Tumour Detection using Convolutional Neural Network on a Lightweight Multi-core Device

T. Hui Teo
Science & Math Cluster / Engineering Product Development
Singapore University of Technology & Design
Singapore
e-mail: tthui@sutd.edu.sg

Wei Ming Tan, and Yi Shu Tan
Engineering Product Development
Singapore University of Technology & Design
Singapore
e-mail: dylan_tan@mymail.sutd.edu.sg, yishu_tan@mymail.sutd.edu.sg

***Abstract*—Convolutional neural networks (CNN) have been the main driving force behind image classification and it is widely used. Large amounts of processing power and computation complexity is required to mimic our human brain as in the image classification. Such complexity may result in large bulky systems. A lack of such, while possible, may result in a rather limited use case and as such constrained functional implementation. One solution is to explore the use of Multicore System on Chips (MCSoC). CNN, however, were commonly built on Graphics Processing Units (GPUs) based machine. In this paper, we reduce the overall size of a CNN while retaining a satisfactory level of accuracy so that it is better suited to be deployed in an MCSoC environment. We trained a CNN model that was validated on detecting malignant tumor cells. The results show significant boost in functionality in the form of faster inference times and smaller model parameter sizes, deploying neural networks in an environment that would have otherwise seemed less practical. Efficient inference networks on lightweight systems can serve as an inexpensive and physically small alternative to existing Artificial Intelligence (AI) systems that are generally costly, bulky and power hungry.**

Keywords- Artificial Intelligence, Convolutional Neural Network, Deep Learning, Machine Learning, Multicore System on Chip

I. INTRODUCTION

Noncommunicable diseases (NCDs) is responsible for the majority (about 71%) of global deaths today. Of NCDs, cancer ranks as the leading cause of death, and is the largest hurdle to improving life expectancy in the 21st century. Female breast cancer comes in at an equally alarming second. It is a highly prominent form of cancer and incidence rates in women far exceeds other cancers in being the most commonly diagnosed, and the leading cause of female cancer deaths [1], [2]. It is prevalent in a woman's lifetime, and the statistics reflects the seriousness of the problem. Regular checks are important, and medical organizations recommend women aged 45 to 54 to get mammograms every year. Women 55 and older should switch to mammograms every 2 years or may continue with yearly screening. Going for mammograms early to identify potentially malignant cancer cells is key to receiving appropriate treatment as soon as possible. Improving the system of identifying cancer cell types could, at a larger level, greatly improve the efficiency in medical industries and in turn reduce time-to-treatment of a potential patient, hence lowering mortality rates. Training a Artificial Neural Network (ANN) to assist in identifying the cancer cells could potentially have many advantages. Given a high and trustworthy level of accuracy, a ANN could help identify cancerous cells almost autonomously. This work focuses on the development of real-time tumour detector using CNN technique for possible handheld medical appliance that one can purchase and used in household environment prior approaching medical center.

CNN has been widely explored and deployed in medical applications. Here in [3], presented a set of experiments to avoid using hand crafted methods and to use CNNs such as the early existing AlexNet [4] that have been designed to identify RGB images. AlexNet had been repurposed to be used to identify the histopathological images and have shown several different methods of handling the network such that training the ANN on a different type of image set and resolution could still yield positive results. It should be noted that their experiments on the BreaKHis [5] dataset showed improved accuracy in classification as compared to traditional methods. However, in applications that have suboptimal physical and/or environmental conditions such as real-time handheld devices, deployment of CNN is challenging especially the electricity power consumption and the physical size. This situation would look ever bleaker for applications that require close to real-time latency. This also extends to areas where network data and internet access may not be easily available or consistent. There is, hence, a need to implement a reliable and functional CNN on a lightweight system and it is in such cases that we would like to quantify the performance of smaller systems.

In recent years, MCSoC have skyrocketed in its power and efficiency. Smaller chips with many cores have been implemented into areas that are able to fully or better utilize its strengths. From multicore chips that exists as powerful processors to Vision Processing Units (VPU) and even Field Programmable Gate Arrays (FPGA), [6]. With better hardware, we can push equally complex workloads onto these chips and achieve levels of satisfaction previously unattainable. Hence able to deploy and implement AI systems into more constrained environments, [7]. Methods such as parameter quantization or network pruning as

978-1-7281-4882-3/19/$31.00 ©2019 IEEE
DOI 10.1109/MCSoC.2019.00020

described in [8] are some common solutions to reduce the complexity of a CNN. This may result in shorter training times, possibly smaller storage sizes or more efficient computation. Such methods are commonly used to help CNN to have shorter inference time with acceptable losses in accuracy.

In this paper, we first demonstrated using existing CNN to classify images from the Breast Cancer Histopathological Image Classification (BreaKHis) dataset that consists of X-ray mammograms. The database has a set that includes malignant and benign tumors in X-ray mammogram pictures labelled appropriately. This existing model was modified to reduce the complexity through Machine Leaning (ML) technique. The model was then deployed onto a chosen low-power MCSoC. The performance of the model on the BreakHis dataset was compared with the existing model.

The paper is organized as follows: Section II introduces the BreaKHis database and the implementation of Deep Learning (DL) using CNN on a lightweight MCSoC. Section III shows the experiments conducted and the process that was taken to reach their accompanying results. Finally, Section IV provides the conclusion based on the results.

II. BreaKHis Database and Implementation of CNN

The BreaKHis composes of 9109 microscopic images of breast tumor tissue collected from 82 patients using different magnifying factors (40X, 100X, 200X, and 400X). To date, it contains 2,480 benign and 5429 malignant samples that were collected from January to December 2014. This database is built in collaboration with P&D Laboratory, Brazil.

TABLE I. Image Distribution of BreaKHis Dataset

Magnification	Benign	Malignant	Total
40X	652	1370	1995
100X	644	1437	2081
200X	623	1390	2013
400X	588	1232	1820
Total no of Images	2480	5429	7909

TABLE I is distributed by the magnification factor and class. The dataset of BreaKHis is divided into 2 main groups: benign tumors and malignant tumors. Histologically benign is a term referring to a lesion that does not match any criteria of malignancy and remains localized. Malignant tumor are lesions that can invade and destroy adjacent structures and cells. The samples were retrieved from surgical biopsy that were then prepared and preserved through the standard clinical process of using paraffin to try as best to maintain the original tissue and structure. These samples that were collected from the patients were then, in short, mounted onto slides and viewed under a microscope. The slides used Hematoxylin and Eosin (HE) to colour the cells, hence improving visibility. After which trained pathologists would visually go through and analyze the tissues, confirming and identifying each sample into a labelled image.

CNN is a class of DL that consists of multiple layers stacked on top of each other, followed by a supervised classifier. A supervised classifier is where the data comes with labels, and the neural network with its ground truth back propagation, can learn from the labels and train its multilayer architecture. This bypasses tougher hand-engineered methods that may prove to be more difficult for such geometrically complex images as mentioned in [9]. These advantages arise because CNN does not use distributed representations but instead, depend on the underlying data-generating distribution having an appropriate componential structure [10]. This not only allows the model to be able to generalize extremely well to new data, but it also makes it easier to predict accurately since the hidden layers allow the model to learn from its inputs in different ways, ways more abstract than otherwise possible.

There are three main types of layers in CNN architectures: a filter bank layer, a non-linearity layer and a feature pooling layer [10]. The filter bank layer, otherwise known as the convolutional layer computes output of neurons, that are connected to the input, each one calculating a dot product between their weights. The set of weights which is convolved with the input is called the filter or kernel. Each filter detects a particular feature at every location on the input. For inputs such as images, typical images are small areas (e.g. 3 X 3, 5 X 5) and each neuron is connected only through this filter. Traditionally, the non-linearity layer consists of tanh() sigmoid function or the logistic sigmoid function, as described in [10]. However, in recent times, the Rectified Linear Unit (ReLU) have seen a huge spike in popularity especially due to its seemingly positive results and computational advantages. A pooling layer is used to reduce the resolution of an output feature map. It is a robust method of decreasing the size of an input while keeping the important features and removing minor variations at the feature locations of the input. This reduces the number of parameters and computations required by the network, making it more computationally feasible. The max pooling layer, for example, applies a window function to the input, and computes the maximum in the surroundings. A fully-connected layer is where neurons have full connections to all activations in the previous layer. It is usually the last fully-connected layer that holds the network output.

In the effort of reducing the complexity of CNN model, number of convolutional layers must be minimized, while keeping ReLU, and max pooling. The details of implementation of the CNN model is discussed in the following paragraph.

A. Training and Deployment of Model

To classify the images in the BreaKHis dataset, we first use an existing network such as AlexNet, [4]. AlexNet was first trained on the BreakHis dataset and gave similar accuracy to the ImageNet [11] dataset at 84.7% in our initial work. AlexNet has approximately 62 million parameters and 1.5 billion flops per forward pass. While 1.5 billion might seem large, it is considered lighter in comparison to another popular network, ResNet152 [12]. We need to modify the existing AlexNet for the BreakHis dataset for deployment on hardware platform.

We reduced the AlexNet's 5 convolution layers and 3 fully connected layers to 3 convolution layers and 2 fully connected layers respectively. While accuracies certainly dipped from the result of untuned parameters, we proposed a simple method of fine tuning the parameters of the new network. With the network trained, the dataset undergoes multivariable linear regression (MLR). MLR is a statistical method that uses least squares to predict the outcome of a variable response. In the case of our application, we used the dataset of twenty trained networks by varying combinations of the 18 input variables to learn the relationship between them. The equation can be seen below at (1) where y is the predicted output of the linear weighted addition of all 18 input variables:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_i x_i + \epsilon \quad (1)$$

With MLR, it is now possible to project a predicted accuracy given our network's 18 variables. TABLE II reflects a portion of the data collected. Four different networks are shown with 18 input variables x_1 to x_{18}. These variables range from input dimensions, to convolution filter size and to number of neurons in the fully connected layer. The accuracy of the network is also recorded for network selection.

TABLE II. SUMMARY OF RECORDED VARIABLES AND OUTPUTS

Net	x_1	x_2	x_3	x_4	x_5	x_6	…	x_{18}	y
A	64	64	3	1	2	2	…	3	0.866
B	64	64	3	1	2	2	…	3	0.845
C	64	64	3	1	3	3	…	3	0.862
D	32	32	3	1	3	3	…	3	0.134

Net: Network

Equation (1) can be used to iterate quickly through varying parameters. A simple iteration through a constrained combination of varying input parameters will give us a list of desirable network configurations. Some output constrains are fixed such as a minimum accuracy, maximum floating-point operations (FLOP) in a forward pass and maximum parameters. The number of parameters and FLOP are calculated based on input parameters through the different layers of our entire CNN. These iterated combinations of constrains output a generated set of parameters that predicts a satisfactory accuracy. Parameters and FLOP were then narrowed down to a lower number. Being deployed on a lightweight network, it is essential that we reduce parameter and FLOP to lower the stress on the system resource and improve its efficiency. Accuracy is limited to a tolerance of 5% from AlexNet. This is to ensure that performance is not forfeit. It is worth taking note that all training thus far was done on a GPU. TABLE III summarizes another four networks' performance and the predicted output. It is from this output that we select the network that best suits our requirements.

TABLE III. SUMMARY OF NETWORKS' PERFORMANCE

Net	x_1	x_2	x_3	x_4	…	Param	FLOP	y
E	32	32	3	1	…	0.15 M	3.62 M	0.850
F	32	32	3	1	…	0.21 M	3.69 M	0.889
G	32	32	5	2	…	0.46 M	3.67 M	0.813
H	32	32	5	2	…	0.15 M	3.36 M	0.804

Net: Network, Param: Parameter

The BreaKHis dataset was divided into two divisions. 80% of the images in the dataset were set aside for training and the remaining balance were left for testing the model. This was consistent throughout the entirety of this work for any training and testing done on each iteration of a neural net datapoint generation.

B. CNN Architecture

The generated output consists of a long list of parameters. We selected the network that yielded the highest score from a normalized average of lowest parameter, FLOP and highest accuracy. The chosen network contains the following layers and parameters:

- Input layer: This layer will load the input (in our case, images). Inputs will go through some data manipulation such as mean-subtraction, or feature scaling. In the case of our input dataset, input image size is (700 x 460). The parameters for the image dimensions are (32 x 32) pixels and 3 channels of RGB. The output will be fed to the Convolution layer.
- Reshape layer: Images input into this layer are flattened. Flattening the image will take the data array and string it into a 1-dimensional vector. In the case of our experiment, images of original size are flattened and reshaped to (1 x 3072).
- Convolutional layer: This layer will convolve the input image data with a set of weights that need to be trained. Each convolution of a filter weight will produce one feature map. In this model, three convolutional layers are used. The kernels are of up to sizes of (3 x 3) with zero padding were used. Stride varies up to 3.
- Pooling layer: The pooling layer will reduce the area dimensions of the input tensor. This is done after each

convolution layer and in our case, uses a Kernel size of up to 3 x 3, with Stride of max 3. Max pooling is used in our experiments.

- ReLU layers: Each convolutional layer includes a ReLU layer. The first fully connected layer also uses the ReLU layer.
- Fully-connected layers: There are two fully-connected layers used in the network. The first fully-connected layer has 384 neurons which all inputs map to. The output is followed with a ReLU. The second fully-connected layer has a Softmax to read the probability.

TABLE IV summarizes the parameters for our network, a modified CNN model from AlexNet. Conv1+Max Pool in the tables refers to the first convolution layer with a Max Pool that follows. Take note here that all Max Pooling is followed by a ReLU activation function applied on outputs.

TABLE IV. SUMMARY OF NETWORKS' LAYER

	Layers					
	1	2	3	4	5	6
Type	RS	C1+ Max pool	C2+ Max pool	C3+ Max pool	FC 1	FC 2
Channels	3	32	32	32	384	2
Filter size	-	3x3	3x3	3x3	-	-
Conv Stride	-	1x1	1x1	1x1	-	-
Pool size	-	(2,2)	(1,1)	(3,3)	-	-
Pool stride	-	(2,2)	(1,1)	(3,3)	-	-
Pad size	-	Same	Same	Same	-	-

RS: Reshape, C1, C2, C3: convolution layer, FC: fully-connected layer

C. Hardware Implementation

There are plenty of lightweight hardware options available in the market. Different hardware platforms provided different benefits. Some of the systems are pure Single Board Computers (SBC) such as the BeagleBone [13] that uses an Arm Cortex A8 processor. Another notable board is the FPGA mixed SBCs such as the Pynq Z2 [14] that utilizes Xilinx Zynq Z7020 chip, programmable using Python. We experimentally used Raspberry Pi 3 B+ as the SBC for user interface, wireless connectivity and etc.

The CNN model was deployed in VPU in this work. VPU, like GPUs, can be a very fundamental component in a DL application especially on the edge device. As described in [15], managing low-latency applications can be very challenging to satisfy even on a powerful SoC running a lean and optimized operating system. This is due to high performance processor not being designed for CNN architectures and complex tensor manipulation. Additionally, running such applications would require majority of the resources on a hardware platform, leaving little to process other functions of an SoC. As such, we would look to VPUs to offload the main bulk of the CNN processing, or act as an acccelerator.

The Myriad X MCSoC by Intel Movidius was the selected VPU for this application. The Myriad X has a Neural Engine and 16 Streaming Hybrid Architecture Vector Engine (SHAVE) Cores that is used specifically for running on-device deep neural network applications [15]. It is an MCSoC that uses a configuration of wide and deep registers together with variable-length long instruction word (VLLIW) to control multiple units at once. This sets apart the SHAVE MCSoC from regular processors and allows the SHAVE to achieve parallelism with a low power consumption. Typically, Myraid can run at 80 to 150 GFLOPS at approximately one Watt of power. This device supports deep learning frameworks such as CAFFE [16], and TENSORFLOW [17] through Intel OpenVINO™ toolkits also.

III. EXPERIMENTS AND RESULTS

Our inference tests were conducted on two systems. The first system was a computer with an average GPU GEFORCE 770M. The second system was mentioned earlier and consists of lightweight hardware: Raspberry Pi 3 B+ and an Intel Movidius Neural Compute Stick USB dongle that utilizes an MCSoC Myriad Chip attached to it.

The CNN model was tested on the BreaKHis set. TABLE V shows the performance and also the utilized resources of the networks of interest. Three networks' results were summarized in this table includes the existing AlexNet from [3]. Netowork-1 has better accuracy but requires much higher FLOP than Network-2, which has a slightly lower accuracy. The predicted value is the weighted outcome from our linear regression equation for Network-2. FLOP and Param are left out for its rows as they are calculated values.

TABLE V. NETWORKS' PERFORMANCE

	AlexNet	**Network-1**	**Network-2**	**Predicted**
Accuracy	0.846	0.866	0.858	0.890
FLOP	1.5 G	100 M	3.6 M	-
Param	62 M	6.6 M	0.21 M	-

Param: Parameter

The performance in term of accuracy for most of the twenty trained networks do not differ much on the BreaKHis test set. What is however interesting is that Network-2 has a vastly smaller number of parameters and an even contrasting number of FLOP per forward

computation. This reduction in size coupled with the maintained accuracy makes Network-2 desirable for hardware deployment. This is especially useful to MCSoC hardware deployment.

The networks were then deployed on our systems to benchmark the inference capabilities of a VPU MCSoC environment against a GPU. TABLE VI shows the performance of the networks on the hardware inference. Utilizing 12 SHAVE processors in the Myraid Chip, Network-2 runs an inference at 2.11 ms as compared to 95 ms on AlexNet. While the GPU inferences are better than MCSoC environments, it is worth noting that Network-2 inference time on the MCSoC is far shorter than AlexNet on a GPU. The bandwidth of Network-2 based on the capability of the MCSoC, is far lower than AlexNet. This may suggest how the MCSoC environment is not as capable with dealing with the large network as comparison with the GPU.

TABLE VI. INFERENCE REQUIRED BANDWIDTH AND TIME

	AlexNet	**Network-1**	**Network-2**
MCSoC Bandwidth (MB/s)	35659.89	5220.9	3645.9
MCSoC Time (ms)	95.05	15.23	2.11
GPU Time (ms)	31.6	4.29	0.77

With a capability of around 150 GFLOPS, our network is well within the capabilities of the hardware and it reflects from the speed of the inferences. Upon running our test set on the two systems, average test accuracy scores are shown in TABLE VII. The performance of Network-2 in term of accuracy is comparable with reference [5]. Parameter-Free Threshold Adjacency Statistic (PTAS) together with Support Vector Machines (SVM) with gaussian kernel or Nearest Neighbour clustering algorithm was used in [5].

TABLE VII. IMAGE INFERENCE ACCURACY BENCHMARK

	System 1 (GPU)	System 2 (MCSoC)	AlexNet [3]	SVM / PTAS [5]
Accuracy	85.80%	82.81%	84.4%	82.23%

The image scores are at a similar level of accuracy as other types of learning methods. From TABLE VII, we can conclude that the lightweight system performs almost as well as its GPU variant also.

IV. CONCLUSIONS

In this work we have presented and compared experiments done using the BreaKHis dataset, deployed on two types of systems amongst other DL methods. We have proposed a lightweight network (CNN model) that can run on the constrained systems with ease and discussed the methods by which the network was generated. The network produced satisfactory levels of classification on the BreakHis dataset and had shown to run efficiently and just as well on the small system. Our experiments have shown that while its latency and accuracy may not exceed its clone GPU implementations, a lightweight MCSoC system with our fine-tuned network can perform comparable profiles to known networks on a large system. CNN application on a MCSoC hardware platform is therefore feasible and it can be used as an affordable alternative to implement machine learned specialized tasks to a satisfactory level.

ACKNOWLEDGMENT

The authors would like to thank Intel for sponsoring and support of the Intel Movidius Myriad VPU for this work under Intel® Movidius™ Neural Compute Stick Research.

REFERENCES

[1] F. Bray et al., "GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *Global cancer statistics 2018,* vol. 68, no. 6, pp. 394-424, 2018.

[2] M. A. Hossain, C. Yidan, J. L. Lee, L. L. Yang, T. Pamela and T. H. Teo, "Breast tumor detection using convolutional neural network to assist medical professionals," *SUTD EPD30.202 Design Report*, Singapore, 2018.

[3] F. A. Spanhol, L. S. Oliveira, C. Petitjean and L. Heutte, "Breast cancer histopathological image classification using Convolutional Neural Networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, 2016.

[4] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, Nevada, 2012.

[5] F. A. Spanhol, L. S. Oliveira, C. Petitjean and L. Heutte, "A dataset for breast cancer histopathological image classification," *IEEE Transactions on Biomedical Engineering,* vol. 63, no. 7, pp. 1455 - 1462, 2016.

[6] T. H. Teo, "Learning Machine Learning by design - an experience sharing using Xilinx SoC (Invited)," in *OpenHW*, Singapore, 2017.

[7] T. H. Teo, "Course notes - introduction to Artificial Intelligience," SUTD Academy, Singapore, 2019.

[8] Y. Cheng, D. Wang, P. Zhou and T. Zhang, "A survey of model compression and acceleration for Deep Neural Networks," *IEEE Signal Processing Magazine*, 2017.

[9] Y. LeCun, B. Y and G. Hinton, "Deep Learning," *Nature,* vol. 521, pp. 436-44, 2015.

[10] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, Paris, France, 2010.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fe, "ImageNet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009.

[12] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[13] “BeagleBoard.org Foundation,” [Online]. Available: https://beagleboard.org/bone. [Accessed 2019].

[14] “Xilinx Corporation,” [Online]. Available: http://www.pynq.io/.

[15] B. Barry et al., “Always-on Vision Processing Unit for mobile applications,” *IEEE Micro,* vol. 35, no. 2, pp. 56-66, 2015.

[16] J. Yangqing et al., “Caffe: Convolutional Architecture for Fast Feature Embedding,” in *Proceedings of the 22Nd ACM International Conference on Multimedia*, Orlando, Florida, USA, 2014.

[17] M. Abadi al. et, “TensorFlow: a system for large-scale Machine Learning,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, Savannah, GA, USA, 2016.