Proceedings of the 4th International Conference on Computing and Data Science (CONF-CDS 2022)
DOI: 10.54254/2755-2721/2/20220583

The Optimization Strategy of Deep Learning Algorithm Based on Image Recognition and Object Detection

Tailai Li

University of Bristol, Beacon House, Queens Road, Bristol, BS8 1QU, UK

pc21827@bristol.ac.uk

Abstract. With the advent of intelligent era, the application of artificial intelligence has penetrated all walks of life in our daily lives and has gradually entered the scope of being understood by the public, such as AlphaGo, which is a human-machine game in Go. Meanwhile, with the increasing complexity of deep neural network algorithms, how to compress and optimize neural network models becomes the key to reducing model storage space and improving model deployment efficiency. This article will complete the INT8 quantization of the model for AlexNet and use the concept of batch normalization (BN) layer to prune the Yolov3 model. Finally, it is proved by experiments that the model quantization makes the image recognition time of the AlexNet about 1/3-1/4 of the original format; and pruning of Yolov3 will significantly reduce the storage space occupied by the model, moreover, make the speed doubled of embedded GPU real-time object detection. However, none of the three experiments reduces the model recognition accuracy. This paper proposes some essential concepts in the main body and applies them to model algorithms and experiments to provide theoretical support and proof for the field of deep learning model compression and optimization. Furthermore, it is possible to explore cutting-edge research directions like Sparse Convolution Net based on the research methods and experimental conclusions of the paper.

Keywords: deep learning, neural network quantization, neural network pruning.

1. Introduction

Nowadays, with the development of computer technology and the hardware computing power, the algorithm of neural network has gradually stepped from shallow to deep, forming deep learning. Deep learning has made great progress and great development space in the fields of image recognition, text processing, speech recognition and object detection. In this regard, Dixit M et al. provide an overview of some of the most used deep learning architectures in an overview of deep learning architectures, libraries and its applications areas, and discuss the various application areas in which deep learning is active [1]. Regardless of the application of artificial intelligence in any field, a neural network model with a small amount of calculation, few parameters, and a fast speed is the goal pursued by enterprises and research institutions. To achieve this goal, neural network optimization techniques such as pruning, and quantization continue to emerge. For neural network model pruning, Zhang Z proposed a unique pruning technique in Model Pruning Techniques for Boosting the Inference Efficiency on Embedded Systems, which can better deploy the algorithm in the embedded device like robots or driverless cars by reducing the storage burden [2]. Afterwards, Liu J et al. conducted a comprehensive survey on the

DOI: 10.54254/2755-2721/2/20220583

acceleration of pruning technology in edge applications in Pruning Algorithms to Accelerate Convolutional Neural Networks for Edge Applications: A Survey[3]. Mousa-Pasandi M et al. proposed a CNN pruning method based on filter attenuation in Convolutional Neural Network Pruning Using Filter Attenuation, in which weak filters are not suddenly removed but attenuated and gradually removed [4]. For neural network model quantization, Zhu F et al. constructed a unified 8-bit (INT8) training framework for common convolutional neural networks in terms of accuracy and speed in Towards Unified INT8 Training for Convolutional Neural Network [5]. In the paper Model-based Weight Quantization for Convolutional Neural Network Compression, Gheorghe Ș et al proposed an adaptive model-based quantization method with a parameterizable number of quantization intervals [6]. For the synthesis of neural network optimization strategies, Liang T et al. studied both pruning and quantization in Pruning and quantization for deep neural network acceleration: A survey to provide a reference for compressed networks [7]; and Han S et al. make pruning, quantization and Huffman coding techniques used to compress deep neural networks, reducing the storage requirements of the algorithm by 35-49 times in Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [8].

The optimization of the neural network model greatly improves the algorithm slimming and hardware deployment efficiency. This paper will study the quantization method of the convolutional neural network AlexNet model and the pruning method of the Yolov3 model, to slim down and accelerate the neural network model at the algorithm level, reduce the amount of calculation and improve the processing speed of hardware deployment. The model before and after optimization is used to conduct comparative experiments in the fields of image recognition and real-time object detection.

2. Algorithm optimization strategy for AlexNet and Yolov3

In order to improve the deployment efficiency of the algorithm to the hardware, the algorithm model is usually lightweight while the model is optimized, so as to achieve the best effect of adapting to the hardware design. For convolutional neural network algorithms such as AlexNet and Yolov3, the cost of increasing the accuracy is the increase in the amount of computation. On the contrary, neural network models need to sacrifice accuracy while reducing the amount of computation. Therefore, without changing the model architecture, reducing the amount of computation while maintaining model accuracy is the ideal direction for model optimization. To achieving the above goals, the optimization strategy of model quantization and pruning is usually adopted. In the model quantization, since the storage accuracy and inference accuracy of the trained model parameters are usually 32-bit floating-point types, the computational complexity and the memory usage are high. Hence, some models with high precision are often difficult to deploy to edge devices with limited computing and memory resources. Therefore, it is necessary to quantify floating-point numbers to fixed-point numbers to reduce the accuracy of parameters while maintaining model accuracy. Model pruning is to delete unimportant neurons/weights. Both methods reduce the memory space occupied by parameters and the calculation amount of the model, thereby improving the efficiency of algorithm deployment to hardware. This chapter will study the algorithm optimization strategies of the convolutional neural network models AlexNet and Yolov3 which provide background and principle support for the following experimental part.

2.1. AlexNet neural network model quantization

The AlexNet model is a convolutional neural network algorithm for image recognition, including 5 convolutional layers, 2 fully connected hidden layers and 1 fully connected output layer, and the sigmoid activation function is changed to a simpler ReLU activation function. The model complexity and parameter quantity are well controlled, and many image augmentations are introduced, such as flipping, cropping, and color changes, which effectively increase the number of data samples, thereby alleviating the occurrence of overfitting. This section will study the quantization method of the AlexNet model to improve the speed of model deployment.

The aim of model quantization is to reduce the precision of parameters, rather than the precision of the model itself, and construct decimals represented by full-precision floating-point numbers into

DOI: 10.54254/2755-2721/2/20220583

integers or half-precision floating-point numbers. The 32-bit floating point number for the quantization mentioned above, its true value "x" is represented by three parts, the sign bit "S", the exponent bit "E" and the mantissa bit "M", the formula is as follows:

$$x = (-1)^S \times (1.M) \times 2^{E-127} \tag{1}$$

Among them, the exponent code E is a 0-255 unsigned number. The above formula is for normalized floating-point numbers. However, the true value "d" of denormalized floating-point numbers represents data that is smaller than normalized floating-point numbers. The formula is as follows:

$$d = (-1)^S \times (0.M) \times 2^{-126}. \tag{2}$$

At present, the chips basically support the operation of normalized floating-point numbers. The true value is greater than the maximum value that can be represented by the dynamic range, which is positive overflow, the output bit is positive infinity, otherwise it is positive underflow, and the output is 0, corresponding to negative overflow and negative underflow. Quantization reduces storage by reducing the number of parameters, so that the amount of data carried by the computing unit for inferring becomes smaller, the throughput increases, the processing speed is improved, and the design of the computing unit becomes simpler. The actual effect of model quantization can also be displayed on the FPGA with limited logic resources, and 32-bit floating-point numbers are quantified to 16-bit floating-point numbers or 8-bit integer numbers on the algorithm. Quantization of floating-point numbers can be achieved by directly converting the data structure. 16-bit half-precision floating-point numbers can double the throughput of the arithmetic unit, and 8-bit integers can increase the throughput of the arithmetic unit by four times. Among them, for the AlexNet model used in the quantization stage in this paper, quantization to 8-bit integers is more suitable for embedded device model inference, because it can significantly improve the inference speed, and its performance in the field of image recognition is the improvement of recognition speed.

2.1.1. Integer quantization. AlexNet model quantization method used in this paper in the later experimental stage is 8-bit integer (INT8) quantization to accelerate inference, which can be embodied as a given Tensor "y", to find the quantified Tensor "y_q", the formula is as follows:

$$y_q = round\big(s \times clip(y, -\alpha, \alpha)\big) \tag{3}$$

where "$round$"stands for rounding, "s"is the scaling factor, and "α"is the range value. Convert the floating-point number to the specified integer number, for INT8, s =127/α, 127 is the maximum value of INT8. The actual embodiment of the model quantization formula can be represented by the multiplication of two Tensor "y" composed of 32-bit floating-point numbers. Each matrix is set to a specific "α". After the scaling factor "s" is obtained, the Tensor "y" is scaled, and the scaling result is rounded. Then the quantified integer Tensor "y_q" is obtained, and finally the multiplication of two new matrices consisting of integers is performed. Since the quantization operation is rounded, if the inverse quantization operation is performed, that is, the matrix multiplication result is restored to the scale according to the scaling factor, and the result obtained will be different from the matrix multiplication result without the quantization operation. Obviously, the quantization of the model is a lossy operation, which may reduce the accuracy of the model, so it is necessary to set an indicator to measure the loss of accuracy. The decisive parameter for accuracy loss is the range value "α", which determines the size of the scaling factor "s", and "s" determines the value of the quantified integer. Therefore, it is necessary to find a reasonable "α" to ensure that the model accuracy is basically unchanged. In the development process of model quantization, there are many methods to solve the loss of quantization accuracy. This paper lists two optimization strategies and chooses one of them as the experimental research method.

The first optimization strategy for AlexNet model quantization is to take the maximum value of the tensor weight parameters as "α", and all parameters after quantization are within the range of "α". The second optimization strategy is to take one of the intermediate values of the tensor weight parameters as

DOI: 10.54254/2755-2721/2/20220583

"α". For the first optimization strategy, if "α" is large, then "s" will be very small, and other weight parameters will be backlogged in a small numerical range after quantization. Therefore, the weight parameter selected as "α" may become an outlier and wasteful. This method still affects the accuracy of the model, and the second optimization strategy does not appear outliers. Therefore, the model quantization in this paper adopts the second optimization strategy to find an optimal intermediate value "α". How to judge and choose "α" is the difficulty to realize the optimization strategy.

2.1.2. KL divergence. As mentioned above, for the selection of the optimal range value, it can be assumed that there is a tensor of FP32, and now the Tensor is expressed by INT8, so the distribution of INT8 after quantization is not an optimal distribution, and there will be certain errors. There are many situations for the value of the range value "α", and there must be a value of "α" that is the closest to FP32 relative to the others. Therefore, a new metric is needed to measure the degree of difference between the different INT8 distributions corresponding to different "α" and the original FP32 distribution. This metric is: relative entropy, also known as KL divergence (Kullback-Leibler divergence), the formula is as follows:

$$D_{KL}(p\|q) = \sum_{i=1}^{N} p(x_i) \cdot \left(log\, p\,(x_i) - log\, q\,(x_i)\right). \tag{4}$$

Among them, "$p(x_i)$" is the original probability distribution, and "$q(x_i)$" is the approximate distribution. This formula expresses the mathematical expectation of the logarithmic difference between the probability distribution of the original data and the approximate distribution. The smaller the KL divergence, the smaller the quantization loss, the smaller the gap between inverse quantization and the original data. Therefore, the KL divergence can be used to calculate how much information the approximate distribution loses compared to the original AlexNet model.

2.2. Yolov3 neural network model pruning

Yolov3 is a convolutional neural network model for real-time object detection. Although it is not the most accurate algorithm, is particularly prominent for the detection of small objects and object parts. The Yolov3 algorithm uses a single neural network to act on the image, divides the image into regions and predicts bounding boxes and probabilities for each region. In this section, we will study the types and methods of pruning to slim down and accelerate the Yolov3 model.

During the development of the human brain, the neurons in the brain and their connection methods will become more and more complex with the development time. To a certain extent, the brain will prune redundant neurons to complete the sparseness of neurons and make thinking more efficient. Neural network pruning is the application of this phenomenon in the field of algorithms, reducing the number of parameters or neurons in the original model and slimming the model. In the field of image recognition or object detection, the processing speed is increased, and the storage space occupied by the model is reduced. The pruning is like quantization operation has an accuracy loss, so when designing the pruning algorithm, it is necessary to consider achieving a high model compression ratio while ensuring the smallest accuracy loss. In recent years, such as VGG-16, ResNet50 and AlexNet, the model compression ratio and theoretical acceleration effect both increase year by year, but the accuracy loss is not varied.

2.2.1. Types of pruning in deep learning. The types of pruning in deep learning are divided into structured pruning and unstructured pruning. Structured pruning is aiming the channel of convolution kernels or feature maps, and unstructured pruning targets specific neurons. The advantage of structured pruning is that it can be accelerated directly on the GPU, because the convolution calculation is equivalent to matrix multiplication. After the feature map and convolution kernel channel are structured pruned, the matrix multiplication and addition operation are not required for the pruned channel so that directly improve the operation speed. The advantage of unstructured is that the pruning granularity is fine, the secondary neurons can be accurately found, the model compression ratio is larger. However, each channel still needs to be calculated during the multiply-add operation, so the acceleration cannot

DOI: 10.54254/2755-2721/2/20220583

be directly reflected on the GPU. Therefore, if need to reflect the intuitive acceleration on the GPU and make the model smaller, it is necessary to design an AI chip to specifically accelerate the pruned model.

2.2.2. Yolov3 model pruning steps and methods. The specific model used in the model pruning experiment in this paper is Yolov3. The entire model pruning process needs to evaluate the importance of neurons or channels at first and pruning unimportant neurons. After that, the accuracy of the model will be lost to a certain extent. Thus, pruned models usually require more training to maintain performance. However, if too many neurons are pruned at one time, the model performance may be degraded. Therefore, the pruning of the model is an iterative process, which is reflected in the alternation of training and pruning, thereby reducing the loss of accuracy after pruning.

In the convolutional neural network, because each batch is convolved to obtain one data distribution of a feature map, and backpropagation needs to adapt to the different distributions of different batches corresponding to each feature map, resulting in slow model convergence and even low-level neural network gradients disappear. Hence, after the convolutional layer and before the activation layer, a BN (batch normalization) layer is added to normalize the distribution of the input value of neuron of each layer of neural network to a standard normal distribution which has mean value of 0 and a variance of 1. Afterwards, the model only needs to adapt to one distribution during backpropagation, which can greatly improve the convergence and training speed. This layer first appeared in the Inception v2 model and can be used in the field of target detection and image recognition. Bjorck N conducted several experiments in Understanding Batch Normalization and showed that BN is mainly able to train with a larger learning rate, which is the reason for faster convergence and better generalization [9]. In the model pruning process, evaluating the importance of neurons or channels is the most important step. In the following pruning experiments, the BN layer-based channel pruning method is used in the Yolov3 model. This method introduces the BN layer scaling factor "γ" to measure the importance of the channel. Each channel of the feature map corresponds to a "γ". When "γ" is small, the channel will be deleted. The formula is as follows:

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i. \tag{5}$$

and

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2. \tag{6}$$

and

$$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}. \tag{7}$$

and

$$y_i \leftarrow \gamma \hat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i). \tag{8}$$

Among them, "μ_B" is the mean value, "σ_B^2" is the variance, "$\hat{x_i}$"is the normalization formula, "γ" and "β" are the parameters of the BN layer, and the normalized value is calculated with "γ" and "β" to obtain the output "y_i". The parameter "γ" and "β" are derivable, which will update themselves with the model when back-propagates. After training, the BN layer parameters "γ" and "β" are the optimal solutions, and finally the model is pruned according to the value of "γ". In general, the Yolov3 structure uses residual blocks for convolutional layer connection and prunes the previous convolutional layer according to the BN layer scaling factor "γ".

3. Experimental part

The deep learning algorithm optimization strategy is classified, discussed, and researched above. The experimental code of this paper uses PyCharm Community Edition 2021.3 to program and modify

DOI: 10.54254/2755-2721/2/20220583

python files. The training, inference and deployment of the experimental part are running in the terminal with the operating system as Ubuntu18.04. This section will complete the following three experiments:

- Image recognition based on AlexNet model quantization
- Video object detection based on Yolov3 model pruning
- Real-time object detection on embedded GPU based on Yolov3 model pruning

3.1. Image recognition based on model parameters quantization of AlexNet algorithm

3.1.1. Experimental purpose and the tool for CNN model quantification. The quantization method used in this experiment is to quantify the 8-bit integer (INT8) of the model AlexNet to accelerate inference, and then use the models before and after quantization to perform image recognition respectively, and observe the optimization brought by quantization to model inference by comparing the image recognition time and accuracy.

For the search range value "α", this experiment uses KL divergence as a measure, and the process of finding KL divergence is complicated, so the NVIDIA rapid deployment tool "TensorRT" is called in the model code to quantify the AlexNet model. This tool can be regarded as a deep learning framework with only forward propagation. It can parse the network models of Caffe and TensorFlow, map them one by one with the corresponding layers in "TensorRT", and convert all models of other frameworks into "TensorRT". In this experiment, TensorRT can convert the model data type to INT8, find the optimal range value "α" while looking for the minimum KL divergence. In the experimental code, the image recognition results of the quantized INT8 model and the original FP32 model are expressed. The results are reflected in the recognized pictures, the comparison is intuitive, and the operation is simple.

3.1.2. Experimental results and analysis. This experiment uses the picture with the target "deer" as the image recognition object. When running the quantization code, it is found that the code runs slowly because the process of finding the range value "$\boldsymbol{\alpha}$" is long, and the KL divergence and "$\boldsymbol{\alpha}$" need to be found at each layer of the AlexNet model. According to the data in the figure and the table below, the image recognition time of the INT8 model after quantization is about 1/3-1/4 of the original format FP32 image recognition time, and the image recognition accuracy before and after quantization is basically the same. It can be seen that the quantization operation can greatly improve the recognition speed of deep learning in image recognition or object detection, and how to choose an optimal model quantization method is a point that algorithm engineers need to consider. For the same model, there is only one range value "$\boldsymbol{\alpha}$" and the scaling factor "$\boldsymbol{s}$", so the "$\boldsymbol{\alpha}$" and "$\boldsymbol{s}$" established after finding the KL divergence can be saved, and there is no need to find the quantization parameter again in the subsequent reasoning process.

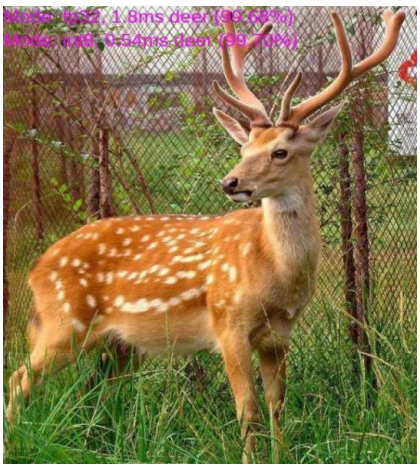

Figure 1. Image recognition results before and after quantization of the AlexNet model.

DOI: 10.54254/2755-2721/2/20220583

This experiment uses the picture with the target "deer" as the image recognition object. When running the quantization code, it is found that the code runs slowly because the process of finding the range value "α" is long, and the KL divergence and "α" need to be found at each layer of the AlexNet model. According to the data in the figure above and the table below, the image recognition time of the INT8 model after quantization is about 1/3-1/4 of the original format FP32 image recognition time, and the image recognition accuracy before and after quantization is basically the same. The quantization operation can greatly improve the recognition speed of deep learning in image recognition or object detection, and how to choose an optimal model quantization method is a point that algorithm engineers need to consider. For the same model, there is only one range value "α" and the scaling factor "s", so the "α" and "s" established after finding the KL divergence can be saved, and there is no need to find the quantization parameter again in the subsequent reasoning process.

In general, the quantized model parameters have almost no effect on the image recognition accuracy. And when the code runs, using "TensorRT" does not cause extra errors and it runs much faster than before the model parameters were quantized. Therefore, experimental results show the effectiveness of Alexnet quantization algorithm and there is no dead loop.

Table 1. Comparison of image recognition performance before and after quantization of the AlexNet model.

Type of data	Recognition time	Recognition accuracy
Before quantization: FP32	1.8 ms	99.68%
After quantization: INT8	0.54 ms	99.70%

3.2. Video object detection based on convolution layer pruning of Yolov3 algorithm

3.2.1. Experimental purpose and BN layer optimization process. In this experiment, a video was selected and the full convolutional network Yolov3 model was used for target detection. At the same time, the model after pruning 60% is used for target detection on the same video, and the results are compared. The recognition principle of Yolov3 is divided into three steps. First, the prediction result is obtained from the feature, then the prediction result is decoded, and finally the frame with the highest probability is selected. In terms of structure, the backbone network of Yolov3 is Darknet53, the important thing is to use the residual network. Each convolution part of darknet53 uses a unique "DarknetConv2D" structure, and regularization is performed at each convolution. After the convolution, perform the BN layer normalization and the activation function.

In this experiment, the pruning process of the Yolov3 model can be divided into basic training - sparse training - pruning, in which sparse training takes a long time. Specifically, the pruning model code needs to obtain the parameters of the original model, and obtain the convolutional layer that can be pruned, that is, the convolutional layer followed by the BN layer. Because this experiment is based on the "γ" parameter of the BN layer for pruning, then sort "γ", prune the convolutional layer channels corresponding to "γ" below 60%, generate the pruned model. The model parameters and model configuration files before and after pruning correspond one by one. The effect of pruning is reflected in the model inference of the experiment, which can reduce the convolution operation, compress the model size, and reduce the logical resource occupation of the hardware during deployment. The Yolov3 model performs object detection on each frame of the video and forms a new video, which can intuitively see the object type and recognition accuracy on the video.

3.2.2. Experimental results and analysis. The above table shows the size comparison of the configuration files before and after pruning of the Yolov3 model. The size of the model after pruning by 60% is six times smaller than the size of the original model. Therefore, pruning can reduce the size of the model and speed up model inference and deployment.

DOI: 10.54254/2755-2721/2/20220583

Table 2. Model size comparison of Yolov3 before and after pruning.

If pruning	Yes	No
Size	41.5 MB (41,465,888bytes)	248.0 MB (248,007,048bytes)

The above table shows the size comparison of the configuration files before and after pruning of the Yolov3 model. The size of the model after pruning by 60% is six times smaller than the size of the original model. Therefore, pruning can reduce the size of the model and speed up model inference and deployment.

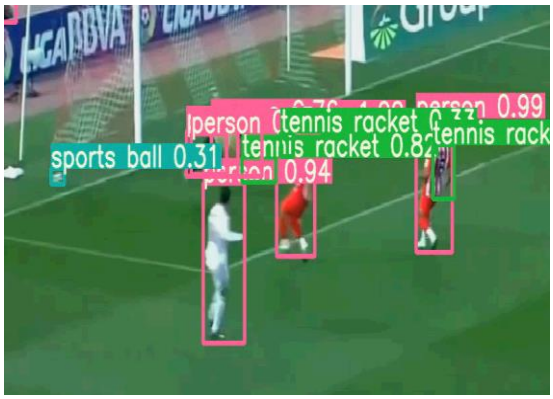

Figure 2. Video object detection before Yolov3 model pruning.

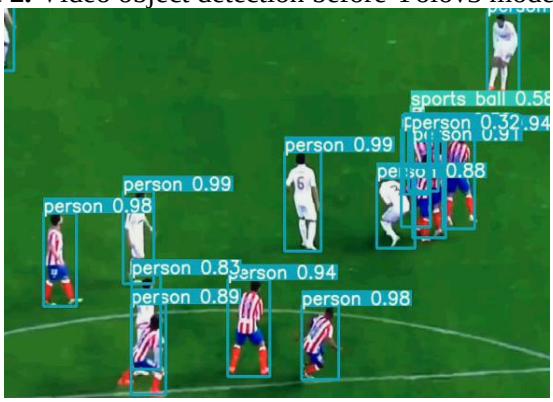

Figure 3. Video object detection after Yolov3 model pruning.

The above picture is a screenshot of the video target detection before and after pruning of the Yolov3 model. Through the accuracy comparison of image units, the recognition accuracy of "people" is basically the same before and after pruning of the two pictures, and there is little difference in the recognition of "objects", and Different objects can be correctly identified. Pruning according to the "γ" parameter of the BN layer will greatly reduce the amount of Yolov3 model parameters, and the model accuracy will hardly be affected.

3.3. Real-time object detection on embedded GPU based on convolution layer pruning of Yolov3 algorithm

3.3.1. Experimental purpose and module operation introduction. In this experiment, the Yolov3 model before and after pruning is used for real-time target detection on the embedded GPU, and the frame rate of target recognition is compared to reflect the optimization effect of pruning on the model hardware deployment. The embedded GPU is "NVIDIA TX2", and the CPU integrated with ARM architecture is

DOI: 10.54254/2755-2721/2/20220583

used to read and write graphics and process IO, thereby realizing a heterogeneous hardware system. This model is small, suitable for building end-to-end testing, and can be used in cameras, drones, sweeping robots, etc. The model is connected to the camera through USB, the resolution is 640*480 pixels, the CPU can install the ubuntu18.04 operating system, and the display is connected to the camera through HDMI to complete real-time object detection.

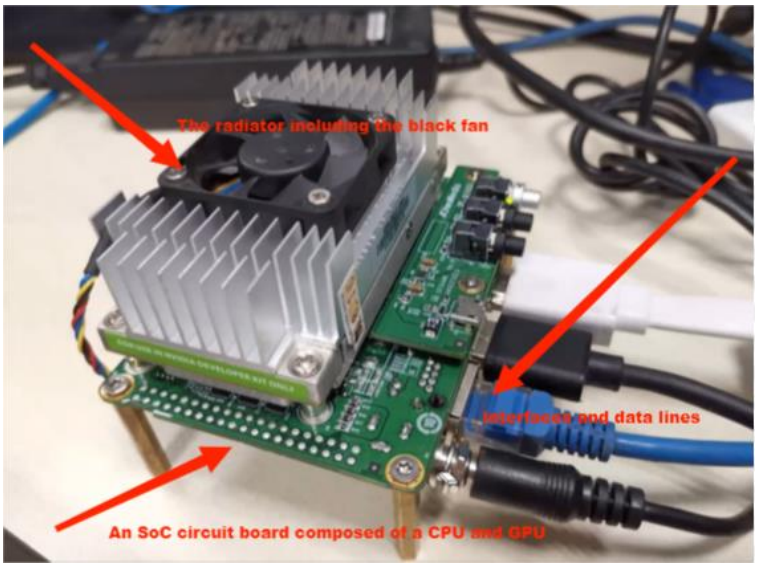

Figure 4. GPU-based SoC (Heterogeneous structure of CPU and GPU) embedded module. (1) The under layer of the embedded module is a GPU-based SoC circuit board. (2) The upper layer is the radiator. The fan in black is used to cool the GPU.

The above picture shows the part of platform built in the laboratory and the embedded GPU "NVIDIA TX2" connected to the radiator. The model has a radiator because of its high-power consumption and requires active heat dissipation. The CPU of the embedded module is interconnected with the server through a data line.

The experimental plan is divided into three steps. First, use the terminal to remotely log in to the "TX2" embedded platform, secondly deploy the target detection model on "TX2", copy the "cfg" and "weight" parameter files before and after pruning to the embedded terminal, and finally deploy the algorithm model before and after pruning to "TX2". Real-time target detection is performed by turning on the camera, and the difference in target detection speed and recognition accuracy before and after pruning is compared.

3.3.2. Experimental results and analysis. The below picture shows the comparison of target detection accuracy before and after pruning. It can be seen from the comparison data that the accuracy of target detection is basically unchanged only after model pruning. Different objects in the picture and even part of the structure of the object can be accurately identified. However, the object detection speed of the model is doubled after pruning, according to the FPS data displayed from the terminal server. This is because pruning can delete some unnecessary neurons and their connections, which greatly reduces the number of parameters and storage space occupied by the model, while the parts that are useful for target detection are still retained. Therefore, the occupancy rate of logical resources can be reduced in the process of deploying to hardware, so that GPU can reduce power consumption while accelerating computing.

DOI: 10.54254/2755-2721/2/20220583

Figure 5. Real-time object detection before Yolov3 model pruning.

Figure 6. Real-time object detection after Yolov3 model pruning.

Table 3. Optimization effects before and after Yolov3 model pruning.

Model: Yolov3	Profile size	Real-time object detection speed	Object Detection Accuracy
Before pruning	248.0MB	Average 5FPS	high
After pruning	41.5MB	Average 10FPS	high

According to the results of experiments 3.2 and 3.3, the above table summarizes the comparison of model size, object detection speed and accuracy in practical applications before and after Yolov3's model pruning. Using the "γ" parameter of the BN layer to prune the Yolov3 model successfully greatly reduces the amount of model parameters and can also improve the frame rate of target recognition without losing model accuracy. Therefore, experimental results show the effectiveness of Yolov3 pruning algorithm and there is no dead loop.The development of pruning technology is not limited to structured pruning and unstructured pruning. New techniques such as sparse pruning, automated pruning and small model pruning will gradually become the focus of technology.

3.4. Summary

In this chapter, the algorithm and model used in this paper are AlexNet and Yolov3 convolutional neural network model, respectively. Through the optimization strategy mentioned in chapter 2, this chapter completed the INT8 quantization of neural network model parameters of AlexNet was carried out based on the concept of KL divergence. Then, the Yolov3 convolutional neural network model pruning is completed based on the search of the "γ" parameters of BN layer after the convolutional layer. Whether it is CNN parameter quantization or pruning of CNN channel or neuron. In the operation of the code, the two algorithms have no extra error although due to the modification of the algorithm such as the BN layer and the use of "TensorRT". There is no code redundancy or inefficiency, and the code runs without

DOI: 10.54254/2755-2721/2/20220583

infinite loop. On the server is reflected in the code running speed, convergence becomes faster. The above three experiments optimize the models of the two algorithms, and both have the effect of accelerating of image recognition and target detection or reducing the model storage space. The optimization results are as expected, so this experiment has played a practical role in the quantization and pruning methods of the two algorithms.

4. Discussion

Reviewing the research methods used in this paper, for the KL divergence mentioned by the AlexNet model quantization, this concept can be applied in many fields. Feng L et al. proposed a new distance metric learning approach in Learning a Distance Metric by Balancing KL-Divergence for Imbalanced Datasets, called distance metric by balancing KL-divergence (DMBK), handles imbalanced datasets, and reduces confusion for machine learning tasks [10]. Compared with model quantization and pruning, sparsity perception will be the frontier of model compression. In this regard, Changpinyo S in The Power of Sparsity in Convolutional Neural Networks generalizes 2D convolution to the use of channel sparse connection structures, disabling the connections between filters in convolutional layers, saving running time and memory for many network architectures [11]. At the same time, Liu Z in Learning Efficient Convolutional Networks Through Network Slimming reduces model size by enforcing channel-level sparsity in the network, reduces the number of computational operations without affecting accuracy, thereby completing the network Slimming [12]. It is also a cutting-edge research field of model compression, but different from pruning and quantization, knowledge distillation refines the knowledge in the model collection into a single model. By building a lightweight small model, using the supervision information of the large model with better performance to train this small model to achieve better performance and accuracy. This method was first proposed by Hinton in the paper Distilling the Knowledge in a Neural Network in 2015 and applied to the classification task [13]. The neural network algorithm will increasingly complex with the increase in the complexity of the realization function. While the number of algorithm parameters continues to increase, the optimization strategy of the neural network model should be continuously optimized, which will also provide help for the development of artificial intelligence in the future.

5. Conclusion and outlook

This paper completes the overview of the quantization of the neural network model AlexNet and the model pruning of Yolov3, discusses the KL divergence and the BN layer. It proves through experiments that the model quantization method based on the KL divergence makes the AlexNet model image recognition time about 1/3-1/4 of the original format, but the recognition accuracy is basically the same. In addition, the model pruning of Yolov3 according to the "γ" parameter of the BN layer will reduce the amount of model parameters while making the video object detection accuracy not reduced. Finally, in the real-time target detection of the embedded GPU, the object detection frame rate of the pruned Yolov3 model is doubled, and the recognition accuracy is almost unchanged. Thus, it can be seen a good neural network model optimization will hardly affect the model accuracy when the algorithm is slimmed down, and at the same time, it will accelerate the deployment efficiency of software to hardware and reduce the logical resource occupation of hardware. In general, pruning and quantization are the basis of model compression and optimization, and advanced neural network algorithms such as knowledge distillation are the frontier areas of artificial intelligence in neural network compression. In addition to model optimization at the algorithm level, the acceleration architecture design of artificial intelligence hardware is also an important part of improving the speed of model deployment. For example, while optimizing the algorithm, combining the optimization strategy of the FPGA to complete the loop optimization and array optimization of the FPGA based on HLS (High-level-synthesis), designing the AI chip to realize the trade-off between hardware data flow processing speed and logical resource occupation. Thus, it can accomplish the acceleration goal of deep learning software and hardware co-design: customization + heterogeneity = high energy efficiency.

DOI: 10.54254/2755-2721/2/20220583

References

[1] Dixit M, Tiwari A, Pathak H, et al. An overview of deep learning architectures, libraries and its applications areas[C]//2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). IEEE, 2018: 293-297.

[2] Zhang Z. Model Pruning Techniques for Boosting the Inference Efficiency on Embedded Systems[C]//2021 2nd International Conference on Computing and Data Science (CDS). IEEE, 2021: 119-124.

[3] Liu J, Tripathi S, Kurup U, et al. Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey[J]. arXiv preprint arXiv:2005.04275, 2020.

[4] Mousa-Pasandi M, Hajabdollahi M, Karimi N, et al. Convolutional Neural Network Pruning Using Filter Attenuation[C]//2020 IEEE International Conference on Image Processing (ICIP). IEEE, 2020: 2905-2909.

[5] Zhu F, Gong R, Yu F, et al. Towards unified int8 training for convolutional neural network[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 1969-1979.

[6] Gheorghe Ș, Ivanovici M. Model-based Weight Quantization for Convolutional Neural Network Compression[C]//2021 16th International Conference on Engineering of Modern Electric Systems (EMES). IEEE, 2021: 1-4.

[7] Liang T, Glossner J, Wang L, et al. Pruning and quantization for deep neural network acceleration: A survey[J]. Neurocomputing, 2021, 461: 370-403.

[8] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[9] Bjorck N, Gomes C P, Selman B, et al. Understanding batch normalization[J]. Advances in neural information processing systems, 2018, 31.

[10] Feng L, Wang H, Jin B, et al. Learning a distance metric by balancing kl-divergence for imbalanced datasets[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 49(12): 2384-2395.

[11] Changpinyo S, Sandler M, Zhmoginov A. The power of sparsity in convolutional neural networks[J]. arXiv preprint arXiv:1702.06257, 2017

[12] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2736-2744.

[13] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015, 2(7).