Facial ~~Expressions~~ Expression Recognition System ~~using~~ Using FPGA-Based Convolutional Neural Network

Tai Nguyen Duong Phuc [1*], Nam Nguyen Nhut [1], Nguyen Trinh [1], Linh Tran [2]

[1] ~~Electrical- Electronics~~ Electrical-Electronics Engineering, Ho Chi Minh City University of Technology VNU HCM, ~~xxxx, xxxx~~Ho Chi Minh, Vietnam

[2] Faculty of Engineering and Technology, Thu Dau Mot University, ~~xxxx, xxxx~~Thu Dau Mot, Vietnam

Abstract

Emotion detection has become one of the most significant aspects to consider in any project related to ~~Affective Computing.~~ affective computing. There are several researches in emotion ~~recognition,where Convolutional Neural Network~~ recognition, where convolutional neural network is considered as an efficient algorithm that achieves the state-of-the-art performance in image recognition. There have been plenty of methods applying CNN model and conducting in software. However, traditional ~~software based~~ software-based computation is not fast enough to meet the demand of real-time image processing. Therefore, we propose an efficient method for ~~expressions~~ expression recognition using FPGA-based. This model is used for classifying seven elementary types of human emotions: angry, fear, disgust, happy, sad and neutral by extracting features through those ~~layers~~ layers, respectively. We also tested the functions of this model on FPGA simulation and achieved over 65% of accuracy using ~~FER2013~~ FER-2013 dataset.

Keywords: ~~convolutional~~ Convolutional neural network, ~~face~~ Face expression recognition, FPGA

Introduction

Emotion recognition is necessary in many practical sectors. For example, enterprises use customer expression detection systems and then adjust to develop service quality and satisfy them with best experiences. However, people vary widely in their accuracy at recognizing the emotions of others. Therefore, using high technology to help them with emotion recognition is a relatively nascent research area. And then results computed by computers can be noted to evaluate satisfaction levels of customers for ~~enterprises.~~ enterprises [1].

CNN is a modern and efficient algorithm for computer vision. It has been applied in many recognition systems and gives considerable accuracy. A general CNN model includes ~~3~~ three main layers: convolutional layer, pooling layer and ~~fully-connected~~ fully connected layer. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a ~~non-linearity.~~ nonlinearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (Softmax) on the last ~~(fully-connected)~~ (fully connected) layer [2].

The CNN algorithm of this recognition system must have high reliability, appropriate in tasks requiring the accuracy. They have high computational requirements such that even modern central processing units (CPUs) are often not fast enough and specialized hardware, such as graphics processing units (GPUs), is needed [3]. We can program the network on different types of FPGAs, providing different processing ~~speed.~~ speeds. Implementation of higher number of convolutional blocks on an FPGA can directly lead to a ~~speed-up~~ speedup in processing [4].

In this paper, we present an architecture implemented on FPGA using hardware description language Verilog where ~~Convolutional, Maxpool, Fully-connected~~ convolutional, maxpool, fully connected blocks are designed and packaged into IP cores to conduct the simulation. Firstly, we prepare dataset for training, ~~Dataset~~ dataset FER-2013, then design CNN architecture (IP core) and implement it on FPGA. Finally, we bring out simulation results from test environment.

Background

Dataset ~~preparation~~Preparation

In this section, we first describe what type of dataset is used for the experiment and how it is modified. ~~DataSet~~ Dataset FER-2013 is published by Kaggle in workshop of ICML 2013 seminar [5]; data include gray ~~multi-layer~~ multilayer scale images of size ~~48x48~~ 48×48 with faces in center and the face scale is adjusted to account for almost image areas. Each image will be labeled into one of the seven types of expression 0 to 6 ~~below:~~ below (Table 1).

However, in ~~DataSet~~ dataset there are 2 points which should be noticed that result in errors in expression detection ~~algorithm, one~~ algorithm: One of them is that the number of images belonging to expression 3 ~~(Happy)~~ (happy) is far greater than those belonging to expression 1 ~~(Fear)~~ (fear) and expression 2 ~~(Angry).~~ (angry). Therefore, in order to improve reliability in results, we will optimize the dataset by balancing ~~DataSet~~ dataset FER-2013, reducing images of expression 3 and merging expression 1 and expression 2 together before ~~training.~~ training (Table 2).

CNN Architecture Design

This paper show CNN model for recognition and classification 7 types of expression. A efficient network architecture for image recognition tasks is determined by several factors [6]; the most crucial factor is to select the number of convolutional modules, pooling layers, FC layers, activation function and to use them properly with topic's features and requires. CNN model initiating for facial expression recognition task is designed with two convolutional modules ~~:~~ (Fig. 1).

Neural network is trained for 200 epochs, and the accuracy is about 65%. It is proceeded ~~Neural Network~~ neural network training of Google Collaboratory's workstation. Collaboratory supports ~~GPU, Therefore~~ GPU. Therefore, it helps increase program computing speed significantly. Training process took about 1h30m where each epoch took 33s in average. Diagram of function showing accuracy and error is ~~conducted in TensorBoard.~~ drawn in TensorBoard (Fig. 2).

Implementation on FPGA

In this part, we present ~~steps respectively~~ steps, respectively, to conduct the simulation of our system with the proposed CNN ~~architecture.~~ architecture (Figs. 3 and 4).

Color ~~image spaces~~Image Spaces

Firstly, with a facial image for experiment, we use ~~Matlab~~ MATLAB tool to crop face part and resize it to ~~48x48.~~ 48×48 . Then we transform it from ~~3 color~~ three-color RGB space to ~~gray scale~~ grayscale space by color-transforming formula [7]:

$$BW = \frac{8*G+5*R+3*B}{16} \quad (1)$$

Once getting facial image in gray scale, we turn it into single ~~floating point 32 bit form then~~ floating point 32-bit form; then, we have bitstream input ready for simulating by hardware description language.

Design IP Cores

In this section, we describe in details the designs of I/O and IP core modules by Verilog. The essential thing for each hardware design is the trade-off between accuracy and device's resource. However, this architectural proposal only simulates with computations in ~~term~~ terms of single floating point 32-bit ~~data.~~ data (Fig. 5).

After training process, the dataset forms values used to compute convolution. Within those values, there is a concept called ~~Kernel~~ kernel matrix and bias coefficient. Kernel matrix is a set of real values weight ~~(w)~~ (w) , of size ~~3x3~~ 3×3 and it and bias ~~(b)~~ (b) coefficient is also real. Then, output ~~(y)~~ (y) of the convolution is determined by input ~~(x):~~ (x) [8]:

$$y_{ijm} = \sum_{k=0}^{K-1}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1}x_{i+p,j+q,k}^{l-1}w_{pqkm} \mathop{\Sigma}\limits_{k=0}^{K-1}\mathop{\Sigma}\limits_{p=0}^{H-1}\mathop{\Sigma}\limits_{q=0}^{H-1}x_{i+p,j+q,k}^{l-1}w_{pqkm} + b_{ijm} \quad (2)$$

Computations are ~~conducting~~ conducting, while ~~Kernel~~ kernel matrix is considered a mask and it strides on every layer of input data from left to right, from top to bottom. Kernel matrix is so-called ~~4-dimension~~ four-dimensional matrix, described ~~at Fig. 6 Where,~~ in Fig. 6, where n is size of input feature map and m is size of output feature map. These real values will be transformed to ~~32-bit~~ 32-bit single floating point form and then taken into ~~simulation.~~ simulation (Fig. 7).

IP ~~Core~~ core of ~~Fully Connected~~ fully connected layer is designed as identically as IP ~~Core~~ core of ~~Conv~~ conv layer.

Finally, we connect designed IP ~~Cores follow order: [Conv1-Relu1-Pool1-Conv2-Relu2-Pool2-Conv3-Relu3-Pool3-Conv4-Relu4-Pool4-FC1-FC2] respectively then we have Top Level IP Core Fig. 10~~cores in the following order: [Conv1-Relu1-Pool1-Conv2-Relu2-Pool2-Conv3-Relu3-Pool3-Conv4-Relu4-Pool4-FC1-FC2], respectively; then, we have top-level IP

core (Fig. 10).

At stage 1, we use 2 memory blocks as ROMs for storing data, the data: The first one stores testing image data, and the other store stores set of weights including weight and bias, bias; both of them have single floating-point floating point format. The set of weights is controlled by 2 port ports, enable_weight and enable_bias enable_bias, from Convolution convolution module to determine convolution data and tranfer transfer them to Convolution convolution module waiting for the next set of weights to execute block convolution. Results after executing convolution are transfered transferred to the next layer Max-pooling, max pooling; in this module module, comparison finding out maximum value is conducted made at the same time in order to minimize size of output volume as Fig. shown in Fig. 8.

At next stages, stage 2, stage 3 and stage 4 is are conducted similarly as Fig. shown in Fig. 9. There is difference from stage 1, input 1: Input volume of these convolution modules is from Max max pooling's output volume of last stage, not from memory block like stage 1. For the rest, order of computing and weights controlling from memory block is as similar as stage 1. Notice, Relu layer's function is integrated to convolutional module since it simply changes negative values to 0 and retains positive values after they were computed convolution. convolution (Fig. 10).

Result

There are three kinds of hardware resource on FPGA: adaptive logic modules (ALMs), digital signal processing (DPSs) and RAM blocks. In this work, we used Quartus Prime version 17.1 to synthesized RTL design for DE-10 standard board. The com-parison comparison result has been received and display is displayed in the following Table VI. Fig. Figure 11 shows the simulation result of our systems. systems (Table 3).

Conclusion

This paper proposes an efficient structure of neural network and data pre-processing preprocessing method and a baseline architecture for facial expression recognition. However, in the original purpose, we decided to simulate this architecture on hardware where the trade-off between accuracy and resources are is concerned. Keep in mind that the designs in this paper is an architecture with many layers composed with Verilog, so it consumes a huge amount of resources as a drawback. Therefore, our next target is to improve the system by applying fixed-point data-type fixed point data type simultaneously and only implement the Conv conv layer on hardware; the other layers will be implemented on software in order to reduce FPGA resource consumption.

490049_1_En_33_Chapter_OnlinePDF.pdf

Acknowledgements

This research is funded by Ho Chi Minh City University of Technology – VNU-HCM under grant number SVKSTN-2019-DDT-02 . SVKSTN-2019-DDT-02. We gratefully acknowledge their support,support.

References

1. M. Babaee, D. Dinh, G. Rigoll (2017). "A Deep Convolutional Neural Network for Background Subtraction" Babaee M, Dinh D, Rigoll G (2017) A deep convolutional neural network for background subtraction

2. Matsumoto David, Hwang Hyi Sung (2011) Reading facial expressions of emotion. Psychological Science Psychol Sci Agenda 25(5):10–18

3. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436

4. K. Sato, C. Young, and D. Patterson, "An Sato K, Young C, Patterson D (2017) An in-depth look at googles first tensor processing unit (tpu)," (TPU) Google Cloud Big Data and Machine Learning Blog, May 2017. [Online]. Available: https://cloud.google.com/blog/big-data/2017/05/ an-in-depth-look-at-googles-first-tensor-processing-unit-tpu https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

5. I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee et al. (2013) "Challenges in representation learning: A

5. Goodfellow IJ, Erhan D, Carrier PL, Courville A, Mirza M, Hamner B, Cukierski W, Tang Y, Thaler D, Lee D-H et al (2013) Challenges in representation learning: a report on three machine learning contests." contests. In: Conference on Neural

Information Processing. neural information processing. Springer, Daegu, Korea. 2013: 117-124

6. Arriaga, M. Valdenegro-Toro, and P. Plöger, "Real-time Convolutional Neural Networks for Emotion and Gender Classification.," CoRR, vol. abs/1710.07557, 2017

7. Kamlesh Lakhwani, Murarka, Narendra Singh Chauhan (2015). "Color Space Transformation for Visual Enhancement of noisy color Image"

8. Cheng Guojian,Ma Wei,Wei Xinshan, Rong Chunlong,Nan Junxiang, Korea, pp 117–124

6. Valdenegro-Toro AM, Plöger P (2017) Real-time convolutional neural networks for emotion and gender classification. CoRR, vol. abs/1710.07557

7. Kamlesh Lakhwani M, Chauhan NS (2015) Color space transformation for visual enhancement of noisy color image

8. Guojian C, Wei M, Xinshan W, Chunlong R, Junxiang N, Research of rock texture identification based on image processing and neural network, Journal of network. J Xi'an Shiyou University(Natural Science Univ (Nat Sci Edition)

9. Hanh Phan-Xuan, Thuong Le-Tien, Sy Nguyen-Tan (2019). "FPGA Platform applied for Facial Expression Recognition System using Convolutional Neural Networks"

10. Amine Horseman. (2007) "SVM for Facial Expression Recognition." Phan-Xuan H, Le-Tien T, Nguyen-Tan S (2019) FPGA platform applied for facial expression recognition system using convolutional neural networks

10. Horseman A (2007) SVM for facial expression recognition. A demonstrate project using SVM.SVM

Fig. 1 Proposed CNN architecture

Fig. 2 Validation accuracy

Fig. 3 Proposed FPGA architecture

Fig. 4 Convolutional Layer layer flowchart

Fig. 5 I/O Conv Layer conv layer IP Corecore

Fig. 6 Kernel Matrixmatrix

Fig. 7 I/O Max Pooling Layer IP Coremax pooling layer IP core

Fig. 8 Connection Stage stage 1

Fig. 9 Connection Stage stage 2

Fig. 10 I/O Top Leveltop level

Fig. 11 Simulation results

Table 1 Classification in Dataset dataset FER-2013

No.	0	1	2	3	4	5	6
Expressionangry	Angryfear	Feardisgust	Disgusthappy	Happysad	Sadsurprise	Surpriseneutral	Neutral

Table 2 Optimized classification in Dataset dataset FER-2013

No.	0	1	2	3	4	5
Expressionangry	Angry	fear & Fear and disgust	happyHappy	sadSad	surpriseSurprise	neutralNeutral

Table 3 Comparison between our design and the reference design

No.	Comparison object	Reference design [9]	Anonymous [10]	Our design
1	Accuracy	60.3%	59.8%	65%
2	DSPs	152		108
3	Clock frequency~~200MHz~~	200 MHz	~~200MHz~~	200 MHz
4	Signal appears output	After 2.404s		After 2.079s