Edge Computing Using Embedded Webserver with Mobile Device for Diagnosis and Prediction of Metastasis in Histopathological Images

Anil Johny¹ · K. N. Madhusoodanan¹

Received: 19 August 2021 / Accepted: 1 November 2021
© The Author(s) 2021

Abstract

Diagnosis of different breast cancer stages using histopathology whole slide images is the gold standard in grading the tissue metastasis. Traditional diagnosis involves labor intensive procedures and is prone to human errors. Computer aided diagnosis assists medical experts as a second opinion tool in early detection which prevents further proliferation. Computing facilities have emerged to an extent where algorithms can attain near human accuracy in prediction of diseases, offering better treatment to curb further proliferation. The work introduced in the paper provides an interface in mobile platform, which enables the user to input histopathology image and obtain the prediction results with its class probability through embedded web-server. The trained deep convolutional neural networks model is deployed into a microcomputer-based embedded system after hyper-parameter tuning, offering congruent performance. The implementation results show that the embedded platform with custom-trained CNN model is suitable for medical image classification, as it takes less execution time and mean prediction time. It is also noticed that customized CNN classifier model outperforms pre-trained models when used in embedded platforms for prediction and classification of histopathology images. This work also emphasizes the relevance of portable and flexible embedded device in real time clinical applications.

Keywords Computer aided diagnosis (CAD) · Edge computing · Internet of medical things (IoMT) · Machine learning · Deep learning · FPGA · CNN · System-on-chip (SoC)

1 Introduction

There are millions of embedded devices connected to the internet today and it is increasing day by day and total devices connected will reach nearly 5.3 billion by 2023 [1]. The number of devices connected to internet are huge so that the network handles enormous amount of data every day. Due to rising number of digitally connected devices, internet traffic will reach 20.6 zeta-bytes (ZB) [1]. Out of the massive data generated in chunks, only fraction of them are saved in database permanently as useful information after data mining and analysis. Traditionally, computing depends on remote cloud architecture to meet the intensive need for data processing. Major application fields are surveillance,

traffic analysis, industrial monitoring, mobile data and other embedded devices referred as internet of things (IoT) and single-board computers. IoT devices merely provide raw data for analysis which is further processed in a cloud computing environment for extracting meaningful information and draw inferences between them [2].

Machine learning (ML) and deep learning (DL) are subsets of the broader field artificial intelligence currently where extensive research and developments takes place during the last decade. The model training with dataset for specific application can be done with the help of cloud or any computing device with GPU support to reduce the training time. The model inferences can be done inside embedded platform as the process requires less computing resource. Deploying pre-trained model directly on mobile devices is inexpedient due to considerable storage requirement and memory bandwidth [3]. Machine learning modules can be incorporated in low resource end-devices but deep learning requires computational power and memory. Such applications are computational resource hungry, so they are executed mostly on cloud servers or personal computers with reasonable hardware

✉ Anil Johny
aniljohny@gmail.com

K. N. Madhusoodanan
madhu@cusat.ac.in

¹ Department of Instrumentation, Cochin University of Science And Technology, Cochin, Kerala 682022, India

support. Transferring resources to edge of the network from the core is the feasible solution as a trade-off between limited computation and reduced latency in communication [4]. In Fog layer [5], most IoT data is analyzed near the devices itself, alleviating traffic from the core network by outsourcing a major share of computational task by creating an array of edge computing devices. The potential for edge nodes are overwhelming so that each node can think and act independently with the help of machine learning tool kit, enables machines to learn and act like humans [6]. Edge Machine Learning (Edge ML) [7], a derivative of edge computing, offers machine learning and deep learning algorithms with the capability of processing data locally (at the device-level or connected local servers) thereby replacing solely cloud based applications. Notwithstanding as alternate extension for the cloud, edge-nodes complements computing near the edge of the network.

In situations where processed data need to be available without delay, which may be hindered by internet traffic or server related issues especially in healthcare applications where edge and fog computing plays an important role. Machine learning based applications in healthcare system devising clinical decision support(CDS) [8] demands reduced latency with growing need for smart as well as intelligent devices. Machine learning enables electronic devices to learn from existing data and to use this acquired knowledge to make assessments and take decisions autonomously [9].

It is possible now to perform machine learning directly on embedded devices so that they can be used in the field for end-to-end workflow. This paved the way for processing the data to make intelligent decisions considering current and previous data which can be implemented on embedded system. The Internet of Medical Things (IoMT) uses IoT for medical and healthcare where patient-centered data is closely monitored, collected and analyzed for research [9]. Digitized healthcare system connects available medical resources in healthcare services and supports follow-up procedures including health statistics. IoMT also enables remote health monitoring and emergency notification via mobile devices [10]. Sensor network can be connected to monitor well being of patients ensuring treatment by collecting and analysing information of subject without delay, even though in adverse conditions. IoMT in healthcare plays a crucial role in disease prevention and control by tracking the patients from a remote location, capturing real time data and analysing patient data using statistical tools. This paper proposes a deep learning approach for malignancy detection in histopathology images using limited resources of embedded platform. A task specific custom model is designed to ensure best performance with minimum execution time. Rest of the paper is organized as follows. Section 2 portrays a detailed study on related works. The proposed methodology

is described in Sect. 3. The implementation results and the findings are narrated in Sect. 4. Section 5 draws the conclusion arrived.

2 Related Works

This section outlines an elaborate study and comparison of various works related to the application. Most of the algorithms available for machine learning and deep learning with the help of cloud computing or any other data mining tools available in repository. Virtual assistants with intelligence are software agents which have become part of our life [11]. Convolutional Neural networks (CNNs) for embedded systems are possible only if the hardware is capable of handling huge data with sufficient memory. For the same reason, available literature mostly refers to either general-purpose or custom embedded device with accelerating hardware support [12, 13]. Edge computing has the potential to overcome these limitations by allocating the computation closer to the network edge and data sources especially in real-time applications [14]. In [15] the authors suggest a novel tree-based algorithm, Bonsai, for efficient prediction in IoT device Arduino Uno board with ATmega328P micro-controller operating at 16 MHz with, 2 KB RAM and 32 KB read-only flash. They have shown that, with this algorithm IoT device can maintain prediction accuracy with minimum model size and make predictions without connecting to the cloud. They obtained prediction time in milliseconds on slow micro-controllers. In [16] authors present a software accelerator that enhances deep learning execution on heterogeneous hardware, including mobile devices. In their work using edge mobile devices, an acquired image is preprocessed and segmentation is performed. They have also classified the image on a remote server running pre-trained convolutional neural network (CNN). Tomasz Szydlo [17] presents the concept of machine learning models executing on the embedded devices with constrained resources by generating the source code of machine learning models. Suresh [18] suggest a solution in which machine learning on the embedded system and performed low power transmission through LoRa using k-nearest neighbors algorithm. They also show that edge processing using energy-efficient ARM Cortex-M0+ processor enables extended battery life while conducting field trial on animals. Lane [19] have implemented *DeepX*, a software accelerator for better deep learning execution with less demand in device resources. They suggest that efficient execution is possible with resource control algorithm in heterogeneous processors. The aforementioned algorithm is implemented in wearable devices [20] as well.

Dundar [21] proposes a method based on scheduled routing topology for deep CNN acceleration hardware to utilize maximum computational resources available. They

have optimized it for embedded platform in such a way to transform high-level representation of deep CNN(DCNN) into operation codes to execute applications in a hardware accelerator tested real-time on field programmable gate array(FPGA). Song Han [3] achieved reduction the storage requirement by compressing the network using three level pipeline: pruning combined with trained quantization and Huffman coding without reduction in accuracy. Their method reduced the storage of popular pre-trained models AlexNet and VGG-16 by a factor of 35x and 49x respectively. This method also enables the model to be deployed on embedded platforms. Gupta et al. [22] proposes k-Nearest Neighbor based algorithm for real time and accurate prediction deployed on resource-scarce devices like Arduino UNO with 2kB RAM. The concept of bringing content closer to end-users can be through content delivery/distribution networks (CDNs) which are information centric. Edge computing technology reduces transmission time and bandwidth consumption for applications by placing computing, storage and other resources on the edge of a network [23]. The concept of extending fog-computing to the edge of the network introduced in [24] incorporates intelligence in IoT devices and utilizes fog networking creating FC-IoT platform. Edge computing offers various IoT based applications in patient monitoring when portability and system robustness are a major concerns. For example, the design of wearable devices offers help to elderly or disabled people round the clock and alerts the response team in case of medical emergency [25]. IoT based healthcare system and challenges involved are portrayed in [26]. The authors also highlight the need for interconnecting all the resources for effective and efficient healthcare service in addition to medical assistance offered. They corroborate that smart devices in healthcare can be used to monitor over all health activity of patients in case of emergency situations.

Verma [27] made initial attempts for utilizing ultra-low-energy computational platforms for seizure-detection. In their system-on-chip (SoC) design, the instrumentation and computation blocks detect anomalies by sampling the EEG channel periodically and processing to derive features at every two second duration. Lee [28] incorporated a flexible accelerator together with a general-purpose CPU which enables feature extraction for a range of physiological signals and clinical applications. The accelerator enables energy and memory-usage space and the model adaptations implemented through the hardware supported embedded active learning. Wang [29] proposed embedded ML which learns statistics of the computational outputs creating an error-aware model for analysis through an embedded feature learning framework. Parker [30] suggests Artificial Neural Network capable of learning using Arduino-Pro Mini micro-processors and significant work has been done in hardware ANN implementations, with notable parallelism in neural

networks. Liu [31] developed recognition system for diet related assessment with pervasive mobile devices. They followed a food recognition system with edge computing than traditional mobile cloud computing which causes system latency.

Alippi [32] used Hidden Markov Model to model relationship between two generic data streams through a sequence of linear dynamic model where the coefficients are used as features. Their model automatically detects changes that occurs in data streams obtained from multi-sensor units due to several reasons by comparing it with probabilistic patterns of normal data. Badan [33] implemented CNN in an embedded system with limited resources by neuro-evolution method thereby optimizing CNNs with respect to the classification error and reduced power consumption. Venieris [34] suggest optimized mapping of Deep neural networks on embedded platform with CNN accelerators to achieve high throughput and low latency. In their work, cascaded classifiers with domain specific knowledge enable the deployment of sophisticated deep learning models on edge mobile and embedded systems. Umuroglu [35] developed a framework for building fast and flexible FPGA accelerators (Finn) using a flexible architecture. They implemented on a ZC706 embedded FPGA platform drawing less than 25 W total power for 12.3 million image classifications per second on the MNIST dataset with 95.8% accuracy. Tsimpouras [36] propose a method for efficient deployment of CNNs in low power processor based architectures used as edge devices in IoT networks to analyze utilization of hardware resources of the edge devices. It is then evaluated using different CNNs in the Intel Movidius Myriad2 low power embedded platform. Lee [37] adopted compression methods for downsizing the CNNs to deploy on resource-constrained embedded systems by pruning technique and knowledge distillation. David [38] used TensorFlow Lite Micro, an open-source ML inference framework, for running deep learning models in embedded systems. FPGA based custom device implementations offers high performance than general purpose computing devices [12, 13, 39]. They can be programmed using hardware description language where several programmable blocks are connected to accomplish the logic [40, 41]. Although many FPGA accelerators have demonstrated better performance over generic processors, the accelerator design area is gradually evolving.

Table 1 compares various existing works using Raspberry-PI and machine learning for different applications including the proposed work. Various edge and embedded system based works using machine learning techniques, in different fields are considered for comparison. From the literature review it is clear that malignancy prediction in histopathology images using Raspberry Pi along with deep learning is less less explored.

Table 1 Comparison of related works

Works	Field	Platform	Tools	Source
[42]	Clay prediction	Raspberry Pi	Machine learning	Multispectral images
[43]	Wearable cognitive assistants	Client-server	ANOVA	EEG, heart rate, galvanic skin response
[44]	Soil fertility prediction	Raspberry Pi	Machine learning	Various sensors
[45]	Construction industry	Intel Joule 1.7 GHz quad core	SIFT	High resolution images using Drones
[46]	Augmented reality	Local server	Response delay	Interactive input for gaming
[47, 48]	Human activity recognition	Cloud and edge	Deep learning	Mobile interface
Proposed work	Malignancy prediction	Raspberry Pi	Machine learning	Histopathology images

Table 2 Overview of base model configuration

Model parameters and hyper-parameters with ranges	
Hyper-parameter	Range
Epochs	500
Batch size	32
Learning rate	$10^{-2} - 10^{-4}$
Optimiser	Stochastic Gradient Descent (SGD)
Loss function	Binary cross entropy
Input shape	96×96
Pooling	Max pooling
Activation	ReLU
Dropout rate	0.25

3 Methodology

This section describes the dataset used, configuration of the proposed model and methodology.

3.1 Dataset

The dataset PatchCamelyon (PCam) [49] is used in our work which contains 96×96 pixel color images (patches) annotated by experts with label indicating presence or absence of metastatic tissue. These patches were extracted from histopathology images of lymph node sections encompassing the benchmark classification dataset-PCam.

Out of 40,000 images in the database, 80% are allotted for training and 20% for the validation corroborating an equal number of images from both classes which is paramount in semi-supervised learning also rule of thumb in literature.

3.2 Configuration of Model

The summary of model configuration with different parameters is given in Table 2, used in tuned CNN model for optimized results.

The base CNN model shown in Fig. 1, has three convolutional layers with pooling layers in between and two fully connected dense layers. Each convolutional layer uses a valid padding to reduce the output feature map dimension proportionate to the filter size used. We used non-linear activation function called ReLU [50] (Rectified Linear Units) in all hidden layers.

The first convolution layer uses 30 filters of kernel size 8×8 to learn larger features, next convolution layer has 100 filters with filter size 5×5 and subsequent convolution layer has 100 filters with size of the filter 2×2 . The stride value is set to unity for convolution operations in all the layers by default. The input to the model is 96×96 images from the standard database [49]. The two fully connected layers in the model with sigmoid activation performs classification based on the features extracted by the convolution layers. Each convolution layer acts together with max pooling, batch normalization, and dropout layers so that features extracted are sub-sampled by pooling layers for dimensionality reduction. Max pooling layers have a 3×3 sliding window and stride set to 2, added after convolution layers to down-sample the feature size. Thus the pooling operation scales down the size of the feature map to reduce the overhead of computational complexity. Batch normalization is applied at the intermediate phase of the convolution stage, except fully connected layers without dropout. Normalizing input layers reduces internal covariate shift [51] while training using mini-batch [52] and reduces over-fitting. The normalization of each scalar feature is performed independently to make zero mean and the unity variance. This step is carried out to replace the random weights initialization which results in vanishing or exploding the gradient otherwise. Algorithm 1 explains pseudo-code for the proposed CNN architecture.

Fig. 1 Block diagram of the base model

Algorithm 1: Pseudo code for proposed CNN model

Data: image, epochs, batch size
Result: Classification with prediction
 Initialization;
while $epochs \leq epochs(max)$ **do**
 extract features;
 for each epoch do
 train CNN();
 for each mini-batch do
 // feature extraction:layer-1
 Extract low-level features;
 Perform dimensionality reduction (max-pooling) ;
 // feature extraction:layer-2
 Extract high-level features;
 Perform dimensionality reduction (max-pooling) ;
 // feature extraction:layer-3
 Extract high-level features;
 Perform dimensionality reduction (max-pooling) ;
 // flatten layer
 Feature vector arranged as one dimensional array;
 // classification layer
 Two fully connected layers performs classification;
 end
 Calculate average loss over each epochs in mini-batch ;
 Backpropagation applied to every iteration;
 end
 Output←(predicted class);
end

Table 3 shows the fine details of CNN models used for our experiments. The model number indicates various models considered and memory consumed by each model is given under model size. All the customized models in the table are trained with various cyclic learning rate [53] strategies. Weights of pre-trained models such as MobileNet [54] and MobileNetV2 [55] are also used in the experiment to compare performance of the method.

Each model is deployed into embedded platform which extracts the features and produces predictions and binary class of test images.

The details of Raspberry-Pi(RPi) used for the experiment is narrated in Table 4 along with stand-alone computer (SAC), for performance comparison.

3.3 Embedded Webserver

Communication is established from mobile to the embedded web-server via TCP/IP protocol over ethernet connection. The query image is loaded from the user end which is send for processing at the other end. The computation algorithm calculates the probability based on the prediction algorithm by the saved model and sends the report as response. Using this method, computing at the edge device is utilized, even though resources are limited. Deploying the model directly into the RPi suffers from following drawbacks such as some display device is required to view the output. Also display device is necessarily required to view the contents of the folder and do the selection of image. Raspberry-Pi acts as the web server and the commands are given from mobile. Communication to RPi is possible through *lighttpd* an open-source web server optimized for speed-critical environments. The low memory footprint (compared to other web servers), small CPU load and speed optimizations make *lighttpd*

suitable for servers that are suffering load problems, or for serving static media separately from dynamic content. The python micro-framework (Flask) [56] is used to script the server side since it is fast and reliable than other general purpose scripting language for web development like PHP. Flask is considered as a micro framework because it does not require particular tools or libraries.

3.4 Deploying the Model into Embedded Platform

Embedded platforms hardly support heavy computations as they are designed for special-purpose computing applications but they can be customized for the application at hand. File size of the pre-trained models restricts feasibility in embedded devices which are designed for dedicated applications. Such portable devices can be custom designed for medical applications by loading only key library functions to take the maximum advantage of resources at the same time minimizing the memory footprint. We used the *save_model* and *load_model* option in keras [57] to save trained model as well as its weights in *.h5* format. Once the converted (*saved_model.h5*) file is ported to embedded device, histopathology images from the database are also loaded into the memory for prediction using mobile device. Next, the essential software are loaded into the RPi to support built-in functions. We also have selected only the best performing model to be ported into the embedded platform to facilitate simple and reliable histo-pathological image classification.

Figure 3 shows the steps involved in deployment of the model. The sequence is initiated by executing the python file which launches the window prompting for loading the image, which then the user can redirect to the folder where images are saved. User is then prompted to load the model required to predict the image and the results are displayed in another window along with ground truth label for comparison. The probability of association with the actual class is also indicated for confirming the predicted class. The code executes the saved model from the RPi and displays the result. CNN-based deep learning with custom model is feasible and easy to port and computationally efficient in contrast to state-of-the-art pre-trained model approaches. The model with best performance metrics is deployed into the RPi thereby reducing false positive(FP) instances. When the user loads new image, the model provides the prediction label along with probability of malignancy in the image. Our proposed CNN

Table 3 Model versions with size

Models	Model size(MB)	Model description
Model1	3.2 MB	Customized model
Model2	4 MB	Customized model
Model3	4.1 MB	Customized model
Model4	3.4 MB	Customized model
Mobilenet [55]	13.1 MB	Pre-trained model
MobillenetV2 [54]	9.3 MB	Pre-trained model

Table 4 Configuration of devices

Device	Processor	Core/CPU	Clock frequency	RAM
Raspberry Pi 3 (Model B+)	Broadcom BCM2837B0	Cortex-A53	1.4 GHz	1GB LPDDR2 SDRAM
Raspberry Pi 4 (Model B)	Broadcom BCM2711	Quad core Cortex-A72	1.5 GHz	2GB LPDDR4 SDRAM
Stand-alone computer (SAC)	Intel	Core i3-6100U	2.30 GHz	4GB

Fig. 2 Representation of the proposed method

model has relatively smaller size (less than 5 MB), which makes it easier to be loaded into portable devices. The image prediction and classification tasks were performed by general purpose embedded device – RPi-Model-3B+ and RPi-Model-4B with 1GB and 2GB RAM respectively.

3.5 Embedded Web-Server Framework

The detailed layout of the proposed method is shown in Fig. 2. The user can type the web address in the URL format

which is the reference to web resource to specify location of devices on a computer network. The proposed model comprises of mobile device, RPi as embedded webserver connected via WLAN with optional intermediate FoG nodes.

Medical practitioner can enter the URL in the mobile device and use the webpage that appears on the screen to select the test image and upload into Raspberry-Pi based web-server as request. Through the same webpage on mobile device, a suitable model can also be selected for prediction and classification of the uploaded image. The web-server

Fig. 3 Communication between client and web-server

performs computation using its own resources and provides the result as response. A single router may limit the range of communication between web-server and mobile device, hence we have used a WiFi router to extend the range so that proximity of the nearby devices can be increased. The complete sequence of communication using local network is depicted in Fig. 3. The framework also can be connected to fog node with limited computation as optional facility if the patient data is stored in local server for later analysis.

Client forwards the request to the server. The request is processed at the server and it sends the response to the client. The RPi which is the host acts as the web-server which communicates with the client through wireless LAN (WLAN) via TCP/IP. The input image is fetched from the mobile with any inbuilt browser and forwarded to embedded server for processing. RPi processes the image using saved model(stored in memory), classifies the image and computes the probability of association with the respective class. Operation is completed when the web-server returns the response to the client side and displays prediction probability and image class in the mobile screen. The system can be enhanced such that if the required response from the server can be saved in the cloud server or fog network for later usage. The medical expert can take the picture of histopathological images in mobile device and send to embedded web-server. The image is then resized, processed and the model classifies it, predicts the class and probability of the image being malignant. At the processing end, images are resized to 96×96 so that saved model can analyze/extract the features of image and predict its class. The system-on-chip single board computer and the user end device under the same network is protected by the firewall so that any attack from other network can be prevented. Moreover the latency in decision making can be reduced especially to ascertain the exact cause of disease. The compact size of the trained model easily fits into limited memory of end-device with supporting libraries. The memory footprint is also less and plug-in features help easy replacement and maintenance of the device. The host communication require no additional display as the client side is a mobile device. Memory usage of proposed configuration of device is minimum since optimization is performed at all implementation levels.

The application is designed to communicate with multiple embedded devices at different instances by changing the URL. It can be configured in such a way that, multi-domain expertise facility can also be implemented by selecting corresponding data-set for model training. In this application, we have spared RPi for dedicated usage of each medical expert. This modality increases the throughput as well as reduces latency. The embedded device activates the application at start up without initial settings so that medical practitioner who is not familiar with device settings or technical know-how can also use it hassle-free with this facility.

Another advantage is that, once the device is configured for a particular network, there is no need to reconfigure it while using under the same network for any period of time. Devices within the range defined by sub-net can be connected to web-server with IP address. The application requires less internet support as it completely runs across any device within the intra-net with access permissions. Multi-user configuration affects the performance of the device as it requires allocation and sharing of resources with limited memory. In future, multi-user environment with separate log-in environment can be implemented for simultaneous usage with multi-device support by configuring the existing web-server to open-source HTTP web-server.

4 Experimental Results

This section portrays the implementation results and main observations of our work. Embedded devices with limited computing resources can be used for prediction and classification of histopathology images by tweaking trained deep CNN model into the device. The work is implemented on various CNN models with different learning strategies and number of neurons in each layer as shown in Table 3. CNN models are then deployed into embedded RPi. The test images are uploaded from mobile device to the local storage of embedded device, for classification and prediction. Thus the memory usage of RPi is minimally utilized. The performance of two variants of Raspberry-Pi with different configurations are compared and values are tabulated. It is clearly evident that the overall execution time using proposed custom models is considerably lesser than that of pre-trained models. This is an added advantage when embedded devices are used for field applications.

Table 5 shows the total execution time taken by different system-on-chip embedded devices to process the given set of test images. Various sets of test images from the same data-set are used to ensure consistency of results. The total time for execution in the RPi web-servers are compared with that of stand-alone computers. From the results in Table 5 it is notable that RPi-4B takes minimal execution time for substantial number of test images.

Figure 4 shows average prediction time per image for prediction across different devices on test images in the data set using each model considered in the work. The results in the figure also agrees with the tabulated values in Table 5 showing that Rpi-4B with limited resource takes lesser mean prediction time per test image. Comparison of performance results on different devices portrayed here shows that the mean prediction time taken by RPi-4B with limited resource is comparable to the performance of a general purpose stand-alone computer while using same model.

Snapshots of interface are shown in Fig. 5 which allows the user to load test image, CNN model and ground truth label from datasets in the front-end. The test image is then analyzed by the trained model which is selected by the user. The interface then displays predicted class of the image and probability of malignancy in the test histopathology images. Provision for remarks from experts is also incorporated in the drop-down menu where actual label is displayed. The application works devoid of internet connection that makes seamless access of the device for medical experts. In case of any glitches, the device can be replaced in minimum time as the components are completely modular and customizable.

A set of histopathology images are selected randomly from the data-set and tested. The obtained results are given in Table 6. The ground truth labels, provided along with dataset, indicates benign or malignant class of the image, annotated by experts. Prediction results with RPi-3B+ and RPi-4B along with that of stand-alone computer are included in the table. In the obtained prediction results, 'M' represents malignancy where 'B' stands for benign nature of test image. All correct predictions in a row indicates perfect prediction score which complements quality in disease

diagnosis thereby simulating medical expertise. The prediction results of the embedded devices are compared with prediction results of stand alone computer and ground truth label. This process is equivalent to consulting two or more experts for second opinion in diagnosing the disease. This step can also be performed as standard procedure in practical scenario, to maximize the likelihood of true positive and true negative results thus reducing false negative cases. The mean prediction time for both the embedded devices shown in Fig. 4 shows that the embedded device can perform well with custom trained model for the classification of histopathology images.

5 Conclusion

Bringing computing capabilities to the edge of the network provides advantage in disease diagnosis and assists health-care systems without dependence on cloud services. In this work an efficient deep CNN based classifier using embedded platform is suggested for histopathology images. The baseline model proposed in our work consist of three layer

Table 5 Execution time in different devices for prediction and classification of histopathology images

Execution time (s)							
Device	Total images	Model1	Model2	Model3	Model4	MobileNet	MobileNetV2
RPi 3B+ (webserver)	400	135.87	120.36	125.75	120.42	176.18	169.58
	300	89.9	85.53	88.88	85.07	125.49	121.19
	200	60.44	57.89	60.44	57.64	84.98	83.81
RPi 4B (webserver)	400	66.32	66.91	65.19	66.97	89.97	89.30
	300	49.77	48.10	49.60	47.66	67.71	65.67
	200	35.18	32.42	33.44	32.48	46.43	45.61
SAC ¹	400	17.30	17.23	17.48	17.07	20.21	20.15
	300	16.57	13.96	14.32	13.82	16.47	16.63
	200	11.16	9.82	10.01	9.73	11.47	11.83

¹for comparison

Fig. 4 Average prediction time per image in different computing platforms

Fig. 5 Snapshots displaying the class and prediction results of histopathology images in mobile screen with the URL

Table 6 Prediction results of Raspberry-Pi and stand-alone computer

Image	Actual	RPi-3B+	RPi4	SAC
#1	M	M	M	M
#2	M	M	M	M
#3	B	B	B	B
#4	M	M	M	M
#5	M	M	M	M
#6	M	M	M	M
#7	M	M	M	M
#8	B	M	B	B
#9	M	M	M	M
#10	B	M	B	M

The letters in bold indicate the predictions which are different from the ground truth label

deep CNN trained with different fully connected layers to produce variants of the model. Cyclic learning rate strategies are adopted to improve performance of models as it reduces over-fitting and converges faster with consistent separability. Since Raspberry Pi is one of the modular and portable embedded devices, it is selected for the work. We have configured the device to interface any mobile device so that user can select and load the test pathology image as well as the model for prediction. The prediction results are then displayed on the mobile screen along with actual label and probability of malignancy.

Performance of the work is tested by considering class prediction probability, mean prediction time and over all execution time. We used two different RPi versions (RPi-3B+ & RPi-4B) and compared mean prediction time with stand-alone computer and it is noticed that mean prediction time and overall execution time is lesser in embedded devices. It is observed that the obtained predictions are consistent with ground truth labels. The application we developed is web-based which eliminates the overhead of software installation in the host device and additional infrastructure at the client side. The method is also found to be secure and platform independent revitalizing machine learning assisted clinical decision support. Clinicians with less technical know-how can effortlessly operate the device without snag. Computing with ubiquitous end devices gives additional mileage to healthcare systems in hospitals as the data is collected on-site by deploying anywhere and interpretation can be done locally. Based on the performance results obtained, we clearly state that embedded devices with custom trained CNN model is well suited for applications to classify benign and malignant histopathology images. The future research direction is presented by processing large-scale whole slide image data in end-devices with multi-user environment.

Author Contributions First author has implemented the work explained in the paper under the guidance of second author.

Funding Information Not applicable

Availability of Data and Materials PatchCamelyon (PCam) is the benchmark database considered for the proposed work. Data and the source code are publicly available at <https://github.com/ajgitfit/edge.git>

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper. Cisco <https://www.cisco.com/c/en/us/>

- [solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html](https://solutions.collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html)
2. Li, W., et al.: A Comprehensive survey on machine learning-based big data analytics for IoT-enabled smart healthcare system. *Mob. Netw. Appl.* **26**(1), 234–252 (2021). <https://doi.org/10.1007/s11036-020-01700-6>. <http://link.springer.com/10.1007/s11036-020-01700-6>
 3. Han, S., et al.: Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding (2016). [arXiv:1510.00149](https://arxiv.org/abs/1510.00149)
 4. De Donno, M., et al.: Foundations and evolution of modern computing paradigms: cloud, IoT, edge, and fog. *IEEE Access* **7**, 150936–150948 (2019). <https://doi.org/10.1109/ACCESS.2019.2947652>. <https://ieeexplore.ieee.org/document/8869772/>.
 5. Verma, P., Sood, S.K.: A comprehensive framework for student stress monitoring in fog-cloud IoT environment: m-health perspective. *Med. Biological. Eng. Comput.* **57**(1), 231–244 (2018). <https://doi.org/10.1007/s11517-018-1877-1>
 6. Kavitha, B., Vallikannu, R., et al.: Delay-aware concurrent data management method for iot collaborative mobile edge computing environment. *Microprocess. Microsyst.* **74**, 103021 (2020). <https://doi.org/10.1016/j.micpro.2020.103021>. <https://www.sciencedirect.com/science/article/pii/S0141933119305666>
 7. Dennis, K., et al.: Edgml: Machine learning for resource-constrained edge devices. <https://github.com/Microsoft/EdgeML>
 8. Lyell, D., Coiera, E., Chen, J., Shah, P., Magrabi, F.: How machine learning is embedded to support clinician decision making: an analysis of fda-approved medical devices. *BMJ Health Care Inform.* **28**(1) (2021). <https://doi.org/10.1136/bmjhci-2020-100301>. <https://informatics.bmj.com/content/28/1/e100301>, <https://arxiv.org/abs/>, <https://informatics.bmj.com/content/28/1/e100301.full.pdf>, <https://informatics.bmj.com/content/28/1/e100301.full.pdf>
 9. Azimi, I., et al.: Hich: Hierarchical fog-assisted computing architecture for healthcare iot. *ACM Trans. Embed. Comput. Syst.* **16**(5s) (2017). <https://doi.org/10.1145/3126501>
 10. Rahmani, A.M., et al.: Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach. *Fut. Gen. Comput. Syst.* **78**, 641–658 (2018). <https://doi.org/10.1016/j.future.2017.02.014>
 11. Huang, C.W., Maas, R., Mallidi, S.H., Hoffmeister, B.: A study for improving device-directed speech detection toward frictionless human-machine interaction. In *INTERSPEECH*, pp. 3342–3346 (2019, September)
 12. Park, S. S., et al.: Implementation of a cnn accelerator on an embedded soc platform using sdsoc, ICDSP 2018, 161–165 (Association for Computing Machinery, New York, NY, USA, 2018). <https://doi.org/10.1145/3193025.3193041>
 13. Zhang, C., et al.: Optimizing FPGA-based accelerator design for deep convolutional neural networks (ACM, 2015). <https://doi.org/10.1145/2684746.2689060>
 14. Zhang, X., et al.: Openai: an open framework for edge intelligence (2019). [arXiv:1906.01864](https://arxiv.org/abs/1906.01864)
 15. Kumar, A., Precup, D., editorTeh, Y. W., et al.: Resource-efficient machine learning in 2 KB RAM for the internet of things. (eds Precup, D. & Teh, Y. W.) *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, 1935–1944 (PMLR, 2017). <http://proceedings.mlr.press/v70/kumar17a.html>
 16. Lane, N.D., et al.: Squeezing deep learning into mobile and embedded devices. *IEEE Pervas. Comput.* **16**(3), 82–88 (2017). <https://doi.org/10.1109/MPRV.2017.2940968>
 17. Szydło, T., Senderek, J., et al.: In: Enabling machine learning on resource constrained devices by source code generation of the learned models 682–694. (publisherSpringer International Publishing, 2018). https://doi.org/10.1007/978-3-319-93701-4_54
 18. Suresh, V.M., Sidhu, R., Karkare, P., Patil, A., Lei, Z., Basu, A.: Powering the IoT through embedded machine learning and LoRa. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 349–354. IEEE (2018, February)
 19. Lane, N.D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L., Kawsar, F.: DeepX: A software accelerator for low-power deep learning inference on mobile devices. In: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 1–12. IEEE (2016, April)
 20. Lane, N.D., et al.: Demo: Accelerated deep learning inference for embedded and wearable devices using deepX, MobiSys '16 Companion, 109 (Association for Computing Machinery, address New York, NY, USA, 2016). <https://doi.org/10.1145/2938559.2949718>
 21. Dunder, A., Jin, J., Martini, B., Culurciello, E.: Embedded streaming deep neural networks accelerator with applications. *IEEE Trans. Neural Netw. Learn. Syst.* **28**(7), 1572–1583 (2017). <https://doi.org/10.1109/TNNLS.2016.2545298>
 22. Gupta, C., Precup, D., Teh, Y. W. (eds) et al.: ProtoNN: Compressed and accurate kNN for resource-scarce devices. (eds Precup, D. & Teh, Y. W.) *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, 1331–1340 (PMLR, 2017). <http://proceedings.mlr.press/v70/gupta17a.html>
 23. Merenda, M., Porcaro, C., Iero, D.: Edge machine learning for ai-enabled IoT devices: a review. *Sensors* **20**(9), 2533 (2020). <https://doi.org/10.3390/s20092533>. <https://www.mdpi.com/1424-8220/20/9/2533>
 24. Alippi, C., et al.: A cloud to the ground: the new frontier of intelligent and autonomous networks of things. *IEEE Commun. Mag.* **54**(12), 14–20 (2016). <https://doi.org/10.1109/MCOM.2016.1600541CM>
 25. Bohr, A., Memarzadeh, K.: In: The rise of artificial intelligence in healthcare applications, 25–60 (publisher Elsevier, 2020). <https://doi.org/10.1016/b978-0-12-818438-7.00002-2>
 26. Birje, M.N., et al.: Internet of things based distributed healthcare systems: a review. *J. Data Inform. Manag.* **2**(3), 149–165 (2020). <https://doi.org/10.1007/s42488-020-00027-x>
 27. Verma, N., Lee, K.H., Shueb, A.: Data-driven approaches for computation in intelligent biomedical devices: a case study of eeg monitoring for chronic seizure detection. *J. Low Power Electron. Appl.* **1**(1), 150–174 (2011). <https://doi.org/10.3390/jlpea1010150>. <https://www.mdpi.com/2079-9268/1/1/150>
 28. Lee, K.H., Verma, N.: A 1.2–0.55 V general-purpose biomedical processor with configurable machine-learning accelerators for high-order, patient-adaptive monitoring. In: 2012 Proceedings of the ESSCIRC (ESSCIRC), pp. 285–288. IEEE (2012, September)
 29. Wang, Z., Lee, K.H., Verma, N.: Overcoming computational errors in sensing platforms through embedded machine-learning kernels. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **23**(8), 1459–1470 (2015). <https://doi.org/10.1109/TVLSI.2014.2342153>
 30. Parker, G., Khan, M.: Distributed neural network: dynamic learning via backpropagation with hardware neurons using arduino chips. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 206–212. IEEE (2016, July)
 31. Liu, C., et al.: A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Trans. Serv. Comput.* **11**, 249–261 (2018)
 32. Alippi, C., Ntalampiras, S., Roveri, M.: An HMM-based change detection method for intelligent embedded sensors. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2012, June)
 33. Badan, F., Sekanina, L.: Optimizing convolutional neural networks for embedded systems by means of neuroevolution. *Lect. Not. Comput. Sci.* 109–121 (2019). https://doi.org/10.1007/978-3-030-34500-6_7

34. Venieris, S.I., Kouris, A., Bouganis, C.S.: Deploying deep neural networks in the embedded space (2018). arXiv preprint [arXiv:1806.08616](https://arxiv.org/abs/1806.08616)
35. Umuroglu, Y., et al.: Finn. Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (2017). <https://doi.org/10.1145/3020078.3021744>
36. Tsimpouras, F., et al.: A design space exploration framework for convolutional neural networks implemented on edge devices. *IEEE Trans. Comput. Aid. Design Integr. Circ. Syst.* **37**(11), 2212–2221 (2018). <https://doi.org/10.1109/tcad.2018.2857280>
37. Lee, J., Lee, S.: Robust cnn compression framework for security-sensitive embedded systems. *Applied Sciences* **11**(3) (2021). <https://doi.org/10.3390/app11031093>. <https://www.mdpi.com/2076-3417/11/3/1093>.
38. David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., ... Rhodes, R.: TensorFlow Lite Micro: embedded machine learning for tinyml systems. In: Proceedings of Machine Learning and Systems, 3 (2021)
39. Xie, W., et al.: An energy-efficient FPGA-based embedded system for CNN application (publisherIEEE, 2018). <https://doi.org/10.1109/edssc.2018.8487057>
40. Qiu, J., et al.: Going deeper with embedded fpga platform for convolutional neural network, *FPGA '16*, 26–35 (Association for Computing Machinery, New York, NY, USA, 2016). <https://doi.org/10.1145/2847263.2847265>
41. Liu, Z., et al.: Automatic code generation of convolutional neural networks in fpga implementation. In: 2016 International Conference on Field-Programmable Technology (FPT), 61–68 (2016)
42. Helfer, G.A., et al.: Multispectral cameras and machine learning integrated into portable devices as clay prediction technology. *Journal of Sensor and Actuator Networks* **10** (3) (2021). <https://doi.org/10.3390/jsan10030040>. <https://www.mdpi.com/2224-2708/10/3/40>.
43. Olguín Muñoz, M., et al.: Impact of delayed response on wearable cognitive assistance. *PLOS One* **16**(3), e0248690 (2021). <https://doi.org/10.1371/journal.pone.0248690>
44. Helfer, G.A., Victória Barbosa, J. L., dos Santos, R., da Costa, A. B.: A computational model for soil fertility prediction in ubiquitous agriculture. *Comput. Electron. Agric.* **175**, 105602 (2020). <https://doi.org/10.1016/j.compag.2020.105602>. <https://www.sciencedirect.com/science/article/pii/S0168169920304853>.
45. George, S., et al.: Towards drone-sourced live video analytics for the construction industry, *HotMobile '19*, 3–8 (Association for Computing Machinery, New York, NY, USA, 2019). <https://doi.org/10.1145/3301293.3302365>
46. Premasankar, G., et al.: Edge computing for the internet of things: a case study. *IEEE Internet Things J.* **5**(2), 1275–1284 (2018). DOI:10.1109/JIOT.2018.2805263. <https://ieeexplore.ieee.org/document/8289317/>
47. Ghosh, A., Grolinger, K.: Edge-cloud computing for IoT data analytics: embedding intelligence in the Edge with deep learning. *IEEE Trans. Ind. Inform.* 1–1 (2020). <https://doi.org/10.1109/TII.2020.3008711>. <https://ieeexplore.ieee.org/document/9139356/>
48. Ghosh, A.M., Grolinger, K.: Deep learning: edge-cloud data analytics for IoT, 1–7 (publisherIEEE, addressEdmonton, AB, Canada, 2019). <https://ieeexplore.ieee.org/document/8861806/>
49. Veeling, B.S., et al.: Rotation equivariant CNNs for digital pathology (2018). [arXiv:1806.03962](https://arxiv.org/abs/1806.03962) [cs.CV]
50. Agarap, A.F.: Deep learning using rectified linear units (relu) (2018) arXiv preprint [arXiv:1803.08375](https://arxiv.org/abs/1803.08375).
51. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015, June)
52. Ioffe, S.: Batch renormalization: towards reducing minibatch dependence in batch-normalized models (2017). [arXiv:1702.03275](https://arxiv.org/abs/1702.03275)
53. Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 464–472 (2017)
54. Sandler, M., et al.: Mobilenetv2: Inverted residuals and linear bottlenecks (2019). [arXiv:1801.04381](https://arxiv.org/abs/1801.04381)
55. Howard, A. G., et al.: Mobilenets: efficient convolutional neural networks for mobile vision applications (2017). [arXiv:1704.04861](https://arxiv.org/abs/1704.04861)
56. Grinberg, M.: Flask web development: developing web applications with python (publisher“O’Reilly Media, Inc.”, 2018)
57. Chollet, F.: Keras. GitHub (2015). Retrieved from <https://github.com/fchollet/keras>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.