**Department of Computer Science**

**BSCCS Final Year Project Report**

**2021-2022**

**21CS108**

**Compression and Acceleration for Neural Networks on Mobile Devices**

**(Volume of )**

**For Official Use Only**

|  |  |  |
| --- | --- | --- |
| Student Name | **:** | **CHEN Zirui** |
| Student No. | **:** | **55715008** |
| Programme Code | **:** | **BSCEGU4** |
|  |  |  |
|  |  |  |
|  |  |  |
| Supervisor | **:** | **Dr LI, Zhenjiang** |
| 1st Reader | **:** | **Dr GUAN Nan** |
| 2nd Reader | **:** | **Prof LIANG Weifa** |

**Student Final Year Project Declaration**

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Compression and Acceleration for Neural Networks on Mobile Devices

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name: | CHEN Zirui |  | Signature: | 图片包含 游戏机, 男人, 虫  描述已自动生成 |
| Student ID: | 55715008 |  | Date: | March 29, 2022 |

**abstract**

In modern society, the importance of mobile electronic devices in people's lives is becoming more and more prominent. It is an excellent choice to install neural networks on mobile terminals to make mobile electronic devices more intelligent, convenient and deal with more problems.

As high-quality solutions for complex issues such as image recognition and classification, many scholars' research has focused on neural networks in recent years. There are many different methods and corresponding research results on improving the speed and reducing the size of neural networks. However, most of the related research is carried out in the PC (GPU) environment, and there is a large gap between the hardware conditions of mobile devices and PCs. The performance of existing methods is still unclear for neural networks deployed on mobile terminals without GPU support and large memory size.

This project will compress and test a typical neural network to observe the model's size, accuracy, and running time on the mobile terminal and compare the advantages and disadvantages of various standard neural network compression methods.

**Table of Content**

[abstract 4](#_Toc99550018)

[1. Introduction 6](#_Toc99550019)

[1.1 background information 6](#_Toc99550020)

[1.2 motivation 7](#_Toc99550021)

[1.3 objective and scope 8](#_Toc99550022)

[2. literature review 9](#_Toc99550023)

[3. Methodology 9](#_Toc99550024)

[3.1 quantization 10](#_Toc99550025)

[3.2 pruning 11](#_Toc99550026)

[3.3 low-rank approximation 11](#_Toc99550027)

[4. Implementation 12](#_Toc99550028)

[4.1 design 12](#_Toc99550029)

[5. Testing result 14](#_Toc99550030)

[5.1 Size and Accuracy 15](#_Toc99550031)

[5.2 Running time on android phone 16](#_Toc99550032)

[6. Conclusion 17](#_Toc99550033)

[6.1 Future improvement 17](#_Toc99550034)

[6.2 Summary of achievement 17](#_Toc99550035)

[reference list 19](#_Toc99550036)

***1. Introduction***

**1.1 background information**

Since the birth of the computer, the miniaturization and portability of electronic equipment have always been development trends. From smartphones owned by everyone to various wearable electronic devices still in development, such as watches, glasses. People hope that these devices can help deal with some problems in daily life, such as analyzing the health status and formulating an appropriate diet plan or assisting colorblind patients in identifying the color of traffic lights. People tend to use more portable and more intelligent products.

Tasks such as recognizing traffic lights in the field of vision and reading handwritten fonts may be easy for our brains. However, this type of problem is very complicated for traditional computer algorithms. It could be hard to Convert these kinds of information into data that computers can read and process. Fortunately, the neural network algorithm can effectively answer this type of problem.

Neural networks are one kind of machine learning method and are the foundation of Deep Learning algorithms, one of the most popular study directions in the last several years [1]. The idea of neural networks came from information transmission between neurons in the human brain. By adjusting the parameters of many nodes in numerous layers, the neural network can cope with some problems that traditional algorithms cannot handle.

Due to the structural characteristics of the neural network (consisting of thousands of neural nodes and a large number of activation functions between them), its training and convergence will be very time-consuming. Compared with manual recognition by human experts, tasks in speech recognition or image recognition may take minutes, even hours. Therefore, many improvements to neural network algorithms have focused on the compression and acceleration of neural networks.

**1.2 motivation**

Neural networks, particularly deep-level neural networks, have evolved a lot over the past few years. With higher accuracy, the complexity of neural network algorithms and the requirements for hardware have also increased a lot [2]. Therefore, the research on compression and acceleration of neural networks has become more meaningful.

Although the hardware performance of the mobile terminal is developing rapidly, the execution of the neural network algorithm with high computational cost in the local of these devices is still quite limited [3]. Nowadays, most of the neural networks on the mobile terminal need to be connected to the Internet, resulting in quite limited applications.

Neural network compression and acceleration techniques, including quantization, pruning, and low-rank approximation, have achieved good performance, effectively compressing the model volume without significantly affecting the model's accuracy and running speed. Nevertheless, most of these techniques are developed with GPUs as the target object because of the high efficiency of GPUs in processing neural network-related computations. The hardware conditions and application requirements on a mobile platform or other embedded systems vary significantly. In the absence of GPU support and minimal memory capacity, it is still uncertain whether these compression methods can play the original effect. Thus, additional tests are necessary to transplant the mobile terminal neural network. [3] The test results can provide a specific reference for the optimization and processing of the neural network model of the mobile terminal in the future.

**1.3 objective and scope**

This project compares each compression method's performance improvements to mobile platforms and suggests which technologies are the most suitable for the specific performance requirements. On the other hand, because acceleration and compression will inevitably lead to data loss and decreased accuracy, understanding the conversion relationship between accuracy and speed helps achieve a better balance to achieve the desired effect.

Neural networks have various branches. Common neural network categories include multi-layer perceptron (MLPs), Convolutional neural networks (CNNs), and Recurrent neural networks (RNNs). Each type would have its structural characteristics and application scenarios.

This project will focus on VGG-16, one popular type of CNN model. We plan to use it to solve the problem of image classification, one of the most classic topics in neural networks. The reason to choose this topic is that it has a certain degree of difficulty but does not have high requirements for computational power, which meets the conditions of mobile devices. The compression and acceleration methods that this project will use will involve quantization, pruning, and low-rank approximation, which have proven their effectiveness in accelerating ordinary neural networks.

***2. literature review***

There has been considerable and relatively mature research on various compression methods of neural networks on GPUs. According to the official documentation of Pytorch, techniques such as quantization can increase the running speed by three times while ensuring that the accuracy loss (original accuracy/compressed accuracy) is less than 1.01 [5].

Regarding the performance of compression techniques on mobile devices, Kaiming Nan, Sicong Liu, Junzhao Du, and Hui Liu conducted related research in 2019. Their study tested the performance of 10 compression methods, including weight factorization and pruning, convolution decomposition, and unique layer architecture designing, on several simple CNN network structures such as LeNet and AlexNet. From the research results, the compressed model, the volume can be reduced to 1/4 of the original while maintaining an accuracy loss below 1.05 and a similar running time[3].

Our project will test the performance of compression techniques such as quantization, pruning, and low-rank approximation on mobile networks. This project will test the more common VGG16 model compared to previous research. Compared with simple models such as LeNet or models specially designed for mobile terminals such as mobilev2, the test of VGG16 can reflect more general results.

***3. Methodology***

This section will review several current effective neural network compression and acceleration techniques that we will apply in this project. We will explain their basic working principles and the positive effects.

**3.1 quantization**

Quantization means using low bit-width numbers to calculate and store, influencing the digital width of inputs, activations, weights, and biases used in the neural network [5]. Generally, these parameters would be represented in float format while the quantization method applies integer even binary numbers. The advantage of this is that, on the one hand, it optimizes the storage of data and allows a more compact network model. On the other hand, it simplifies the calculation process and enables the neural network to perform more efficient calculations (compatible with hardware devices that only support integers). Some research proved that quantization could improve the latency of the CPU and hardware accelerator and, at the same time, hardly reduce the accuracy of the model [6].

图表, 折线图

描述已自动生成

Image 1: Schematic diagram of quantization principle

**3.2 pruning**

In a neural network, the valid layers and key nodes that play an indispensable role during the calculation process do not account for a high proportion of the entire neural network structure. Based on this view, the concept of pruning is to Simplify the neural network structure by removing unimportant layers or nodes. According to the different pruning objects, the genres of pruning algorithms incorporate Filter-level Pruning (corresponding to direct channel reduction and filter reduction) and Fine-grained Pruning (adjusting a single weight) [9]. Many scholars have proven the effect of pruning on reducing network complexity and preventing overfitting [10].

手机屏幕截图

中度可信度描述已自动生成

Image 2: Schematic diagram of the pruning principle

**3.3 low-rank approximation**

Low-rank approximation improves the network by using the low-rank matrix to represent a similar original high-rank matrix. Therefore, we can reduce parameter storage and computation. On the other hand, decomposition minimizes the need for parameters of neural networks. Reducing the number of parameters means a more compact neural network and lower memory requirements.

CP-decomposition is a standard Low-rank approximation method in the computer field [11]. Its advantage is that as a primary and mature tensor linear algebra implementation, many algorithms such as NLP can implement CP-decomposition relatively simply. Like CP-decomposition, as a relatively mature decomposition method, it is accessible to implement Tucker-decomposition on neural networks.

图表

描述已自动生成

Image 3: Schematic diagram of decomposition principle

***4. Implementation***

**4.1 design**

In the project's first phase, we built the VGG-16 network through Pytorch on the PC and used the CIFAR-10 data set for training. The CIFAR-10 dataset is a dataset of 60,000 images with ten categories. As an officially supported dataset, many studies have used the cifar10 dataset, which provides a large amount of data to compare with our tests.

The mobile phone model used in the test is the HUAWEI P30 Pro, equipped with a Kirin980 processor (with 8 CPUs) and 8GB of memory capacity. As a processor released in 2018, it can better represent the computing power of an average mobile phone.

**4.2 Compression methods**

The compression methods we will test include: 1. static quantization and static quantization aware training (since CNN type networks do not support dynamic quantization). 2. Prune units in a tensor by removing useless nodes, 3: Tucker decomposition method and CP decomposition method.

The quantization method provided by PyTorch performs the quantization test. Two new layers, QuantStub and DequantStub, are added to the original network to apply quantization. Different static quantization configurations are achieved by using different qconfigs. (default config corresponds to static default quantization, "fbgemm"("qnnpack") corresponds to static optimal quantization.)

Torch-pruning package could assist in getting the model pruned. The operation of the pruning tutorial provided by the official PyTorch documentation can only set the parameters of the nodes with lower weights to 0. The nodes themselves will remain so that the model's size cannot stay unchanged. In contrast, torch-pruning can describe the dependencies between layers by building a dependency graph of the model and creating layers with less influence.

The implementation of decomposition will go through tensorly. Tensorly package can provide very convenient CP-decomposition and tucker decomposition transformation.

After completing the pre-training and experiment of each method on the PC side, we will deploy the network to the android mobile terminal through the android studio and compare their performance differences under the android terminal conditions. We will count the time it takes for the compressed model to process 1000 images and compare it with the original model.

图片包含 游戏机

描述已自动生成

Image 4: Test Program User Interface

***5. Testing result***

**5.1 Size and Accuracy**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Origin | Static (Default) | Static (optimall) | QAT | Prun  (20%) | Prun  (40%) | CP | Tucker |
| Accuracy (%) | 94.18 | 93.83 | 94.09 | 98.833 | 93.6 | 92.2 | 82.76 | 83.87 |
| Size (MB) | 134.65 | 33.78 | 33.86 | 33.99 | 112.17 | 93.79 | 106.7 | 107.1 |

Table 1: Model Size and accuracy result comparison

We can observe from the table that the quantized model's volume dropped to about 1/4 of the original. Given that the principle of quantization is to store data (weights) held initially in 32 bits in 8 bits, the change in model volume is within the expected range. Compared with the original model, the accuracy of the quantized model does not show a significant decline. The most prominent result is the QAT method, whose accuracy exceeds the original model.

Since the pruning ratio only acts on the convolutional layer, the model's pruning will be slightly larger than the result calculated directly from the pruning ratio. In the two tests with a pruning ratio of 20% and 40%, the model's accuracy after direct pruning dropped to about 70%. After retraining, their accuracy return to up to 90%. For a higher pruning ratio of 50%, the model's accuracy still cannot reach 90% after retraining.

The effects of CP-decomposition and Tucker-decomposition are similar. The model volume declined to about 106 MB. The accuracy rate has also dropped to about 82%, which is the most sacrificed in accuracy among the three methods.

In summary, quantization maintains the highest accuracy in the case of the highest compression ratio. Pruning under a certain percentage can reduce the model's size while maintaining an accuracy similar to the original model. Compared with the above two methods, decomposition's effect in reducing volume and maintaining accuracy is not outstanding.

**5.2 Running time on android phone**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Origin | Quantization | Pruning 40% | Decompostion |
| Running time (ms) | 1198 | 1375 | 7775 | 684 |
| Size (MB) | 269.1 | 67.6 | 187.5 | 214.2 |

Table 2: Running time on mobile terminal result comparison

For the model to run on an android phone, the optimize\_for\_mobile function is necessary to adjust the model. The processed model doubled in size compared to its original size. (Because there is no apparent gap between different quantization and decomposition methods, they are classified into two categories: quantization and decomposition.)

We can find that the model after decomposition takes the least running time when comparing the running time. The reason may be that smaller matrix operations are more advantageous on mobile.

The model processed by quantization does not increase much in operation time. The low-bit operation is more in line with the computing method of the mobile processor.

The response time of the model after pruning increased dramatically. The possible reason is that it does not compress to the characteristics of mobile computing. At the same time, the optimize\_for\_mobile function may not be compatible with torch-pruning, so the pruned model cannot be adequately optimized.

***6. Conclusion***

**6.1 Future improvement**

The test of this project selects the commonly used VGG model structure. Different models have different fitness for each compression method. The applicability of the test results of this project may be limited to the CNN model. The selection of compression methods for other model structures still needs further research.

Combining each compression method is also a subject worthy of further study. Since Pruning, Quantization, and low-rank approximation have different principles and act on other parts of the neural network, it is feasible to implement them in sequence on the same network. However, the test results show that although the multi-step compressed model can show good performance on the PC, it cannot run successfully on the mobile terminal. Optimizing the compression technology to match each other and be compatible with mobile programs is also a developing direction.

**6.2 Summary of achievement**

This project aims to test the performance of standard neural network compression techniques: quantization, pruning, and decomposition on mobile networks. The experiments use the VGG16 network as a representative model, train on the cifar10 dataset, and test on Huawei 980 mobile phones.

The test data shows that quantization, the most commonly used method for optimizing the neural network to the mobile terminal, has the best comprehensive effect. Quantization can use 8 bits instead of 32-bit data to reduce the model’s size to about 1/4 of the original while maintaining high accuracy. Since the low-bit data fits the computing model of the mobile processor, quantization will not bring too much delay.

The pruned model has better performance on the PC side and can maintain a high accuracy rate when compressing the model volume (before excessive pruning). However, testing on mobile shows that the pruned model takes more time to run than the original. Follow-up compatibility and optimization are still required to apply pruning technology on the mobile network.

Compared with the first two kinds of decomposition, the accuracy of its model has dropped significantly, and the reduced volume cannot be compared with quantization. However, after the decomposition process, the running speed of the model on the mobile terminal has been dramatically improved. Low-rank decomposition can be used as a choice for pursuing model response speed.

In general, the test report of this project can provide a reference for the application of neural networks on mobile terminals in the future. According to different requirements (the smallest model size or the fastest response speed), developers can choose a specific compression scheme. Some deficiencies of the existing compression technology reflected in the testing process can also point out some directions for the future development of neural network technology.

***reference list***

[1] IBM Cloud Education. (n.d.). What are neural networks? IBM. Retrieved October 17, 2021, from <https://www.ibm.com/cloud/learn/neural-networks>.

[2] Cheng, J., Wang, P., Li, G., Hu, Q., & Lu, H. (2018). Recent Advances in Efficient Computation of Deep Convolutional Neural Networks.

[3] Nan, K., Liu, S., Du, J., & Liu, H. (2019). Deep model compression for mobile platforms: A survey. Tsinghua Science and Technology, 24(6), 677–693. https://doi.org/10.26599/TST.2018.9010103 [4] Nielsen, M. “Neural networks and deep learning” Determination Press, 2015

[5] Quantization. (n.d.). Retrieved October 20, 2021, from <https://pytorch.org/docs/stable/quantization.html>

[6] post-training quantization: TensorFlow Lite. (n.d.). Retrieved October 20, 2021, from https://www.tensorflow.org/lite/performance/post\_training\_quantization

[7] Wikipedia contributors. (2021, October 19). Sparse dictionary learning. In Wikipedia, The Free Encyclopedia. Retrieved 15:45, October 20, 2021, from <https://en.wikipedia.org/w/index.php?title=Sparse_dictionary_learning&oldid=1050680408>

[8] Bhattacharya, S., & Lane, N. (2016). Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM <https://doi.org/10.1145/2994551.2994564>

[9] Mao, H., Han, S., Pool, J., Li, W., Liu, X., Wang, Y., & Dally, W. J. (2017). Exploring the Regularity of Sparse Structure in Convolutional Neural Networks. Retrieved November 11, 2021, from <https://arxiv.org/pdf/1705.08922v3.pdf>

[10] Han, S., Mao, H., & Dally, W. J. (2015). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.

[11] Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition. Retrieved November 11, 2021, from <https://arxiv.org/pdf/1412.6553v3.pdf>

[12] Yuzhenhan （n.d.）Quantize\_Pytorch\_Vgg16AndMobileNet Retrieved March 30, 2022, from <https://github.com/Forggtensky/Quantize_Pytorch_Vgg16AndMobileNet#readme>

[13] Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition.

[14] Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning Filters for Efficient ConvNets.

[15] Gongfan Fang (n.d.) Torch-Pruning Retrieved March 30, 2022, from <https://github.com/VainF/Torch-Pruning>

[16] Tensorly (n.d.) Retrieved March 30, 2022, from <https://github.com/tensorly/tensorly>

[17] Jean Kossaifi (n.d.) tensorly-notebooks Retrieved March 30, 2022, from <https://github.com/JeanKossaifi/tensorly-notebooks>

[18] Jacob Gildenblat (n.d.) pytorch-tensor-decompositions Retrieved March 30, 2022, from <https://github.com/jacobgil/pytorch-tensor-decompositions>

[19] Pytorch mobile android (n.d.) Retrieved March 30, 2022, from <https://pytorch.org/mobile/android/>