Optimization and Implementation of Image Compression Algorithm Based on Neural Network

Jing Hu[1,2,a*], Xianbin Xu[1,b], Xuefeng Pan[2,c], Lingmin Liu[2,a]

[1]School of Computer, Wuhan University, Wuhan, China

[2]Luojia College,Wuhan University, Wuhan, China

[a]jhu1983@whu.edu.cn, [b]xbxu@whu.edu.cn, [c]panxuefeng800103@sina.com

Keywords: Video compression, Neural networks, Linear approximation

Abstract. This paper presents an image compression algorithm based on neural network with almost real-time response, to address the difficulty of real-time video transmission and compression. After giving the structure of the neural network based video compression system, the self-learning algorithm of neural network is presented. Then, the neuron activation function is optimised using a linear approximation and the designs of the typical modules based on FPGA are proposed. Finally, the overall performance of the image compression algorithm was verified on DSPbuilder and Matlab.

Introduction

The main problems faced by digital image processing are the huge data to be processed, especially for video images. In practical applications, such video images are needed to be transmitted and exchanged among different users, for which large files should be compressed in order to reduce the cost of network resources [1]. Neural network is composed of a large number of simple neuron models to simulate the function of the human brain, e.g., the function of associative memory and reasoning thinking. It is a nonlinear adaptive power system, but also a signal and information processing system, in order to solve the problems which are difficult or impossible for conventional information processing methods to solve. The research on the theory and method of image compression based on neural network is of great significance to both academic value and practical application [2,3]. With more and more widespread applications of video transmission, most of the industries require higher real-time quality of video transmission. However the bandwidth of mediums of video transmission is limited, which places great restrictions on the real-time quality of video transmission. So choosing a better real-time processing platform for video compression and regeneration is imperative. [4,5]

This paper presents an image compression algorithm based on neural network, and the system based on this algorithm is implemented on FPGA. Video images are sent to the transmitting terminal and compressed in FPGA. Then, the processed video images are transmitted via internet. Finally, the processed video images are regenerated on the FPGA of receiving terminal. The weighting coefficients of neural network on transmission terminal are updated periodically and the weighting coefficients on receiving terminal are updated simultaneously.

The Neural Network Based Video Compressing Algorithm

In this paper, BP (Back Propagation) neural network is used, which includes input layer, hidden layer and output layer. BP neural network is one of the most widely used neural network model, which provides data compression capability directly. This section introduces video compression system structure first and then introduces self-learning algorithm of neural network.

Usually, neural network compression system consists of compression layer (input layer - first intermediate layer), transmission layer (first intermediate layer - second intermediate layer) and regeneration layer (second intermediate layer - output layer). The basic principle of this system is that the input layer and output layer of the neural network are composed of the same number of units. In the learning process of the network, the learning model of input layer and the teaching model of output layer use the same digital image signal. Since the number of units of the

intermediate layer is much less than the input layer and output layer, after learning, the network will effectively presents the image model through intermediate layer units and by the model transmit signals. Because the network output layer and the network input layer have the same image model, the network output layer can easily represent the input image model. In this process, conversion between the input layer and the intermediate layer can be viewed as the process of encoding of compression, and conversion between the intermediate layer and the output layer can be viewed as the decoding process [6,7].

In this paper, the system is different from the conventional compression system. It adds a weighting coefficient module which is self-updated at the transmission terminal based on the conventional compression system. The specific structure shows in Fig. 1. The whole updated module has a sub-module similar to the regeneration structure and also has a self-learning weighting coefficient algorithm module. Regeneration structure-like module can simulate the regeneration of video at the transmission terminal to provide reliable data for self-learning module. The weighting coefficient in the regeneration structure-like module keeps updated with weighting coefficients of the other networks to ensure that the receiving terminal can reflect the actual situation. In this way, we can construct a complete self-learning neural network at the transmission terminal.

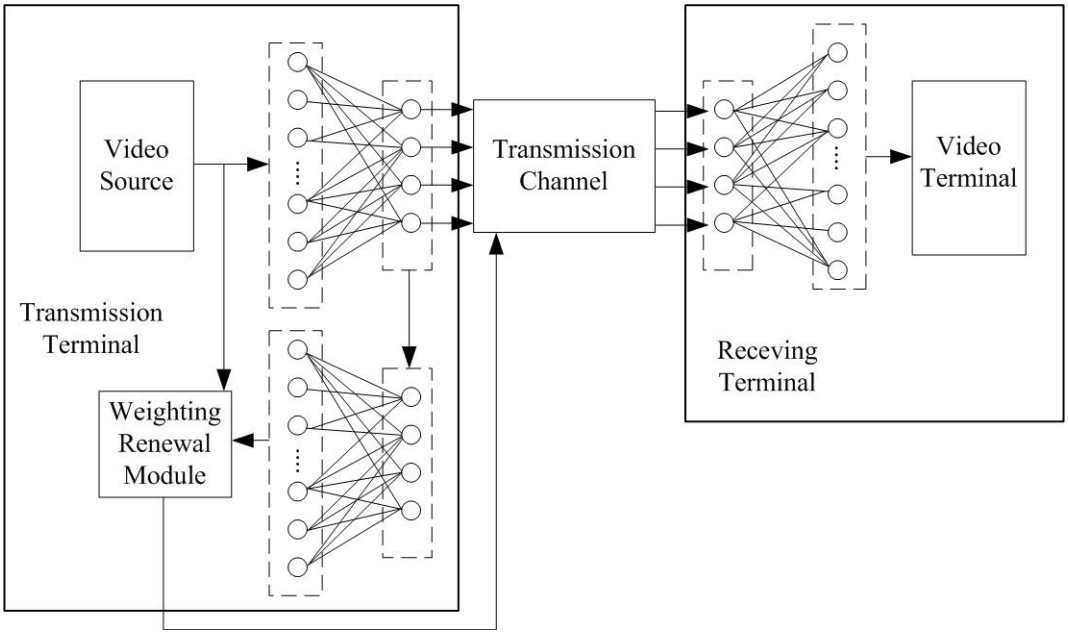

Figure 1. The network structure of video transmission system

In this network architecture, the image compression steps are as follows:

1) Once the hardware is built, the system need to be initiated. The concrete process is: gain some video image from the input terminal and correct the weighting coefficient of neural network combining the weighting coefficient of self-renewal module. If we get content result, we can renewal the weighting coefficient of receiving terminal and begin the video transmission.

2） In the process of video, transmission the network of transmit terminal compresses the video image and submit to the network transmission module. After receiving terminal gains the compressed data from transmit terminal, it will regenerate the compressed data and then display and store it.

3）At the same time of image compression, transmission and regeneration, the weighting coefficient self-renewal module of transmit terminal will periodically renew the weighting coefficient of network. The module will replace its weighting coefficient with the renewed one and synchronously update the one of receiving terminal via internet to make sure the consistence of two terminals.

While performing the above steps, segmentation and pixel’s gray value of each frame must be standardized. Here in this paper, the gray value of the pixel is set to [-1, +1] and the image is

divided into small segments, each containing 16 pixels. The intermediate layers has 4 pixels and image compression ratio is 16/4 = 4.

We mark the self-learning neural network with three flag: *j, i, l* .The network' activation function is common-used, which is:

$$y(x)=\frac{1}{1+e^{-x}} \tag{1}$$

The purpose of self-learning is to minimize the mean square error of input and output of network, so we can take the performance function:

$$E_k=\sum_{x=1}^{4}\sum_{y=1}^{4}\left(f(x,y)-f'(x,y)\right)^2 \tag{2}$$

Function *f(x,y)* is the gray value of pixels sampling from image before compressed. Function *f'(x,y)* shows the gray value of pixels sampling from compressed image. E_k is the mean square error between images compressed or not.

The corrected weighting coefficient according to gradient descent methods, which seek and adjust the negative gradient of weighting coefficient in line with *E(k)* and attach a inertia item to ensure rapid convergence to a global minimum.

$$\triangle w_{li}(k)=-\eta\frac{\partial E_k}{\partial w_{li}(k)}+\alpha\triangle w_{li}(k-1) \tag{3}$$

In the formula，η is the learning ratio；α is inertia coefficient。

When we consider something like that performance function isn't the explicit function to the weighting coefficient, we will find that it is hard to calculate partial derivative. So we prepare to use the chain rule from calculus to get partial derivative.

As for the chain rule of output layer:

$$\frac{\partial E_k}{\partial w_{li}(k)}=\frac{\partial E_k}{\partial O_l(k)}\cdot\frac{\partial O_l(k)}{\partial net_l(k)}\cdot\frac{\partial net_l(k)}{\partial w_{li}(k)} \tag{4}$$

$O_l(k)$ and $net_l(k)$are the input and output of output layer, respectively.

And calculate apart the three terms in the formula, we will get:

$$\frac{\partial E_k}{\partial O_l(k)}=2O_l(k) \tag{5}$$

$$\frac{\partial O_l(k)}{\partial net_l(k)}=g'(net_l(k)) \tag{6}$$

$$\frac{\partial net_l(k)}{\partial w_{li}(k)}=O_i(k) \tag{7}$$

So we can conclude the self-learning algorithm for the weighting coefficient of output layer:

$$\triangle w_{li}(k)=\alpha\triangle w_{li}(k-1)-2\eta O_l(k)g'(net_l(k))O_i(k) \tag{8}$$

In a similar way, we also can get the self-learning algorithm for the weighting coefficient of hidden layer:

$$\triangle w_{ij}(k)=\alpha\triangle w_{ij}(k-1)+2\eta O_l(k)g'(net_l(k))w_{li}g'(net_i(k))O_j(k) \tag{9}$$

And $g(.)=g(.)(1-g(.))$， $O_j(k)$is the input of neural network.

The implementation of system' FPGA

Due to the structure of FPGA, operations such as division, logarithm operation and exponential operation are difficult to implement or occupy a large amount of resources. However, the activation function in neural network involves a lot of division and exponential operation, so we need to improve the activation function of neural network and use shared resources as far as possible on the basis of satisfying time constraint.

The traditional activation function is non-linear, and its linear approximation is as follow [8]:

$$y=\begin{cases}\dfrac{1}{2}+\dfrac{\hat{x}}{4} & ,x<=0\\ \dfrac{}{2^{|(x)|}} & \\ 1-y(-x), x>0 & \end{cases} \tag{10}$$

In this function, (x) is the integer part of x， $\hat{x}$ is the sum of x and $|(x)|$.

The division operation in this approximation will be replaced with shifting, which means that in the Eq. 10, we just need to use the operations of shifting and adding. As mentioned before, the gray value is normalized to the section of -1 and 1. Such normalization will provide a benefit for the calculation of neural network, because in that section the activation function will easily displaced by a substitution, which is:

$$y=\frac{1}{2}+\frac{x}{4} \tag{11}$$

We can find that, through simulation, under the circumstance of appropriate weighting coefficient, the value from system input sent to the activation function is lower than 1, and then we can replace activation function with the above Eq. 11.

In Fig. 2 we give the comparison of three kind of activation function. The constant line is Eq. 1, and the point line representing Eq. 11. We also can see that the constant line and dashed line is approximating while point line in the section of -1 and 1 is close to them, too.

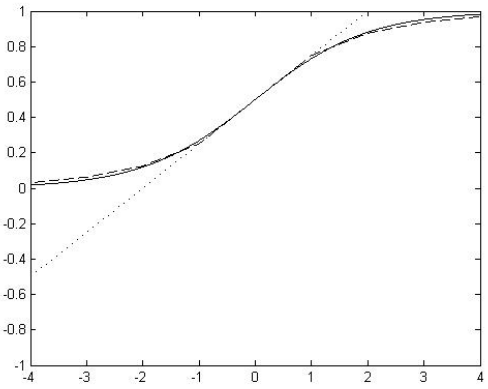

Figure 2. The comparison of three kind of activation function

The system design based on FPGA mainly pays attention to two aspects: time and range constraint. On the premise of meeting time constraint, we should try to save the use of range. We will analyze the single main function module and calculate the time they used and then considering the features of video transmission give the repeating times for these function' modules.

The calculation of $O_l(k)$, $O_i(k)$, Δw_{li} and $\triangle w_{ij}$ occupy the most time during the process of video compression and in the following statements we will show the modules of the values.

For the output of $O_i(k)$， each one need 16 input to be multiplied with weighting coefficient and is used as the input of neuron. Through the activating effect of activation function, we can gain the general calculating module, which is showed in Fig. 3. Register 1 to register 16 stores each input of neural network and these inputs via the choosing action of multiplexer 1 arrive at multiplying unit.

Multiplexer 2 choose to send corresponding weighting coefficient. By doing that, for the calculation of one output value, we need 16+2=18 clock period, this number '2' -2 clock period, is consuming on the outputting and the resetting. If we maximize this kind of modules, the whole time is 18*4=72 clock periods to calculate the output of hidden layer. In a same way, for the calculation of output $O_l(k)$ of output layer，we will know that the time for each value' calculation is six clock periods, and the whole output time is 6*16=96 time periods. So, 168 clock periods have been used on the image compression and regeneration, and for a FPGA running with 50M HZ, it is just 3.36μs.

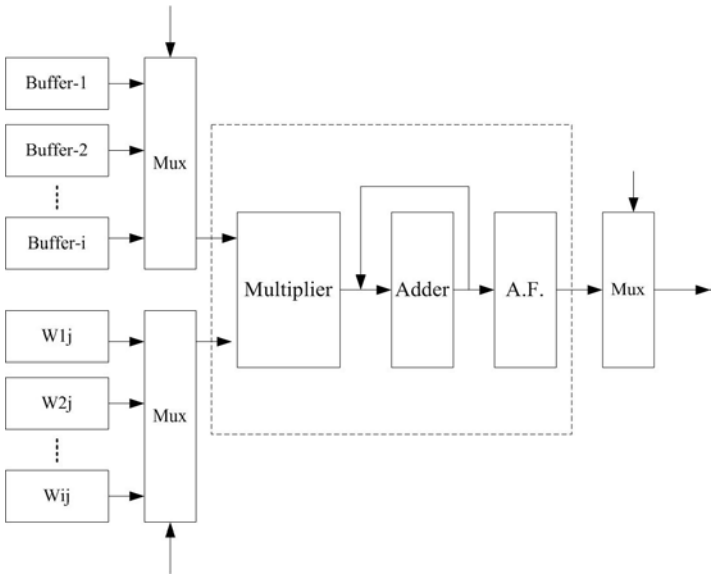

Figure 3. The general calculation model for self-learning weighting coefficient

Eq. 8 and Eq. 9 give us the equation of self-learning method for weighting coefficient, we can find that the second item in this two formulas have constant multiplications. In order to save used region, as the Fig. 4 presents, we can resolve this problem by using the tired multiplier of totalizer. So, for the each $\triangle w_{li}$，it means 5+2=7 clock periods，and this "2"means the same thing we mentioned yet. In a similar way, we need 7+2=9 clock periods to cover the calculation of one $\triangle w_{ij}$. In this network, which have 64 w_{li} and 64 w_{ij}, and subsequently we will use 16*64 clock periods to self-renewal weighting coefficient. And for a FPGA running with 50M HZ, it is just 20.48μs.

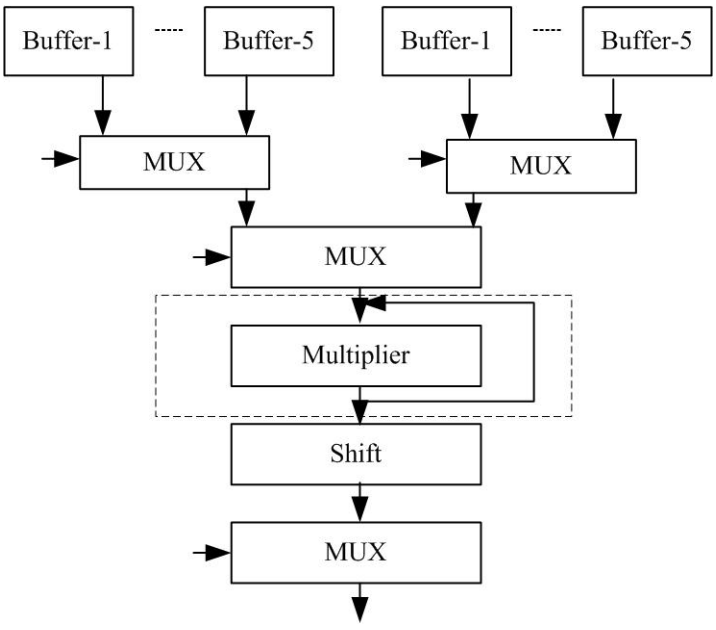

Figure 4. The general calculation model for the self-learning weighting coefficient

Simple video have a frame rate of 30 fps, and the remaining time for each frame's operation including compressing, transmitting and regenerating is about 33ms. For a video with a resolution of 256*256, a single frame need be processed 4096 times and the time for each frame is approximate 14ms. This calculated time meet the requirements of system, which is important. We are going to use the method of renewing periodically to update the weighting coefficient of system. Every renewal need to transmit 4*16=64 weighting coefficient and to ensure the use ratio of network bandwidth, and we are using the strategy of handling n times and weighting coefficient learning one time, in the meantime, transmit data one time. On the foundation of time and bandwidth, we plan to maximize the increase of the strength for the neural network's renewal.

Experiments

We simulate the Verilog module by DSP Builder in MATLAB and ALTERA. DSP Builder can compile the Verilog module that users write on their own into MATLAB functional module, so that it can be used in Simulink to test the function of the module [9].

In the experiment we use an unscheduled delay between compression layer and regeneration layer to replace the network transmission procedure, and the others are the same as introduced in part 2 in this paper. When the module is running, an image is read first and then the system corrects the weighting coefficients to make the error within 0.001. After correction, update the weighting coefficients of regeneration layer to initialize the module and then video transmission starts. We design the system to update weighting coefficients every 128 processes.

Fig. 5 shows the error variation in the entire simulation process. We can find the correction is finished at 0.04s and before this moment the error is a little big. After the correction the error approaches 0.001. As mentioned above, we get the whole calculation time of compression, regeneration and self-learning of weighting coefficients to be 23.84μs, which means the learning in the correction takes 23.84μs. During the correction of initialization, it takes about 1700 repeated calculations. After 0.04s, the error keeps within 0.001 when the video is being transmitted, which turns out that the system structure with its modules can meet the requirements of video transmission.

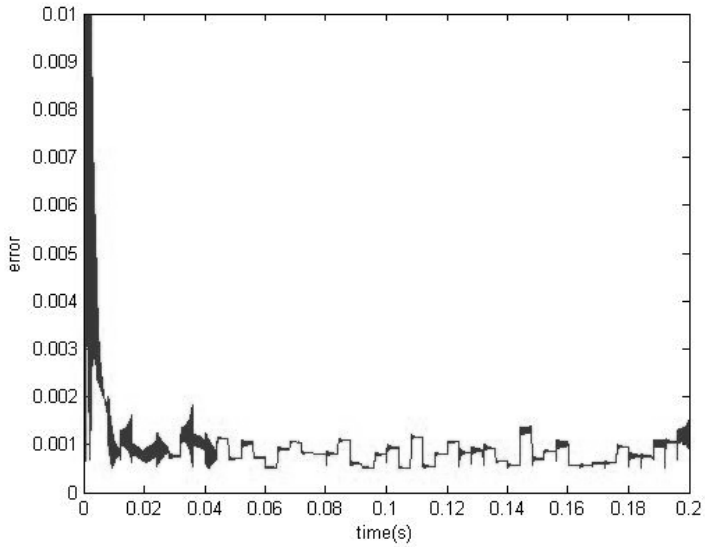

Figure 5. The error change of video image

Conclusion

In order to solve the difficulties of real-time video transmission, we proposed a new video compression method based on neural network using FPGA. We introduce a video compression neural network, a self-learning algorithm of weighting coefficients and put forward a linear approximation of the non-linear neuron activation function. We analyze an implementation method

of video transmission on FPGA and finish designing and writing verilog module. And finally we verify the Verilog module through DSP Builder and MATLAB and get the results. In the research, the implementation of video transmission based on neural network on FPGA is completed, which makes high-quality video transmission possible.

References

[1] Ohm J., Sullivan G.J. . High Efficiency Video Coding: The Next Frontier in Video Compression. IEEE Signal Processing Magazine, v 30, n 1, 152-8, Jan. 2013

[2] Teoh E. J., Tan K. C., Xiang C. Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. IEEE Trans. Neural Netw., 2006, 17(6):1623–1629

[3] Goh C.K., Teoh E.J., Tan K.C.. Hybrid Multiobjective Evolutionary Design for Artificial Neural Networks. IEEE Trans. Neural Netw., 2008,19(9): 1531~1547

[4] Nayak A. , Biswal B., Sabut S.K. . Evaluation and comparison of motion estimation algorithms for video compression. International Journal of Image, Graphics and Signal Processing, v 5, n 10, p 9-18, Aug. 2013

[5] Tresa L.E., Sundararajan M. Comparative analysis of different wavelets in DWT for video compression. 2014 International Conference on Circuit, Power and Computing Technologies (ICCPCT 2014), p 1512-17, 2014

[6] Salama M.M.A. , Bartnikas R. . Determination of neural-network topology for partial discharge pulse pattern recognition. IEEE Transactions on Neural Networks, v 13, n 2, p 446-56, March 2002

[7] Taylor C.M. , Agah, A. . Evolving neural network topologies for object recognition. World Automation Congress (WAC) 2006, p 1067-72, 2007

[8] Acosta Nelson, Tosini Marcelo, Custom Architectures for Fuzzy and Neural Networks Controllers[J]. JCS&T, 2002, Vol. 2, No. 7: pp. 9-15.

[9] Xue Dingyu,Chen Yangquan.MATLAB Solution to Mathematical Problems in Control[M]. Beijing: Tsinghua University Press, 2007, pp. 356-370.

