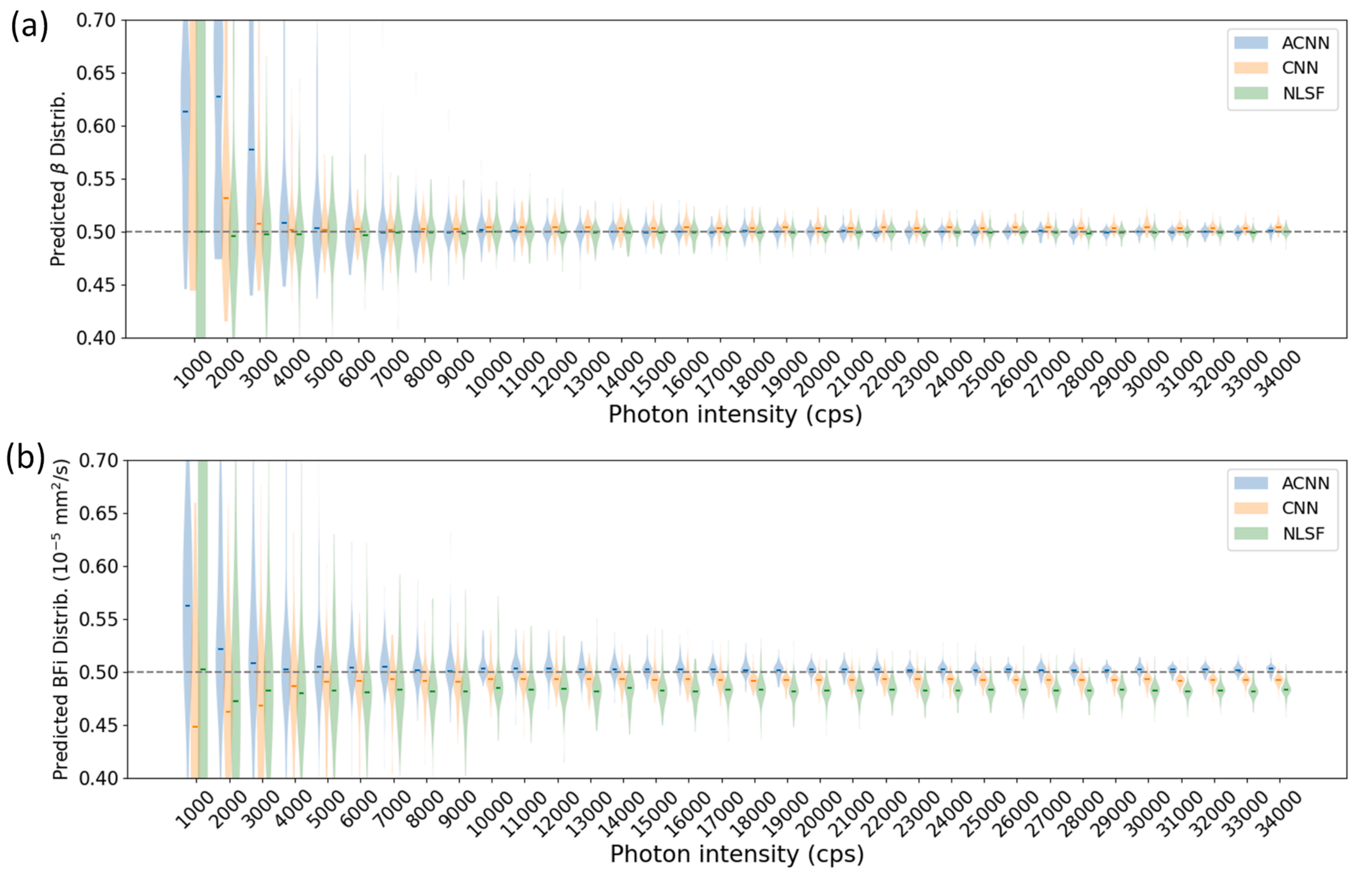

Fig. 10. Accuracy evaluation of ACNN and CNN under different levels of photon rate in the noise model. (a) and (b) reconstructed β and BFi.

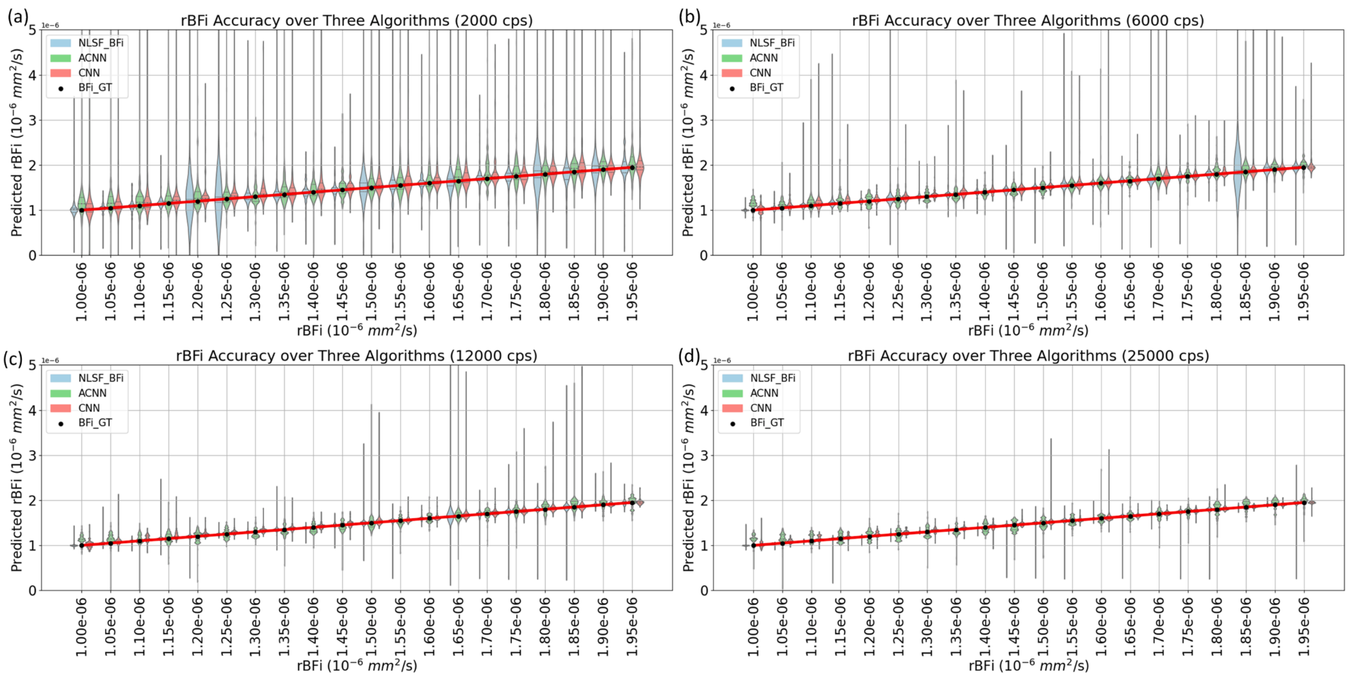

Fig. 11. rBFi evaluation of ACNN, CNN, and NLSF in (a) 2,000, (b) 6,000, (c) 12,000, and (d) 25,000 cps.

transfer learning using the corresponding datasets defined by ρ, consuming around 4 min. When $\rho = 20$ mm, NLSF presents the most significant % error ($E_{\alpha D_B}$) in αD_B. Although ACNN achieves the smallest $E_{\alpha D_B}$, it shows a slightly higher E_β. Although CNN is more accurate than ACNN regarding β, there are more outliers than ACNN and NLSF in αD_B estimation. Three algorithms show similar error distributions when $\rho =$ 10 and 30 mm, where CNN presents a higher E_β than ACNN and NLSF; ACNN and CNN show similar $E_{\alpha D_B}$. As ρ increases, the standard deviation

Table 4
Specification of the experiment for phantom measurement.

Parameters	Property
Detector fibre	Single mode
Detector	Hamamatsu, C1366-1350GD
Laser module	CrystalLaser, DL785-120-S, 785 nm long coherence (>5m)
ρ	10mm, 20 mm, 30mm
dilution ratio	Water: milk = 2:1 (milk: 1.7% Fat, 3.5% Protein)
μ_a	0.0027 mm^{-1}
μ_s'	1 mm^{-1}
Correlation scheme	Multi-tau

from each algorithm also increases. $E_{\alpha D_B}$ and E_β are computed using $(\hat{y} - y_{ave}/y_{ave})$%, where $\hat{y}$ and y_{ave} are the predicted values and pseudo reference of β and αD_B, respectively.

6. Accelerator architecture

The ACNN accelerator was implemented on programmable logic (PL) on cost-optimized Zynq-7000 SoC and high-end Zynq-UltraScale+ MPSoC using Vivado high-level-synthesis (HLS) 2018.2. Alongside the accelerator, the computation of ACF was implemented on the ARM-based processing system (PS). The following reasons are for producing the ACF module on PS: firstly, implementing Eq. (1) on the FPGA requires taking thousands of frames and conducting element-wise vector multiplications and divisions; it is challenging due to limited DSP slides and on-chip memory. The onboard double-data-rate synchronous DRAM (DDR) with hundreds of megabytes can efficiently accommodate raw data and bridge the data transfer between the ACF module on PS and BFi reconstruction on PL. Secondly, in theory, the multiplication operations in the numerator are followed by an averaging operation that can be implemented as adder-trees and end up with a subsequent divider. Although the overhead of parallelizing the adder-tree for averaging is negligible, vector multiplications and divisions are computationally expensive in FPGAs. To address this concern, we leverage Neon and

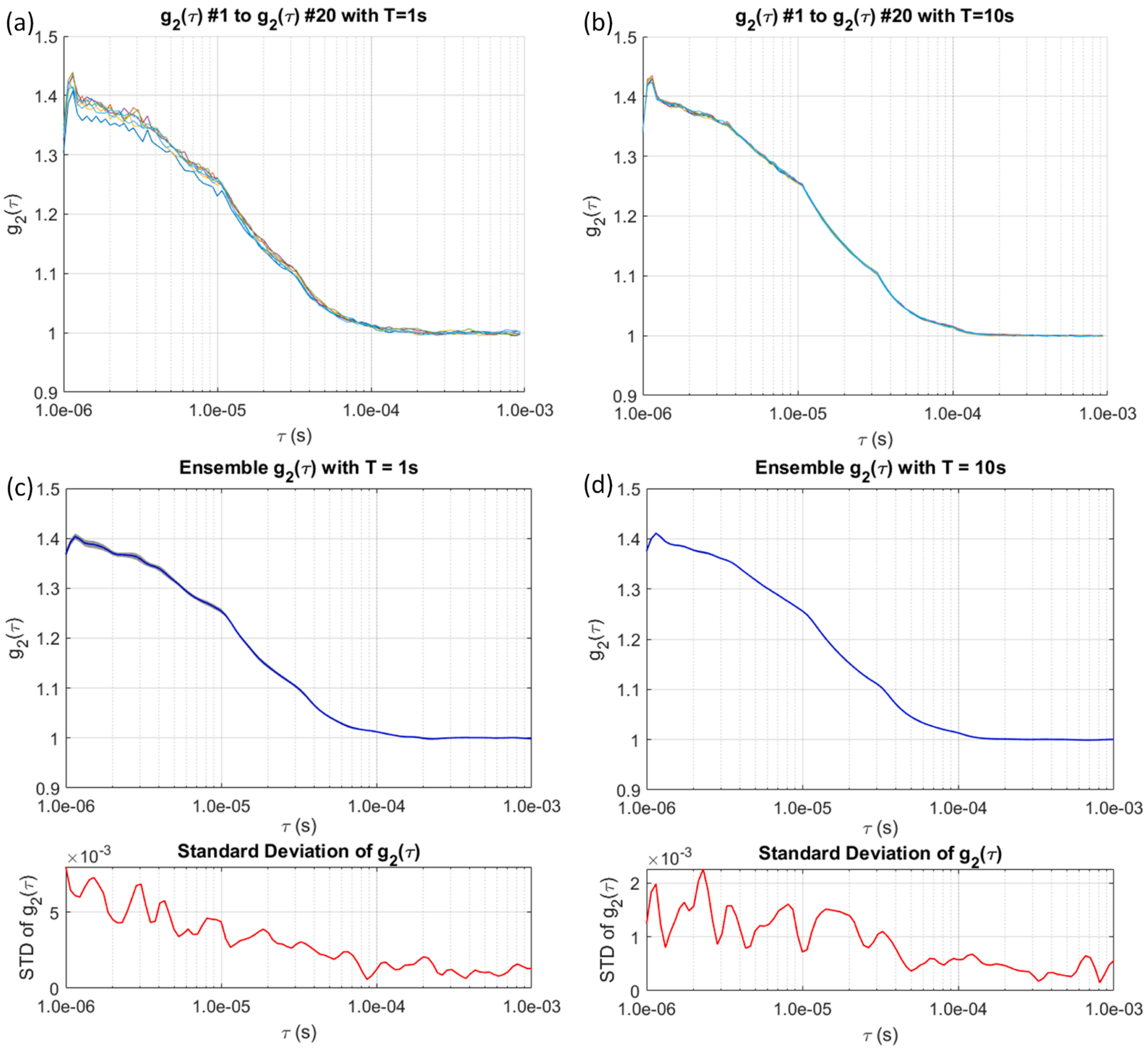

Fig. 12. $g_2(\tau)$ curves of phantom experiement. (a) and (b) 20 curves measured with T = 1 s and 10 s. (c) and (d) Averaged ensambld $g_2(\tau)$ (blue) and *std* (red).

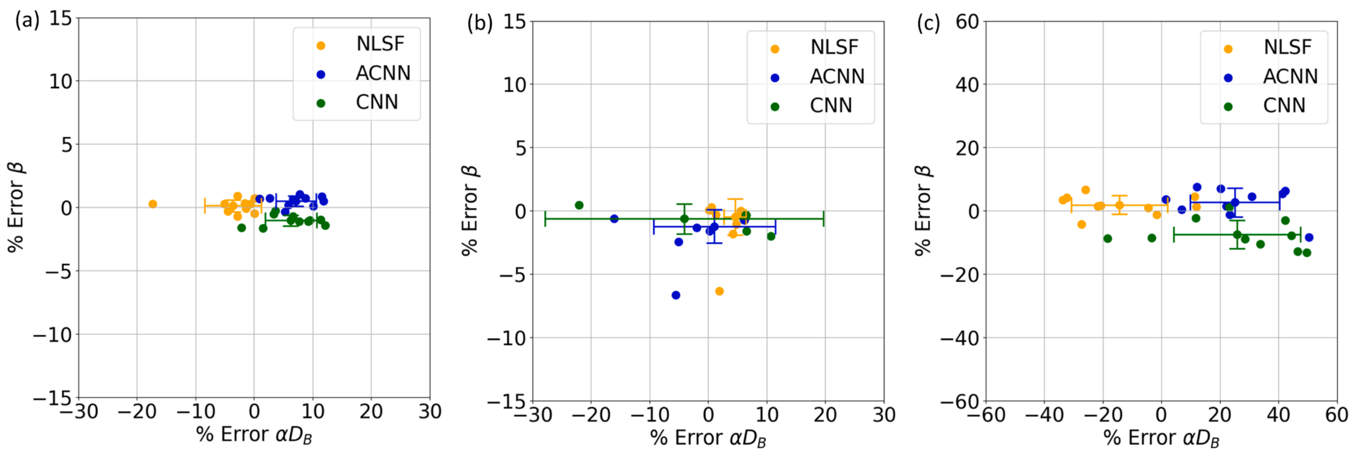

Fig. 13. Error distribution of reconstructed β and αD_B from three algorithms evaluated with milk. (a-c) ρ = 10, 20, 30 mm. Error cars show the mean of the error distrobution and mean values are at the centre of the crosses. The initial value of β and αD_B before the fitting is 0.4 and $1{\times}10^{-6}$ mm^2/s.

VFPU [47] technologies in ARM CPUs to accelerate vector multiplications in the numerator, as the arithmetic logical unit (ALU) in ARM Cortex-A9 and -A53 CPU cores on our FPGA boards contain vectorized dividers and multipliers optimized with single instruction multiple data (SIMD), which offloads ACF processing from PL. We yielded 11-fold acceleration compared to the speed without enabling VFPU. Therefore, allocating workloads to PS and PL achieves a trade-off between performance and hardware utilization. Meanwhile, we can reserve more hardware resources for regular vector-addition and accumulation (VAA) and vector-multiplication (VM) operations for DL accelerators.

The processing pipeline in PS is summarized in Fig. 14(a). To validate the functionality, we initially preloaded intensity data in the DDR. Multiple bit-width fixed-point (FXP) quantization methods were adopted to alleviate the hardware overhead and timing latency, and the floating-point (FLP) to FXP conversion was implemented as shift operations and divisions on PS. Once enough ACFs are computed and a predefined number corresponding to the number of cores specified in Table 5 is reached, the quantized and normalized ACF are dispatched to the PL accelerators through an AXI-full interface. Similarly, the FXP to FLP conversion was implemented similarly for printing readable BFi and β. *malloc(·)* was used to store quantized and dispatch ACF vectors to AXI's Master port. *MACROS* parameterizes the pointer addressing space and the length of ACFs. A general-purpose timer driver was employed to measure the time-consumption when DL accelerators reconstruct BFi and β.

As shown in Fig. 14(b), our ACNN accelerator is a scalable, multi-core architecture, where each DL-core can process one ACF and reconstruct BFi and β simultaneously. The details about each DL-core are depicted in Fig. 14(b); the input feature go through a series of UAC modules, where the for-loops of output channels and kernel windows are unrolled to improve the parallelism. The *Reshape* flattens the feature and allocates the data in channel dimensions for channel-wise convolution afterwards. Unrolling operations in UAC are decomposed in Fig. 14(c). The BRAMs (storing learned parameters) are partially partitioned

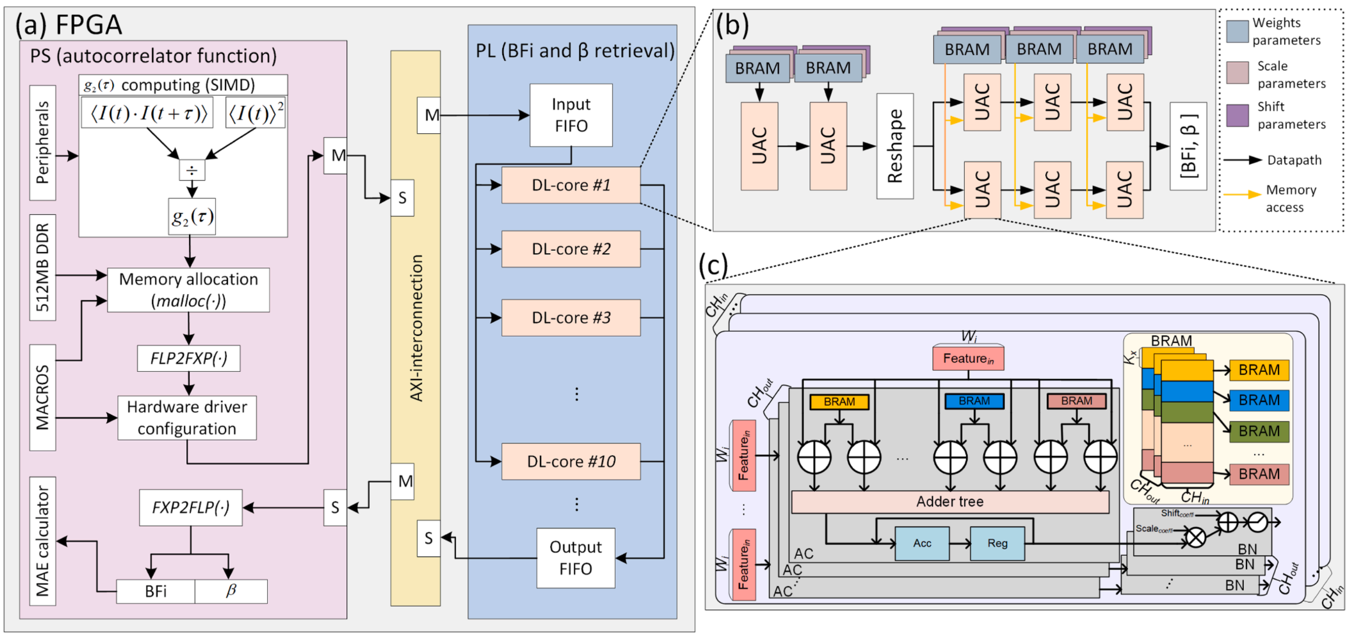

Fig. 14. Hardware architecture integrates intensity temporal ACF computation and ACNN accelerators (For example, 10 cores on Zynq 7000 and 15 cores on Zynq UltraScale+). (a). The architecture overview illustrates the data transfer and functionalities of each module. (b). Detailed architecture of each DL core, data path and memory access were depicted with back and yellow arrows, respectively. (c). Detailed structures of each UAC illustrate parallelism in the input channel, output channel, and kernel size.

Table 5
Evaluation results of ACNN accelerators with different quantization bit-width on Zynq-7000 and Zynq-UltraScale+ FPGA.

	# Core	Data type	DFF	LUT	LUTRAM	BRAM_18K	DSP	PPMS
Zynq-7000	5	Fixed<18, 9>	18.99% (20,203 out of 106,400)	50.56% (26,898 out of 53,200)	7.20% (1,253 out of 17,400)	36.07% (50.50 out of 140)	29.55% (65 out of 220)	21.74
		Fixed<24, 12>	19.00% (20,212 out of 106,400)	49.89% (26,541 out of 53,200)	7.20% (1,253 out of 17,400)	36.07% (50.50 out of 140)	29.55% (65 out of 220)	17.86
	10	Fixed<18, 9>	34.75% (36,973 out of 106,400)	91.93% (48,588 out of 53,200)	12.55% (2,183 out of 17,400)	75.36% (105.5, out of 140)	59.09% (130 out of 220)	41.67
		Fixed<24, 12>	34.81% (37,043 out of 106,400)	90.07% (47,919 out of 53,200)	12.54% (2,182 out of 17,400)	75.36% (105.5, out of 140)	59.09% (130 out of 220)	35.01
Zynq UltraScale+ MPSoC	10	Fixed<18, 9>	5.88% (27,083 out of 460,800)	21.96% (50,603 out of 230,400)	1.35% (1,375 out of 101,760)	35.58% (111 out of 312)	7.52% (130 out of 1,728)	38.46
		Fixed<24, 12>	5.68% (26,162, out of 460,800)	22.25% (51,253 out of 230,400)	1.35% (1,375 out of 101,760)	35.58% (111 out of 312)	7.52% (130 out of 1,728)	33.21
	15	Fixed<18, 9>	8.35% (38,473, out of 460,800)	32.28% (74,375 out of 230,400)	1.82% (1,855 out of 101,760)	53.21% (166 out of 312)	11.28% (195 out of 1,728)	48.49
		Fixed<24, 12>	8.05% (37,101, out of 460,800)	32.74% (75,438 out of 230,400)	1.82% (1,855 out of 101,760)	53.21% (166 out of 312)	11.28% (195 out of 1,728)	44.37

accordingly to satisfy the bandwidth requirement of UACs. As BNs are involved in each UAC, they are implemented as VMs after ACs. In theory, BN at the inference phase can be transformed into a matrix multiplication with two constants, *scale* and *shift* [27,48,49]:

$$x_{BN}(i) = scale(i) \times x(i) + shift(i), \tag{6}$$

where i means the i^{th} output channel,

$$\begin{cases} scale(i) = \dfrac{\gamma(i)}{\sqrt{\delta(i)^2 + \varepsilon}} \\ shift(i) = \omega(i) - \dfrac{\gamma(i)\theta(i)}{\sqrt{\delta(i)^2 + \varepsilon}} \end{cases}, \tag{7}$$

γ and ω are trainable parameters that are parsed off-line, δ and θ are the statistical standard deviation and mean of $x(i)$, ε is a constant to avoid dividing by zero (0.0001 by default). Once these parameters from the pre-trained model are extracted, they are preloaded on BRAMs, as the dashed box shows in Fig. 14(b).

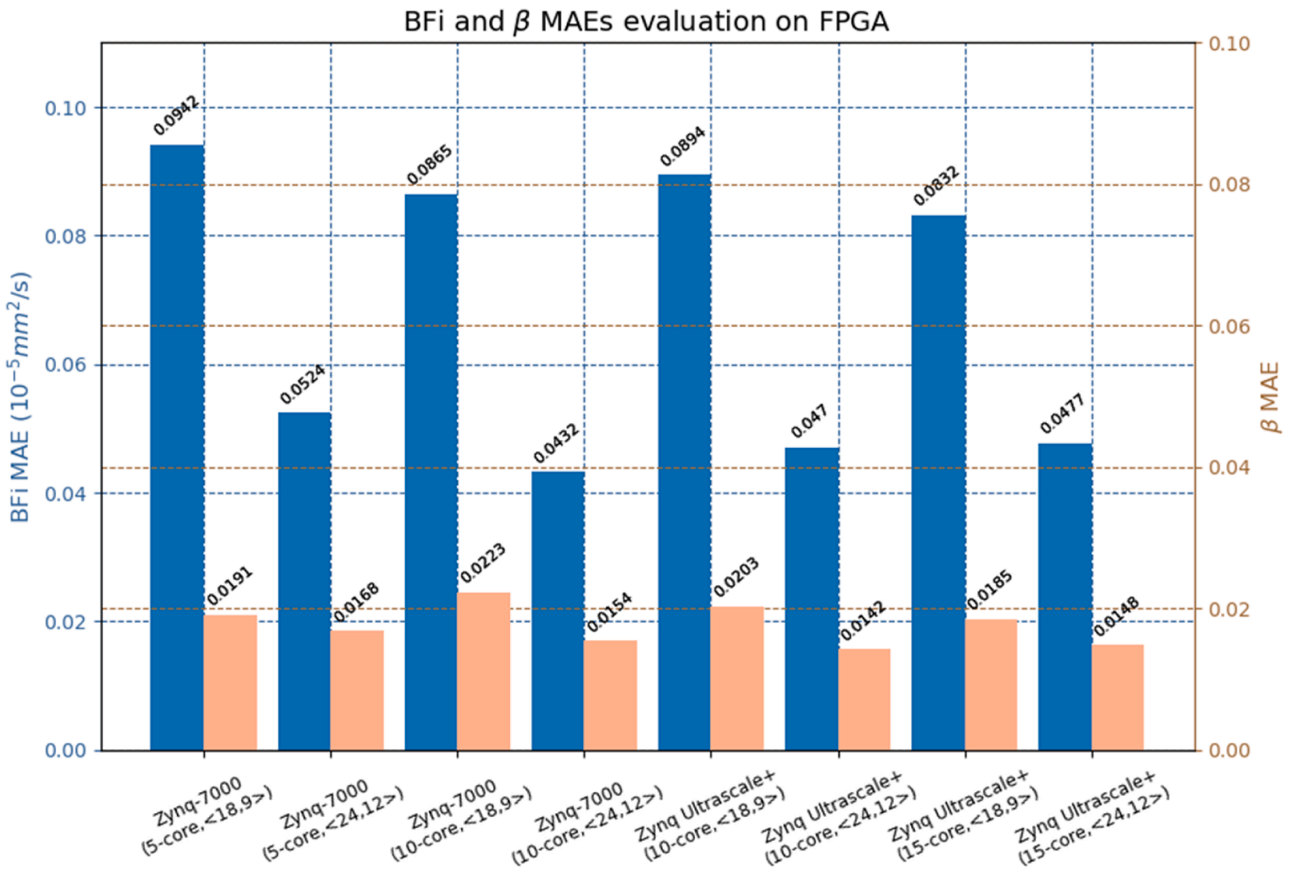

Fig. 15. DL cores implementation on Zynq-7000 and Zynq-UltraScale+ with different quantization FXP schemes, and their corresponding MAE of BFi and β.

6.1. Accelerator evaluation

We implemented the ACNN accelerator on Zynq-7000 and Zynq-UltraScale+ to investigate its performance. To validate the scalability, we implemented five and ten DL cores on Zynq-7000, and ten and fifteen on Zynq UltraScale+. Each scheme contains two sets of FXP bit-width, 24 bit-width containing 12 fractional bits, and 18 bit-width containing 9 fractional bits. Hardware consumption and speed are demonstrated in Table 5. Considering our accelerator targets single-photon detector arrays, we use pixel per millisecond (PPMS) to assess the speed of simultaneously processing ACFs. Timer functions on the PS measure the time-consumption of the ACNN accelerator. According to Table 5, PPMS increases while the number of DL cores increases. Besides, we evaluated the reconstruction accuracy of BFI and β of the eight schemes in terms of mean absolute error (MAE) that is directly output from the FPGA board. As shown in Fig. 15, there is no considerable fluctuation from different numbers of cores and FXP bit-widths. In theory, the accuracy of FPGA results should match that of the GPU, and the ratio between <18, 9> and <24, 12> should remain consistent. However, slight deviations may occur due to quantization errors, underflow or overflow errors, or timing violations. These errors are acceptable within the context of our work. We define the computational efficiency using the latency of batches over power consumption, as shown in Table 6. We measured the power consumption of accelerators on FPGAs using Xilinx Power Estimator [50]. We also used NVML [51] and PyJoules [52] APIs to measure the power consumption of GPU and CPU. FPGA-based SoC platforms obtain the highest computational efficiency across different batch sizes compared to the CPU and GPU.

7. Discussion

We present a holistic hardware platform integrating a $g_2(\tau)$ generator with a multiplication-free, DNN-based hardware accelerator for BFi reconstruction. The current implementation assumes that photon intensity data is received from a single detector. However, averaging $g_2(\tau)$ across multiple parallel detectors can provide a higher signal-to-noise ratio (SNR). Additionally, the intensity data was pre-loaded into on-chip memory for functional evaluation, assuming that the hardware interface has sufficient data transfer bandwidth to receive data from the detector. This hardware platform presents three key aspects for the future work:

1. we will explore how to connect the $g_2(\tau)$ generator module to the detector with an efficient interface design.
2. a module could be developed to compute average ensemble $g_2(\tau)$ data from individual detectors, enhancing the SNR. Another hardware improvement is the power consumption; 90% of total power consumption is from PS ($g_2(\tau)$ generator) as the CPU cores are more power-demanding than PL side logic fabric. Given a previous study implementing a $g_2(\tau)$ generator in PL [7], an ideal solution is to integrate a $g_2(\tau)$ generator and DNN accelerator on PL to save more energy if the hardware resource is sufficient.
3. the PYNQ-Z2 and ZCU104 MPSoC FPGAs have 2 and 4 CPU cores, respectively; however, we only enabled one core to implement the $g_2(\tau)$ generator. In the future, it is worthwhile to explore enabling multiple cores to accelerate the computation of ensemble $g_2(\tau)$ calculations.

Apart from potential hardware improvements, there are limitations in the simulated data generated by both *in-silico* and analytical models. According to Eq. (4), noise generation for $g_2(\tau)$ requires averaging time T and photon intensity I. While the present total number of photons in a simulated voxel is available in MCX, configurable I in the temporal dimension of $g_2(\tau)$ is not available in MCX, nor is T. Therefore, the analytical noise model in MCX is not applicable for producing clear $g_2(\tau)$ data from MCX simulations. Additionally, regarding the analytical model used in this work, we adopted a semi-infinite homogeneous model instead of a multi-layer model, which limits the generalization of our network to complex tissues, such as a multi-layer brain model. Recent research [45] has introduced a three-layer analytical model, which could enhance data-driven methods for multi-layer structures. As a future direction, we plan to integrate the multi-layer analytical model into our network training pipeline to generalize our approach to more complex tissue structures.

8. Conclusion

This study reports a compact, hardware friendly DNN to reconstruct BFi and β in DCS. The correctness of our analytical model was verified by comparing *in-silico* MC simulations. According to Fig. 8, we ensure that our DNN architecture achieves higher accuracy than conventional CNNs yet has simpler operators. We used a real liquid phantom experiment to evaluate ACNN, CNN, and NLSF. The results reveal that ACNN estimates αD_B and β values closest to the reference values. By leveraging the miniaturized architecture, we implemented our scalable DL clusters on an SoC-based FPGA to assess their accuracy, hardware utilization, and computational efficiency. Enhanced with different bit-widths of quantization schemes, we evaluated these quantized versions on cost-optimized and high-end FPGAs. To achieve end-to-end processing, we encapsulated $g_2(\tau)$ generation and BFi reconstruction on a single SoC-based FPGA, providing different on-chip processing solutions for modern DCS sensing systems. Given that current FPGA mounted on modern SPAD arrays are mainly middle-end devices that only contain PL, our heterogenous hardware architecture provides a foundation for future integration of high-end FPGA and SPAD arrays.

Table 6
Performance comparisons of CPU, GPU, and FPGA-based SoC when processing different numbers of pixels (ACFs) for each batch.

	5 pixels/batch			
	i7-12800H CPU	RTX A1000 GPU	Artix-7 and 1 thread ARM-Cortex-A9, <18, 9>	Artix-7 and 1 thread ARM-Cortex-A9, <24, 12>
Power (W)	3.586	4.438	**1.962**	**1.981**
Latency (ms/batch)	3.753	1.311	**0.183**	**0.174**
Efficiency (W/ms)	0.956	3.385	**10.721**	**11.385**
	10 pixels/batch			
	i7-12800H CPU	RTX A1000 GPU	Artix-7 and 1 thread ARM-Cortex-A9, <18, 9>	Artix-7 and 1 thread ARM-Cortex-A9, <24, 12>
Power (W)	3.534	4.346	**2.549**	**2.579**
Latency (ms/batch)	3.927	1.356	**0.258**	**0.267**
Efficiency (W/ms)	0.900	3.205	**9.880**	**9.659**
	15 pixels/batch			
	i7-12800H CPU	RTX A1000 GPU	UltraScale+ and 1 thread ARM-Cortex-A53, <18, 9>	UltraScale+ and 1 thread ARM-Cortex-A53, <24, 12>
Power (W)	**3.549**	4.236	4.826	4.846
Latency (ms/batch)	5.115	1.524	**0.283**	**0.295**
Efficiency (W/ms)	0.694	2.780	**17.053**	**16.427**

CRediT authorship contribution statement

Zhenya Zang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Data curation, Conceptualization. **Quan Wang:** Software. **Mingliang Pan:** Visualization, Validation, Methodology, Data curation. **Yuanzhe Zhang:** Visualization, Validation, Software, Methodology, Data curation. **Xi Chen:** Software. **Xingda Li:** Software. **David Day Uei Li:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by the EPSRC (EP/T00097X/1): the Quantum Technology Hub in Quantum Imaging (QuantiC), Datalab, Photon Force, Ltd., Xilinx, and the University of Strathclyde.

References

[1] T. Durduran, A.G. Yodh, Diffuse correlation spectroscopy for non-invasive, micro-vascular cerebral blood flow measurement, Neuroimage 85 (2014) 51–63.

[2] S.A. Carp, M.B. Robinson, M.A. Franceschini, Diffuse correlation spectroscopy: current status and future outlook, Neurophotonics 10 (1) (2023) 013509.

[3] Y. Shang, T. Li, G. Yu, Clinical applications of near-infrared diffuse correlation spectroscopy and tomography for tissue blood flow monitoring and imaging, Physiol. Meas. 38 (4) (2017) R1.

[4] W. Liu, et al., Fast and sensitive diffuse correlation spectroscopy with highly parallelized single photon detection, APL Photonics 6 (2) (2021).

[5] Y. Shang, T. Symons, T. Durduran, A.G. Yodh, G. Yu, Effects of muscle fiber motion on diffuse correlation spectroscopy blood flow measurements during exercise, Biomed. Opt. Express 1 (2) (2010) 500–511.

[6] C.G. Bangalore-Yogananda, R. Rosenberry, S. Soni, H. Liu, M.D. Nelson, F. Tian, Concurrent measurement of skeletal muscle blood flow during exercise with diffuse correlation spectroscopy and Doppler ultrasound, Biomed. Opt. Express 9 (1) (2018) 131–141.

[7] F.M. Della Rocca, E.J. Sie, R. Catoen, F. Marsili, R.K. Henderson, Field programmable gate array compression for large array multispeckle diffuse correlation spectroscopy, J. Biomed. Opt. 28 (5) (2023) 057001.

[8] M.A. Wayne, et al., Massively parallel, real-time multispeckle diffuse correlation spectroscopy using a 500\times 500 SPAD camera, Biomed. Opt. Express 14 (2) (2023) 703–713.

[9] D. Tamborini, et al., Portable system for time-domain diffuse correlation spectroscopy, IEEE Trans. Biomed. Eng. 66 (11) (2019) 3014–3025.

[10] E.J. Sie, et al., High-sensitivity multispeckle diffuse correlation spectroscopy, Neurophotonics 7 (3) (2020) 035010.

[11] C. Zhou, G. Yu, D. Furuya, J.H. Greenberg, A.G. Yodh, T. Durduran, Diffuse optical correlation tomography of cerebral blood flow during cortical spreading depression in rat brain, Opt. Express 14 (3) (2006) 1125–1144.

[12] C.J. Stapels, et al., A scalable correlator for multichannel diffuse correlation spectroscopy, in: Proceedings of the Advanced Biomedical and Clinical Diagnostic and Surgical Guidance Systems XIV, SPIE, 2016, pp. 106–112.

[13] S.O. Rice, Mathematical analysis of random noise, Bell Syst. Tech. J. 23 (3) (1944) 282–332.

[14] L. Dong, L. He, Y. Lin, Y. Shang, G. Yu, Simultaneously extracting multiple parameters via fitting one single autocorrelation function curve in diffuse correlation spectroscopy, IEEE Trans. Biomed. Eng. 60 (2) (2012) 361–368.

[15] H. Zhao, E. Sathialingam, E.M. Buckley, Accuracy of diffuse correlation spectroscopy measurements of cerebral blood flow when using a three-layer analytical model, Biomed. Opt. Express 12 (11) (2021) 7149–7161.

[16] D. Mazumder, M.M. Wu, N. Ozana, D. Tamborini, M.A. Franceschini, S.A. Carp, Optimization of time domain diffuse correlation spectroscopy parameters for measuring brain blood flow, Neurophotonics 8 (3) (2021) 035005.

[17] W. Lin, D.R. Busch, C.C. Goh, J. Barsi, T.F. Floyd, Diffuse correlation spectroscopy analysis implemented on a field programmable gate array, IEEE Access 7 (2019) 122503–122512.

[18] P. Zhang, Z. Gui, L. Hao, X. Zhang, C. Liu, Y. Shang, Signal processing for diffuse correlation spectroscopy with recurrent neural network of deep learning, in: Proceedings of the IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), IEEE, 2019, pp. 328–332.

[19] C.S. Poon, F. Long, U. Sunar, Deep learning model for ultrafast quantification of blood flow in diffuse correlation spectroscopy, Biomed. Opt. Express 11 (10) (2020) 5557–5564.

[20] Z. Li, Q. Ge, J. Feng, K. Jia, J. Zhao, Quantification of blood flow index in diffuse correlation spectroscopy using long short-term memory architecture, Biomed. Opt. Express 12 (7) (2021) 4131–4146.

[21] J. Feng, M. Jiang, J. Bai, K. Jia, Z. Li, Cerebral blood flow monitoring using a ConvGRU model based on diffuse correlation spectroscopy, Infrared Phys. Technol. 129 (2023) 104541.

[22] Agner Fog, Instruction Tables-Lists of Instruction Latencies, Throughputs and Micro-Operation Breakdowns For Intel, AMD, and VIA CPUs, Technical University of Denmark, 2022 [Online]. Available: [Online]. Available: https://www.agner.org/optimize/instruction_tables.pdf.

[23] J. Dong, R. Bi, J.H. Ho, P.S. Thong, K.C. Soo, K. Lee, Diffuse correlation spectroscopy with a fast Fourier transform-based software autocorrelator, J. Biomed. Opt. 17 (9) (2012) 097004.

[24] Y. Shang, G. Yu, A Nth-order linear algorithm for extracting diffuse correlation spectroscopy blood flow indices in heterogeneous tissues, Appl. Phys. Lett. 105 (13) (2014).

[25] S. Xu, et al., Imaging dynamics beneath turbid media via parallelized single-photon detection, Adv. Sci. 9 (24) (2022) 2201885.

[26] J. Buchholz, et al., FPGA implementation of a 32×32 autocorrelator array for analysis of fast image series, Opt. Express 20 (16) (2012) 17767–17782.

[27] Z. Zang, D. Xiao, Q. Wang, Z. Jiao, Y. Chen, D.D.U. Li, Compact and robust deep learning architecture for fluorescence lifetime imaging and FPGA implementation, Methods Appl. Fluoresc. 11 (2) (2023) 025002.

[28] D. Xiao, et al., Dynamic fluorescence lifetime sensing with CMOS single-photon avalanche diode arrays and deep learning processors, Biomed. Opt. Express 12 (6) (2021) 3450–3462.

[29] D.A. Boas, L. Campbell, A.G. Yodh, Scattering and imaging with diffusing temporal field correlations, Phys. Rev. Lett. 75 (9) (1995) 1855.

[30] M. Seong, Y. Oh, K. Lee, J.G. Kim, Blood flow estimation via numerical integration of temporal autocorrelation function in diffuse correlation spectroscopy, Comput. Methods Programs Biomed. 222 (2022) 106933.

[31] R.R. Alfano, W.B. Wang, L. Wang, S.K. Gayen, Light propagation in highly scattering turbid media: concepts, techniques, and biomedical applications. Photonics, John Wiley & Sons, Ltd, 2015, pp. 367–412, https://doi.org/10.1002/9781119011804.ch9.

[32] C. Cheung, J.P. Culver, K. Takahashi, J.H. Greenberg, A. Yodh, *In vivo* cerebrovascular measurement combining diffuse near-infrared absorption and correlation spectroscopies, Phys. Med. Biol. 46 (8) (2001) 2053.

[33] T.J. Farrell, M.S. Patterson, B. Wilson, A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties *in vivo*, Med. Phys. 19 (4) (1992) 879–888.

[34] A. Siegert, On the Fluctuations in Signals Returned by Many Independently Moving Scatterers, Radiation Laboratory, Massachusetts Institute of Technology, 1943.

[35] L. He, Y. Lin, Y. Shang, B.J. Shelton, G. Yu, Using optical fibers with different modes to improve the signal-to-noise ratio of diffuse correlation spectroscopy flow-oximeter measurements, J. Biomed. Opt. 18 (3) (2013) 037001.

[36] R. Bi, J. Dong, K. Lee, Deep tissue flowmetry based on diffuse speckle contrast analysis, Opt. Lett. 38 (9) (2013) 1401–1403.

[37] X. Cheng, H. Chen, E.J. Sie, F. Marsili, D.A. Boas, Development of a Monte Carlo-wave model to simulate time domain diffuse correlation spectroscopy measurements from first principles, J. Biomed. Opt. 27 (8) (2022) 083009.

[38] S.A. Carp, et al., Diffuse correlation spectroscopy measurements of blood flow using 1064 nm light, J. Biomed. Opt. 25 (9) (2020) 097003.

[39] J.D. Johansson, D. Portaluppi, M. Buttafava, F. Villa, A multipixel diffuse correlation spectroscopy system based on a single photon avalanche diode array, J. Biophotonics 12 (11) (2019) e201900091, https://doi.org/10.1002/jbio.201900091.

[40] Q. Fang, D.A. Boas, Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units, Opt. Express 17 (22) (2009) 20178–20190.

[41] Z. Zang, et al., Hardware inspired neural network for efficient time-resolved biomedical imaging, in: Proceedings of the 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2022, pp. 1883–1886, https://doi.org/10.1109/EMBC48229.2022.9871214.

[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[43] K. Simonyan, A. Vedaldi, and A. Zisserman, 'Deep inside convolutional networks: visualising image classification models and saliency maps', *ArXiv Prepr. ArXiv13126034*, 2013.

[44] H. Chen, et al., AdderNet: do we really need multiplications in deep learning?, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 1465–1474, https://doi.org/10.1109/CVPR42600.2020.00154.

[45] Q. Wang, M. Pan, Z. Zang, D.D.U. Li, Quantification of blood flow index in diffuse correlation spectroscopy using a robust deep learning method, J. Biomed. Opt. 29 (1) (2024) 015004.

[46] T. Lindbergh, I. Fredriksson, M. Larsson, T. Strömberg, Spectral determination of a two-parametric phase function for polydispersive scattering liquids, Opt. Express 17 (3) (2009) 1610–1621.

[47] 'NEON version: 1.0 programmer's guide'. 2013. [Online]. Available: [Online]. Available: https://static.docs.arm.com/den0018/a/DEN0018A_neon_programmers_guide_en.pdf.

[48] S. Liang, S. Yin, L. Liu, W. Luk, S. Wei, FP-BNN: binarized neural network on FPGA, Neurocomputing 275 (2018) 1072–1086, https://doi.org/10.1016/j.neucom.2017.09.046.

[49] Z. Xu, R.C.C. Cheung, Binary convolutional neural network acceleration framework for rapid system prototyping, J. Syst. Archit. 109 (2020) 101762, https://doi.org/10.1016/j.sysarc.2020.101762.

[50] ‘Xilinx power estimator user guide (UG440)’. 2023. [Online]. Available: [Online]. Available: https://docs.xilinx.com/r/en-US/ug440-xilinx-power-estimator?tocId=nnrf2odl4xIaqGp3~WtIBA.

[51] ‘NVML API reference guide’. 2023. [Online]. Available: [Online]. Available: https://docs.nvidia.com/deploy/nvml-api/index.html.

[52] ‘pyJoules’. 2024 [Online]. Available: [Online]. Available: https://pyjoules.readthedocs.io/en/latest/, 2020.