THE DEEP DISTRIBUTED LEARNING IMPLEMENTATIONS WITH A LARGE DATASET ON THE DESKTOP GRID

I.I. Kurochkin[1,a]

[1]*Institute for Information Transmission Problems of Russian Academy of Sciences, Bolshoy Karetny per. 19, build.1, Moscow, 127051, Russia*

E-mail: [a] qurochkin@gmail.com

Machine learning methods and in particular deep neural networks are often used to solve the problem of image classification. There is a trend towards an increase in training data and an increase in the size of neural network architectures. The process of training a deep neural network with millions of parameters can take thousands of hours on modern computing devices. Distributed computing systems can be used to reduce training time. The wide scalability of grid systems and the ease of connecting new computational nodes can significantly reduce the training time for deep neural networks. But at the same time, you should take into account the peculiarities of data exchange between the nodes of the desktop grid system. Methods of organizing distributed deep learning are proposed using the example of the image classification problem. The technique of random formation of a local dataset is proposed. The results of using synchronous and asynchronous approaches to distributed learning are shown.

Keywords: deep neural networks, distributed computing systems, desktop grid, large dataset, image classification, distributed deep learning

Ilya Kurochkin

1. Introduction

Machine learning methods and deep neural networks are quite relevant and effective for solving various problems of image classification in applied problems [1]. Training deep neural networks is computationally hard. Deep learning with large datasets can take weeks or even months on modern personal computers or servers. With the help of modern video cards (GPU), training time can be significantly reduced compared to using only the CPU. To minimize the training time for deep neural networks, high-performance computing systems are used. Unfortunately, not all researchers have access to supercomputers. As a result, they use either reduced datasets or simplified deep neural network architectures.

Therefore, the use of distributed computing systems is a logical step to reduce training time. Distributed computing systems have a number of features [2]: heterogeneity of computing nodes; possible failure of nodes and communication channels; occurrence of errors and delays in calculations and data transmission; autonomy of calculations at the nodes and the difficulty of coordination.

Some types of distributed computing systems have a control node (server). Such centralized distributed systems include desktop grid systems and mobile grid systems. There are various software for organizing such grid systems, for example: BOINC [3], HTCondor, Grid Engine and etc.

BOINC is one of the popular platforms for organizing distributed computing and voluntary distributed computing. On the basis of this platform, more than 100 projects have already been deployed to solve various scientific problems. BOINC consists of a client and a server. The computing node of the desktop grid (BOINC client) periodically requests new tasks on the server and sends the results [3]. The server part is deployed under the Linux operating system and consists of several separate applications (daemons). The BOINC backend has a shared MySQL database or a MariaDB distributed database. These daemons can be run asynchronously. Main BOINC daemons: work generator, scheduler, feeder, transitioner, validator, assimilator, file deleter and database purger. If necessary, each of their daemons can be modified or changed. For example, the validator's algorithm can be changed in validator-daemon to check the correctness of the results obtained.

Data exchange between the nodes of the desktop grid, bypassing the server, is not expected. This imposes additional restrictions on the formation of input data for subtasks and scheduling the distribution of tasks among computational nodes. It is necessary to adapt the deep learning for computing on a desktop grid system. The separation of one computational learning task into many autonomous subtasks should occur both between computational nodes (parallel execution) and over time (sequential execution) [4]. With time separation, coordination between nodes becomes possible via the server.

Deep neural networks can be used not only to determine the presence of objects in images, but also to classify texture images. In images where there are no objects with clear contours or the number of objects is so large that it is difficult and meaningless to select all the objects separately. In these cases, it is best to use all the information in the image and define it as a texture. This approach is often used for multiple images obtained using optical microscopy techniques.

2. Big datasets

One of the main goals in preparation for training deep neural networks is the dataset creation. For the problem of image classification, a *dataset* is a set of images of the same size. In this case, only one class value can be assigned to one image.

There are reference datasets with texture images. They can be used to check the adequacy of the chosen architecture of a deep neural network. The reference classification problem can be used to test scheduling systems and the quality of deep learning in a desktop grid system. For example, the following datasets can be cited: Kylberg Texture Dataset (28 classes; image size: 576x576 pixels) [5], Brodatz (112 classes; image size: 512x512 pixels), KTH-TIPS2 (11 classes; image size: 200x200

pixels). The image size in these datasets is small and does not exceed 600x600 pixels. This is due to the size of the input layer on popular deep neural network architectures. Such architectures for solving problems of classification of texture images include: VGG16, AlexNet, T-CNN [6], InceptionV3, ResNet50.

As a rule, when obtaining images by various microscopy methods, the size of the images is significantly higher, for example: 2272x1704 or 1920x1080 [7]. This means that the original image should be divided into fragments. At the same time, it should be taken into account that important information can get to the border of the fragment. This situation can be corrected by the method of splitting into fragments, where fragments from the original image will be taken with an overlap of 30% -50% (Figure 1).

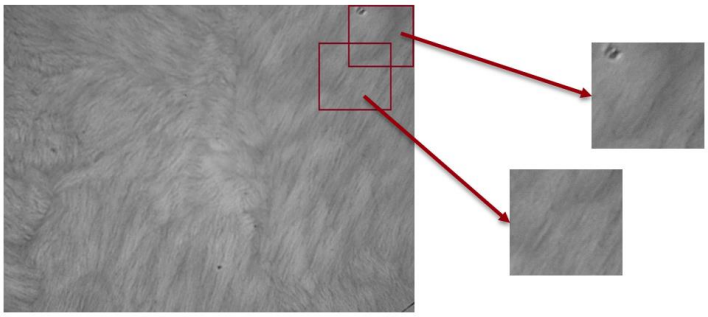

Figure 11. Dividing the image into fragments

As a result, the number of images for the dataset increases significantly. In addition, it should be taken into account that the orientation of the images obtained using microscopic methods is not known. This means that you should place not only fragments of the original images, but also fragments of images rotated at different angles. As a result, from one original image of size 2272x1704, you get: 192 fragments, taking into account 50% overlap in each coordinate, and 6972 fragments, taking into account the rotation angle from 0 to 355 degrees with a step of 5 degrees.

As a result of augmentation, a dataset of 4,559,688 fragments is obtained from 654 source images. The number of rotation angles during augmentation may be less, but the size of the dataset will still be large. On disk, such a dataset will occupy about 65 GB.

3. Distributed deep learning

To train most architectures of convolutional neural networks, the SGD (Stochastic Gradient Descent) method and its variations are used, for example: ISGD, Adagrad, Adam [4]. Several approaches are used to distribute train the same deep neural networks. First of all, distinguish between the approaches of partitioning by model and partitioning by data. This paper assumes that the training model will be the same on all nodes of the desktop grid. The use of a large dataset does not make it possible to transfer it to all nodes of the desktop grid, which makes the data partitioning approach the only possible one. In accordance with this approach, the results of training local models will be aggregated on the server. The process of training a local model on a computational node consists of sequential iterations. The input data for each iteration will be the current gradients from the aggregated global model on the server. According to the methods of aggregating local gradients, a synchronous and asynchronous approach to learning is distinguished.

In the synchronous approach (all reduce), each iteration ends up collecting all the results of the local models. In the case of a heterogeneous distributed computing system, fast nodes will wait for slow nodes to finish. After the completion of the iteration, the local results are aggregated and a global gradients are formed. This set of gradients will be the input data for the next iteration of the local training models. The synchronous approach provides a relatively fast learning rate. The disadvantages of the synchronous approach include the irrational use of computing resources – small values of

utilization of computing nodes. This disadvantage can be partially compensated for when the results are collected only from a certain percentage of nodes (for example, 80% [8]).

In contrast to the synchronous approach, in the asynchronous approach there is no synchronization when aggregating the results of local models. The global model gradients on the server are updated asynchronously. But because of this, there is an effect of "aging of gradients", which significantly slows down learning process. There are modifications of asynchronous methods, including gradient sparsification [9].

Despite significant differences in the synchronous and asynchronous approaches, both of them can be implemented on the desktop grid. With any approach, the problem of a large dataset remains. Its location on the BOINC server will make it a bottleneck when transferring data and results. To unload the BOINC server, it is necessary to place the dataset on several FTP servers (Figure 2). Then each node has the opportunity to download data from one of the FTP servers, and at the same time it will not waste the resources of the main server of the grid system.

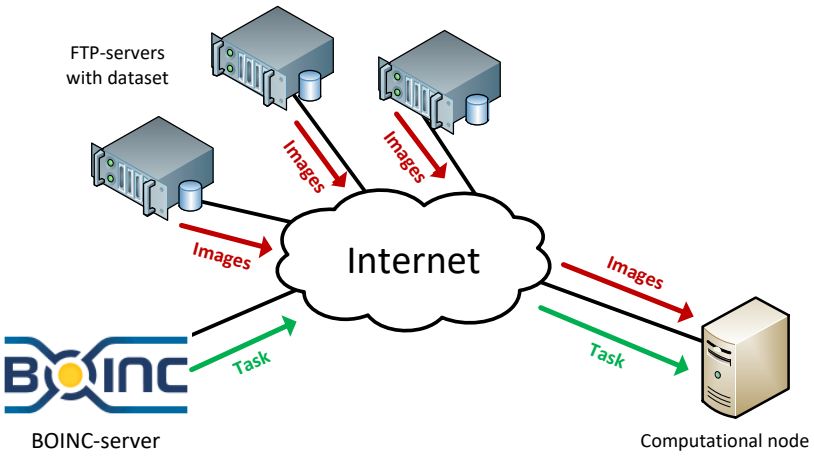

Figure 12. Data exchange schema

Taking into account the features of the desktop grid system, a logical step was the formation of a random local dataset for training the neural network at one iteration. The size of the local dataset can vary, and the presence of a large number of nodes in the desktop grid will allow you to cover the entire dataset. However, the dataset consists of fragments of the original images, therefore, when dividing into training and test samples, a restriction should be introduced. So for the formation of a local test sample, fragments of images that are currently participating in the training sample cannot be used. The balance requirement for all classes for the local dataset was taken into account.

4. Results

Experiments on distributed deep learning for solving the image classification problem were carried out. Synchronous (Table 1) and asynchronous (Table 2) approaches have been implemented. SGD algorithm and its modification DGS (Deep Gradient Compression) were implemented. Distributed learning implementations varied significantly among themselves. Various datasets were used: for the synchronous approach, a combined dataset was used (12 classes from Kylberg Texture Dataset + 3 class from selfmade big dataset [8]), and for the asynchronous approach, one of the CIFAR reference datasets was used.

Table 4. Results with synchronous distribute learning

Algorithm	Number of iterations	Number of nodes	Size of local train set	Size of local test set	Accuracy
synSGD	18	5	6 000	2 700	0.792

Table 5. Results with asynchronous distribute learning (reference dataset)

Algorithm	Number of iterations	Number of nodes	Size of local train set	Size of local test set	Accuracy
SGD	50	6	50 000	10 000	0.939
DGS	50	6	50 000	10 000	0.926

Despite the small number of nodes in test grid systems, the training results allow us to conclude that it is possible to conduct distributed deep learning on desktop grid systems. The overhead costs for the local datasets creation have been moved to the computational nodes side.

5. Conclusion

The implementation of synchronous and asynchronous approaches of distributed deep learning showed the possibility of using desktop grid systems to solve image classification problem. Placing a large dataset on independent FTP-servers relieved the load on the desktop grid server. The proposed method of forming local datasets will allow for a desktop grid with a large number of nodes to cover a dataset of a big size.

6. Acknowledgement

The reported study was funded by RFBR according to the research projects No. 18-29-03264 and No. 19-07-00802

References

[1] Y. LeCun, Y. Bengio, G. Hinton. “Deep learning” // Nature 521.– 2015.– pp. 436–444.

[2] Ian Foster, Carl Kesselman. “The Grid: Blueprint for a New Computing Infrastructure”, SF.: Morgan Kaufmann Publishers, 1998.

[3] Anderson, D.P. “BOINC: A Platform for Volunteer Computing.” // Journal of Grid Computing , 2020. DOI: 10.1007/s10723-019-09497-9.

[4] Ben-Nun, Tal, and Torsten Hoefler. "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis." ACM Computing Surveys (CSUR) 52(4), 2019. pp.1-43.

[5] Kylberg, Gustaf. Kylberg Texture Dataset v. 1.0. Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, 2011.

[6] Andrearczyk, Vincent, and Paul F. Whelan. "Using filter banks in convolutional neural networks for texture classification." Pattern Recognition Letters 84 (2016): pp. 63-69.

[7] Kolosova, Olga Yu, et al. "Cryostructuring of polymeric systems. 58. Influence of the H2N-(CH2) n-COOH–type amino acid additives on formation, properties, microstructure and drug release behaviour of poly (vinyl alcohol) cryogels." Reactive and Functional Polymers, Vol. 167, 2021. 105010. DOI: 10.1016/j.reactfunctpolym.2021.105010

[8] Kurochkin I.I., Kostylev I.S. Solving the Problem of Texture Images Classification Using Synchronous Distributed Deep Learning on Desktop Grid Systems. In: Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science, Vol. 1331. Springer, Cham, 2020. DOI: 10.1007/978-3-030-64616-5_55

[9] Y. Lin, S. Han, H. Mao et al. “Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training” In ICLR, arXiv:1712.01887, 2018.