Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40)
April 29 – May 2, 2019, Leuven, Belgium

FPGA Platform applied for Facial Expression Recognition System using Convolutional Neural Networks

Hanh Phan-Xuan[a], Thuong Le-Tien[b,*], Sy Nguyen-Tan[c]

[a]*HCMC Uni. of Technology, 268 Ly Thuong Kiet, Dist10, HoChiMinh City (postcode:70000), VietNam*
[b]*HCMC Uni. of Technology, 268 Ly Thuong Kiet, Dist10, HoChiMinh City (postcode:70000), VietNam*
[c]*HCMC Uni. of Technology, 268 Ly Thuong Kiet, Dist10, HoChiMinh City (postcode:70000), VietNam*

Abstract

Emotion being a subjective thing, leveraging knowledge and science behind labeled data and extracting the components that constitute it. With the development of deep learning in computer vision, emotion recognition has become a widely-tackled research problem. In this work, we propose a Field Programmable Gate Array (FPGA) architecture applied for this task using independent method called convolutional neural network (CNN). The emotion recognition block receives the detected faces from a video stream by using VITA-2000 camera module and process the image data with the trained CNN model. The architecture is implemented on a Zynq-7000 All Programmable SoC Video and Imaging Kit. Once we have trained a network, weights from the Tensorflow model will be convert as C-arrays, to be used in Vivado HLS. After having the weights as C arrays, they can be implemented to FPGA system. We can also test the functionality of the CNN entirely, by compiling the design with C++ compiler. This method was trained on the posed-emotion dataset (FER2013). The results show that with more fine-tuning and depth, the CNN model can outperform the state-of-the-art methods for emotion recognition. We also propose some exciting ideas for expanding the concept of representational landmark features and sliding windows to improve its performance.

Keywords: CNN; Tensorflow model; Vivado HLS; FPGA; FER2013.

1. Introduction

The exponential growth of data science during the recent decade motivates for innovative methods to extract high semantic information from raw sensor data such as videos, images and human speech sequences [1][2]. Among the

* Thuong Le-Tien. Tel.: +84-903-787-989.
E-mail address: thuongle@yahoo.com

proposed methods, Convolutional Neural Networks (CNNs) [3][4] have become the real standard by delivering good accuracy in many applications related to machine vision supported by machine learning or deep learning (e.g. classification [5], detection [6], segmentation [7]) and speech recognition [8]. Although these models are limited in the ability to only identify some basic emotions in millions of combinations daily complex emotions of people, discrete emotions models are still preferred and most common. In this paper, we use a model of Deep Learning to be able to identify the emotions simultaneously deployed on an FPGA platform. CNN [4] is one of the popular methods to best serve the field of image processing and biometric identification. Use of the CNN in facial recognition opens up opportunities for deep learning development. Model implemented on FPGA system will reduce the cost of computation to meet real-time processing needs.

Field-Programmable Gate Arrays (FPGA) are applied to reduce computational cost with hardware parallelism, therefore a deep learning model can be implemented. The majority of the traditional methods have used hand-crafted features or shallow learning. However, the natural emotion contests in the world are organized. These competitions have collected a wealth of diverse and comprehensive data on human emotions that promote the introduction of new effective methods of identifying emotions such as CNN. In the meanwhile, due to the dramatically increased chip processing abilities (e.g., GPU units) and well-designed network architecture, studies in various fields have begun to transfer to deep learning methods, which have achieved the state-of-the-art recognition accuracy and exceeded previous results by a large margin. Likewise, given with more effective training data of facial expression, deep learning techniques have increasingly been implemented to handle the challenging factors for emotion recognition in the wild. This growth comes at the price of a large computational cost as CNNs [4]. As a result, dedicated hardware is required to accelerate their execution. Graphics Processing Units (GPUs), are the global used platform to implement CNNs [4] as they offer the best performance in terms of pure computational throughput. Nevertheless, in terms of energy consumption, Field-Programmable Gate Array (FPGA) solutions are known to be more power efficient (vs GPUs). As a result, numerous FPGA-Based CNN [4] accelerators have been proposed, targeting embedded systems. While GPU implementations have demonstrated state-of-the-art computational performance, CNN acceleration is incontinently moving towards FPGAs for two reasons. Firstly, recent motivation in FPGA technology put FPGA performance within striking distance to GPUs with a reported equivalent performance. Second, recent trends in CNN [4] development increase the performance of CNNs and use energy efficient model. As a result, next generation CNN [4] accelerators are expected to have better computational throughput.

To achieve both goals of recognizing facial expression from candid images and setting a FPGA Based neural network, we propose to use deep learning methods. In our learning based approach, we propose to use convolutional neural networks (CNNs) without extra features such as histogram of oriented gradients (HOG), which in the past have been proved to be effective in image classification. Compared with the feature based approach, since the features are automatically learned from images, we expect that the CNN-based approach would be able to keep performance in recognition. It uses only the raw pixels of images for training, or if it's better to feed some extra information to the CNN (such as face landmarks or HOG features). The results show that the CNN can perform well in FPGA. The algorithm was implemented on a Zynq-XC7Z020 FPGA, and this architecture can recognize 500 faces in a frame, which make it possible for a real time system. As a summary, this paper has the following contributions: A FPGA platform for facial expression recognition using convolutional neural networks. The results have been shown the effective approach in recognizing facial expression of candid images.

1.1. Hardware Description Language and Register Transfer Level

These processes describe combinational logic, basic arithmetic operations as well as registers, and are driven by the rising and falling edges of a clock signal. RTL descriptions are very close to the logic gates and wires that are actually available in the underlying FPGA or ASIC technology, and therefore the hardware that results from RTL synthesis can be closely controlled.

1.2. High-Level Synthesis

High-Level Synthesis (HLS) tries to lower this barrier to entry by enabling designers to specify their algorithms in a high-level programming language such as C, C++ or SystemC. Many implementation details are abstracted away and handled by the HLS compiler, which converts the sequential software description into a concurrent hardware description, usually at RTL level.

2. Proposed system

2.1. Dataset

For emotion recognition, several datasets are available for research, varying from a few hundred high resolution photos to tens of thousands smaller images. The main we will discuss is the FERC-2013 [9] [10].

The data of FER consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

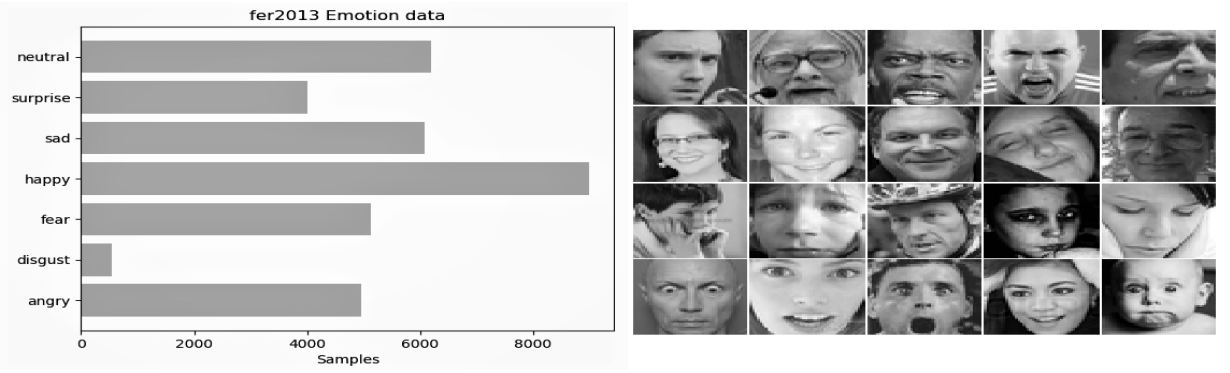

Fig. 1. Structure the training set.

2.2. Convolution Neural Network

2.2.1. Convolutional Layers and Matrix Multiplication

The central operation to be accelerated is the 2D convolution of multiple input feature maps with a number of small filter kernels. The two-dimensional convolution of an input image and a filter can be intuitively understood as the result from sliding the filter over the input image, and taking the dot product between the filter and the pixels underneath at each possible filter position. For a filter of $< A, B >= \sum_{i=0}^{n-1} A_i.B_i =$ size k × k, each dot product $A_0 \cdot B_0 + A_1 \cdot B_1 + \cdots + A_{n-1} \cdot B_{n-1}$ requires k^2 multiplications and additions. The 2D convolution between k × k filter F and H × W input image I yields output image O with

$$O_{(y,x)} = \sum_{j=-k/2}^{k/2} \sum_{i=-k/2}^{k/2} I_{(y-j,x-i)}.F_{(j,i)} \tag{1}$$

under the assumptions that k is an odd integer and the input image is appropriately zeropadded, i.e. $I_{(y,x)} = 0$ for all pixels outside of the valid image area W × H. In convolutional layers there is not a single input image, but a three-dimensional stack of chin input images called input feature maps $I^{(ci)}_{(y,x)}$. The convolutions then produce a stack of ch_{out} output images, called the output feature maps $O^{(co)}_{(y,x)}$ by applying a bank of filters $F^{(ci,co)}$. Under the above assumptions, a convolutional layer computes

$$O^{(co)}_{(y,x)} = \sum_{ci=0}^{ch_{in}-1} \left(\sum_{j=-k/2}^{k/2} \sum_{i=-k/2}^{k/2} I^{(ci)}_{(y-j,x-i)}.F^{(ci,co)}_{(j,i)} \right) = \sum_{ci=0}^{ch_{in}-1} \langle I^{ci}_{\substack{y+[k/2]...y-[k/2] \\ x+[k/2]...x-[k/2]}}, F^{(ci,co)} \rangle \tag{2}$$

for every output pixel (y, x) and every output channel co, which amounts to a total of $n_{MACC} = H \times W \times ch_{in} \times ch_{out} \times k^2$ multiplications and accumulations. Despite requiring a high computational effort, the mathematical operations behind convolutional layers are not complex at all, and offer a lot of opportunities for data reuse and parallelization.

This approach transforms the 2D convolution into one large matrix multiplication. For this, each local input region (the image region underneath each possible filter location) is stretched out into a column vector, and all the column vectors are concatenated to form a matrix C. Since the filter's receptive fields usually overlap, every image pixel is replicated into multiple columns of C. The filter weights are similarly unrolled into rows, forming the matrix R. The 2D convolution is then equivalent to a matrix product RC, which can be calculated very efficiently using highly optimized linear algebra (BLAS) routines which are available for CPUs, GPUs and DSPs. For a small 3×3 filter, matrix C is already blown up by a factor of 9 compared to the original input image. This makes it necessary to split the problem into a number of overlapping tiles, and later stitch the results back together, which artificially increases the complexity of the problem. On FPGAs and in ASICs, matrix multiplications can be efficiently implemented with a systolic architecture. A suitable systolic array consists of a regular grid of simple, locally-connected processing units. Each of them performs one multiplication and one addition, before pushing the operands on to their neighbors. Thanks to the locality of computation, communication and memory, these architectures are very hardware-friendly.

The Matrix Multiplication is well suited for general-purpose architectures such as GPUs. They are especially efficient for large problem sizes and batched computation. However, their additional memory consumption and the resulting need for tiling and re-stitching introduce artificial memory and computation requirements, which reduce the resource efficiency of the architecture. Our focus on the regular, well-optimized CNN further eliminates the need to support all kinds of different parameter combinations. Therefore, we believe the direct 2D convolution approach as formulated in eq. (1)(2) to be the most efficient way to implement an FPGA-based accelerator, regarding both memory and computational requirements.

2.2.2. CNN architecture

Fig. 2 shows the architecture designed, which consists of three convolutional layers, 2 max-pooling layers and 2 fully connected NNs layers.

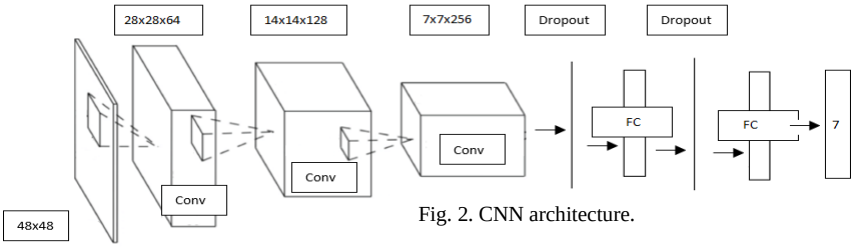

Fig. 2. CNN architecture.

3. FPGA implementation

3.1. The proposed Implementation

The face recognition architecture was implemented on a Zynq-7000 All Programmable SoC Video and Imaging Kit. A Zynq-7000 All Programmable SoC Video and Imaging Kit contains a Zynq-XC7Z020 FPGA with 220 MHz dual-core Cortex-A9 microprocessor, DDR3 component memory, a tri-mode Ethernet PHY, general purpose I/O, and two UART interfaces. Other features can be supported using VITA-57 FPGA Mezzanine Cards (FMC) attached to either of two Low Pin Count FMC connectors (LPCs). Fig. 3 shows the proposed algorithm.

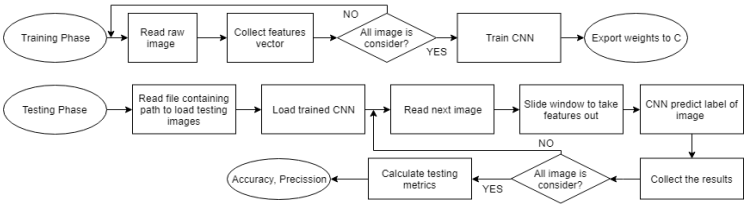

Fig. 3. Algorithm log of the proposed implementation.

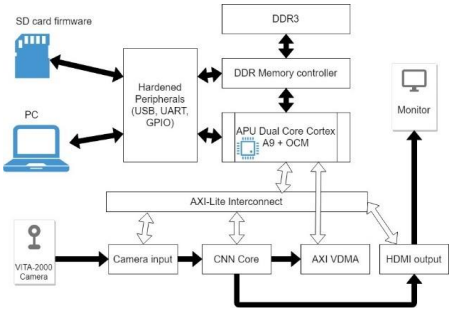

Fig. 4. FPGA architecture

The architecture of the proposed system is built-up from eight main parts as can be seen in Fig. 3. These are the ARM Processor, Camera input, High Definition Multimedia Interface (HDMI) output, the Advanced Extensible Interface (AXI) Video Direct Memory Access (VDMA) Controller, the AXI-Lite Interconnect, the Memory Controller, DDR3 and the CNN Core.

The ARM Processor communicates with the host computer via Gigabit Ethernet and controls the data-flow on the AXI-Lite and AXI buses. Furthermore, it pre-calculates the required trained weights for the CNN Core. The Memory Controller and the DDR3 are responsible to store the human faces extracted from the video stream, trained weights and also the sub and final results of the classification process. The data transfer between the CNN Core and the Board Memory is handled by the AXI VDMA Controller. In the memory, the data is stored sequentially. The CNN Core computes the algorithmic steps of the CNN method. In the first step it calculates the face features to detect faces in a frame. In the second step the face will be serialized into a vector. As the third step the result vector is computed after passing CNN Core. In the fourth step the softmax function is computed and label will appear. In recognition mode, label of input image will be calculated. Then, images after passing the CNN core processor block are labeled and displayed on screen. In the CNN Core only the data for actual step is stored, to minimize the BRAM memory usage. The AXI-Lite Interconnect provides the connection between parts of the system.

3.2. FPGA simulation

This is the last and most difficult process of using the training model. Xillinx module is an embedded system that cannot meet the requirements of high-level languages such as python. Vivado HLS allows the use of a number of built-in functions to simulate training and deployment models in FPGA systems by extracting the IP core Fig. 5. The requirement now is that this model can be understood by the FPGA system. The whole weight of the trained model to hexadecimal so that Vivado HLS can be used independently while facilitating the deployment of the trained model to the hardware. Vivado HLS included as a no cost upgrade in all Vivado HLx Editions, accelerates IP creation by enabling C, C++ and System C specifications to be directly targeted into Xilinx programmable devices without the need to manually create Register Transfer Language (RTL). We decided to use Vivado HLS to perform simulation results for demonstrating that the system with the required precision could work well in the FPGA platform and compare results to other platforms. Firstly, Vivado HLS and Vivado execute each IP block for processing system, then the result after passing the data through the IP block was simulated by creating a test file consisting of data inputs and declarations of inputs and outputs port. Data will be passed through processing block. If the output is satisfactory, the results show that the IP block can work well in the FPGA environment.

After the network has been fully trained we call the network back then start pouring the weight down to the header file (.h). Hardware parameters such as target frequency, target FMpS are specified for simulations performed on Vivado HLS. The overall architecture of parameterized modules is described as below. VITA Receiver contains a task of reading image or data from VITA camera, which generates addresses to read the input data in each clock cycle, and a scheduler, which caches the input data and feed the data into computation kernels. The main computation kernel is image pipe. This task is received the image data from VITA camera through VITA receiver and starting Emotion Recognition process. DDR memory has an address generator for storing output data into RAM. APU Dual Core Cortex A9 + OCM is the main scheduler of this system. The combination of three blocks APU Dual Core Cortex A9 + OCM, AMBA Switches and AXI Lite Interconnect is used to operate the system.

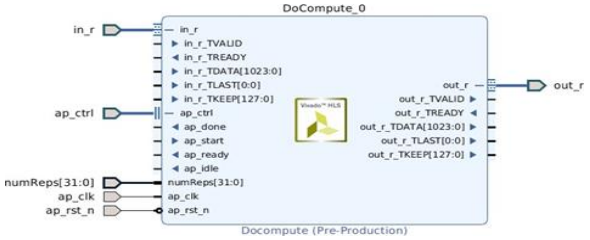

Fig. 5. Vivado HLS IP core.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{4}$$

$$F_{score} = \frac{2 * Recall * Precision}{Recall + Precision} \tag{5}$$

$$Precission = \frac{TP}{TP + FP} \tag{6}$$

The main processor Arm core receive data from master and write control or data signal to slave. The AXI Lite Interconnect connects one or more AXI memory-mapped master devices to one or more memory mapped slave devices. After finishing all task, processed data will be transfer to HDMI Output block to display to HDMI screen.

The current version Vivado HLS 2016.2 seems to introduce a bug into the RTL code for floating-point multipliers and adders, which causes the OPMODE input port oft the DSP slices involved to contain undefined values. While the functionality of the simulation model is not impaired, the undefined values cause the simulator to issue hundreds of warnings per clock cycle. This slows the co-simulation so much that even the smallest designs take hours to simulate.

4. Experimental result

This section is presented some metrics for evaluating the model. The result of classification will be in 4 possible cases, namely True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

In these four metrics, the Accuracy represents a general information of the models performance. However, in anomaly detection problems, the number of happy samples are typically greatly bigger than the all other ones. Consequently, if the model is simply set in a way that all of inputs are classified as happy, it can reach a spectacular accuracy. Therefore, Precision and Recall are exploited to overcome the shortcoming of Accuracy eq. (3). To be more clear, the Precision reflects how many samples that are exactly happy among samples indicated as happy, whilst Recall eq. (4) highlights the ratio of samples predicted as happy inside definite happy samples. Besides, we hope that there is a unique metric, representing ability of a model in the problem of skewed distribution detection, instead of two these metrics of Precision and Recall.

4.1. Testing

In the training process, the main target is to train the neural network so that it can learn optimal parameters itself in order to classify data into 7 classes. In this task, the momentum optimizer is used with the learning rate of 1e-1. After 50 epochs, the result is shown in Table 1. As it can be seen, the model does not occur overfitting phenomenon. This is caught by the dropout followed layers in the neural network. In the time of training, within each iteration, some of neurons in a layer will be randomly chosen to be deactivated. Moreover, their weights and biases are also not updated by back-propagation. This will lead to a fair training that no neuron is too active while the others are inactive. As a result, the training loss and accuracy will fluctuate because of the deactivation.

Table 1. Training and testing metrics

Metric	Training	Testing
Accuracy	60.5%	60.4%
Precision	61.2%	60.71%
Recall	60.43%	59.86%
F-Score	59.6%	59.85%

Table 2. FPGA used resource

Module	BRAM(18K)	DSP48E	FF	LUT
AddLast_1u_s	0	0	1138	334
AppendZeros	0	0	68	154
CONV2D_ACT_NoP	31	33	4160	10430
CONV2D_ACT_NoP_1	25	41	3526	7749
CONV2D_ACT_NoP_2	31	33	4159	10390
DENSE_ACT	25	33	7526	14316
CONTROL_S_AXI	0	0	74	104
ExtractPixels	0	0	1130	250
POOL2D_NoP	5	6	1076	3176
POOL2D_NoP_1	5	6	1075	3167
ReduceWidth_1	0	0	598	1193
Total Required/ Available	118/140	152/220	24524/106400	36924/53200
Utilization	84%	69%	12.3%	69%

4.2. FPGA resource

There are four kinds of hardware resources on FPGA: Look-up Tables Unit (LUT), Flip-Flop Unit (FF), Block-RAM Unit (BRAM) and Digital Signal Processing Unit (DSP). The result has been received and display in the following Table 2. The resource requirement of the CNN Core can be seen in Table 2. To maintain a low Block RAM memory usage, the required matrices are loaded from the DDR3 memory in every step and used immediately. The FPGA implementation of the CNN method has been tested with the FER dataset. Using these face images 60.3% recognition efficiency can be achieved, with a 400 faces/second throughput on 200 MHz clock frequency. The system costs of proposed works can be seen in Table 2. The system works with a very heavy and deep model to detect emotion, but a system like the XC7Z020 can still be found to show that the system has the cost advantage in computing. In addition, the system achieves the desired accuracy when compared to other methods that have been processed on a computer that is not an embedded system, which shows that the system has the structural advantage.

4.3. Compare to different model

After testing the proposed model, we continue comparing our performance to the others. The conventional methods, Support Vector Machine (SVM) [12], along with another CNN network [13] are chosen to make a detection comparison. This comparison uses accuracy metric obtained when testing on the FER2013 database. In Table 3, the proposed method stands at the second rank, which over-comes conventional one and left behind the SVM [14]. As regarding to the CNN methods, Shima et al. [15] used a powerful CNN model with 14 layers and a SVM classifier at the end of pipeline as well as they utilized the whole FER database for training and testing. However, our model implemented is not too different from the above model with the same standard set of data. On the other hand, our model has been deployed on FPGA and for satisfactory results shows that with a not too strong hardware we were able to successfully build a model with CNN not so deep but could reach high precision. FPGAs can continue to reduce computation time by parallel processing so the model has shown that it is possible to achieve higher efficiency. Our model just requires around 400 seconds for training on the CPU. Besides, because of the narrowness in the architecture, our proposed model is probably faster than other Deep networks in both of training and testing process, this network can be applied to an embedded system such as FPGA, while it can keep a high accuracy. Fig. 6 shows the simulation result of our systems. We installed the control signals and clock signals for the system, then the data was put into the Intellectual Property Core (IP) block to get the output signal. After 2.404s since the start signal is set to 1, the signal appears at the output. This shows that the system performs a snapshot of the image to meet the demand in real time. Processing speed is over 25 frames per second.

Table 3. Comparison between methods on FER2013 dataset.

Method	Accuracy	Number of test images	Number of Neurons
Shima [15]	62.4%	3589	256,512
Anonymous (SVM)[14]	59.8%	3589	-
Proposed	60.3%	3000	1568

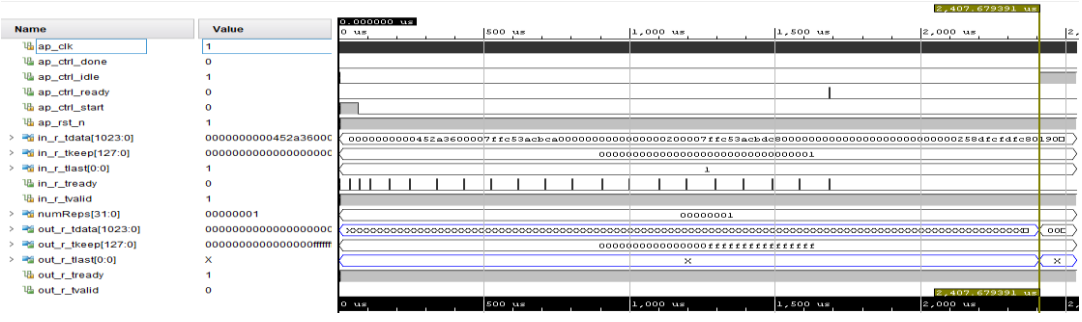

Fig. 6. Simulation results

5. Conclusion

In this paper, we proposed a low computational-cost and effective model to solve the problem of facial expression recognition which can be deployed in a FPGA. In which, a fully CNN network, along with relating components (e.g., activations, initialization, normalization, optimizer), was designed to classify emotions inside a face image. By conducting a light pattern, we can apply this model with different hardware as a specific FPGA. Besides, in this experiment, we compare our model to some baselines, including an existed CNN model [14] and SVM [15] method. Our model can achieve a noticeable detection accuracy of 60.3%. Although our model is not very high in the standard dataset but compared to other methods the accuracy is not too different and still good. In addition, our model demonstrates that it can be applied on an FPGA basis, thus opening up the development of deep learning models on similar embedded platforms. In the future we will apply some new methods of training CNN to improve the accuracy of the network and minimize the amount of computing, such as the application of human foot-prints and the export HOG features, along with the available features of the sample dataset. As a result, we expect the CNN architecture to be more complete and expandable for different hardware.

Acknowledgements

This research is funded by Ho Chi Minh City University of Technology – VNU-HCM, under grant number T-SDH-2019-1680943.

References

[1] D. Ververidis, and C. Kotropoulos. (2004) “Automatic speech classification to five emotional states based on gender information " *Proceedings of the EUSIPCO2004 Conference, Austria*: 341-344.

[2] Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., and Taylor. (2001) “Emotion recognition in human computer interaction " *IEEE Signal Processing magazine*, **Vol**. 18, **No**. 1: 32-80.

[3] Yann LeCun, Yoshua Bengio, and Georey Hinton. (2015) “Deep learning.” *Nature*. **Vol**.521: doi 10.1038/nature14539.

[4] Xavier Glorot, Yoshua Bengio. (2010) “Understanding the difficulty of training deep feedforward neural networks " *Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy. **PMLR9**: 249-256.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein. (2012) “Imagenet large scale visual recognition challenge.” *International Journal of Computer Vision*.: 211-252.

[6] Ross Girshick. (2015) “Fast R-CNN.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Boston, MA, USA, **CVPR**.15: 1440-1448.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. (2015) “Fully Convolutional Networks for Semantic Segmenta.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* Boston, MA, USA. **CVPR**.15: 3431-3440.

[8] Ying Zhang, Mohammad Pezeshki, Philmon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courvill. (2016) “Towards end-to-end speech recognition with deep convolutional neural networks.” *Proceedings of Interspeech conference*. San Francisco, USA.

[9] 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 report on three machine learning contests.” *Conference on Neural Information Processing. Springer, Daegu, Korea. 2013*: 117-124.

[10] B. Martinez and M. F. Valstar. (2017) “Advances, challenges, and opportunities in automatic facial expression recognition.” *Advances in Face Detection and Facial Image Analysis.Springer.2016*: 63-100.

[11] V. Nair, E. Hinton. (2010) “Rectified Linear Units Improve Restricted Boltzmann Machines” *Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel*, **ISBN**: 978-1-60558-907-7, 807-814

[12] H. Gunes, and B. Schuller. (2013) “Categorical and dimensional affect analysis in continuous input: Current trends and future directions.” *Image and Vision Computing, Butterworth-Heinemann Newton, MA, USA.* **Vol**.31: 120-136.

[13] Fuhai Li, Jinwen Ma, and Dezhi Huang. (2005) “MFCC and SVM based recognition of Chinese vowels.” *Proceedings of the 2005 international conference on Computational Intelligence and Security, Xi'an, China.* **Vol.2**: *812-819.*

[14] Amine Horseman. (2007) “SVM for Facial Expression Recognition.” *A demonstrate project using SVM*.: https://github.com/amineHorseman/facial-expression-recognition-svm/commit/aecd525367f6d77e5d274de7c6f0166d5bfa4bb9.

[15] Shima A., Azar F. (2016) “Convolutional Neural Networks for Facial Expression Recognition.” *Proceedings of the Stanford University report.*: 3-4. http://cs231n.stanford.edu/reports/2016/pdfs/005_Report.pdf.