Convolutional Neural Network Hardware Implementation for Flower Classification

Trang Hoang, Thinh Do Quang
Department of Electronics, Faculty of Electrical and Electronics Engineering,
Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10,
Ho Chi Minh City, Vietnam
Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District,
Ho Chi Minh City, Vietnam
hoangtrang@hcmut.edu.vn

***Abstract*— Flower classification becomes more and more important as the medical and industrial world grows. Based on that emergency, Convolutional Neural Network (CNN) proposed a way for computer to recognize flowers in place of human as the data becomes enormous. This study proposes the hardware architecture for CNN which is tested with FPGA. Numbers and type of layers, as well as their properties are also proposed for effective hardware implementation. Math functions that engine the CNN are also well-cared for the smoothness of both feed forward and back propagation processes. Measurements were taken on the proposed CNN; its accuracy and yield were verified. It also appeared that the classification accuracy of the CNN is strongly affected by the training conditions as well as the flower characteristics. This indicates that further image pre-processing can improve the accuracy of the CNN, which can be implemented separately with the CNN or embedded in CNN's first layers by controlling the weights.**

Keywords*— *AI, CNN, flower classification, hardware implementation, feed forward, back propagation.

I. Introduction

Much research in recent years has focused on using various Convolutional Neural Network (CNN) for object classification [1]. Object classification is the process of distinguishing one object from other objects, which provides a systematic view for scientists to identify them for later usage and for avoidance of misjudgment. CNNs, which mimic human neurons and brains, are designed to give the computer learning ability through their adjustable internal properties. However, it has still been a huge gap from CNN's intelligence to that of a person, hence currently CNN has been used for classification of homogenous objects only, such as human faces, animal, or vehicle classification [2][3][4].

Among those homogenous objects, flower classification plays an important role in plant pathology, medicine, and agriculture. They have been considered as subjects for many CNN research and applications, as they are easy to capture images and extract feature. For example, Busra et al. provided a CNN model that use image processing techniques to extract flower features before classification [5]. Another research from Chhaya et al. used transfer learning technique to make use of the existed trained CNNs [6]. Despite the fact that many CNNs has been built, most of the current CNNs for flower classification to the best of our knowledge has been based on software only, which has hindered the ability of acceleration as well as efficiency optimization of the hardware.

This paper proposes a fully hardware implementation of a CNN for flower classification. Our CNN architecture contains 16 layers and processes on RGB images of 10 different kinds of flower. ReLU is selected as the activation function since its match well with hardware implementation, while max pooling is selected as the pooling method because of the color-dependent characteristic of flower [7]. After implementation on real hardware, the proposed CNN seemed to perform the classification with great accuracy. However, the accuracy of the classification does vary in different training conditions (different batches and epochs), and the particularly characteristic of each kind of flower does affect the result as well. Further image pre-processing is also a factor that makes an impact on the performance since it affects the feature extraction process. Hence, the classification results are presented specifically for each scenario and evaluations are given for further research later.

The rest of this paper is organized as follows. Section II presents the methodology for construction of the proposed CNN. The hardware implementation, which includes structural blocks, the state machines, and the hardware operation, is described in section III. Section IV contains the experimental results, and the conclusion of this study is presented in the last section.

II. Methodology

In this section, we provide a systematic view of how the proposed CNN is constructed and what makes it work. Basic components of the CNN like layers and nodes, as well as classification operation are given with specific properties, which were carefully considered from different scenarios for hardware optimization.

A. Classification

An identical CNN contains a number of layers as shown in Fig. 1. These layers are divided into 2 groups: feature learning and classification. The convolutional and pooling layers are responsible for feature learning. Convolutional layers use weight to calculate convolution with the image matrix to extract

its feature (called feature map), while pooling layers reduces that feature map even smaller without losing its characteristic. The latter layers are used for classification purpose. Minimized feature map from the feature learning is flattened into a vector (through flatten layer); then that vector is multiplied with another weighted vector to produce a vector that has the same size as the number of types of objects required to be classified. Finally, that vector is passed through a softmax function [8] to produce result in the form of probability. The nearer the result to 1, the higher the probability of classification the input object into the corresponding type is.

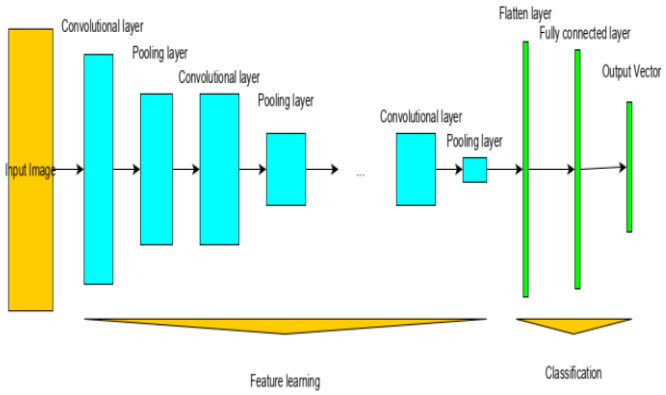

Fig. 1. An identical CNN architecture.

B. Activation function

Let us define the input data as X, the weights (internal properties) of the CNN as W, and the output of the CNN as Y. Y is a vector with a width of n elements; each element represents a type of object that the CNN classifies. To get Y, a function which includes X and W are performed. The purpose of this function is to find the "borders" that separate every "region" representing types of objects, and that function is called the activation function. Since the CNN's structure is divided into layers, so the activation function can also be broken into smaller activation functions for each layer. Therefore, those activation functions need to be non-linear, for the "borders" of object types are scarcely linear and a summation of linear functions is just a bigger linear function and CNN's layers are meaningless then.

Several activation functions have been provided by other previous research [9][10][11]. The most used ones are sigmoid and ReLU function; each one has its own pros and cons. These two functions are non-linear, which fulfill the first requirement of an activation function. While sigmoid function is well-bounded in the range of [-1;1], ReLU function does not have an upper bound in the positive number domain. On the other hand, sigmoid function is not easy to compute (contains exponential), whereas ReLU calculation is simple even when finding the derivatives for the back-propagation process.

C. Gradient descent

To determine error or loss of a CNN, loss function is defined to show how much precisely the CNN can classify; the smaller the loss function, the more accurate the classification is. Based on that point, gradient descent is an optimization algorithm for finding a local minimum of that loss function of the CNN.

Gradient descent is based on basic mathematic theorem. The sign of the derivative of the function at an arbitrary point can tell if it is on the right (larger) or on the left (smaller) of the minimum point. Gradually adjusting the test point until it reaches the minimum point is how gradient descent works. The adjustment step is called learning rate and it determines how fast a CNN can learn.

As gradient descent is used to find local minimum of the loss function, the loss function itself also needs to be easy to find the derivative. Prior research did provide many loss functions and each one of them has its own strength. Hichame et al. did a research that compares the most common loss functions [12]: weighted cross-entropy loss, Huber loss, and sparseMax loss. The weighted cross-entropy loss is strong for awareness of class imbalance while the other two are more effective in learning speed. For the proposed CNN, we determined to use weighted cross-entropy loss since hardware implementation does accelerate the speed and we can focus more on the accuracy performance.

D. Image pre-processing in CNN

Theoretically, CNN's weights should be updated after each training time (through back-propagation) and can be randomly initialized. However, we did notice that if the initialized weights are too far from the optimized ones, it takes more time for the training process. And there is a chance that two nodes from the same layer may extract the same feature from the original image, especially at the first layers of the CNN, since the weight initialization for them can be nearly the same. This results in lower performance as some features may be concentrated on too much, while some other ones are not extracted. And in the case of flower, which has many endemic cases, this means some flower classes give better accuracies since the matching feature are extracted, whereas some classes give worse results.

As stated before, the convolutional layers are used for feature extraction, using their kernel matrices. Previous research has determined some specific kernel matrices for specific purposes on the feature extraction [13][14], which can include edge detection, blur, sharpening. This study hence provides a way to reconfigure the weights (which also mean the kernels) of first layers of the CNN to observe the result and then compares to the randomly initializing one. Some kernels of the first layers are configured for edge detection and sharpening while some others are randomly initialized to allow the CNN to detect other features.

III. Hardware Implementation

In this section, we present the specific hardware structure of the proposed CNN, which includes the controllers and the layers. State machines are described for the control operation while structural blocks give a view of the data processing between layers.

A. CNN structure

To implement the CNN for flower classification in this study, we proposed a CNN of 16 layers, which is based on:

- Input: RGB image of 508x508x3 pixels.
- Output: A 10x1 vector, each element correspondingly represents a flower class.
- Training set: includes 650 images with 65 images for each type of flower, as well as their labels.
- Test set: includes 150 images with 15 images for each type of flower.

The properties of each layer are shown in Table I as below. The input size of the images, which is 508x508x3 is actually determined backward from the last pooling layer for integral calculation. The original size of the input images does vary, since they are separated collected by different means and different peoples, so they are cropped to a considerable size for uniform input of the proposed CNN. The crop process is described in section IV.

TABLE I. CNN LAYER PROPERTIES.

No	Layer type	Input size	Output size (feature map)	Number of feature map
1	Input	None	508x508	3
2	Convolutional	508x508	504x504	6
3	Pooling	504x504	252x252	6
4	Convolutional	252x252	248x248	6
5	Pooling	248x248	124x124	6
6	Convolutional	124x124	120x120	12
7	Pooling	120x120	60x60	12
8	Convolutional	60x60	56x56	24
9	Pooling	56x56	28x28	24
10	Convolutional	28x28	24x24	24
11	Pooling	24x24	12x12	24
12	Convolutional	12x12	8x8	24
13	Pooling	8x8	4x4	24
14	Flatten	4x4(x24)	384x1	N/A
15	Fully connected	384x1	100x1	N/A
16	Output	100x1	10x1	N/A

B. Classification threads

The data moving within the CNN can be divided into four threads:

- Initiating weights.
- Feed forward.
- Back propagation.
- Updating weights.

Three threads including initiating weights, feed forward and updating weights are processed from input layer to output layer, while back propagation is the only thread that flows the opposite way.

The initiating weights thread is process only once at the start of the training set to randomly generate weights for the CNN (Gauss distribution) [15]. After this thread is done, feed forward is process on input datum to extract its feature and calculate the output vector. That vector is included in the loss functions to estimate errors, which starts the back propagation thread (gradient descent). At last, the updating weights thread is run to update the weights based on results of the back propagation thread.

C. Control blocks

At input layer and output layer, two control blocks are defined to control all threads, which we called startpoint and endpoint controller. These two blocks contain two state machines, which interact with the input and output layer to determine when a thread needs to be run, and when to stop it. They are also responsible for how training process can be done, through controlling given batch and epoch. Their state machine graphs are shown in Fig. 2.

The startpoint controller will switch its state to S0 if it detects input datum, then if it is enabled to train or encounters a reset signal, it switches to S1 for initiating weight thread. Then, the startpoint controller will go to S2 and S3 as the feed forward thread process. It then waits for the back propagation thread to finish to start the updating weights thread by switching to S5 state. The cycle then begins with another input datum.

The endpoint controller is simpler as it just needs to notice the startpoint controller about the completion of the feed forward, initiating weights and updating weights process so that the startpoint controller can continue.

D. Layers

Because the CNN in this study processes four internal threads, the design of each layer is also divided into four blocks for each thread.

As the convolutional kernel "moves" throughout the input matrix, it is simple to use counters to change the location (or address) where the kernel starts convoluting after each clock tick. The convolution results in one number that is also guided by those counters to fit in the output feature map. The design of this convolution cell is shown in Fig. 3.

The full designs of the convolutional layers are presented in Fig. 4. Firstly, Fig. 4.a shows the design for the feed forward thread, in which a counter is used to select input feature maps, choose the corresponding kernel for calculating the convolutional result by using the convolutional cell as stated above. The results are passed through a ReLU activation function and pasted into the output matrix (output feature map). Secondly, the design of convolutional layer for back propagation thread is in Fig. 4.c, is nearly the same as the design for feed forward process, however, the kernel is "flipped", and the input matrix is not the feature map but the error matrix. Finally, the design for initiating and updating weights are presented in Fig. 4.b and 4.d, in which the kernels are generated or adjusted.

The design for pooling layers is much simpler as they do not contain the weights. Only designs for feed forward and back propagation are required, and they are shown in Fig. 5. Subtraction blocks are used to compare the pixel values within the a 2x2 region of the feature map, which then results in the maximum value. For the back propagation thread, the saved

signed values from feed forward thread are used to return the derivative to its corresponding address through the demuxes.

IV. Results

This section gives the classification results of the proposed CNN implemented on real hardware. We carefully evaluated the results in different scenarios, as well as considered the accuracy for each particular flower class, from which we summarized the impact of them on the flower classification. The image-processing kernels are also taken into account for observing theirs influence on the classification.

A. Datasets

This paper used the well-known flower dataset of Oxford University [16], which contains 17 categories of flower with 800 images of each class. As stated before, only 10 of these flower classes were selected for the proposed CNN model, which are Daffodil, Snowdrop, Bluebell, Tiger Lily, Fritillary, Sunflower, Daisy, Cowslip, Buttercup, and Windflower.

However, the images in the dataset were different in size, so a function process in Matlab was run to crop these images into the size of 508x508 pixels. To maintain the focus of the image (avoid flower features being lost), feature extraction technique was used to keep the most important region, including the petals, pistil, and stamens. From the original image, the background is discarded using segmentation techniques [17], then the texture and shape features are extracted [18] to determine the location of the pistil and stamen (usually at the center of the flower), which will then be the center of the cropped images. After all the original images are cropped, the dataset was ready as uniform input of the CNN.

For random ordering the images in the dataset into the CNN to avoid same flower characteristic being repeated consecutively, a random function was calculated to rename the name of the original images into different ordering numbers using the commonly used pseudo-random generator, linear-feedback shift register [19]. The images were also divided into the training set and test set, which were 650 and 150 images respectively, of each class.

B. Simulation

Before real hardware implementation, the behavior of the CNN was pre-tested through simulation on Questa Sim 10.6c. Questa Sim 10.6c is a software that effectively support SystemVerilog and provides good assistance for hardware testbench as well as other verification processes.

SystemVerilog randomization feature was used to first-time establishment of the CNN's weights. Input images from the datasets were read as hex files using SystemVerilog system task $readmemh. The output classification results as well as trained weights were stored also as hex files for later evaluation.

C. Hardware setup

In order to implement the proposed CNN in real hardware model, Virtex-6 FPGA of number XC6VSX475T of Xilinx was used for high signal processing capability. Vivado Design Suite HLx software was selected to synthesize and implement the SystemVerilog code of the CNN.

The Vivado Design Suite HLx also provided a tool to generate a block memory with the initial content as a given image [20], so this was how the FPGA board processed through the training and test set. Moreover, the output vectors were programmed to connect to the FPGA LEDs for a visionary view of the classification result. And all weights values were stored in the provided block RAMS with patterned addresses for each layer's access.

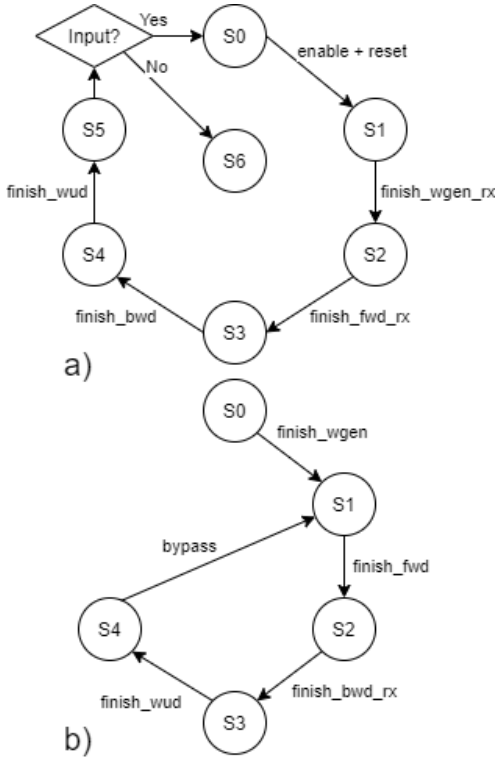

Fig. 2. Controller state machines.

D. Training and test results

To evaluate the impact of different scenarios on the classification result, we selected several numbers of epoch and batch. In Table II, the comparison of accuracy in each scenario are shown. We found that as the epoch was increased, the classification result became more accurate, however, if the epoch is too large, the CNN seemed to be overfitting and it became less precise. The same thing happened as the batch increased; larger batch increased the accuracy by avoiding overfitting (avoid adjusting weights too much) but too large batch lowered the efficiency since the CNN had to recalculated weights to match too many input images.

At another aspect, the classification results were also different for each class. Flower class with special characteristics tended to result in better classification. For example, as shown in Table III (at epoch = 3 and batch = 3), class 9 (buttercup) had a high accuracy of 97.33, maybe because of its plain color and clearly separated petals. Class 6 (sunflower) also achieved better results, thanks to its easy-

recognizable disk florets at the center. On the other hand, class 2 (snowdrop) and class 8 (cowslip) had lower accuracy due to the same tube-shaped of both flowers and due to the diversly camera angles of the training images.

Finally, the study evaluates the impact of using image-processing kernel in the CNN's first layers instead of randomly initializing them. The accuracy results can be seen in Table IV. It was shown that using the pre-determined kernels did improve accuracy for some kinds of flower. The used kernels are for edge detection and sharpening so it results best with specific texture like fritillary and snowdrop. However, the color features of the flowers seemed to be lowered as in the case of Windflower and Daffodil. These two flowers have very similar shape but different colors, and the CNN in this scenario often mistook between these two flowers. It indicates that more specific kernels should be added to first layers to be able to extract more features. However, some features cannot be easily extracted using kernel but using much more complicated image processing algorithm. Later study may find a way to combine both image pre-processing and CNN for a more efficient way to solve the classification problem.

TABLE II. CLASSIFICATION ACCURACY AT DIFFERENT EPOCHS AND BATCHES.

Batch	Epoch = 1	Epoch = 2	Epoch = 3	Epoch = 5
1	88.47	89.13	89.20	88.27
3	90.40	92.13	93.27	91.20
5	87.20	88.73	90.13	88.40

TABLE III. FLOWER CLASSES ACCURACY COMPARISON AT EPOCH=3 AND BATCH=3.

Class	Name	Characteristics	Accuracy
1	Daffodil	Plain color, separated petals	94.67
2	Snowdrop	Tube-shaped, upside-down	83.33
3	Bluebell	Grows in bunches, large pistils	92.00
4	Tiger Lily	Spots, parabolic petals	94.67
5	Fritillary	Spots, bell-shaped, upside-down	93.33
6	Sunflower	Large disk florets, many petals	98.00
7	Daisy	Many petals, plain color	94.67
8	Cowslip	Tube-shaped, grows in bunches	88.00
9	Buttercup	Plain color, separated petals	97.33
10	Windflower	Plain color, separated petals	96.67

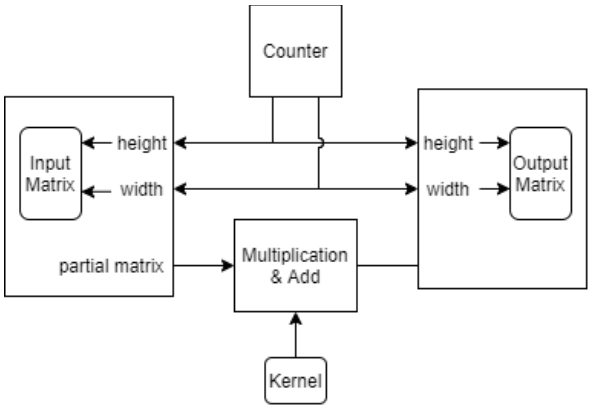

Fig. 3. Design of the convolutional cell.

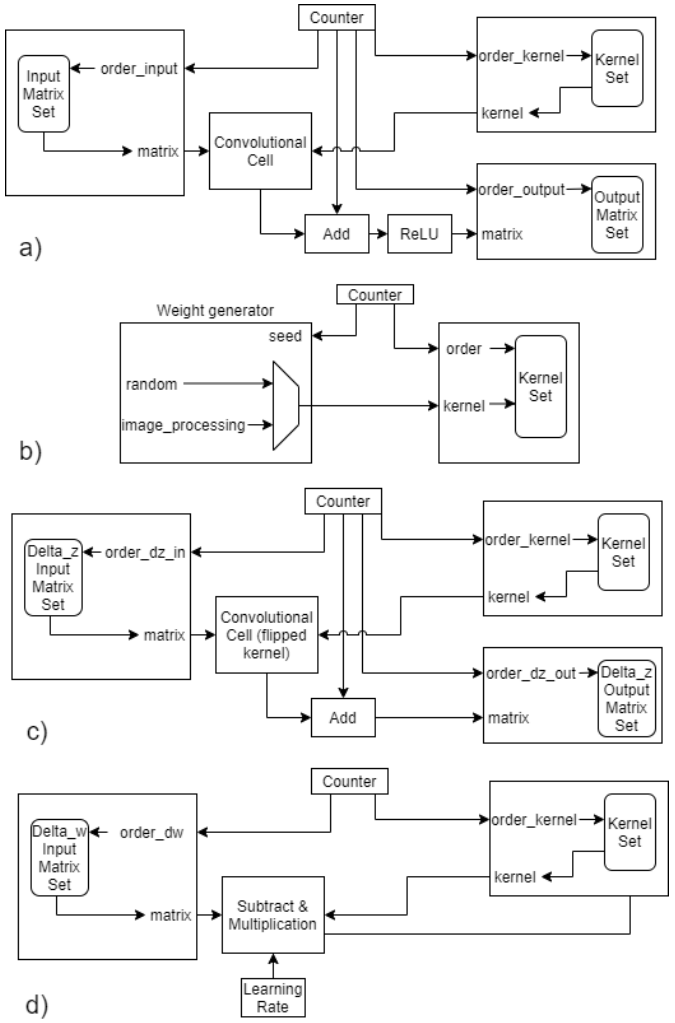

Fig. 4. Design for convolutional layers based on the threads.

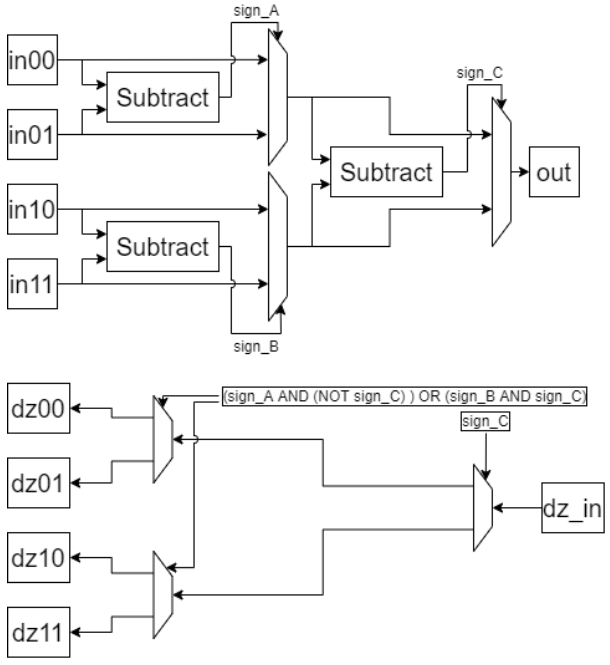

Fig. 5. Design for max pool layers.

TABLE IV. ACCURACY COMPARISON FOR THE CNN WITH IMAGE-PROCESSING KERNELS.

Class	Name	Accuracy of CNN with randomly initializing weights	Accuracy of CNN with image-processing kernels
1	Daffodil	94.67	87.33
2	Snowdrop	83.33	90.67
3	Bluebell	92.00	90.33
4	Tiger Lily	94.67	95.67
5	Fritillary	93.33	94.33
6	Sunflower	98.00	90.33
7	Daisy	94.67	90.33
8	Cowslip	88.00	92.67
9	Buttercup	97.33	93.67
10	Windflower	96.67	91.33

V. CONCLUSION

Prior work has built many CNN models for flower classification, and they performed well in term of accuracy, by using different techniques, not only from the artificial intelligence field but also from the image processing field. However, these CNN models are not hardware-based so they may not result in the best efficiency. This study proposed a fully hardware-based CNN in which all components and numbers are considered for hardware application. We found that the implemented model achieved the criteria for accuracy in different scenarios. We also noticed that special characteristic of each flower class does make an impact on the results, especially flower with endemic ones. This indicates that further image processing should be done before going into the CNN, or the CNN's weights of the first several layers should be pre-determined for specific feature extraction. Later study can focus on a combination of image processing and CNN to make use of both of their advantages.

ACKNOWLEDGMENT

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

REFERENCES

[1] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9,** 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

[2] Ben Fredj, H., Bouguezzi, S. & Souani, C. Face recognition in unconstrained environment with CNN. *Vis Comput* 37**,** 217–226 (2021). https://doi.org/10.1007/s00371-020-01794-9

[3] Z. Cao, J. C. Principe, B. Ouyang, F. Dalgleish and A. Vuorenkoski, "Marine animal classification using combined CNN and hand-designed image features," *OCEANS 2015 - MTS/IEEE Washington*, 2015, pp. 1-6, doi: 10.23919/OCEANS.2015.7404375.

[4] D. Zhao, Y. Chen and L. Lv, "Deep Reinforcement Learning with Visual Attention for Vehicle Classification," in *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 4, pp. 356-367, Dec. 2017, doi: 10.1109/TCDS.2016.2614675.

[5] B. R. Mete and T. Ensari, "Flower Classification with Deep CNN and Machine Learning Algorithms," *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1-5, doi: 10.1109/ISMSIT.2019.8932908.

[6] C. Narvekar and M. Rao, "Flower classification using CNN and transfer learning in CNN- Agriculture Perspective," *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 660-664, doi: 10.1109/ICISS49785.2020.9316030.

[7] T. Tiay, P. Benyaphaichit and P. Riyamongkol, "Flower recognition system based on image processing," *2014 Third ICT International Student Project Conference (ICT-ISPC)*, 2014, pp. 99-102, doi: 10.1109/ICT-ISPC.2014.6923227.

[8] Liang X., Wang X., Lei Z., Liao S., Li S.Z. (2017) Soft-Margin Softmax for Deep Classification. In: Liu D., Xie S., Li Y., Zhao D., El-Alfy ES. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science, vol 10635. Springer, Cham. https://doi.org/10.1007/978-3-319-70096-0_43

[9] Szandała, Tomasz. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks.

[10] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning", arXiv:1811.03378, 2018.

[11] da S. Gomes, G.S., Ludermir, T.B. & Lima, L.M.M.R. Comparison of new activation functions in neural network for forecasting financial time series. *Neural Comput & Applic* 20**,** 417–439 (2011). https://doi.org/10.1007/s00521-010-0407-3

[12] H. Yessou, G. Sumbul and B. Demir, "A Comparative Study of Deep Learning Loss Functions for Multi-Label Remote Sensing Image Classification," *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 1349-1352, doi: 10.1109/IGARSS39084.2020.9323583.

[13] H. Takeda, S. Farsiu and P. Milanfar, "Kernel Regression for Image Processing and Reconstruction," in *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349-366, Feb. 2007, doi: 10.1109/TIP.2006.888330.

[14] Taichi Joutou and Keiji Yanai, "A food image recognition system with Multiple Kernel Learning," *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 285-288, doi: 10.1109/ICIP.2009.5413400.

[15] Deng, Z.; Cao, Y.; Zhou, X.; Yi, Y.; Jiang, Y.; You, I. Toward Efficient Image Recognition in Sensor-Based IoT: A Weight Initialization Optimizing Method for CNN Based on RGB Influence Proportion. *Sensors* 2020, *20*, 286. https://doi.org/10.3390/s20102866

[16] Maria-Elena Nilsback and Andrew Zisserman, Oxford University, Oxford, United Kingdom. 17 Category Flower Dataset. Available: https://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

[17] Najjar, Asma & Zagrouba, Ezzeddine. (2012). Flower image segmentation based on color analysis and a supervised evaluation. International Conference on Communications and Information Technology - Proceedings. 10.1109/ICCITechnol.2012.6285834.

[18] Mabrouk, Amira & Najjar, Asma & Zagrouba, Ezzeddine. (2014). Image Flower Recognition based on a New Method for Color Feature Extraction. VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications.

[19] J. Savir and W. H. McAnney, "A multiple seed linear feedback shift register," *Proceedings. International Test Conference 1990*, 1990, pp. 657-659, doi: 10.1109/TEST.1990.114080.

[20] Xilinx, Vivado Design Suite User Guide, UG973, June 3, 2020.