مجلة العلوم البحتة والتطبيقية
**Journal of Pure & Applied Sciences**
www.Suj.sebhau.edu.ly ISSN 2521-9200
Received 23/03/2020 Revised 16/08/2020 Published online 05/10/2020

# Capsule Network Implementation On FPGA

*Salim A. Adrees , Ala A. Abdulrazeg
Department of Computer Engineering, College of Engineering, Omar Al-Mukhtar University, Libya
*Corresponding author: Salim.ali@omu.edu.ly

**Abstract** A capsule neural network (CapsNet) is a new approach in artificial neural network (ANN) that produces a better hierarchical relationship. The performance of CapsNet on graphics processing unit (GPU) is considerably better than convolutional neural network (CNN) at recognizing highly overlapping digits in images. Nevertheless, this new method has not been designed as an accelerator on field programmable gate array (FPGA) to measure the speedup performance and compare it with the GPU. This paper aims to design the CapsNet module (accelerator) on FPGA. The performance between FPGA and GPU will be compared, mainly in terms of speedup and accuracy. The results show that training time on GPU using MATLAB is 789.091 s. Model evaluation accuracy is 99.79% and the validation accuracy is 98.53%. The time required to finish one routing algorithm iteration in MATLAB is 0.043622 s and in FPGA it takes 0.00065s which means FPGA module is 67 times faster than GPU.
**Keywords:** Artificial intelligent, Capsule neural network, Deep learning, FPGA, Image recognition.

# تطبيق الشبكات الكبسولية على FPGA

*سالم علي ادريس و علاء علي عبد الرازق
قسم الحاسوب–كلية الهندسه–جامعة عمر المختار، ليبيا
*للمراسلة: Salim.ali@omu.edu.ly

**الملخص** الشبكة العصبية الكبسولية (CapsNet) هي مفهوم جديد في الشبكة العصبية الاصطناعية (ANN) التى تقدم تجانس افضل في بنية النموذج. أداء الشبكة العصبية الكبسولية على وحدة معالجة الرسومات (GPU) يعتبر أفضل بشكل كبير من الشبكة العصبية الالتفافيه (CNN) في التعرف على الارقام المتداخلة للغاية في الصور، غير أن الشبكة العصبية الكبسولية لم تصمم على مصفوفة البوابات المنطقية القابلة للبرمجة (FPGA) لقياس سرعة التنفيذ و دقة النتائج و مقارنتها بوحدة معالجة الرسومات. هذه الورقة تهدف الي تصميم نموذج لشبكة عصبية كبسولية على مصفوفة البوابات المنطقية القابلة للبرمجة ، بعد ذلك سيتم المقارنة بين اداء مصفوفة البوابات المنطقية القابلة للبرمجة و وحدة معالجة الرسومات من حيث الدقة و سرعة الأداء، النتائج بينت بأن وقت التدريب على وحدة معالجة الرسومات بأستخدام MATLAB هو 789.091 ثانية، دقة تقييم النموذج هي 99.79% و دقة تحقق النموذج هي 98.53%، الوقت المطلوب لانهاء دوره واحده لخوارزمية التوجيه في MATLAB هو 0.043622 ثانية و على مصفوفة البوابات المنطقية القابلة للبرمجة الوقت المطلوب 0.00065 ثانية مما يعني أن نموذج مصفوفة البوابات المنطقية. القابلة للبرمجة 67 مره اسرع من وحدة معالجة الرسومات.
**الكلمات المفتاحية:** الذكاء الأصطناعي، الشبكات العصبية الكبسولية، التعلم العميق، FPGA، التعرف على الصور.

## I. Introduction

Nowadays, Machine learning powers different fields such as websites, cameras, and smartphones. Machine learning [1] was implemented to identify objects in the image and extract them to apply further processing. The general idea of machine learning is to take a real dataset and apply any machine learning algorithms on the dataset such as deep learning [1], neural networks[2], Perceptron algorithm [3], K-nearest neighbour algorithm [4], decision tree[5], etc. In addition, the most dominant technique is deep learning.

Convolutional neural network (CNN) has controlled the field of deep learning because it is very good to extract and analyze data to form the relationships between inputs and outputs. Unfortunately, CNN has a drawback in its basic architecture. This drawback causing it to be not effective [6-8]. The main drawback is the max pooling function in the convolutional layer where extracting the information for the image is accrued [1]. After applying convolution on the input image using feature extracting filters, the max pooling will take a place as shown in Fig. 1.Max polling function detects the presents of a certain object only and ignores other important features such as precise location, rotation, and the relationships to the other objects in the same image [6-8].

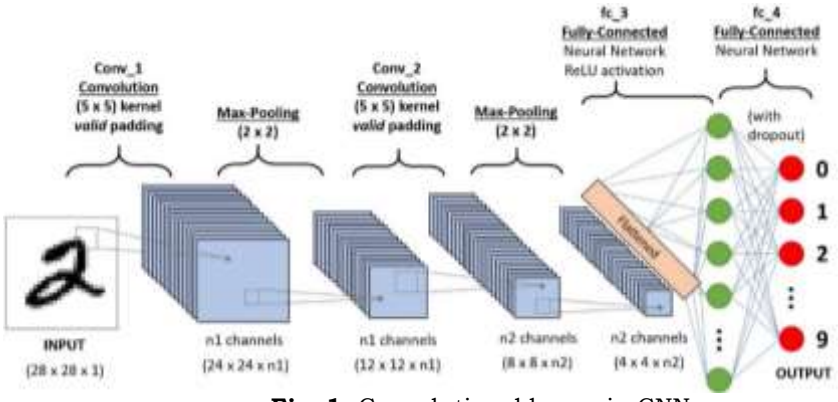


**Fig. 1:** Convolutional layers in CNN

The Capsule network is a new approach of machine learning field. Hinton and his team in 2017 [6] introduced a dynamic routing mechanism for capsule networks. The approach has proved that it reduces the MNIST dataset error rates and size of the training dataset. Results were estimated to be better than the CNN on highly overlapped digits. CNN learn to classify objects by passing the image through convolutional layers to extract low-level features in first layers and then grouping up features learned through the first layers to extract more complex features in deeper layers [1]. CNN tries to overcome the limitation by using a large training dataset. The fundamental issue in CNN is using the max pooling function to extract features and fed them to the next layer [9]. For this reason, CapsNet is developed to replace invariance concept with equivariant. Equivariant means that if the input appears in the image but in different rotation or position the model will consider them, while invariance means that the model aims to identify the presence of the object ignoring other related properties.

In the last few years, there was a race between GPU and FPGA vendors in terms of the amount of computation that can be handled efficiently with high speed. In addition, the popularity of machine learning and its complex computation demands makes it one of the important comparisons between GPU and FPGA [10].

In this paper, we have implemented CapsNet on FPGA and GPU and compare the accuracy and the speedup performance. Vivado is used in FPGA implementation and MATLAB is used in GPU implementation. The organization of this paper is as follows. Section II includes details about the CapsNet algorithm. The FPGA implementation is described in Section III. Section IV presents the results in GPU. Section V presents the results in FPGA. Finally, Section VI concludes the paper.

## II. capsule neural network

The hierarchy in this algorithm is divided into three main blocks as shown in Fig. 2. Each block has sub operation and they are [6]:

1. Primary capsules:
a) Convolution.
b) Reshaping process.
c) Squashing function.
2. Higher level capsules:
a) Routing between capsules.
3. Loss calculation:
a) Margin loss.

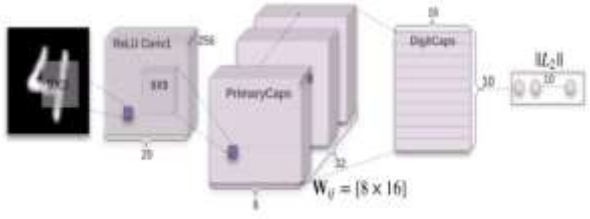


**Fig. 2***:* Structure of the capsule networks

b) Reconstruction loss

The key feature of this algorithm is the Routing algorithm which represents the mid-section of the CapsNet algorithm. The hardware part will be designed for the Routing algorithm. The mathematical representation of this algorithm is shown in Procedure 1.

This procedure will receive $\hat{u}$ which the output of the primary capsules, the number of layers $l$ and the number of epochs $r$.

**Procedure 1** Routing algorithm

1: **procedure** Routing **(**$\hat{u}_{j|i}$ , $r$ , $l$**)**
2: **for** all capsule $i$ in layer $l$ and capsule $j$ in layer $(l + 1)$: $b_{ij} \leftarrow 0$.
3: **for** $r$ iterations **do**
4: for all capsule $i$ in layer $l$: $c_i \leftarrow$ softmax$(b_i)$ ▶ softmax computes Eq. (1)
5: for all capsule $j$ in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij}\, \hat{u}_{j|i}$
6: for all capsule $j$ in layer $(l+1)$: $v_j \leftarrow$ squash$(s_j)$ ▶ squash computes Eq. (2)
7: for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{i+}\hat{u}_{j|i} \cdot v_j$
8: **return** $v_j$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \qquad (1)$$

$$v_j = \frac{\|s_j\|}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \qquad (2)$$

First, all initial logics *b* assigned to zero and the coupling coefficient is calculated using *softmax function*, equation (1), then the sum $S_j$ of all inputs in each digit capsule is calculated, the final step is applying *squashing function* on $S_j$, equation (2), then the initial logics *b* are updated and this is the end of the first epoch. This process will be repeated for *r* times. The complexity of the routing algorithm is the reason for choosing it to be implemented in the hardware part. This will be proved in the result section. The dataset used in this algorithm is Modified National Institute of Standards and Technology (MNIST). The dataset consists of 70,000 images (28 X 28). The training images are 60,000 images, and testing images are 10,000 images [11]. This part will be implemented in MATLAB. Profiling will be implemented on the code to find the timing of each part. The code will be executed on GPU.

**III. FPGA implementation**

The hardware implementation is done in the prediction mode, not in the training mode. First, the primary capsule layer is implemented in the MATLAB and the output of this layer will be adopted to the hardware implementation. Fig. 3 shows the flowchart of the hardware design.

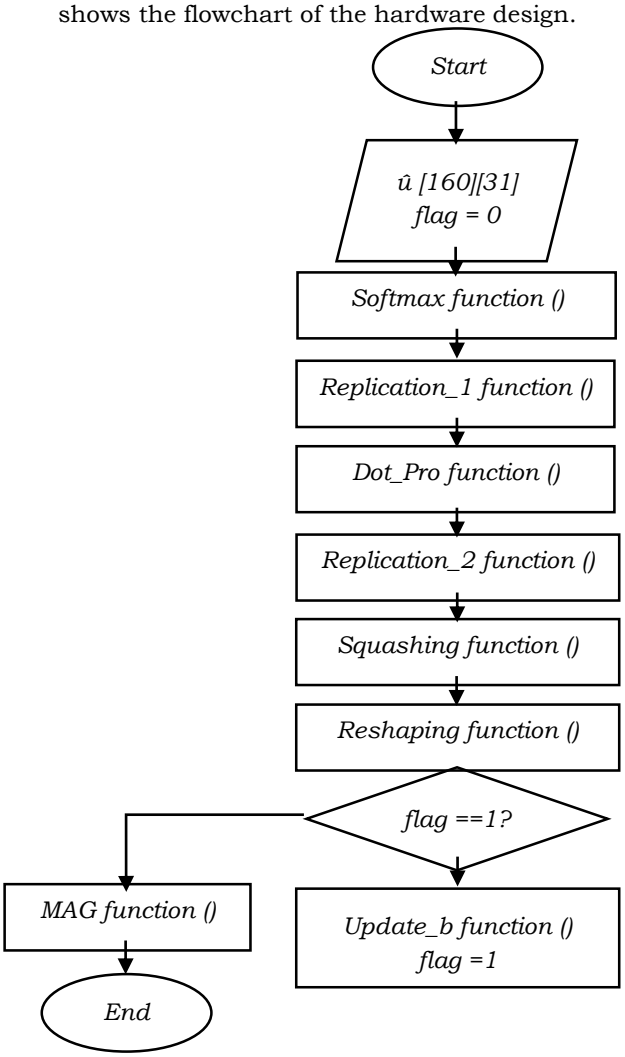


**Fig. 3:** Flowchart of the hardware design

The fixed-point representation is used in this design with 16-bit accuracy. This high number of bits reserved for the accuracy is because it is one of the aims of this research. A matrix *û [160][31]* and *flag* for the loop are the inputs of this flowchart. The main obstacles of the hardware design are represented in the *square root function* which is used in calculating the magnitude in *MAG function*, and the *exponential function* used in *softmax function*. First, the *square root function* is adopted from previous work [12]. Second, the exponential function is implemented on *b* matrix. The simulation in MATLAB shows that the values in this matrix almost equal to zero except for one row for all images in MINTS dataset. By replacing *exp(b)* with $2^b$ the results will not be affected and many clock cycles will be saved.

The software used in this part is Xilinx Vivado. The software will provide the timing simulation of the hardware design.

**IV. GPU Results**

CapsNet is implemented on GPU. The module is trained using MNIST dataset. The results show that the error rate and loss after each epoch are close to zero. Fig. 4 shows the results in MATLAB simulation.

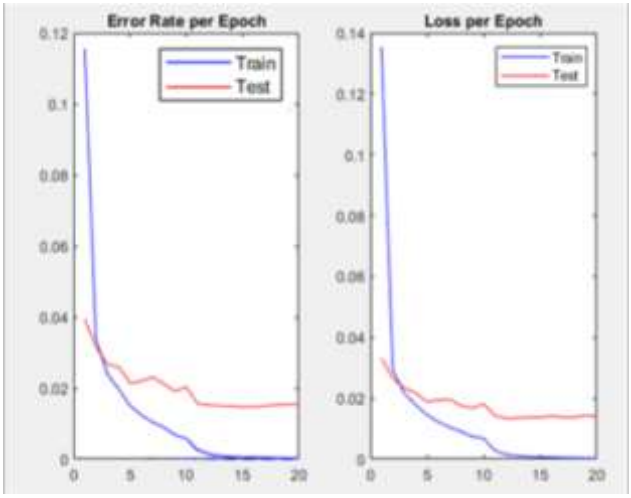


**Fig. 4:** Error rate and loss after each epoch

The module evaluation accuracy is 99.79% and the validation accuracy is 98.53%.

The time analyzing of the code by implementing profiling on MATLAB code shows that the maximum amount of time from overall execution time is taken by the routing algorithm which is equal to 0.043622 s. Fig. 5 shows the profiling summary of the code in MATLAB.

| Function Name | Calls | Total Time | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| RunAll | 1 | 789.091 s | 3.653 s | |
| Main_MNIST_MLP_DynamicRoutingCapsules | 1 | 785.438 s | 2.871 s | |
| Main_Template | 1 | 770.200 s | 1.271 s | |
| TrainingScript | 1 | 730.117 s | 49.238 s | |
| train_net | 20 | 545.034 s | 15.915 s | |
| dynamic_routing_linear_capsule | 6160 | 320.339 s | 268.712 s | |
| net_ff | 3280 | 301.835 s | 10.106 s | |
| net_bp | 2880 | 215.576 s | 1.853 s | |
| linear_layer | 12320 | 146.276 s | 25.206 s | |
| selective_sgd | 20 | 101.271 s | 0.227 s | |
| select_learning_rate | 2 | 101.043 s | 2.789 s | |
| Convolution2D>Convolution2D.forward | 6560 | 79.030 s | 0.117 s | |
| tion2D>Convolution2D.forwardNormal | 6560 | 78.871 s | 0.190 s | |

**Fig. 5:** Profiling summary of the code

Based on this profile summary, the part that should be implemented in FPGA is the routing by agreement algorithm due to its time consuming.

## V. FPGA Results

Routing algorithm will be implemented in FPGA. The input $\hat{u}j|i$ is taken from MATLAB implementation. The prediction of this $\hat{u}j|i$ in MATLAB in digit 5. The hardware design must give the same result for the same $\hat{u}j|i$ or the design will be wrong.

The clock frequency is 100 MHz (20 ns for one clock cycle). The modelling style used in this research is the behaviour modelling style. This type of modelling is considered as the highest level of abstraction provided by Verilog. A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details such as how many adders, multipliers and divider are required.

Vivado software tool will be used to write ASM chart for the complete design.

the module in Vivado is correctly synthesized and it is ready to be tested using $\hat{u}j|i$ as a primary input for the module. After implementing the $\hat{u}j|i$ to the module, the result is shown in Fig. 6.

The values are stored in the memory in binary, and the representations of this binary numbers in decimal are 41, 156, 110, 170, 20, 62881, 47, 143 ,0 and 33. By multiplying these number with 2-16, the actual floating number will be obtained. This is because of using fixed point numbers with 16-bit precision. The output shows that the digit appeared in the image is 5 and it is clear that both of the MATLAB and FPGA results are equal.

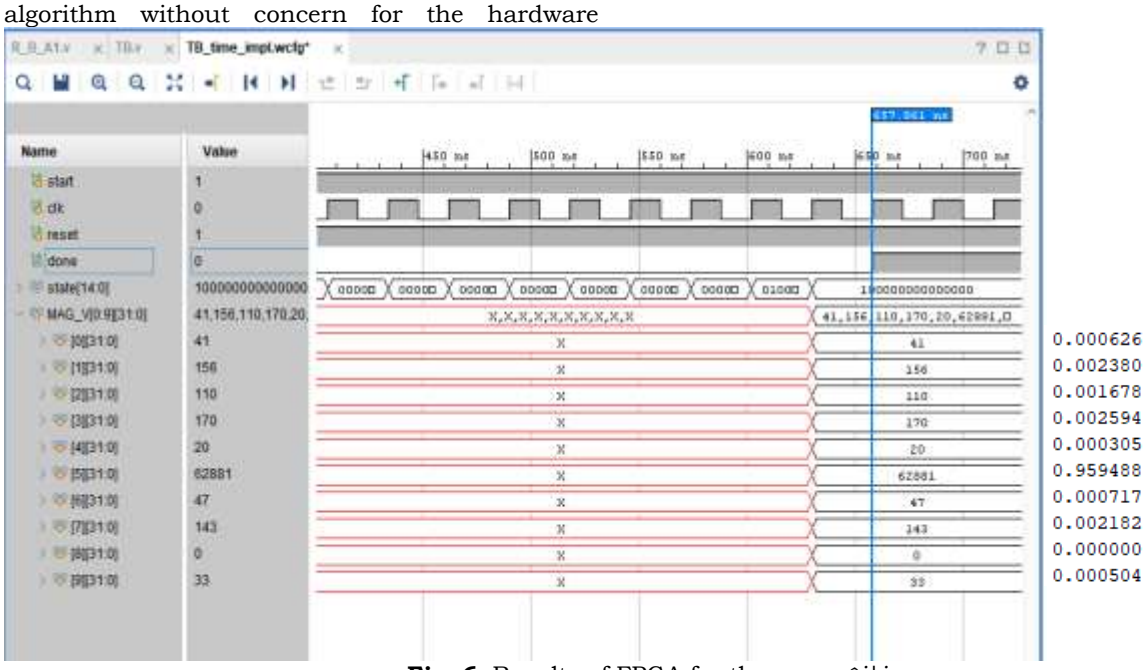


**Fig. 6***:* Results of FPGA for the same $\hat{u}j|i$

The timing simulation in FPGA shows that the delay of the output is 650 ns. The results show the significant improvement in terms of timing simulation. The second objective has been addressed at this point.

Table 1 shows the differences in timing simulation between GPU and FPGA. In addition, the accuracy of prediction is not affected even by repeating the process with ten different values of $\hat{u}j|i$.

**Table 1: Summary of timing simulation and the accuracy in FPGA and GPU.**

| | GPU | FPGA |
|---|---|---|
| Timing | ($0.043622*10^6$ ns) 2181100 clock cycles | (650 ns) 32 clock cycles |

## VI. Conclusion

First of all, in this research the most challenging part is how to extract the inputs from MATLAB implementation. This is because of the complexity of the code and CapsNet itself. Second, fixed point numbers are easier and more efficient than the flouting point number. To implement the floating-point numbers in hardware, IP cores are needed, these IP cores consume many clock cycles. In addition, fixed point numbers are not affecting the results and the error is negligible.

The FPGA module is designed with behavioural modelling style. Resources utilized by the module are not addressed in this type of modelling style. The software is responsible to find how many adder, multiplier and register are needed. Which is one of the advantages of this type of modelling style. However, the drawback of behaviour design is that it is so difficult for software tool to synthesise behaviour design and translate the Verilog code to the hardware design. Thus, if the module is so complex, behavioural modelling style is not the good choice.

Finally, routing by agreement algorithm is the most time-consuming part in the CapsNet algorithm. Implementing this part in FPGA will minimize output delay without effecting the accuracy of the output.

## References

[1]- Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, no. 7553, p. 436, 2015.

[2]- A. Muthuramalingam, S. Himavathi, and E. Srinivasan, "Neural network implementation using FPGA: issues and application," *International journal of information technology,* vol. 4, no. 2, pp. 86-92, 2008.

[3]- Y. Taright and M. Hubin, "FPGA implementation of a multilayer perceptron neural network using VHDL," in *ICSP'98. 1998 Fourth International Conference on Signal Processing (Cat. No. 98TH8344)*, 1998, vol. 2, pp. 1311-1314: IEEE.

[4]- Z.-H. Li, J.-F. Jin, X.-G. Zhou, and Z.-H. Feng, "K-nearest neighbor algorithm implementation on FPGA using high level synthesis," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2016, pp. 600-602: IEEE.

[5]- R. Narayanan, D. Honbo, G. Memik, A. Choudhary, and J. Zambreno, "Interactive presentation: An FPGA implementation of decision tree classification," presented at the Proceedings of the conference on Design, automation and test in Europe, Nice, France, 2007.

[6]- S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, 2017, pp. 3856-3866.

[7]- G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," 2018.

[8]- A. Ahmad, B. Kakillioglu, and S. Velipasalar, "3D Capsule Networks for Object Classification from 3D Model Data," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 2225-2229: IEEE.

[9]- A. Shahroudnejad, P. Afshar, K. N. Plataniotis, and A. Mohammadi, "Improved explainability of capsule networks: Relevance path by agreement," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 549-553: IEEE.

[10]- E. Nurvitadhi *et al.*, "Can FPGAs beat GPUs in accelerating next-generation deep neural networks?," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 5-14: ACM.

[11]- L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 141-142, 2012.

[12]- Y. Li and W. Chu, "A new non-restoring square root algorithm and its VLSI implementations," in *Proceedings International Conference on Computer Design. VLSI in Computers and Processors*, 1996, pp. 538-544: IEEE.