Retinal Blood Vessel Segmentation for Macula Detachment Surgery Monitoring Instruments

Mohsen Hajabdollahi[1], Nader Karimi[1], S.M. Reza Soroushmehr[2,3], Shadrokh Samavi[1,3], Kayvan Najarian[2,3,4]

[1]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran
[2]Michigan Center for Integrative Research in Critical Care, University of Michigan, Ann Arbor, 48109 U.S.A
[3]Department of Emergency Medicine, University of Michigan, Ann Arbor, 48109 U.S.A
[4]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, 48109 U.S.A

SUMMARY

In recent years image processing has improved detection and diagnosis in medical application. Image processing applications are now embedded in medical instruments such as MRI and CT. In the case of retinopathy, fast extraction of blood vessels can allow the physician to view injury regions during surgery. Macula detachment surgeries, or computer-aided intraocular surgeries, require precise and real-time knowledge of the vasculature during the operation. Use of artificial neural network (ANN) has produced good results in image processing applications, but its implementation may not be suitable for real-time applications in small, embedded hardware. Due to error resiliency of the neural network, its structure can be pruned and simplified. In this paper an efficient hardware implementation of neural network for retinal vessel segmentation is proposed. We simplify the neural network structure in such a way that the accuracy of the results is not altered significantly. Simulation results and FPGA implementation show that our proposed network has low complexity and can be applied for segmentation of retinal vessels with acceptable accuracy. This makes the proposed method a good candidate to be implemented in any device such as a binocular indirect ophthalmoscope (BIO).

Keywords: Hardware implementation; artificial neural networks (ANN); error compensation; retinopathy; retinal vessel segmentation.

1. INTRODUCTION

In some medical applications, efficient detection of injury location plays a vital role in the process of diagnosis. In recent years image processing techniques are considered as a useful method for detection of medical abnormalities in the human body. Retinal blood vessel detection is very important in diagnostic approach for retinopathy. For example as illustrated in Fig. 1, diabetic retinopathy occurs when blood vessels in the eye become leaky resulting in blood and other fluids to flow into the retinal tissue [1,2].

For retinal bleeding in diabetic patients it is necessary to recognize the location of the bleeding vessels. Vessel segmentation and extraction by image processing techniques can be useful before and during surgical operations of retina. In some operations related to retina, vessels must be detected as soon as possible and segmentation algorithm must be real-time. Hardware implementation of retinal vessel segmentation method in surgical vision instruments, such as binocular indirect ophthalmoscope (BIO) and can be very useful.

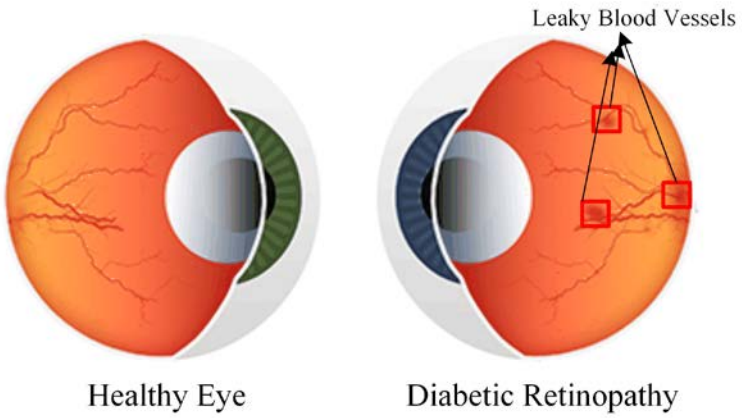


Fig. 1. Diabetic Retinopathy [1]

There are many studies to extract and segment the retinal blood vessels. In [3] a method for detection of retinal vessels based on linear combination of weak classifiers (Ada boost) is proposed. For each pixel, a 41-D feature vector including, intensity structure, spatial properties, and multiple scale geometry is used. Feature vectors are constructed by various filters such as 2-D Gabor wavelet transform. Also, likelihood of structures such as edges and ridges and numerical estimation of the differential properties of the intensity are used. Although this feature vector contains a rich description of vessel regions but it can be very complex. In [4], ensemble system of bagged and boosted decision trees, are proposed. Feature vector based on the orientation analysis of gradient vector field, morphological transformation, line strength measures, and Gabor filter responses are used. In [5] an approach using deep max-pooling convolutional neural networks with GPU implementation is proposed. In [6] a method for the map and localization of retinal vessels for intraocular surgery in real-time is proposed. A recursive template matching algorithm is performed which includes a sparse retinal vessel initialization followed by a tracing algorithm. Candidate vessel points are detected along a coarse grid. Then these points are used to detect other points in both directions along the vessel. Although small fraction of the pixels is analyzed, but in order to detect each series of vessel points, the algorithm needs to trace the candidate points iteratively. In [7] an algorithm for retinal vessel extraction is proposed using local dynamic convolutions and morphological operations. This methodology finds the boundary of the blood vessels using an active contour method, which is called pixel-level snakes (PLS). Edge detection and some simple preprocessing operations are performed which provide some points to guide PLS evolving. This algorithm runs iteratively and all of the image pixels must be traversed during the run which is a time consuming task. In [8] probability of spatial dependency, using an adaptive local thresholding, is used to coarsely segment the vessel map. The result is then refined through curvature analysis and morphological reconstruction. In [9] vessel pixels are segmented in an iterative process and in each process the threshold is adapted using residual image generated by masking out the existing segmented vessel pixels. The new vessel pixels are then extended based on region growing into the existing vessels. In [10] adaptive histogram equalization and a 2D Gabor wavelet are used to enhance the vessels. Also, an anisotropic diffusion filter is used for smoothing the image and preservation of vessel boundaries. Retinal vessels are obtained by using a region-based active contour model. In [11] multi-wavelet kernels, in the form of matched filters, are used for vessel enhancement. Binary map of the vasculature is identified using a threshold based on the vessel edge information. In [12] filters are convolved in

different directions and directional responses of a pixel are considered as a vector. The combination of the statistical measures of the response vectors and its local maxima provide the seeds for the region growing procedure. In [13] vessel-like patterns are detected using a filter by computing the weighted geometric mean of the responses of DoG filters with collinearly aligned supports.

Among the previous studies, artificial neural network (ANN) based approaches have led to efficient designs and good accuracies. In [14] a new deep neural network architecture for retinal vessel extraction is proposed. A Four-stage convolutional neural network is used to create the probability map of vessel boundaries. After that, for improving the results of probability map, a conditional random field layer over the all convolutional neural network outputs, is applied. In [15] a deep convolutional neural network is applied to find a relation between a retinal image and a vessel map. This network architecture can produce a label map as well as a single label value without increasing the normal size of the patches. After that, vessels are segmented by using Bayes probability theory. This means that the number of neighboring pixels participating to vessel detection is increased without increasing the patch size. It can be useful to reduce computational complexity of the algorithm. In [16] a lattice neural network with dendritic processing is used for retinal blood vessel extraction. This process includes preprocessing, feature computation, classification and post processing. Preprocessing is performed by removal of the background light and vessel enhancement. Invariant moments and intensity are used for feature extraction. Train based on merging is used for lattice neural network with dendritic processing. A median based filtering method is applied in the post processing. In [17] a segmentation based on combination of the convolutional neural network and random forest is proposed. The convolutional neural network works as a feature extractor while ensemble random forest works as a classifier. Although neural network studies have good accuracy and scalability, but their architectures are very complex, which is not suitable for real-time applications. Some previous studies on cellular neural networks focused on time and speed up more than any other factors [18, 19, 20]. In [18] retinal vessel tree is extracted in three stages. In the first step a preprocessing is performed based on histogram equalization and opening. After erosion and edge detection in second step, a contour is initialized. Using an active contour and the information in the first step, retinal vessels are detected. In [19] and [20] a retinal vessel segmentation algorithm based on cellular neural networks is proposed which is implementable on hardware. A 15×15 neighborhood with various directions is used for line detection. The accuracy of the proposed algorithm is not comparable to the state-of-the-art algorithms. For its implementation, large amount of hardware resources are required. There are many studies for the acceleration of medical image processing algorithms using hardware-accelerators such as FPGAs and GPUs [21]. Also in portable onsite biomedical diagnostic applications hardware implementation for real-time responses is necessary [22]. In [23] a system for retinal image processing is proposed which combines high quality with high computational efficiency. Vessels are detected based on the local Radon transform, and efficient implementation on GPU is proposed [23]. Although an efficient implementation on GPU is proposed, but many steps are required in local radon transform and in the preprocessing and post processing stages. In [24] a fast and accurate method for retinal vessel segmentation which is suitable for real-time

implementation on GPU is proposed. This work which is based on vessel tracing is included a preprocessing and a contour tracing stages. Preprocessing is performed by linear filtering and morphological operations. Then in the contour tracing stage data from the preprocessing stage form the initial contours. Although parallel processing is done by GPU, but morphological operations as well as contour tracing need to be done iteratively. On the other hand, the high power consumption and high hardware complexity of GPUs discards these for embedded systems [25]. In [22] and [26] high performance parallel hardware architecture for acceleration of the retinal vessel extraction is proposed. A matched filtering based procedure is used to segment the blood vessels. The design is implemented on FPGA. Although in [22] an acceptable speed up is achieved but high hardware utilization makes inappropriate for medical devices such as binocular indirect ophthalmoscope. In [27] an FPGA implementation with SIMD architecture for retinal-vessel tree detection is introduced. A pre-segmentation procedure feed information to a region growing algorithm which is done by cellular active contour algorithm. Pixel processing is done in a parallel manner and independently in several split sub images. This algorithm runs iteratively and its delay is not acceptable for real-time applications.

Macula detachment surgeries, or computer-aided intraocular surgeries, require precise and real-time knowledge of the vasculature. This knowledge is important during retinal procedures such as laser photocoagulation or vessel cannulation [6]. Although in some conditions a single computer works near the medical equipment, but there is a demand for embedding a fast and small hardware module with low power consumption in medical devices [28]. This embedding can be done in portable and hand-held devices or in devices which the real-time implementation is vital. As it is shown in Fig. 2, in binocular indirect ophthalmoscope and digital fundus fluorescein a fast extraction of blood vessels enables clinical personal to view injury regions during surgical operations or in portable devices.

Digital Fundus Fluorescein Angiography [29]

Binocular indirect ophthalmoscope[30]

Fig. 2. Retinal Diagnostic Equipment

From previous studies we see that ANN is an effective and complex system. Hence, it is not a suitable candidate for portable medical devices with real-time applications such as BIO. In this paper a simplified ANN is proposed to detect and segment the retinal blood vessels. For

this purpose, at first an ANN is configured to give an accurate segmentation results. In order to simplify the network structure in the second step, network weights as well as its biases are quantized. To reduce and compensate the quantization error, an optimization method is proposed. Nonlinear activation function is approximated by a pricewise linear function. All approximations are performed in such a way that the hardware structure becomes simple and efficient. Finally, for each part of modified neural network a hardware structure is proposed. By analyzing final design complexity the number of required hardware resources is determined.

The rest of this paper is organized as follows. In Section 2, neural network simplification approaches are presented. In Section 3 hardware architecture of the proposed method is explained. Section 4 is dedicated to simulation results, and in Section 5 concluding remarks are presented.

2. Simplified Artificial Neural Networks

ANNs are considered as general tools for learning of structures of different images. Usually multilayer perceptron structures with nonlinear activation functions are used. Although by using ANN accurate results are achieved but their multilayer structures with many neurons and complex activation functions make their implementations complex. In this regard, research publications are interested in simplification of the main components of neural network including multiplier [31], adders [32] and activation functions [33]. On the other hand, some recent papers are interested on ANN signal representations and operations. In [34, 35, 36, 37] a simple binary representation with logical operations during training is used. Thanks to its signal representation, all multiplication operations as well as activation functions are performed by simple and level-one logical operations. The binary and ternary signal representation used for their implementation makes such structure too deep and wide. The logical operation and binary signal representation necessitate the increasing of number of neurons and layers to generate accurate results. In Fig. 3, the overview of proposed method for simplification of ANN is shown. The network is learned and optimization is performed on its different parts. For each network neuron, its inputs are multiplied by corresponding weights and are added by a tree-adder. These results are fed into an activation function which generates inputs of the next layer. As illustrated in Fig. 3, there are three main components in each ANN layer including multiplier, tree-adder, and activation function. Optimization and simplification, which are performed on the structure shown in Fig. 3, are as follows.

2.1. Weight quantization and error compensation

As illustrated in Fig. 3, a typical multi-layer ANN requires elements which multiply each input by a corresponding weight. In typical implementation of an ANN with M inputs, W nodes in the hidden layer and N output nodes, the number of multipliers is

$$number\ of\ multiplier = \ M \times W \times N$$

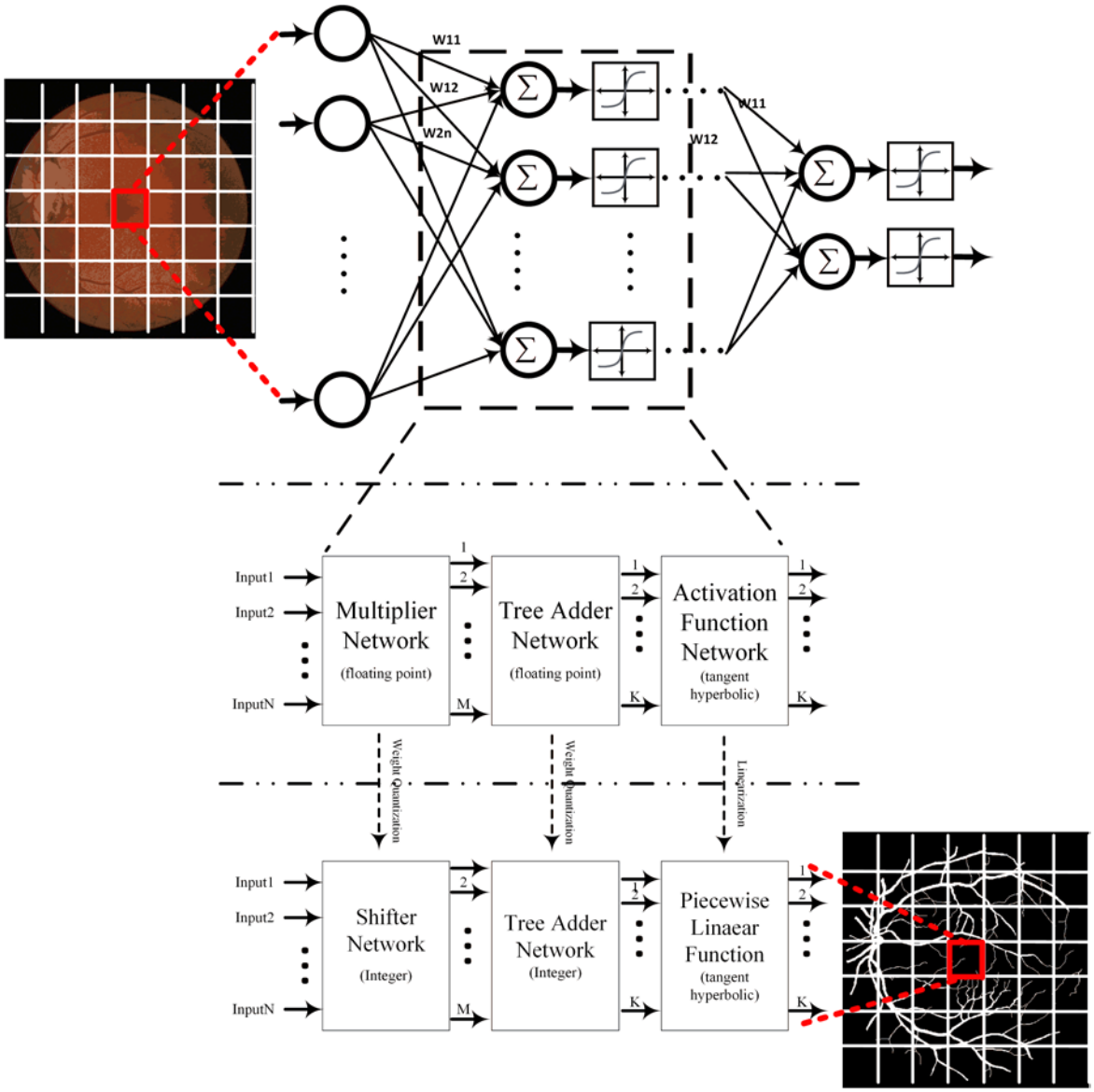


Fig. 3. Overview of ANN structure optimization

Given that all multiplies must be floating points, high computational effort could be imposed for computational operations in each layer. An alternative way is to implement multipliers with shift and add operations. This reduces the complexity of calculations. Considering the complexity of the design hardware implementation, weights are quantized assuming that at most a limited number of shift and add operations are allowed. Hence, the nearest number to the original one is selected as its quantized value. For example suppose that we have two numbers 0.8735 and 0.3811 as weights in an ANN and at most 3 number of shift and add operations are allowed. These numbers can be represented in a new form of addition and shift operations as follows

$$0.8735 \cong 0.8750 = 2^{-1} + 2^{-2} + 2^{-3}$$

$$0.3811 \cong 0.3750 = 2^{-2} + 2^{-3}$$

By using this form, all weight become in the form of sum of power 2 numbers which can be implemented by shift and add operations. In this way, each required multiplier module in

ANN is converted to several adders and shifter modules. Although a simple quantization with respect to numbers of power of 2 operations reduce the computational complexity but an error is generated which may be a problem in some conditions. In the following an error compensation procedure is proposed to deal with this problem.

2.2. AVERAGE QUANTIZATION ERROR REDUCTION

In the common type of quantization, weights are quantized by only considering their values. This could create quantization errors that are accumulated and could lead to significant loss of accuracy. Hence, a compensation error method is proposed. Each quantization may lead to a loss of accuracy. But there is similarity in each image region and the loss of accuracy due to a weight quantization can be compensated in the later weight quantization. In this way average error as well as the loss of accuracy can be reduced. To do this after quantization of each weight, the created error is diffused in the next weight quantization. Let us consider the following example. Three weight coefficients of 0.8000, 0.4250 and 0.4050 are considered and at most 3 number of shift and add operations are allowed. The nearest quantized value in form of add and shift number and their quantization error are illustrated.

$$0.8000 \cong 0.7500 = 2^{-1} + 2^{-2} \Rightarrow quantization\ error = 0.8000 - 0.7500 = +0.0500$$
$$0.4250 \cong 0.3750 = 2^{-2} + 2^{-3} \Rightarrow quantization\ error = 0.4250 - 0.3750 = +0.0500$$
$$0.4050 \cong 0.3750 = 2^{-2} + 2^{-3} \Rightarrow quantization\ error = 0.4050 - 0.3750 = +0.0300$$

Hence, average quantization error becomes

$$average\ error = \frac{+0.0500+0.0500+0.0300}{3} = +0.0433$$

Average quantization error can be reduced by diffusion of each quantization error in the next steps of weight quantization. In the case of our example we have

$$0.8000 \cong 0.7500 = 2^{-1} + 2^{-2} \Rightarrow error = 0.8000 - 0.7500 = +0.0500$$

$$0.4250 \xrightarrow{\substack{error \\ difusion}} = 0.4250 + 0.0500 = 0.4750 \cong 0.5000 = 2^{-1} \Rightarrow error = 0.4250 - 0.5000 = -0.0750$$

$$0.4050 \xrightarrow{\substack{error \\ difusion}} = 0.4050 + 0.0500 - 0.0750 = 0.3800 \cong 0.3750 = 2^{-2} + 2^{-3} \Rightarrow error = 0.4050 - 0.3750 = +0.0300$$

$$average\ error = \frac{+0.0500-0.0750+0.0300}{3} = +\frac{0.0050}{3} = +0.0016$$

All quantization errors in previous stages are considered in the current quantization stage. As it was observed, in the case of 0.4250, the quantization error of +0.0500 from the previous stage is considered in the current quantization stage. Hence, the current value 0.4250 is increased by +0.0500. Also for the quantization of 0.4050, previous quantization errors (+0.0500 and – 0.0750) are considered in the current quantization. This means that the current value 0.4050 is added to +0.0500 and – 0.0750.

In this example the average error is reduced due to considering the previous quantization errors in the current weight quantization. By this technique the total error caused by quantization can be reduced.

2.3. ACTIVATION FUNCTION LINEARIZATION

In ANN, hyperbolic tangent is the widely used activation function with the following form.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Hence, an exponential operation as well as a floating point division must be calculated. Simplification and linearization of activation function especially in the case of large networks can be an efficient way to reduce total computation volume. In this paper the domain of $tanh(x)$ function is divided into four intervals, and in each interval a linear approximation function is used based on (1).

$$Picewise\ Linear\ Approximation\ of\ \tanh(x) = \begin{cases} \pm x & for \quad 0 \le \pm x < 0.5 \\ \pm\left(0.5 + \frac{(ABS(x) - 0.5)}{2}\right) & for \quad 0.5 \le \pm x < 1 \\ \pm\left(0.75 + \frac{(ABS(x) - 1)}{4}\right) & for \quad 1 \le \pm x < 2 \\ \pm 1 & for \quad \pm x \ge 2 \end{cases} \tag{1}$$

By using this pricewise linear function, computation is performed without any multiplication and division and all operations are in form of addition and shift. In Fig. 4, $tanh(x)$ and its approximation are depicted visually. From Fig. 4, it is observed that the proposed piecewise linear function is approximately near the original function. A comparison between approximate and accurate structure is provided in experimental results.

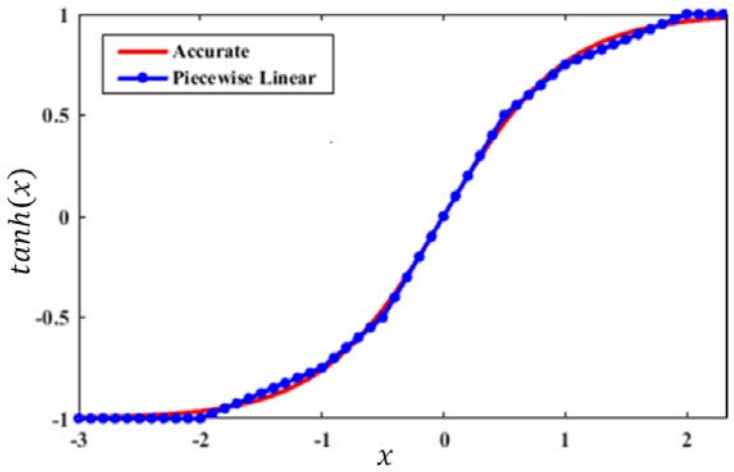


Fig. 4. Piecewise linearization of tanh(x)

2.4. POST PROCESSING

After neural network segmentation, there are some pixels inside of the vessels which may be not segmented as vessel points. Always these pixels are located on thick vessels. Hence, after the segmentation by ANN, its accuracy is improved by a post processing stage. Always all gaps are filled out by a morphological filling operation. Since morphological operations are

iterative and create delay in the system, a simple filling operation is proposed. For a 5×5 image block, any pixel with at least T neighbors is classified as vessel point.

3. Hardware Architecture of Simplified Neural Network

In this section, efficient hardware architecture is designed for the proposed simplified neural network. The major part of hardware architecture is the implementation of the weights of the layers as well as the activation function. To simplify the neural network we try to optimize the architecture of each part. The hardware structure of the proposed simplified neural network is illustrated in Fig. 5. This hardware architecture does not need any multiplication and division and all operations are in the form of add and shift. Three major parts of this structure are designed as follows.

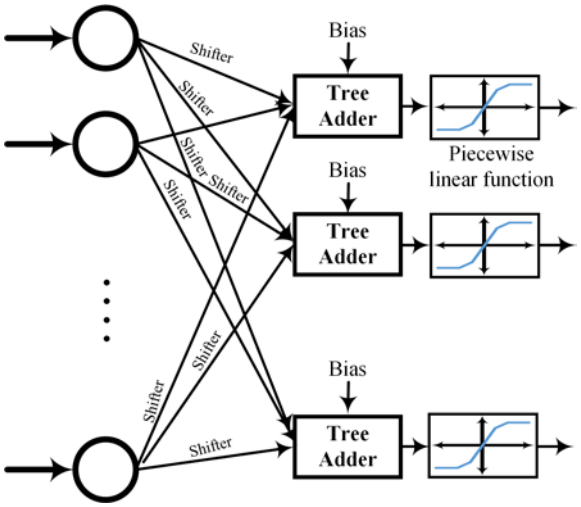


Fig. 5. Hardware structure of the simplified ANN

3.1. Weights Multiplication Layer

It was mentioned in Section 2.1 that a floating point operation is necessary for multiplication of an input by a corresponding weight. Using the proposed weight quantization it is possible to perform a simple shift and add instead of a complex floating point operation. As shown in Fig. 5, a network of shift operations in each layer instead of complex multiplications.

3.2. Tree Adder Layer

After multiplication of network weights in each layer, in the form of proposed method, their results must be summed up. For this purpose a tree-adder with Integer representation is used for each neuron to add the multiplication results from previous stages as well as neuron bias.

3.3. PIECEWISE LINEAR ACTIVATION FUNCTION

In the case of nonlinear activation function implementation, one of the approximate and flexible methods is look up table (LUT) [38]. In the proposed ANN, LUT structure with 256 entry in each of them a floating point number is required. Hence, total size of LUT for each activation function become 256 × 32 bit.

An approximation of an exponential function in form of piecewise linear makes its implementation simple and efficient. This is expressed by a multi-part function and can be obtained by a series of simple logical operations as illustrated in Fig. 6. The three conditions of (1) are checked by three comparators which are fed from the absolute calculator module. Then appropriate outputs, which are provided by shifter and adder modules, are selected by the multiplexer. Finally, a sign correction module, corrects the sign of final result. Two shifter modules are used in which one and two right shifting operations are performed.

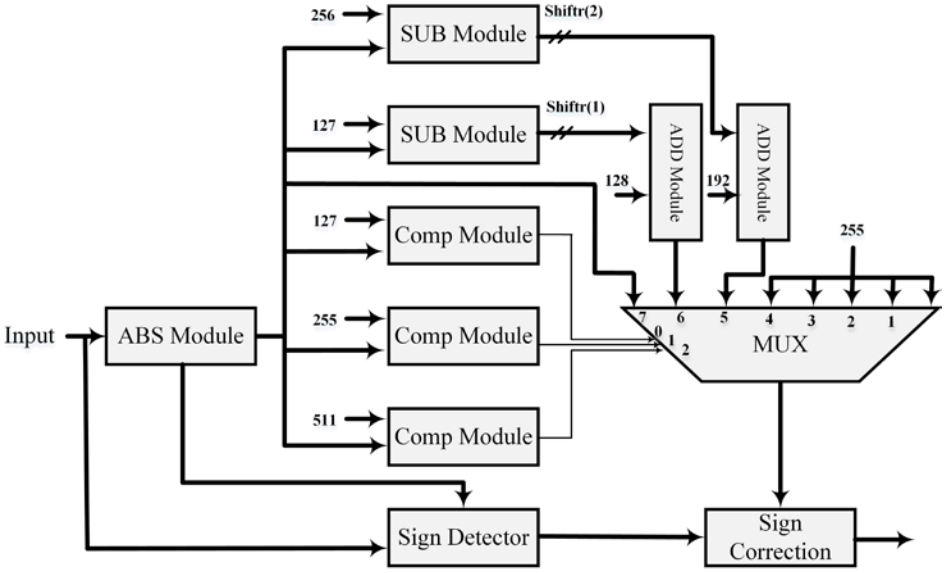


Fig. 6. Piecewise linear activation function module

3.4. POST PROCESSING HARDWARE

Post processing is performed on the segmentation mask which is a binary image. In Fig. 7, the hardware structure of this post processor is illustrated. Pixels of 5×5 image block are shown as P1 to P25. A 25:5 compressor module is required for counting the number of ones in the image block. For hardware implementation of 25:5 compressor, two 15:4 compressors as well as a 4 bit adder are used. Then the generated binary value is compared with a threshold value (T) by a comparator module.

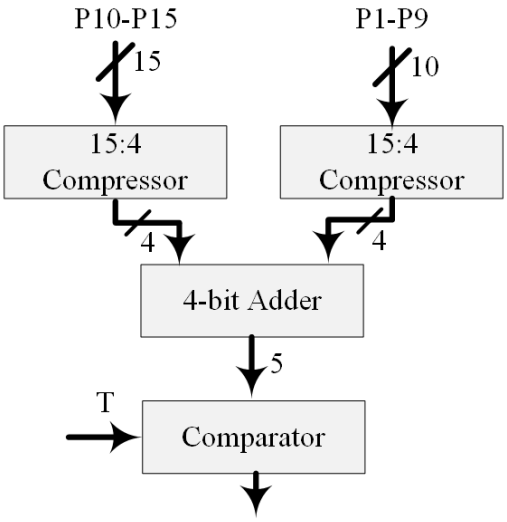


Fig. 7. Post processing module

4. EXPERIMENTAL RESULTS

The performance and the accuracy of the proposed method have been tested using the publicly available digital retinal images from databases of (DRIVE) [39] and STARE [40]. DRIVE database contains 40 images and STARE contains 20 images. The performance of the proposed method is evaluated separately using 4-fold cross validation method. Each pixel of the image is analyzed by feeding the neural network with a 7×7 image block that surrounds that pixel. The threshold value for post processing is set to 10.

4.1. ANN CONFIGURATION SELECTION

Different neural networks are tested with different configurations on two aforementioned datasets. Four scores which are used for comparison are accuracy (ACC), Dice score ($DICE$), sensitivity (SEN) and specificity (SPE), which are as follows.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \qquad DICE = \frac{2TP}{2TP + FP + FN} \qquad SEN = \frac{TP}{TP + FN} \qquad SPE = \frac{TN}{FP}$$

where $TP, TN, FP,$ and FN are true positive, true negative, false positive and false negative rates respectively. Three best networks are selected and their results are compared in Table 1 for DRIVE and in Table 2 for STARE. In these configurations, networks with three layers in the form of 20-6-2, 10-6-2 and 8-4-2 nodes are used. The three networks that are used are: 1) network with no quantization (accurate), 2) network with common weight quantization, and 3) network using quantization with error compensation.

It is observed that in the network 20-6-2 there is not significant improvement in accuracy but it has higher complexity than the other configurations. The performance comparison of these two structures in the case of DRIVE and STARE are provided in section 4.2. On the other

hand, it is observed that although the 8-4-2 network configuration has lower complexity but the loss of accuracy due to quantization is high.

Table 1- Performance Comparison in Different Configuration on DRIVE

	ANN 8-4-2	*ACC*	*DICE*	*SEN*	*SPE*
DRIVE	Quantization	0.9121	0.5454	0.5253	0.9555
	Quantization and error compensation	0.9493	0.6833	0.5586	0.9928
	Accurate	0.9580	0.7783	0.7460	0.9817
	ANN 10-6-2	*ACC*	*DICE*	*SEN*	*SPE*
	Quantization	0.9220	0.6111	0.6207	0.9555
	Quantization and error compensation	0.9421	0.7413	0.8319	0.9545
	Accurate	0.9588	0.7811	0.7426	0.9830
	ANN 20-6-2	*ACC*	*DICE*	*SEN*	*SPE*
	Quantization	0.9469	0.6529	0.5147	0.9950
	Quantization and error compensation	0.9533	0.7534	0.7198	0.9794
	Accurate	0.9590	0.7843	0.7531	0.9820

Table 2- Performance Comparison in Different Configuration on STARE

	ANN 8-4-2	*ACC*	*DICE*	*SEN*	*SPE*
STARE	Quantization	0.9495	0.5870	0.5325	0.9886
	Quantization and error compensation	0.9504	0.6602	0.6171	0.9810
	Accurate	0.9573	0.7088	0.6734	0.9830
	ANN 10-6-2	*ACC*	*DICE*	*SEN*	*SPE*
	Quantization	0.9394	0.6702	0.7335	0.9574
	Quantization and error compensation	0.9440	0.6872	0.7538	0.9608
	Accurate	0.9515	0.6975	0.7060	0.9735
	ANN 20-6-2	*ACC*	*DICE*	*SEN*	*SPE*
	Quantization	0.9199	0.6225	0.7192	0.9377
	Quantization and error compensation	0.9436	0.6853	0.7446	0.9610
	Accurate	0.9432	0.6913	0.7632	0.9589

4.2. Performance Comparison

The network with 10-6-2 configuration is considered as the best one for both two datasets. Results of 10-6-2 configuration for DRIVE and STARE are illustrated in Table 1 and Table 2 respectively. These results are for three cases of accurate, quantization without error compensation and with error compensation. It is observed that the loss of accuracy due to quantization with error compensation is negligible. In Fig. 8 and Fig. 9, the visual results of these networks on DRVE and STARE are shown respectively. The simulation results are for the cases of b) after quantization without error compensation, and c) with error compensation, d) without network quantization and e) ground truth. These results show that the visual quality degradation due to quantization with error compensation is acceptable. Also visual

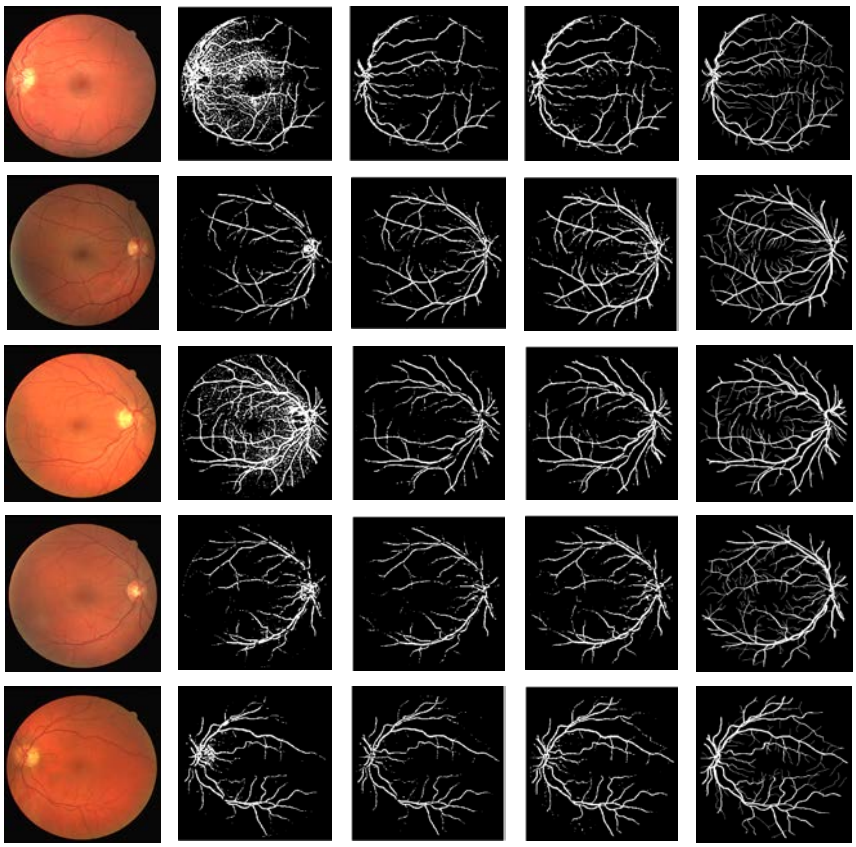


Fig. 8 - Visual results of ANN on DRIVE database.

results show that the quantization without any compensation creates bad effects on segmentation results.

Also performance of the proposed 10-6-2 network architecture with error compensation is compared to related work in Table 3 for the DRIVE and STARE image dataset. In [20, 22, 24, 26] retinal vessel segmentation is studied considering hardware acceleration techniques. With respect to the software accuracy, the presented method may not be the best approach. However with comparison to the computational complexity of the different methods, the proposed method is simplified which is easy to hardware implementation.

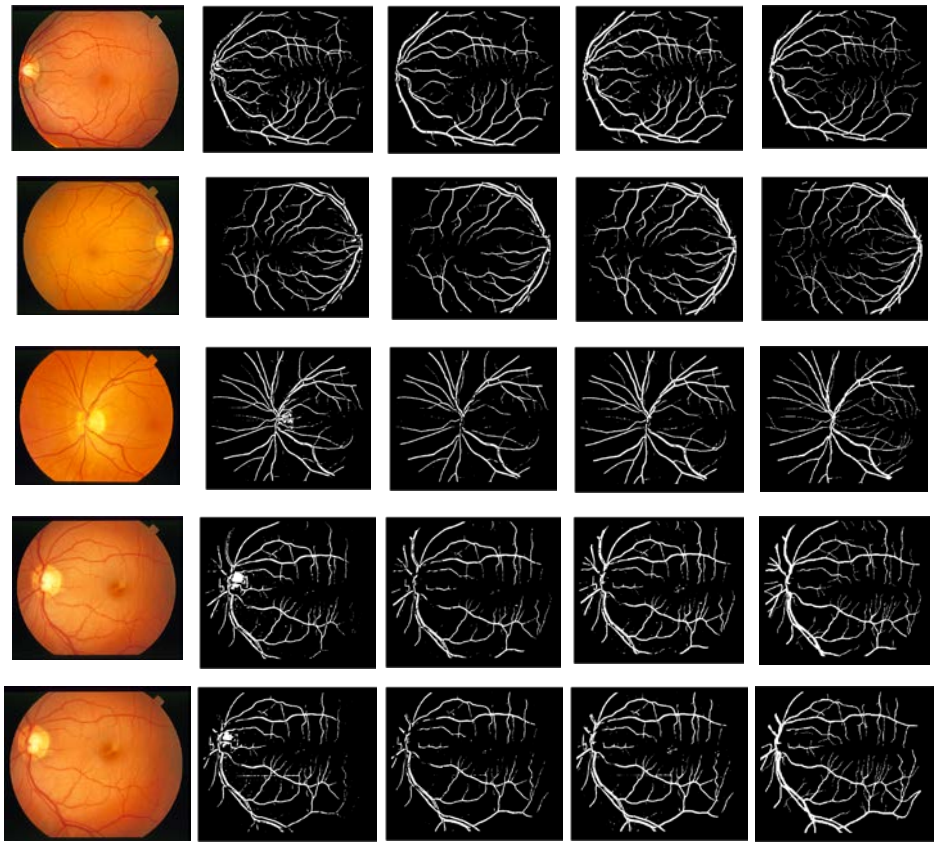


Fig. 9 - Visual results of ANN on STARE database.

Table 3- Performance Comparison ANN 10-6-2

	DRIVE			STARE		
Method	SEN	ESP	ACC	SEN	ESP	ACC
[24] F. Arguello, et al.: (2014)	0.7209	0.9758	0.9431	0.8951	0.9384	9448
[22] D. Koukounis, et al.: (2013)	--	--	0.9019	--	--	0,9009
[26] D. Koukounis, et al.: (2014)	--	--	0.9240	--	--	0.9240
[20] G. Costantini, et al. (2010)	--	--	0.9261	--	--	--
[4] MM. Fraz, et al. (2012)	0.7406	0.9807	0.9480	0.7548	0.7548	0.9534
[5] M. Melinscak, et al. (2015)	0.7276	--	0.9466	--	--	--
[14] H. Fu, et al. (2016)	0.7603	--	0.9523	0.7412	--	0.9585
[8] LC. Neto, et al.: (2017)	0.7806	0.9629	--	0.8344	0.9443	---
[9] S. Roychowdhury, et al.: (2015)	0.7390	0.9780	0.9490	0.7320	0.9840	0.9560
[10] YQ. Zhao, et al.: (2014)	0.7354	0.9789	0.9477	0.7187	0.9767	0.9509
[11] Y. Wang, et al.: (2013)	--	--	0.9461	--	--	0.9521
[12] I. Lázár, et al.: (2015)			0.9458			0.9492
[13] G. Azzopardi, et al.: (2015)	0.7655	0.9704	0.9442	0.7716	0.9701	0.9497
Proposed	0.8319	0.9545	0.9421	0.7538	0.9608	0.9440

4.3. Complexity Analysis

In this section, simplified network structure of segmentation system on DRIVE is analyzed from the complexity point of view. Multiplication of weights in the input stage and computation of activation function impose most computational complexity. In this paper network weights which are in the interval of -3 to 3 are quantized into 48 intervals with 0.125 steps. Weight coefficients are quantized in the form of numbers that their multiplications can be implemented by shifts and additions. By considering a 7×7 image patch as neural network input, weights and required module for computations in the input layer are listed in Table 4.

Since a 10-6-2 structure is used, in the input layer we have 10 neurons and 49 pixels are fed into each neuron. Hence, we have 490 weights at the first layer. These weights are numbers within -3 to 3 interval. To form each weight a number of additions and shifts are required. Considering the proposed configuration of the network structure, the required number of hardware modules for a common implementation is very high. For example as it is represented in Table 4, sum of all required add modules are 207, while for non-quantized implementation we need 490 floating point multipliers. An analysis of other input weights in other layers can be used to determine all of the required hardware modules in other layers.

Table 4- Weights in the first layer of ANN

Weight coefficients	Number of coefficients	Weight coefficients	Number of coefficients	Number of additions	Total # of adders
0	57	-	-	0	0
0.125	41	-0.125	60	0	0
0.25	35	-0.25	43	0	0
0.375	22	-0.375	36	1	58
0.5	27	-0.5	26	0	0
0.625	23	-0.625	16	1	39
0.75	15	-0.75	13	1	28
0.875	8	-0.875	6	2	28
1	8	-1	9	0	0
1.125	7	-1.125	7	1	14
1.25	4	-1.25	4	1	8
1.375	3	-1.375	2	2	10
1.5	3	-1.5	1	1	4
1.625	1	-1.625	3	2	8
1.75	0	-1.75	0	2	4
1.875	0	-1.875	0	3	0
2	2	-2	0	0	0
2.125	1	-2.125	1	1	2
2.25	0	-2.25	1	1	1
2.375	0	-2.375	0	2	0
2.5	0	-2.5	2	1	2
2.625	0	-2.625	0	2	0
2.75	0	-2.75	0	2	0
2.875	0	-2.875	0	3	0
3	0	-3	1	1	1

Here, a 7×7 image block is fed into the input layer. The size of the input layer is a dominant factor affecting the network complexity. In this regard in the proposed ANN architecture it is attempted to design a network with minimum input size. Design summary of all required hardware modules in proposed simplified architecture versus non-simplified (floating point)

are reported in Table 5. As it is illustrated in Table 5, number of arithmetic units and their structures are reduced significantly. It is proved that ANN uses less clock cycle and area when a fixed point format is used versus floating point [38]. Moreover the in the proposed representation by using shift and add integer operation, multiplier modules are removed.

Table 5- Design summary of the proposed 10-6-2 ANN

Layer name		# of multiplier	# of addition	Activation function modules
Layer1	Simplified (Ours)	Non	207 Integer	10 Piecewise Linear
	Non-simplified (Floating-Point)	490 Floating Point	490 Floating Point	10 Non Linear
Layer2	Simplified (Ours)	Non	51 Integer	6 Piecewise Linear
	Non-simplified (Floating-Point)	60 Floating Point	60 Floating Point	6 Non Linear
Layer3	Simplified (Ours)	Non	18 Integer	2 Piecewise Linear
	Non-simplified (Floating-Point)	12 Floating Point	12 Floating Point	2 Non Linear

The computational time of our segmentation method also makes it proper to retinal diagnostic system. On average, it required 1.63 s to detect blood vessels for an image from DRIVE, and 1.92 s for one from STARE. The method was implemented in MATLAB 2016a using a PC configured by an Intel(R) Core(TM) i7-4790 CPU 4.00 GHz and 32GB of RAM. By using a simple neural network with low computational complexity, the segmentation method can be applied in commodity hardware devices.

4.4. FPGA Implementation

For identifying the performance of the proposed method on hardware, an FPGA implementation of segmentation system on DRIVE is performed. Proposed architecture is described in VHDL and is implemented on a XILINX Spartan6 family 6slx75fgg676-3 device. As illustrated in Table 6, low FPGA resources are utilized and an image with size 512×512 can be processed in 49 ms. Hardware simulation results in Table 6, show the proposed hardware ANN structure can be implemented on hardware with acceptable area and speed. . We have also compared our method with another method which has a hardware implementation.

Table 6- Hardware implementation specification of the proposed Method

Method	Device	# of Slice LUTs	# of Bonded IOB	Processing Time
[26] D. Koukounis, et al. (2014)	Xilinx FPGA (6XC65LX150T)	39625 out of 92152	--	52 ms (on 768×584 image size)
Proposed Method	Xilinx FPGA (6slx75fgg676-3)	7981 out of 46648	363	49 ms (on 512×512 image size)

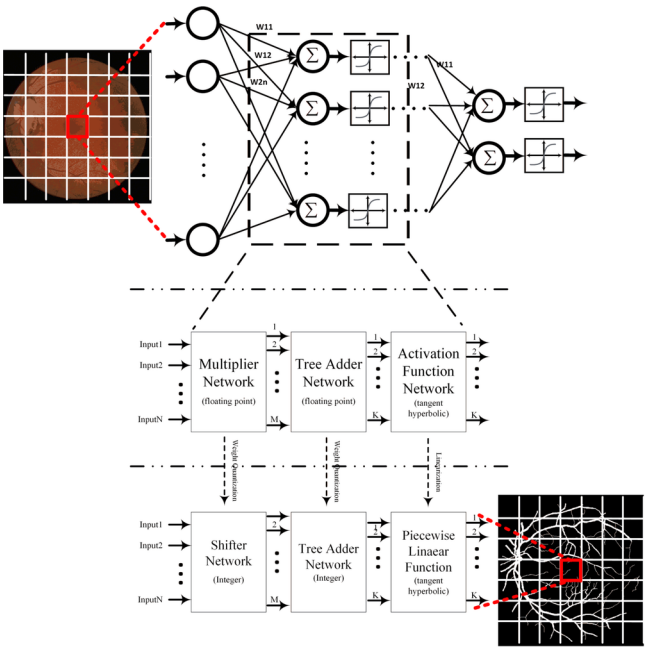
5. CONCLUSION

Real-time retinal vessel segmentation was proposed. The main goal was to make it implementable for medical devices for cases such as intraocular procedures. This could be highly helpful for surgical monitoring systems and portable ophthalmoscopy devices. For segmentation purposes a neural network, as a general learning tool, was designed with a limited number of layers and neurons. All of its parts were simplified for real-time implementation. Activation functions were linearized and weights were quantized. It was observed that using only weight quantization could not lead to an acceptable accuracy, and an error compensation method was hence proposed. Simulation results indicated that the proposed real-time method had simple architecture and loss of accuracy due to simplification was negligible. By using the proposed low complexity structure its hardware architecture can efficiently be embedded in medical instruments.

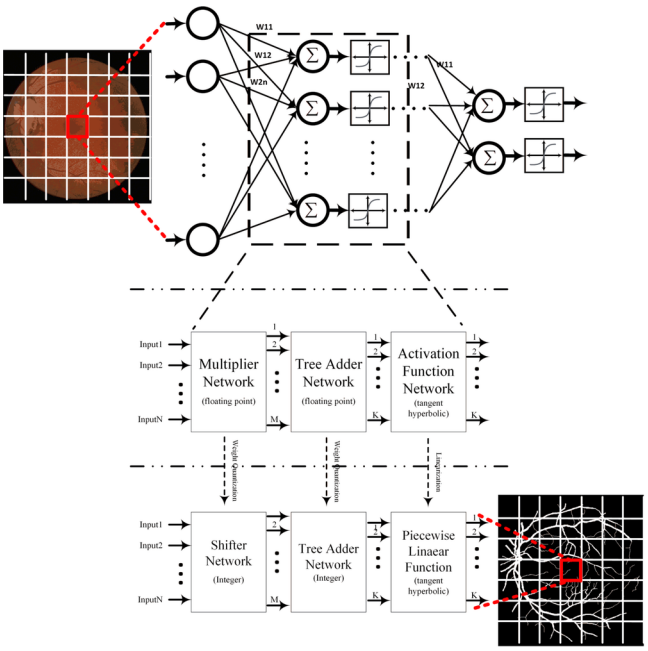
REFERENCES

1. Laser Care & Research Center, available from : http://lasereyecenter.ae/services/diabeticretinopathy/
2. MD. Abràmoff, MK. Garvin, M. Sonka, Retinal imaging and image analysis. IEEE reviews in biomedical engineering. 3 (2010) 169-208.
3. CA. Lupascu, D. Tegolo, E. Trucco, FABC: retinal vessel segmentation using AdaBoost. IEEE Transactions on Information Technology in Biomedicine. 14(5) (2010) 1267-74.
4. MM. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, AR. Rudnicka, CG, Owen, SA. Barman. An ensemble classification-based approach applied to retinal blood vessel segmentation. IEEE Transactions on Biomedical Engineering. 59 (9) (2012) 2538-48.
5. M. Melinscak, P. Prentasic, S. Loncaric. Retinal vessel segmentation using deep neural networks. VISAPP 10th International Conference on Computer Vision Theory and Applications. (2015).
6. BC. Becker, CN. Riviere. Real-time retinal vessel mapping and localization for intraocular surgery. IEEE International Conference on Robotics and Automation (ICRA). (2013) 5360-5365.
7. C. Alonso-Montes, DL. Vilarino, P. Dudek, MG. Penedo. Fast retinal vessel tree extraction: A pixel parallel approach. International Journal of Circuit Theory and Applications. 36 (5-6) (2008) 641-51.
8. LC. Neto, GL. Ramalho, JF. Neto, RM. Veras, FN. Medeiros. An unsupervised coarse-to-fine algorithm for blood vessel segmentation in fundus images. Expert Systems with Applications. 78 (2017) 182-192.
9. S. Roychowdhury, DD. Koozekanani, KK. Parhi. Iterative vessel segmentation of fundus images. IEEE Transactions on Biomedical Engineering. 62.7 (2015) 1738-1749.
10. YQ. Zhao, XH. Wang, XF. Wang, FY. Shih. Retinal vessels segmentation based on level set and region growing. Pattern Recognition 47.7 (2014) 2437-2446.
11. Y. Wang, G. Ji, P. Lin, E. Trucco. Retinal vessel segmentation using multiwavelet kernels and multiscale hierarchical decomposition. Pattern Recognition 46.8 (2013) 2117-2133.
12. I. Lázár, A. Hajdu. Segmentation of retinal vessels by means of directional response vector similarity and region growing. Computers in biology and medicine. 66 (2015) 209-221.
13. G. Azzopardi, N. Strisciuglio, M. Vento, N. Petkov. Trainable COSFIRE filters for vessel delineation with application to retinal images. Medical image analysis. 19.1 (2015) 46-57.
14. H. Fu, Y. Xu, S. Lin, DW. Wong, J. Liu. Deep Vessel: Retinal Vessel Segmentation via Deep Learning and Conditional Random Field. Springer International Conference on Medical Image Computing and Computer-Assisted Intervention. (2016) 132-139.
15. Q. Li, L. Xie, Q. Zhang, S. Qi, P. Liang, H. Zhang, T. Wang. A supervised method using convolutional neural networks for retinal vessel delineation. IEEE International Congress on Image and Signal Processing (CISP). (2015) 418-422.
16. R. Vega, G. Sanchez-Ante, LE. Falcon-Morales, H. Sossa, E. Guevara. Retinal vessel extraction using lattice neural networks with dendritic processing. Computers in biology and medicine. 58 (2015) 20-30.
17. S. Wang, Y. Yin, G. Cao, B. Wei, Y. Zheng, G. Yang. Hierarchical retinal blood vessel segmentation based on feature and ensemble learning. Neurocomputing. 149 (2015) 708-17.
18. C. Alonso-Montes, DL. Vilarino, MG. Penedo. CNN-based automatic retinal vascular tree extraction. IEEE International Workshop on Cellular Neural Networks and Their Applications. (2005) 61-64.
19. R. Perfetti, E. Ricci, D. Casali, G. Costantini. Cellular neural networks with virtual template expansion for

retinal vessel segmentation. IEEE Transactions on Circuits and Systems II: Express Briefs. 54 (2) (2007) 141-5.
20. G. Costantini, D. Casali, M. Todisco. A hardware-implementable system for retinal vessel segmentation. 14th WSEAS International Conference on Computers. (2010) 2.568-573.
21. JJ. Koo, AC. Evans, WJ. Gross. 3-D brain MRI tissue classification on FPGAs. IEEE Transactions on Image Processing. 18 (12) (2009) 2735-46.
22. D. Koukounis, C. Tttofis, T. Theocharides. Hardware acceleration of retinal blood vasculature segmentation. 23rd ACM international conference on Great lakes symposium on VLSI. (2013) 113-118.
23. M. Krause, RM Alles, B. Burgeth, J. Weickert. Fast retinal vessel analysis. Journal of Real-Time Image Processing. 11 (2) (2016) 413-22.
24. F. Arguello, DL. Vilarino, DB Heras, A. Nieto. GPU-based segmentation of retinal blood vessels. Journal of Real-Time Image Processing. (2014) 1-0.
25. A. Nieto, V. Brea, DL. Vilarino, RR. Osorio. Performance analysis of massively parallel embedded hardware architectures for retinal image processing. EURASIP Journal on Image and Video Processing. (1) (2011) 10.
26. D. Koukounis, C. Ttofis, A. Papadopoulos, T. Theocharides. A high performance hardware architecture for portable, low-power retinal vessel segmentation. INTEGRATION, the VLSI journal. 47 (3) (2014) 377-86.
27. A. Nieto, VM. Brea, DL. Vilarino. FPGA-accelerated retinal vessel-tree extraction. IEEE International Conference on Field Programmable Logic and Applications. (2009) 485-488.
28. TR. Savarimuthu, A. Kjær-Nielsen, AS. Sørensen. Real-time medical video processing, enabled by hardware accelerated correlations. Journal of Real-Time Image Processing. 6 (3) (2011) 187-97.
29. Ophthalmic Photographers Society, available from : http://www.opsweb.org/?page=FAequipment
30. Kerala Society of Ophthalmic Surgeons available from : ksos.in/ksosjournal/journalsub/Journal_Article_18_292.pdf
31. U. Lotric, P. Bulic. Applicability of approximate multipliers in hardware neural networks. Neurocomputing. 1;96 (2012) 57-65.
32. Y. Kim, Y. Zhang, P. Li. An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems. IEEE International Conference on Computer-Aided (2013) 130-137.
33. A. Armato, L. Fanucci, EP Scilingo, D. De-Rossi. Low-error digital hardware implementation of artificial neuron activation functions and their derivative. Microprocessors and Microsystems. 35 (6) (2011) 557-67.
34. YD. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:1511 (2015) 06530. 2015.
35. S. Han, H. Mao, WJ. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510 (2015) 00149.
36. DD. Lin, SS. Talathi, VS. Annapureddy. Fixed point quantization of deep convolutional networks. arXiv (2015).
37. P. Gysel, M. Motamedi, S. Ghiasi. Hardware-oriented Approximation of Convolutional Neural Networks. arXiv preprint 1604 (2016) 03168.
38. AW. Savich, M. Moussa, S. Areibi. The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study. IEEE transactions on neural networks. 18(1) (2007) 240-52.
39. Drive. Digital Retinal Images for Vessel Extraction, available from: http://www.isi.uu.nl/Research/Databases/DRIVE/
40. AD. Hoover, V. Kouznetsova, M. Goldbaum. Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Transactions on Medical imaging. 19.3 (2000) 203-210.



CTA_2462_GTOC.tif



CTA_2462_GTOC.tif