International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 3, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v3n2.31
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Research and Implementation of a Facial Emotion Recognition System Based on ZYNQ

Jiali Jiang

School of Electronic Information, Southwest University for Nationalities, China

ABSTRACT

Emotion is the psychological and physiological response of a person or animal to a specific stimulus or event, and it can also have an impact on cognition, decision-making, and behavior. Research on emotion recognition has wide applications in various fields, such as dialogue and communication with machines, and in the medical field. At present, most emotion recognition systems are based on PC or embedded platforms. Compared with traditional emotion recognition platforms, FPGA has good parallel computing ability and can process multiple data streams simultaneously, making it suitable for complex pattern recognition in emotion recognition. This article chooses ZYNQ as the development platform, combined with the collaborative ability of FPGA and ARM processors, to build a complete hardware application system that can carry neural networks. Secondly, in response to the problems of complex computation and high storage resource consumption in existing neural network computing frameworks, this paper adopts a lightweight neural network MobileNetV3 and implements and improves it. The model is encapsulated and ported using the Vivado HLS development tool and C++. The hardware IP core is called on the hardware platform to achieve ZYNQ based facial emotion classification and recognition, and the power consumption of hardware resources is analyzed.

KEYWORDS

Emotional recognition; ZYNQ; Face detection; Neural networks

1. INTRODUCTION

Since the 1990s, researchers engaged in facial studies have shown increasing enthusiasm for the field of facial emotion recognition. Some research institutions in this field have also received funding support. Economically developed countries led by the United States and some developing countries have established specialized research institutions to conduct facial emotion recognition research. Currently, overseas research methods for facial emotion recognition and analysis can be categorized into several aspects: image-based methods, geometric feature-based methods, model-based methods, and motion-based extraction and recognition methods. Due to its low power consumption and high speed, FPGA has significant advantages in emotion recognition and has thus received widespread attention. Many relevant research achievements have emerged abroad. For instance, Su et al. implemented a facial emotion processing system on the FPGA platform. The Har team connected the FPGA platform with a facial clarity controller, sending collected facial emotions to the corresponding IP for processing and obtaining emotional classification outputs. Meanwhile, foreign scholars have implemented an expression recognition system on the FPGA platform through matrix linear analysis methods. Jin et al. applied a linear regression model on mobile devices, achieving a new application of facial emotion recognition and accelerating the development of emotion recognition technology. The Saini team accomplished emotion recognition based on support vector machines, using CK+ emotion data for training and testing. Sauram and Jask, among others, categorized facial expressions

into happy, sad, and neutral classes on an FPGA-based system, achieving good hardware acceleration effects. This system is currently applied in clinical medicine and facial detection fields.

2. RELATED WORK

The main focus of this chapter is to discuss and validate algorithms relevant to identifying corresponding emotions. This includes both facial detection algorithms and emotion recognition algorithms. These studies are essential for developing a robust system. The appropriate algorithms for facial emotion detection have a significant impact on obtaining corresponding experimental results. The paper explores the principles and applications used in these algorithms and provides a detailed analysis.

2.1. Traditional Method

Figure 1 depicts the traditional approach to facial emotion recognition, which combines research problems with feature extraction. Useful features are manually extracted and inputted into a classifier for learning.

Figure 1. Flow chart of traditional face recognition

Firstly, using image processing techniques or machine learning algorithms, the image undergoes facial detection to determine the position and size of detected faces within the image. For each detected face, facial expression features representing the expression are extracted from the image. Utilizing these extracted facial features, along with prior knowledge or trained models, facial expressions are analyzed and classified to determine the corresponding emotion category. Based on the analysis of features, each facial image is assessed to identify the expressed emotion, enabling emotion recognition and classification.

2.2. Deep Learning Technology

The core idea of deep learning involves multilayered nonlinear transformations. By extracting features across multiple layers, deep models automatically capture more abstract and diverse feature information from raw data, enabling more accurate and reliable tasks such as classification and prediction.

Deep learning technology encompasses various models, each with distinct structures and characteristics suitable for different tasks and application domains.

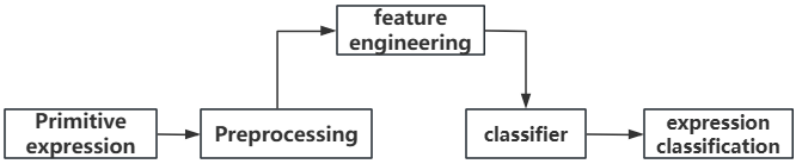

Figure 2. Deep learning emotion recognition process

2.3. Emotion Recognition Algorithm

The CNN-LSTM algorithm utilizes a CNN network to extract spatial features from facial expression images, which are then input into an LSTM network for sequence modeling to capture temporal information, as depicted in Figure 3.

Specifically, within the CNN-LSTM model, CNN is employed for spatial feature extraction, while LSTM leverages temporal information to capture sequential correlations. This structure effectively breaks down the entire 2D image into smaller blocks, utilizing LSTM to address temporal issues within each block's time series. During training, the model parameters are continuously adjusted to optimize network structure, ultimately enabling accurate prediction of the emotional content corresponding to input images. Compared to other algorithms, the CNN-LSTM model has demonstrated strong performance in static image emotion recognition, particularly on datasets such as FER (Facial Expression Recognition). It serves as an excellent baseline algorithm in this domain. Furthermore, the CNN-LSTM model can be extended through methods like multitask learning, simultaneously recognizing emotions and postures, thereby enhancing recognition efficacy. The CNN-LSTM process is illustrated in Figure 3.

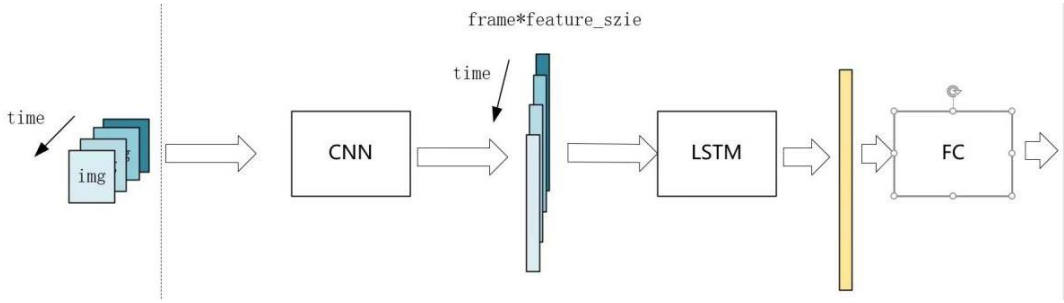

Figure 3. CNN-LSTM flowchart

2.4. Data Set Introduction

Figure 4. Partial images of FER2013 dataset

The PubFig dataset is a facial recognition dataset used in the field of facial recognition research. Released in 2004 by Olivier Duchenne et al., it comprises over 58,000 facial images of celebrities from various domains such as actors, politicians, musicians, and writers.

Each individual has multiple images sourced from the internet, depicting different angles and expressions. Each image is annotated with human-labeled tags identifying specific individuals. Due to its extensive size and high-quality annotations, the PubFig dataset has become a crucial resource in facial recognition research, providing robust support for evaluating and comparing various algorithms' performance. The PubFig dataset is illustrated in Figure 4 above.

3. MOBILENETV3 FACIAL EMOTION RECOGNITION

The core of this chapter focuses on the design and implementation of a facial emotion recognition system based on MobileNetv3. Utilizing the MobileNetv3 network architecture enables lightweight feature extraction and classification. By processing facial images, the system achieves rapid and accurate recognition of facial emotions on the PyCharm platform.

3.1. CNN Image Preprocessing

(1) Read images.

(2) Resize or crop: Scale images to a fixed size or crop them as needed to ensure uniformity in size and aspect ratio for each image.

(3) Image normalization: Process images by shifting pixel values to have a mean of 0 and a standard deviation of 1, thereby standardizing them into a normal distribution.

(4) Data augmentation: Expand the dataset by applying desired operations to images. Data augmentation can include adding noise, adjusting brightness and contrast, among other techniques.

(5) Batch processing: Group images into batches of size batch size and feed them into the model for training.

(6) Encapsulate preprocessing steps: Wrap the image processing operations into a function and call it during training.

(7) Implement image preprocessing: Use APIs provided by TensorFlow, Keras, or other deep learning frameworks to implement CNN image preprocessing.

(8) Train the model.

These preprocessing steps enhance model accuracy and generalization performance, enabling convolutional neural network models to better adapt to various images.

3.2. MobilenetV3 Network Introduction

MobileNetV3 incorporates a series of optimization strategies tailored for lightweight network designs, including more efficient activation functions, improved network structures, and meticulously designed modules. It demonstrates versatile performance in tasks such as object detection and semantic segmentation. Due to its efficient computation and lightweight characteristics, MobileNetV3 is highly suitable for real-time applications like real-time image processing and recognition on mobile devices. Additionally, it introduces linear bottlenecks inverse residual blocks and integrates SE (Squeeze-and-Excitation) lightweight attention. MobileNetV3 consists of specific components including CONV, DW (Depthwise Convolution), PW (Pointwise Convolution), pooling layers, fully connected layers, and others, as illustrated in the diagram.

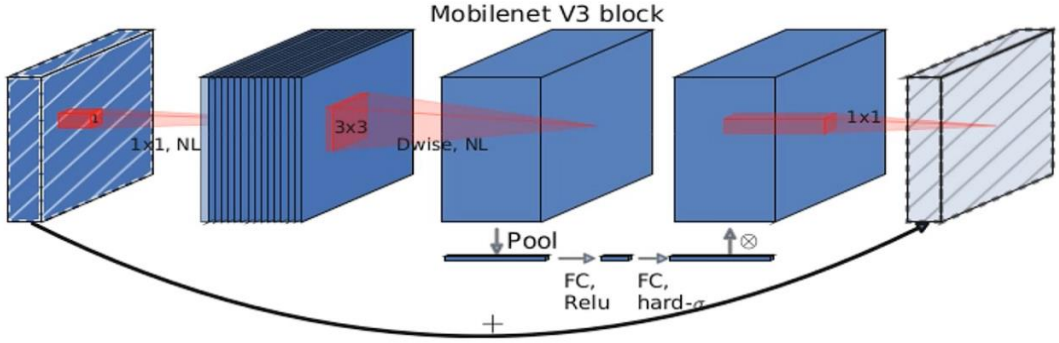

Figure 5. Mobile NetV3 Inverse Residual Structure Diagram

The SE (Squeeze-and-Excitation) module first performs global average pooling on each channel of the output feature matrix, resulting in a vector of the same number of channels. Next, these vectors pass through a squeeze layer (typically a fully connected layer) that reduces the size to one-fourth of the original number of channels. Subsequently, an excitation layer (another fully connected layer) follows, outputting a vector of the same size as the number of channels. Additionally, the activation functions used in these two fully connected layers are ReLU and h-swish.

The specific convolutional neural network structure of MobileNetV3 is detailed in Table 1.

Table 1. Convolutional Neural Network Structure of Mobile Net v3

Input	Operation	Ascending channel	output channel	SE	NL	Strides
$224^2 \times 3$	Conv2D, 3×3	-	16	-	HS	2
$112^2 \times 16$	Bneck, 3×3	16	16	√	RE	2
$56^2 \times 16$	Bneck, 3×3	72	24	-	RE	2
$28^2 \times 24$	Bneck, 3×3	86	24	-	RE	1
$28^2 \times 24$	Bneck, 5×5	96	40	√	HS	2
$14^2 \times 40$	Bneck, 5×5	240	40	√	HS	1
$14^2 \times 40$	Bneck, 5×5	240	40	√	HS	1
$14^2 \times 40$	Bneck, 5×5	120	48	√	HS	1
$14^2 \times 48$	Bneck, 5×5	144	48	√	HS	1
$14^2 \times 48$	Bneck, 5×5	288	96	√	HS	2
$7^2 \times 96$	Bneck, 5×5	576	96	√	HS	1
$7^2 \times 96$	Bneck, 5×5	576	96	√	HS	1
$7^2 \times 96$	Conv2D, 1×1	-	576	√	HS	1
$7^2 \times 576$	Pool, 7×7	-	-	-	-	1
$1^2 \times 576$	Conv2D, 1×1	-	1280	-	HS	1
$1^2 \times 1280$	Conv2D, 1×1	-	K	-	-	1

The primary reason for choosing MobileNetV3 to build a facial emotion recognition model is its lightweight network structure, which is well-suited for efficient operation on mobile devices. Designed with consideration for the limited computational resources of mobile devices, MobileNetV3 can maintain high accuracy while reducing computational burden, making it an ideal choice. This enables real-time facial emotion recognition on mobile devices without significantly impacting device performance.

3.3. Processing and Analysis of Data Sets

Here, comparative tests were conducted on different emotion datasets. Specifically, 70% of each dataset was used as the training set, with the remaining 30% split evenly between a validation set and a test set. The network was trained for 500 epochs with a batch size of 64 images. The initial learning rate was set to 0.01, and every 5 epochs after the first 30 epochs, the learning rate was decayed by 5%.

ReLU activation function was selected for the network. During training, the entire network underwent optimization through gradient descent. Throughout the training process, the model with the lowest validation set loss was chosen for testing on the test data [47]. Finally, experiments were conducted on PubFig, CK+, JAFFE, AffectNet, and Emotionet datasets, and recognition rates and accuracies across different emotion categories were obtained as shown in Table 2.

According to the experimental results, the emotion recognition system of this model performs exceptionally well on the PubFig dataset, achieving an average recognition rate of 97.58%. Compared to other models, this model exhibits higher recognition rates and more significant improvements.

Table 2. Accuracy of emotion recognition on different datasets based on MobilenetV3

Expression	PubFig (%)	CK+(%)	JAFFE (%)	AffectNet (%)	Emotionet (%)
Angry	97.28	92.16	91.65	89.75	90.16
Happy	98.23	95.14	90.50	93.20	85.09
Sad	98.45	95.60	92.50	91.25	89.23
Surprise	97.12	93.15	86.35	94.11	94.11
Fear	95.54	94.20	89.54	79.15	79.15

4. VIVADO HLS MOBILENET V3 NETWORK HARDWARE IP DESIGN

In the previous chapter, we completed emotion classification using the MobileNetV3 neural network system. In this chapter, we need to print and output the weight parameters used by the neural network. Subsequently, we will combine hardware description language (HDL) and the Vivado HLS tool to encapsulate the entire neural network into modules required for hardware. Configuration of the ports of the hardware neural network IP core in HLS will lay the foundation for implementing facial emotion classification on the Vivado platform.

4.1. Network Hardware IP Design Ideas

First, we need to extract the corresponding weight parameters of MobileNet V3. Using the HLS (High-Level Synthesis) tool, we'll convert the neural network model into hardware description language (Verilog HDL in this case). In this step, we must consider the computational logic, data flow, and memory requirements of each layer of the neural network.

Next, in Vivado, we'll create a top-level design connecting the ZYNQ processor with the neural network hardware IP core. We'll establish appropriate interface protocols (such as AXI bus) to enable communication between the processor and the hardware IP core. Input and output ports of the hardware IP core need to be configured in the HLS tool and optimized as necessary to fit the ZYNQ hardware IP structure. Ensuring correct port definitions is crucial for data transfer and control.

Additionally, in the HLS tool, we'll synthesize, optimize, and validate the hardware IP core to ensure correct functionality and meet performance requirements. Parameter adjustments will be made to enhance performance and efficiency.

Afterwards, in Vivado, we'll synthesize, implement, and route the entire design to generate bitstream files suitable for ZYNQ. Through these steps, we'll ensure that the design can successfully map onto the FPGA. The synthesized bitstream file will be loaded into the ZYNQ FPGA, enabling deployment and execution of the hardware IP core.

Finally, in XSDK, we'll write code to facilitate communication between the ZYNQ processor and the hardware IP core. This ensures that input data is correctly passed to the neural network hardware IP core for obtaining and processing classification results.

In summary, this approach successfully encapsulates the neural network into the ZYNQ hardware IP structure, enabling acceleration of neural network tasks on the FPGA. The overall architecture is depicted in Figure 6.

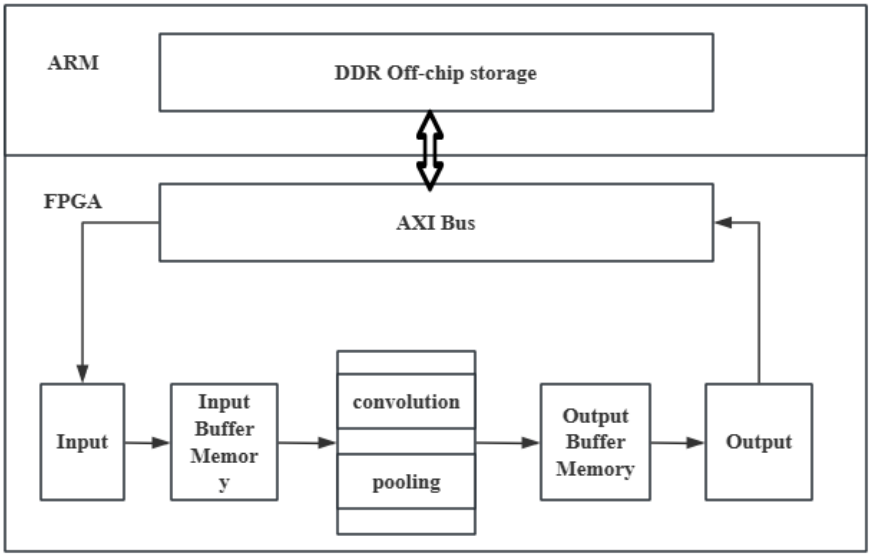

Figure 6. Hardware IP design structure diagram

4.2. Print and Adjust the Training Parameters of the Network

```
tensor([[[[ 9.5803e-02,  5.5239e-02,  1.0288e-01,  7.5048e-02,  1.6195e-01],
          [ 5.1581e-03,  2.3891e-01,  2.9071e-01,  7.2389e-02,  3.2870e-02],
          [ 1.2892e-01, -2.4238e-02,  2.2148e-02,  4.4590e-02, -1.7992e-01],
          [-8.1017e-02, -2.2018e-01, -1.9814e-01, -1.4945e-01, -1.9766e-01],
          [ 4.8322e-02, -2.7193e-03, -1.1966e-01, -4.9346e-02, -1.5816e-02]],

         [[-1.0290e-01, -2.5498e-02, -4.0403e-02,  1.3622e-02, -5.6298e-02],
          [ 2.1140e-02,  1.4635e-01, -1.2658e-02,  9.6755e-02,  8.4838e-03],
          [ 1.5997e-02, -3.9567e-02, -9.2227e-02, -1.3738e-01, -1.9495e-02],
          [-1.4476e-01, -2.8756e-02, -1.8236e-01, -6.7748e-02, -2.7482e-02],
          [-4.3342e-02, -9.1109e-02,  7.4808e-02,  3.6240e-02,  8.8675e-02]],

         [[-1.6678e-01, -2.8875e-03, -1.2045e-01,  4.2056e-02, -1.0014e-01],
          [ 9.2429e-02,  7.2163e-03,  1.5210e-01,  8.0952e-02,  7.4164e-02],
          [ 6.2761e-02,  1.4735e-01, -2.9008e-02,  1.0105e-01, -1.2899e-02],
          [-1.0417e-01, -1.2973e-02,  2.5826e-02, -8.4108e-02, -7.9515e-02],
          [-8.8542e-03,  3.5951e-02,  2.2815e-02,  7.3007e-03,  7.6190e-02]]],
            ......
```

Figure 7. Example of printed neural network parameters

By printing the training parameters of the MobileNet V3 network, we ensure that when deploying the model onto a hardware platform, the network structure and weights can be loaded correctly. This process also aids in further optimizing the model to meet specific deployment requirements. Here, we save data in float format, with each weight stored in 4 bytes. We separate the weight files for each layer, saving them as independent files named weightx.dat and biasx.dat. This approach facilitates using the #include directive in HLS development to read weight files, among other benefits. Figure 7 illustrates the parameters of the neural network printed for reference.

4.3. Hardware IP Design of Convolutional Module

The convolutional modules in MobileNet V3 include three types of operations: traditional CONV, DW, and PW. CONV represents standard convolution using conventional kernels to convolve input data, considering both depth and spatial information. DW refers to Depthwise Convolution, a component of depth separable convolution. DW performs independent spatial convolutions on each input channel, meaning each channel has its own convolution kernel without inter-channel interaction. This reduces parameter count and computational load. PW, or Pointwise Convolution, is another

component of depth separable convolution. PW involves applying 1x1 convolution kernels to combine output channels from the depth convolution. Pointwise convolution allows for feature combination and adjustment across different channels while preserving dimensions. Here, we encapsulate MobileNet V3's convolution modules—CONV, DW, and PW—into three distinct hardware IP cores based on their respective functionalities.

4.4. Hardware IP Design of Pooling Module

The convolutional neural network hardware IP interacts with the ARM via the AXI bus. Prior to launching the convolutional neural network hardware IP, the ARM terminal writes the network's structural information through the AXI_SLAVE bus. This includes parameters for the pooling layer such as LayerType, kernel size (kernel_size), stride (kernel_trim), and padding. The pooling operation module executes different operations based on the network structure information. Common pooling methods include max pooling and average pooling.

The pooling operation module uses a four-channel input, first determining the number and type of input data channels. It typically employs three color channels (red, green, blue) from RGB color images and an additional channel (such as depth information), considering inter-channel correlations and relevance. Features from different channels may vary in importance or correlation, allowing for channel weighting or selective processing of certain channels. Hardware parallelism is utilized to process data from multiple channels simultaneously, with FPGA employing parallel computing units or pipeline structures for handling data across different channels.

Next, the data transmission path and processing sequence within the hardware are determined to minimize data transmission and processing delays. Data is selectively read and written as needed to avoid unnecessary data transfers, employing strategies such as local caching or data reuse to reduce accesses to external memory. This approach optimizes memory usage and minimizes data transmission bandwidth consumption.

The design result of the pooling module in Vivado HLS is depicted in Figure 8, illustrating considerations for data storage in memory to maximize efficiency.

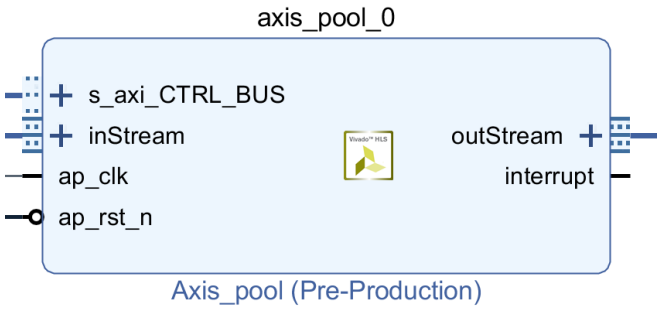

Figure 8. Design diagram of Vivado HLS IP for pooling module

4.5. Hardware IP Design of Fully Connected Module

Firstly, the fully connected layer computation unit is responsible for executing matrix multiplication and bias addition operations within the hardware design. Custom hardware logic is typically employed to accelerate matrix multiplication, enhancing computational performance and efficiency. Following the fully connected layer, the computed results are typically passed through a sigmoid activation function. The design of the data path and storage units involves handling data transfer and storage, including loading input data, transmitting intermediate results, and storing final outputs. Well-designed data paths can improve processor throughput and reduce latency. This approach optimizes the neural network's input processing flow, simplifies data management and maintenance,

and enhances overall system performance as illustrated in Figure 9, which shows the feature map input unit.

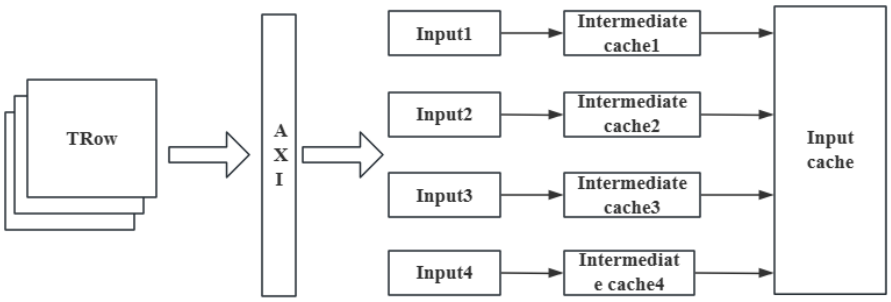

Figure 9. Input unit of feature map

To optimize the data interaction process between the ARM processor and the CNN hardware IP, several methods are employed. Before starting the CNN hardware IP, the ARM processor uses the AXI_SLAVE bus to write the address of the input interface. It calculates the number of data reads and the length of each read to fetch feature map data from DDR (Double Data Rate) SDRAM and store it in cache. Increasing the burst read length effectively reduces transfer time. Techniques such as pipelining optimized instructions, utilizing high-speed caches, and maximizing data transfer lengths are utilized to achieve more efficient data read and storage processes. This optimization aims to enhance the overall system performance and efficiency.Figure 10 in Vivado HLS illustrates the design result of the fully connected module.

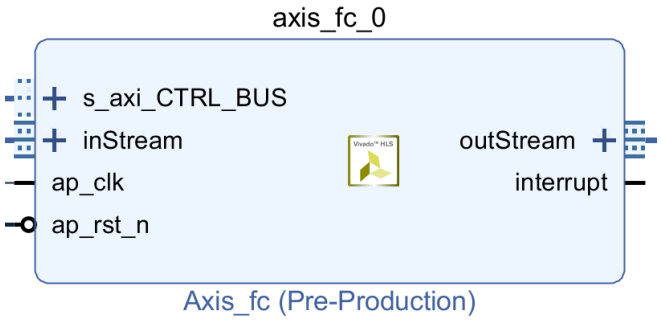

Figure 10. Design diagram of Vivado HLS IP for pooling module

5. EXPERIMENTS

5.1. The Test Result of the Upper Computer

After 500 epochs of training iterations, both the model's loss and accuracy exhibit a stable trend, as depicted in Figure 11. Subsequently, the results from training the Mobilenetv3 neural network model on a PC are shown in Figure 4.8, where the input image depicts a person smiling. Following computation through the entire neural network, probability values are obtained, classifying the image into the "happy" category, indicating detection of a person in a joyful state. Thus, within the Mobilenetv3 neural network structure, favorable classification results for images are achieved, enabling emotion detection functionality.

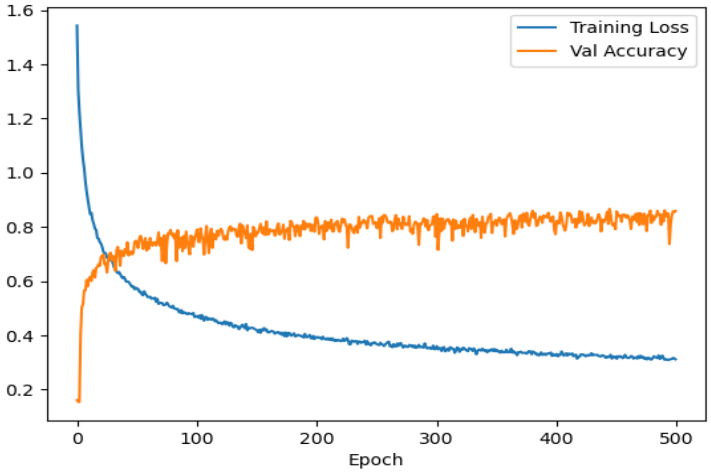

Figure 11. Loss and Accuracy

5.2. Image Capture and Display

First, initialize platform parameters, including setting up relevant hardware platforms and devices. Next, initialize the lookup table, which involves preparing the data tables or indices needed for use. Then, call HLS to generate RTL IP core files. When using the SDK, the generated driver functions and associated calling methods are fixed. Essentially, it initializes relevant structures and then calls functions to obtain results. If you need to find related driver functions, you can search within the SDK header files. Data passed in Xilinx driver functions typically uses the u32 (unsigned integer) data type. For floating-point data, it needs to be converted to unsigned integers for transmission because integers are more commonly used to represent data in hardware design.

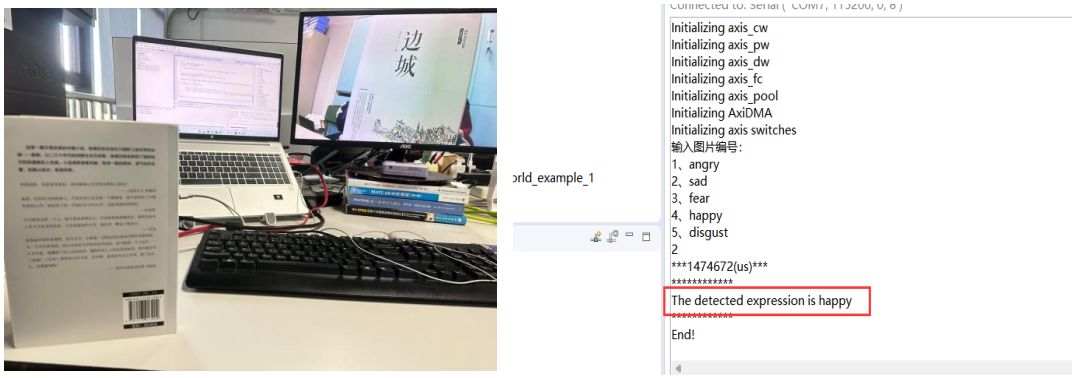

Figure 12. Image acquisition and display system experiment

5.3. ZYNQ Testing Result

This paper implements a facial emotion detection system based on ZYNQ, utilizing the PubFig dataset for emotion prediction and constructing the MobileNetV3 model based on convolutional neural networks. To enhance prediction accuracy and speed, the research team refined the MobileNetV3 algorithm.

In the experiments, the team trained and tested the model using the PubFig dataset, widely used in the field of emotion recognition, which includes facial expression images under various conditions and from diverse sources. Experimental results demonstrate that the team's improvements to the MobileNetV3 algorithm yielded satisfactory outcomes in emotion recognition. For expressions such as anger, happiness, sadness, surprise, fear, and undetected faces, the average recognition accuracy exceeded 70%, achieving a recognition speed of 32 frames per second. Compared to running the network in PyCharm, the accuracy improvement was negligible.

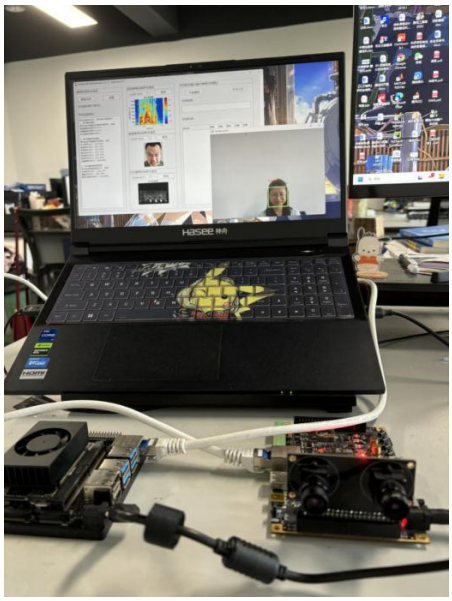

Figure 13. experimental result

5.4. Analysis of System Resources and Power Consumption

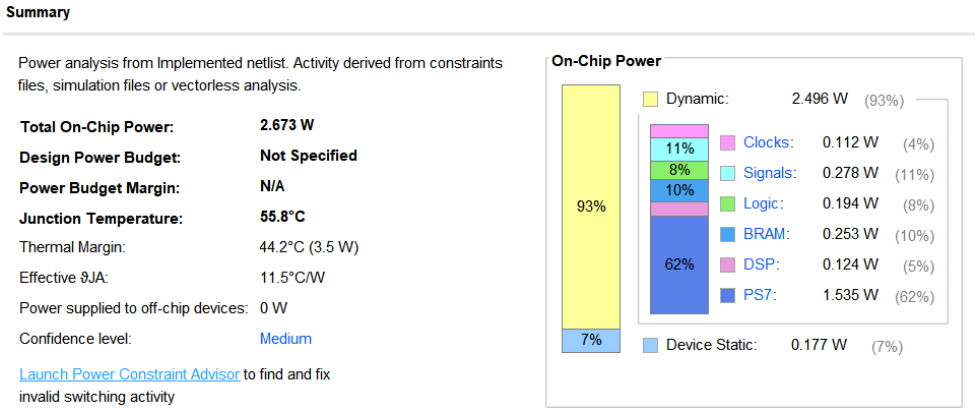

Figure 14. System resources and power consumption

The figure clearly shows the power consumption of the resources utilized during the experiment. The total power consumption is noted as 2.496W. Detailed breakdowns of various components are available in the report, highlighting that dynamic and static power consumptions are 93% and 7% respectively. Within the dynamic power consumption, the PS section accounts for 1.535W, constituting 62% of the total. Moreover, DSP and BRAM also consume relatively high-power resources.

The following table presents the resource utilization report generated after completing logic synthesis and layout in Vivado for configuring the accelerator's SOC system. The report details the usage of different resources including BRAM, LUT, FF, LUTRAM, and BUFG resources.

Table 3. Resource Occupation of ZYNQ7020

Resource	Utilization	Available	Utilization%
LUT	45127	53200	84.83
LUTRAM	1542	17400	8.86
FF	39280	106400	36.92
BRAM	92	140	65.71
DSP	148	220	67.27
BUFG	1	32	3.13

6. CONCLUSION

This paper designs a facial emotion recognition system based on Zynq. By employing FPGA hardware acceleration technology, it enhances image processing speed and integrates ARM for porting the Mobilenet V3 network model. The system achieves facial emotion recognition on FPGA hardware with effective results. The innovation lies in utilizing Zynq for image preprocessing, employing Vivado HLS for building and porting convolutional neural network models, thereby achieving hardware acceleration and accelerating development through software-hardware integration.

The paper first applies and studies the Viola Jones detector and CNN-LSTM emotion recognition algorithm in neural networks. Furthermore, it analyzes and compares Mobilenet V3 neural networks on multiple facial emotion datasets to achieve optimal performance. Subsequently, using Vivado HLS tools, the Mobilenet V3 neural network is designed as hardware IP, breaking down functionalities into CONV, DW CONV, PW CONV, pooling modules, and fully connected modules on the hardware platform. Building upon this, the ZYNQ7020 emotion recognition system is further configured, utilizing Vivado platform to invoke IPs like VDMA and VTC. The Zynq heterogeneous processor is macroscopically controlled for system operations, utilizing image data captured by the OV5640 camera to identify facial regions through face detection algorithms.

REFERENCES

[1] Alexander Kroh, Oliver Diessel. Efficient Fine-grained Processor-logic Interactions onthe Cache-coherent Zynq Pla tform [J]. ACM Transactions on Reconfiguable Technologyand Systems(TRETS), 2019, 11 (4).

[2] Yong yang Zou, Ming Chen, Kanglin Wei. Design of Custom AXI4 IP Based on AXI4Pro tocol [J]. Applied Mechanics and Materials, 2014, 3634.

[3] Chen S, Hsu C, Kuo C, et al. EmotionLines: an Emotion corpus of multi-party conversations [J]. arXiv preprient. 2018, ariv:1802-8379.

[4] Busso C, Bulut M, Lee C, et al. IEMOCAP: interactive emotional dyadic motion capture database [J]. Language Res: urces and Evaluation. 2008, 42:335-359.

[5] LaBar KS, Cabeza R. Cognitive neuroscience of emotional memory [J]. Nature Reviews Neuroscience, 2018, 7(1):54-64

[6] Lehrer S F, Xie T. The bigger picture. Combining eoconomericos with analytics improves forecasts of movie success J Management Science, 2022, 68(1:189-210.

[7] Abouelela, A. Abbas, H.M., Eldeeb, H., Wahdan, A.A. Nassar, S.M.: Automated vision system for localizing structural defects in textile fabrics. Pattern recognition, 2019:64-73

[8] Xinyu Li, Guangshun Wei, Jie Wang, et al. Muli-scale joint feature network for micro-expression recognition [J]. Computatonal Visual Media, 2021, 7(3):407-417.

[9] Cerisara C, Jafaritazehjani S, Oluokun A, et al. Multi-task dialog act and sentiment recognition on mastodon [C] IPrcceedings of the 27th International Conference on Computational Linguistics. Santa Fe, PA:ACL, 2018 :745-754.

[10] zahis, Choi J. Emotion detecion on T7l showltranscripts with sequence-based convolutional neural networks J. arXiv preprient. 2019, arXiv:1708-4299.

[11] Citeje, cupla U, Chnmakot Ml K, et al Undertanding emotions in fext using deep leaming and big data [J]. computers in Human Behavior. 2019, 93:309-317.

[12] Chatteje A, NarahariKN, Jonshi M et al SemEval-2019 Task 3: emocontext conextual emotion delection in text [C] JIProceedings of the 13th International Workshop on Semantic Evaluation. Minneapolis, PA:ACL, 2019:39-48.

[13] Micewn G, last r M. Ccowie R, t al The SEMAINE Databese: anmotated mutimnodal recoris of emotionaly colored conversations between a person and a limited agent [J]. IEEE Transactions on Affective Computing. 2019, 3(1):5-17.

[14] Poia S, Hazarika D, Majunder N t al MELD: a mutimodal murt-party dataset for emotion recogrition in conversationsS [C] Proceedings of the 57th AnualMeeting of the Association for Computational Linguistics. Florence, PA:ACL, 2019 :527-536.

[15] Saha T, Patra A, Saha S, et al bowerds emotion-aided muli-modal dialogue act clssfitationC]JIProcedings of the 5ith Anul Meeting of theAssociation for Computational Linguistics.Online, PA: ACL, 2020: 4361-4372.

[16] Dou, Pengyu, and John Morris. "Low-Power FPGA Design for Real-Time Facial Emotion Recognition." 2019 IEEE International Symposium on Smart Electronic Systems (iSES). IEEE, 2019.

[17] A. J. Atoum and M. H. A. Jaafar, "Real-Time Facial Expression Recognition on FPGA," in IEEE Access, vol. 6, pp. 66225-66237, 2018.

[18] Zhang, S., Wang, H., & Yang, F. (2016). Real-time facial expression recognition based on ZYNQ. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1333-1337). IEEE.

[19] Zhang, B., Huang, Y., & Zhang, L. (2018). An efficient FPGA-based system for real-time facial expression recognition. Journal of Real-Time Image Processing, 15(4), 813-826.

[20] Hu, J., Chen, X., & Peng, S. (2019). Implementation of real-time facial emotion recognition system on ZYNQ SoC. In Proceedings of the 2019 3rd International Conference on Electronics Engineering and Informatics (ICEEI) (pp. 185-189). ACM.