2019 International Conference on Unmanned Systems and Artificial Intelligence（ICUSAI)

Research on Intelligent Target Detection and Coder-decoder Technology Based on Embedded Platform

Xiaodong Zhao
Unmanned System Research Institute
Northwestern Polytechnical University
Xi'an, China
xdzhao@nwpu.edu.cn

Xunying Zhang
Unmanned System Research Institute
Northwestern Polytechnical University
Xi'an, China
zhangxy@nwpu.edu.cn

Xuemei Cheng
Institute of 365
Northwestern Polytechnical University
Xi'an, China
xmcheng@ nwpu.edu.cn

Fayang Chen
Flight Control Department
AVIC Xi'an Flight Automatic Control Research Institute
Xi'an, China
chen0805@126.com

Zuofeng Zhou
Xi'an Institute of Optics and Precision Mechanics
Chinese Academy of Sciences
Xi'an, China
zfzhou@opt.ac.cn

Tao Xu
Unmanned System Research Institute
Northwestern Polytechnical University
Xi'an, China
taoxu@nwpu.edu.cn

***Abstract*—In order to meet the embedded application requirements of machine learning algorithm, the intelligent target detection and recognition algorithm based on convolutional neural network and corresponding optimal process are studied. Detailed network structure analysis and network performance analysis are carried out. Based on GPU embedded platform, TensorRT technology is used to accelerate the embedded application of intelligent target detection and recognition algorithm, including fp16 and int8 inference modes. Satisfactory verification results are achieved on embedded platform. In addition, an integrated system of real-time machine learning and H.265 encoding and decoding technology is realized. Firstly, the compressed image data sent by the camera is received by embedded platform and decoded in real time in H.265 format. Then the real-time intelligent target detection and recognition algorithm basing on TensorRT technology is done for RGB data obtained by hardware decoding process. Finally, the data is compressed in H.265 format, and subsequently storage and data transmission are carried out. The experimental results show that TensorRT technology can improve the inference speed of neural network in embedded platform. The network structure optimized by TensorRT technology can achieve three times the speed increase, with limited accuracy loss. Hardware coding and decoding of H.265 can also cause corresponding delay to program inevitably.**

Keywords—Convolutional Neural Network; target detection and recognition; network optimization; coder-decoder system; embedded platform

I. INTRODUCTION

Deep learning is the future direction of technology development in the field of intelligent target detection and recognition. Convolutional Neural Network (CNN) [1] has been proved to be the best structure in deep learning and has made remarkable recognition result compared with traditional methods. By training and learning features of color, edge, texture, shape and image topology, the structure is accurately described. In addition, Recurrent Neural Network (RNN) [2] and Long Short-Term Memory (LSTM) [3] have unique advantages over CNN in the context of target acquisition. There are also some algorithms based on RNN and LSTM for intelligent target detection and recognition, however, the results are significantly less than those based on CNN.

The main directions of intelligent target detection and recognition algorithms based on CNN are divided into two categories: one field is region-based algorithms and the other field is regression-based algorithms. The former include region-CNN, Fast region-CNN [4] , Faster region-CNN [5] and region-based Fully Convolution Network [6]. The latter include YOLO (You Only Look Once, YOLO) [7] , YOLO v2, YOLO 9000 [8], YOLO v3 [9], SSD (Single Shot MultiBox Detector, SSD) [10] and DSSD (Deconvolutional Single Shot Detector, DSSD) [11]. Region-based algorithms divide object recognition and location into two steps respectively, which shows low recognition error rate and low leak recognition rate. These algorithms are slow and cannot meet the real-time scene detection requirements. Regression-based algorithms do not need recommendation step, and directly generate probability and coordinates of objects, which can achieve better time efficiency.

R-CNN innovatively combines target candidate area with deep learning to achieve target detection. Fast R-CNN extracts feature maps of candidate area directly from depth feature map and uses it to achieve subsequent recognition and boundary box regression. Faster R-CNN proposes a new network called RPN (Region Proposal Network, RPN), which make the process of target detection and recognition unified in a deep network, to achieve end-to-end mapping, greatly improving the speed of the algorithm. In recent years, more algorithms have been improved in this framework, including Bayesian optimization, structural prediction, location prediction, semantics segmentation and other models to improve accuracy and location results. However, from the point of view of real-time, it is difficult to obtain satisfactory results. For example, by adding location information before pooling layer, R-FCN algorithm assigns different levels to detect different locations of targets. R-FCN replaces VGG16 with ResNet to reconstruct network, which improves the accuracy, but greatly increases the complexity of the algorithm.

YOLO divides picture into several parts, and directly judges whether there is a target in each part through deep neural network, and then predicts the target category and boundary box. Compared with YOLO, its improved YOLO v2 and YOLO v3 algorithms achieve higher detection accuracy and faster speed. Among them, YOLO v3 adopts

Darknet-53 network structure, which enhances the recognition ability of small targets, improves prediction accuracy and maintains speed advantage. The effect of SSD is inferior to YOLO v3 in terms of detection accuracy and detection speed. SSD combines anchor mechanism and uses multilayer features of different feature maps to detect targets. DSSD is the most famous branch of SSD, which replaces VGG with Resnet-101 in feature extracting. Since the network structure is more complex than SSD, the detection speed of DSSD is much slower than SSD, which is slightly inferior in accuracy and speed balance compared with YOLO v3.

Since resources on embedded system is limited, it is necessary to study how to accelerate deep learning algorithms. The complexity of multi-layer neural network algorithm is relatively high, and it cannot meet real-time requirement. Intelligent computing optimization technology [12] must be combined to make the original network with higher complexity and more redundancy of parameters reduce to network structure with low complexity and small parameters, which is suitable for hardware inference. TensorRT [13] is a high-performance inference C++ library produced by Invidia, which is specially used for inference of edge devices. TensorRT can decompose and fuse the already trained models, and the fused models have high degree of aggregation. TensorRT supports low-level inference, and the computing speed can be significantly improved. TensorRT is optimized in the following aspects: Merge some layers, data process supporting FP16 and INT8 [14] [15] [16], kernel auto-tuning, dynamic tensor memory and multiple parallel operations [17] [18] supporting.

In addition, in order to obtain higher compression ratio, the hardware system usually uses compression format to transmit image data. At present, the most effective compression format is H.265 [19]-[25]. With the same picture quality and bit rate, H.265 occupies 50% less theoretical storage space than H.264 [26] [27]. If the storage space is equal, it means that at the same bit rate, H.265 will have a higher theoretical value of 30%-40% than H.264. The birth of H.265 standard is to transmit higher quality network video with limited bandwidth. Besides retaining some of the original technologies, it adds related technologies that can improve the relationship between bit stream, coding quality, delay and algorithm complexity. Embedded platform receives compressed data and decodes it in real time, and calculates deep learning algorithms in real-time. At the same time, data can be transmitted to other platform such as FPGA.

This paper implements a highly integrated embedded data processing system, which supports real-time deep learning calculation of images. What's more, the system integrates H.265 coding and decoding system. Firstly, aiming at the received camera data, H.265 decoding process is performed. Secondly, real-time intelligent target detection of the decoded image data is carried out. Real-time inference is carried out on the embedded platform using TensorRT, and the results are displayed in real time. In addition, after intelligent calculation, the processed image data would be coded by H.265, and be stored in embedded platform, and data can be transmitted to other platforms in real-time.

In this paper, we firstly introduce intelligent target detection and recognition methods, and then we analyze the network structure of YOLO v3 in detail. Secondly, we introduce neural network optimization technology, and analyze fp16 and int8 reference in detail. Thirdly, we introduce coder-decoder real-time framework of H.265. Finally, we integrate all the above technologies on embedded platform, and give the experiment results.

II. Intelligent Target Detection and Recognition

A. *Analysis of YOLO v3 Network Structure*

The original framework of YOLO v3 algorithm is darknet. Darknet is a lightweight open source deep learning framework based on CUDA. Its main characteristics include no dependencies, good portability, and supporting CPU and GPU computing methods. YOLO v3 can balance speed and accuracy perfectly, which is suitable for engineering applications. YOLO v3 network structure contains three parts of output. After obtaining the three final feature maps, classification and position regression are performed. The three feature maps are used to detect small, medium and large targets respectively. This is one of the improvements mentioned in algorithm, namely predictions across scales. The improved point uses FPN (Feature Pyramid Networks, FPN) for reference, and uses multi-scale ideas to detect targets of different sizes. The finer the grid cell, the finer the object can be detected.

In terms of feature extraction, YOLO v3 adopts Darknet-53 network structure, which uses ideas from residual network for reference and sets up shortcut connections between layers. Convolution form of YOLO v3 network shows in Fig. 1. For YOLO v3, Batch Normalization and Leaky Relu are already inseparable parts of convolution layer, expect for the last layer, which together constitute the smallest component.

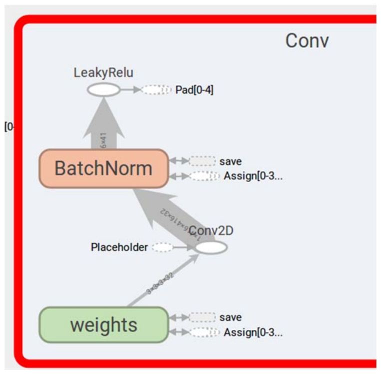
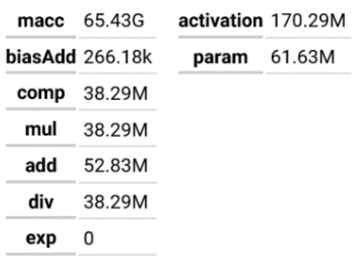


Fig. 1. Convolution form of YOLO v3 network

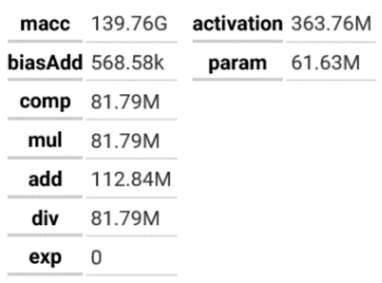
B. *MAC Calculation Quantity of YOLO v3 Network*

When input image size of network is 416x416 and 608x608, analyze the calculation quantity under the input condition. As shown in Fig. 2 and Fig. 3, where the former part shows MAC calculation quantity under condition of 416x416, and the latter part shows MAC calculation quantity under condition of 608x608. This represents the requirement of YOLO v3 neural network for hardware resource utilization.



macc	65.43G	**activation**	170.29M
biasAdd	266.18k	**param**	61.63M
comp	38.29M		
mul	38.29M		
add	52.83M		
div	38.29M		
exp	0		

Fig. 2. MAC calculation quantity under 416x416 input size condition



macc	139.76G	activation	363.76M
biasAdd	568.58k	param	61.63M
comp	81.79M		
mul	81.79M		
add	112.84M		
div	81.79M		
exp	0		

Fig. 3. MAC calculation quantity under 608x608 input size condition

C. Bounding Box Prediction of YOLO v3 Network

As show in Fig. 4, when predicting border, assume c_x and c_y are distances relatively to the upper left corner. The length of each cell is 1. Assume t_x and t_y output offset between 0-1 through sigmoid function respectively. Then the position of final center point of bounding box can be calculated by adding with c_x and c_y. Assume p_w and p_h are anchor width and height manually set, then b_w and b_h can be calculated by (1). The final goal is to make the predicted four parameters close to the real ground truth. The calculation formulas are shown as (1).

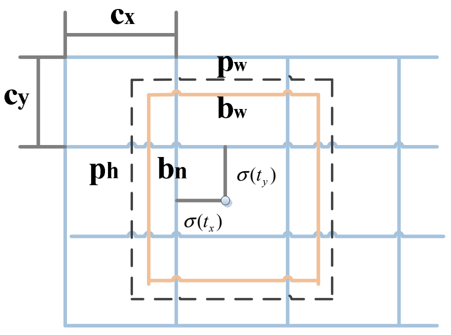


Fig. 4. Bounding box prediction sketch map

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \quad & b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \quad & b_h &= p_h e^{t_h} \end{aligned} \tag{1}$$

III. NEURAL NETWORK OPTIMIZATION TECHNOLOGY

A. Deployment process and optimization modes

TensorRT is a high-performance deep learning support engine, and is not open source. TensorRT is an acceleration technology of neural networks, which provides an acceleration solution based on GPU platform for neural network deployment. The deployment process of TensorRT shows in Fig. 5. During deployment process, TensorRT combines and quantifies the network to form a more compact network structure with less hardware resource requirements. TensorRT technology first generates the intermediate engine as show in Fig. 5 through optimization technology, and then deploys the optimized engine to realize the real-time application of neural networks under constrained resources.

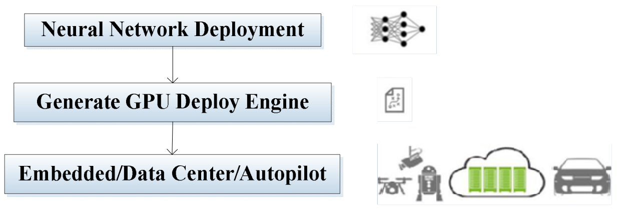

Neural Network Deployment
Generate GPU Deploy Engine
Embedded/Data Center/Autopilot

Fig. 5. Neural Network deployment process based on TensorRT

For the original network, TensorRT optimizes both in vertical direction and horizontal direction. The comparisons between original network and network after optimized in vertical direction show in Fig. 6.

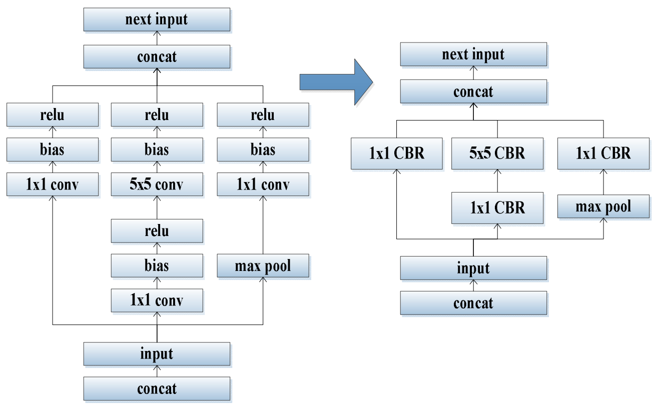


Fig. 6. Original network and network after optimized in vertical direction

The comparisons between original network and network after optimized in horizontal direction show in Fig. 7.

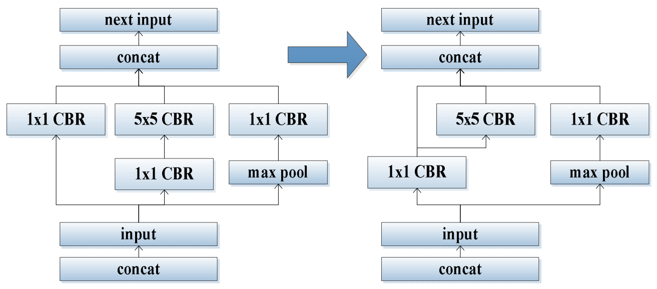


Fig. 7. Original network and network after optimized in horizontal direction

B. FP16 Inference and INT8 Inference

16-bit and 8-bit inference are useful in applications which can handle larger data sets or achieve better performance by manipulating less precise data. Some large-scale neural network models are limited by GPU storage bandwidth. For fp16 inference, Nvidia GPU implements IEEE 754 floating-point standard, which definition includes 1-bit symbol, 5-bit index bit and 10-bit accuracy. Half2 structure stores two half values in a 32-bit value, as shown in Fig. 8.

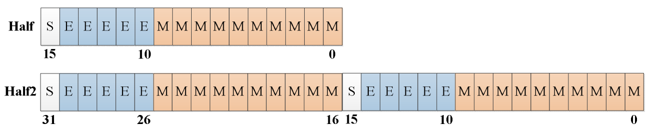

Half
S E E E E E M M M M M M M M M M
15 10 0
Half2
S E E E E E M M M M M M M M M M S E E E E E M M M M M M M M M M
31 26 16 15 10 0

Fig. 8. Half2 and Half structure

For int8 reference, relative entropy is used to represent difference between two distributions before and after quantification. Formula of information entropy is shown as (2), and formula of K-L divergence is shown as (3). The bigger the difference is, the bigger the relative entropy is.

$$H(X) = -\sum_{x\in\chi} p(x)\log p(x) \tag{2}$$

$$D_{KL}(p\,\|\,q) = \sum_{i=1}^{N} p(x_i)(\log p(x_i) - \log q(x_i)) \tag{3}$$

C. *Design Framework flow chart*

The programming framework is shown in Fig. 9. Firstly, create a builder, and then create a network using builder. Secondly, analyze Caffe model and specify which tensor is the output tensor. Thirdly, create TensorRT engine and serialize the engine. Finally, deserialize engine and create runtime and context, and then do the data operations.

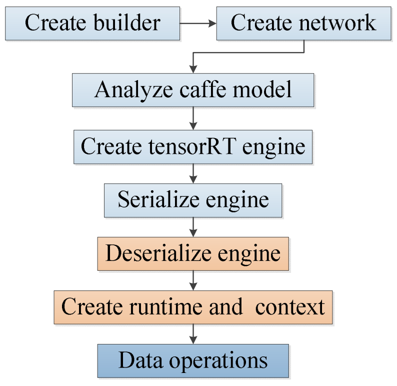


Fig. 9. Programming framework

IV. CODER-DECODER REAL-TIME FRAMEWORK

A. *Organizational structure of the system*

The overall system structure, which combines H.265 encoding and decoding with real-time deep learning process, is shown in Fig. 10. The whole system include receiving module, decoding module, intelligent algorithm processing module, result display module and coding module.

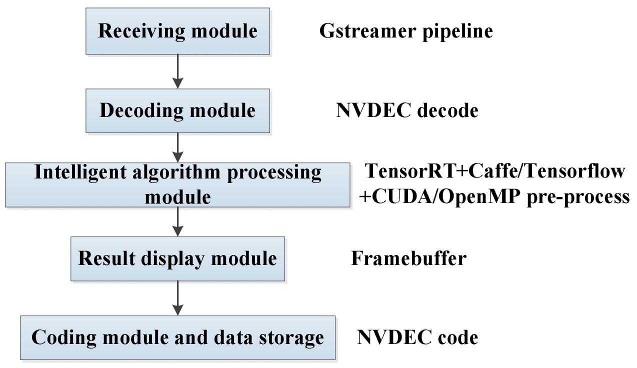

Receiving module
Gstreamer pipeline
Decoding module
NVDEC decode
Intelligent algorithm processing module
TensorRT+Caffe/Tensorflow +CUDA/OpenMP pre-process
Result display module
Framebuffer
Coding module and data storage
NVDEC code

Fig. 10. Overall system structure

Receiving module is responsible for receiving image data streaming using GStreamer pipeline. Decoding module is responsible for H.265 hardware accelerated decoding. Intelligent algorithm processing module is responsible for intelligent algorithm application in real-time. Result display module is in charge of framebuffer process, showing processed result in real-time. Coding module is in charge of H.265 hardware accelerated coding and data storage. Among these five modules, it is especially worth mentioning that TensorRT design method can be combined into Caffe or TensorFlow framework to accelerate the inference speed.

In addition, CUDA can also be combined into the whole process framework. CUDA is both architecture and language. In terms of architecture, it includes hardware architecture such as Fermi, Kepler, hardware computing capability and principle of mapping CUDA program to GPU. In terms of languages, CUDA provides various functions of computing with GPU. Image pre-processing based on CUDA can accelerate process speed remarkably, such as Image denoising and enhancing pre-processing. Especially, image processing program performance can be significantly improved by using multiple CUDA streams. The scheduling process of multiple CUDA streams is shown as Fig. 11.

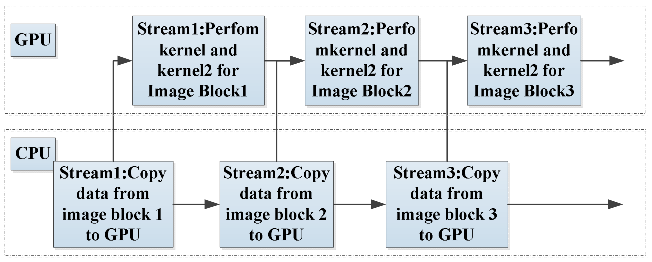

GPU
Stream1:Perfom kernel and kernel2 for Image Block1
Stream2:Perfo mkernel and kernel2 for Image Block2
Stream3:Perfo mkernel and kernel2 for Image Block3
CPU
Stream1:Copy data from image block 1 to GPU
Stream2:Copy data from image block 2 to GPU
Stream3:Copy data from image block 3 to GPU

Fig. 11. Scheduling process of multiple CUDA streams

In the process of pre-processing, OpenMP can also be combined into the whole calculation framework. When working in OpenMP mode, the whole image can be separated into several small image blocks. All the blocks can be sent to multiple CPUs to do pre-processing calculation steps. The process of OpenMP is shown in Fig. 12.

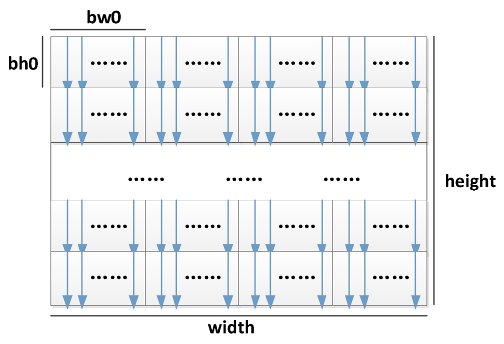


Fig. 12. Process of OpenMP

B. *GStreamer Pipeline in Real-time*

GPU embedded platform uses GStreamer framework to process video in real-time. As a streaming media application framework, GStreamer adopts a plugin-based and pipeline-based architecture. All functional modules in framework are implemented as pluggable components. What's more, it can be easily installed on any pipeline mechanism. It's easy to use various plugins to assemble a well-functioning multimedia application. Structure of GStreamer is shown as Fig. 13.

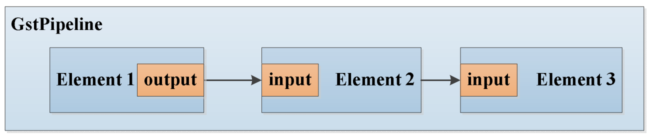


Fig. 13. Structure of GStreamer

By connecting the pads of different components in turn to form a media processing pipeline, image data can be processed normally by each component in the process of flowing through pipeline, and finally achieve specific multimedia functions.

C. *H.265 Hardware Accelerated Coding-decoding*

Hardware accelerated coding and decoding uses the API provided by hardware chip manufactures for coding-decoding. These codecs have been integrated into hardware bottom layer. The advantage is that the speed is very fast. However, the disadvantage is platform-related, and not flexible enough. The flow chart of the decoding process is shown in Fig. 14, and the encoding process is similar. For hardware accelerated decoding process, the video acceleration engine directly controls NVDEC to achieve decoding acceleration. What's more, the encoding process is the same.

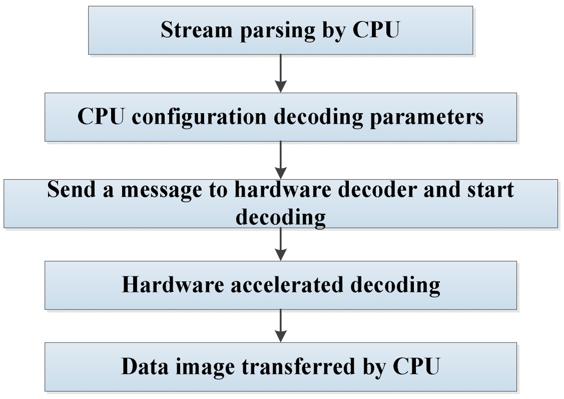

Stream parsing by CPU
CPU configuration decoding parameters
Send a message to hardware decoder and start decoding
Hardware accelerated decoding
Data image transferred by CPU

Fig. 14. Decoding process

D. Framebuffer Mechanism

Framebuffer is an interface provided by Linux for display device, which can display buffer content, and shield the bottom difference of image hardware. Framebuffer allows upper application program to read and write display buffer directly in graphical mode. Users do not need to care about the specific buffer location, which is all accomplished by framebuffer device driver. The typical display mechanism is shown as Fig. 15.

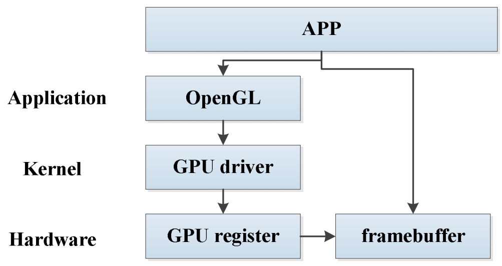


Fig. 15. Framebuffer display mechanism

Framebuffer abstracts graphic hardware structure into a series of data structures, which can be directly operated by read and write API. Users can take framebuffer as an image of memory.

V. Experiment Results

In order to verify the acceleration ability of proposed algorithm process and the coder-decoder ability of H.265, Nvidia Jetson Xavier is selected as the verification embedded GPU platform. Xavier is a heterogeneous embedded GPU platform with 8 cores of ARM64 architecture and 512 CUDA cores. After 20 simulation tests, the optimal simulation results are selected. The simulation results are shown in Tab. I. What's more, through the interface provided by TensorRT, the inference time required for all layers to run can be printed.

TABLE I. SIMULATION VERIFICATION RESULTS

Simulation Category	*Network input Size*	*Frame Frequency*	*Acceleration Type*
darknet	416x416	8 frame/s	Not acceleration
TensorRT	416x416	18 frame/s	FP16
TensorRT	416x416	25 frame/s	INT8

Based on TensorRT technology of Nvidia, the inference speed of the neural network can be improved three times on the embedded GPU platform, with limited accuracy loss.

The delay of H.265 coder-decoder system based on embedded GPU is shown in Tab. II. It can be seen that in the encoding and decoding process of GPU embedded platform, the coding and decoding system has 120ms delay time, which is uncontrollable. The specific delay time is determined by the hardware configuration of the platform. In most case, it cannot be improved by program optimization. H.265 coding and decoding based on other platforms, such as FPGA, will significantly improve the delay.

TABLE II. DELAY TIME VERIFICATION RESULTS ON EMBEDDED GPU PLATFORM

Mode	*Input Size*	*System delay time of coder-decoder*
H.265	1080p	120ms

The paper realizes the GPU embedded platform application with the functions of code-decoder and real-time deep learning processing. Other deep learning algorithms can also be implemented in this framework. The next research direction of this paper is neural network compression [28]-[30] algorithm of convolutional neural network, quantitative hardware resource evaluation of neural network, and further validation of compression algorithm for GPU embedded platform, to provide technical basic reserve for domesticated platform with independent property rights.

Acknowledgment

This work is funded by related pre-research project. All of the authors are very grateful for their support in the implementation of the project.

References

[1] Y. S. Zhang, Y. Ming, R. Q. Zhang, "Object Detection and Tracking based on Recurrent Neural Networks," 2018 14th IEEE International Conference on Signal Processing (ICSP), pp. 338-343, Aug 2018, Beijing, China.

[2] Y. F. Wang, D. X. Li, L. Lin, B. H. Chen, L. H. Wang, M. Zhang, "Feature Aligned Recurrent Network for Causal Video Object Detection," 2019 IEEE International Conference on Image Processing (ICIP), pp. 3900-3904, Sept, 2019, Taipei, Taiwan.

[3] Y. Yuan, X. D. Liang, X. L. Wang, D. Y. Yeung, A. Gupta, "Temporal Dynamic Graph LSTM for Action-Driven Video Object Detection," 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1819-1828, Oct. 2017, Venice, Italy.

[4] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Dec. 2015, Santiago, Chile.

[5] S. Q. Ren, K. M. He, R. Girshick, J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, Issue. 6 pp. 1137-1149, 2017.

[6] J. F. Dai, Y. Li, K. M. He, J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," 30th Conference on Neural Information Processing Systems (NIPS), vol 29, pp. 1-11, 2016.

[7] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, real-time object detection," IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, June 2016, Las Vegas, NV, USA.

[8] J. Redmon, A. Farhadi, "YOLO9000: Better, Faster, Stronger," IEEE Conference on Computer Vision and Pattern Recognition, pp. 6517-6525, July 2017, Honolulu, HI, USA.

[9] J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement,” IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg, “SSD: Single Shot MultiBox Detector,” European Conference on Computer Vision (ECCV), pp. 21-37, 2016.

[11] C. Y. Fu, W. Liu, A. Ranga, A. Tyagi, A. C. Bergs, “DSSD: Deconvolutional Single Shot Detector,” IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[12] J. Q. Xu, B. Wang, J. B. Li, C. Hu, J. Pan, “Deep learning application based on embedded GPU,” 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), pp. 1-4, June 2017, Harbin, China.

[13] Y. W. Yao, B. Dong, Y. K. Li, W. Q. Yang, H. Q. Zhu, “Efficient Implementation of Convolutional Neural Networks with End to End Integer-Only Dataflow,” pp. 1780-1785, July 2019, Shanghai, China.

[14] R. G. Xu, F. Han, Q. Ta, “Deep Learning at Scale on NVIDIA V100 Accelerators,” 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pp. 23-32, Nov. 2018, Dallas, TX, USA, USA.

[15] J. X. Wu, C. Leng, Y. H. Wang, Q. H. Hu, J. Cheng, “Quantized Convolutional Neural Networks for Mobile Devices,” IEEE Conference on Computer Vi-sion and Pattern Recognition, pp. 4820-4828, June 2016, Las Vegas, NV, USA.

[16] B. Jacob, S. Kligys, B. Chen, M. L. Zhu, M. Tang, “Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference” IEEE Conference on Computer Vision and Pattern Recognition, pp. 2704-2713, 2018, Salt Lake City, UT, USA.

[17] N. Zhang, Y. S. Chen, J. L. Wang, “Image parallel processing based on GPU,” 2010 2nd International Conference on Advanced Computer Control, pp. 367-370, March 2010, Shenyang, China.

[18] T. Li, Y. Dou, J. F. Jiang, J. Gao, “Efficient parallel implementation of morphological operation on GPU and FPGA,” Proceedings 2014 IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), pp. 430-435, Oct. 2014, Wuhan, China.

[19] R. Yousfi, M. B. Omor, T. Damak, M. A. B. Ayed, N. Masmoudi, “JEM-post HEVC vs. HM-H265/HEVC performance and subjective quality comparison based on QVA metric,” 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), March 2018, Sousse, Tunisia.

[20] R. Elleuch, F. Belgith, M. A. B. Ayed, N. Masmoudi, “Statistical analysis on coding unit partition of HEVC encoded HD videos,” 2016 International Image Processing, Applications and Systems (IPAS), pp. 1-5, Nov. 2016, Hammamet, Tunisia.

[21] J. Sharma, T. Choudhury, S. C. Satapathy, A. S. Sabitha, “Study on H.265/HEVC against VP9 and H.264 : On Space and Time Complexity for Codecs,” 2018 International Conference on Communication, Computing and Internet of Things (IC3IoT), pp. 106-110, Feb. 2018, Chennai, India.

[22] Y. Lu, Q. Zhan, B. Wei, “Real-time CPU based H.265/HEVC encoding solution with x86 platform technology,” 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 418-421, Feb. 2015, Garden Grove, CA, USA.

[23] T. Uhl, J. H. Klink, K. Nowicki, C. Hoppe, “Comparison Study of H.264/AVC, H.265/HEVC and VP9-Coded Video Streams for the Service IPTV,” 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Sept. 2018, Split, Croatia.

[24] S. Verde, L. Bondi, P. Bestagini, S. Milani, G. Calvagno, S. Tubaro, “Video Codec Forensics Based on Convolutional Neural Networks,” 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 530-534, Oct. 2018, Athens, Greece.

[25] M. Rapczynski, P. Werner, A. AI-Hamadi, “Effects of Video Encoding on Camera Based Heart Rate Estimation,” IEEE Transactions on Biomedical Engineering.

[26] N. Zotos, G. Xilouris, A. Kourtis, D. Renzi, B. Shao, “Performance evaluation of H264/SVC streaming system featuring real-time in-network adaptation,” 2011 IEEE Nineteenth IEEE International Workshop on Quality of Service, June 2011, San Jose, CA, USA.

[27] B. C. Liu, Q. K. Chen, “Implementation and optimization of intra prediction in H264 video parallel decoder on CUDA,” 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), pp. 119-122, Oct. 2012, Nanjing, China.

[28] M . Courbariaux, Y Bengio, J. P. David, “BinaryConnect: Training Deep Neural Networks with binary weights during propagations,” Conference on Neural Information Processing Systems (NIPS), vol 28, pp. 1-9, 2015.

[29] Z. W. Cai, X. D. He, J. Sun, N. Vasconcelos, “Deep Learning with Low Precision by Half-wave Gaussian Quantization,” IEEE Conference on Computer Vision and Pattern Recognition, pp. 5406-5414, July 2017, Honolulu, HI, USA.

[30] C. Leng, Z. Dou, H. Li, S. H. Zhu, R. Jin, “Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM,” 32nd AAAI Conference on Artificial Intelligence, pp. 3466-3473, 2018.