Algorithm for Fast Image Compression on Heterogeneous Computing Devices

Vadim V. Matskevich, Viktor.V. Krasnoproshin
Belarusian State University
Minsk, Belarus
matskevich1997@gmail.com, Krasnoproshin@bsu.by

***Abstract*—In spite of the intensive development of computer technology, the problem of efficient use of computing resources remains an urgent issue [1], [2]. Computing devices may be heterogeneous (different in architecture and power). In this case, when solving applied problems, it becomes necessary to efficiently load them [3]–[6].**

The paper proposes an algorithm for loading heterogeneous devices, which allows speeding up the data processing process when solving the problem of image compression.

***Keywords*—Parallel computing, computing devices, system performance, neural network, training, dataset.**

I. Image compression using neural network (NN)

In the general case, a color raster image is described by a vector, each coordinate of which reflects its particular characteristic. Coordinate values can, for example, define [6]:

- the number of pixels in the image;
- color wavelength values;
- coordinates of specific pixels (if the image has an arbitrary shape);
- and etc.

Storage of color raster images in the form of a vector requires, as a rule, large amounts of memory [7]. There is a problem of their reduction, i.e. It is necessary to solve the problem of image compression.

Consider the process of compressing color images using a forward passing neural network (NN) [7] (see figure 1).

In general, the process of compressing images using neural networks consists of the following main steps.

Stage 1. The format of the original image is determined. Usually the image is considered as raster.

Stage 2. Vector $(x_1, x_2, ..., x_n)$, describing the original image is formed. For color images, each pixel is usually described by three coordinates of the vector. One coordinate determines the content of red, the second – the green and the third - the blue.

Stage 3. The neural network configuration is selected to implement the image compression process. The number of neurons in the input layer always corresponds to the number of coordinates in the vector $(x_1, x_2, ..., x_n)$. The output layer determines the size of the output vector $(y_1, y_2, ..., y_m)$.

Stage 4. As a result of neural network data processing, a vector is constructed $(y_1, y_2, ..., y_m)$, which describes the original image already in a compressed form $(m \leq n)$.

When solving the problem, the NN architecture of the following configuration was used.

The sizes of the input and output layers consisted of 48x48x3 and 12x12 neurons, respectively. Neighboring layers were represented as full bipartite graphs. In all neurons (except the input layer), a bipolar sigmoid function of activation with a unit coefficient was used.

$$f(S) = \frac{2}{1 + e^{-cS}} - 1 \quad (1)$$

The initial weights of the NA were generated uniformly distributed over the interval of values

$$\left[-\frac{16}{\sqrt{l_1}}, \frac{16}{\sqrt{l_1}}\right] \quad (2)$$

where l_1 – is the number of neurons in the first layer (when learning, the output signals of this layer are considered as input data for training).

Using the same formula, the values of the fictitious autoencoder weights were generated.

The learning rate (lr) for each layer throughout the training was a constant value and was calculated by the formula:

$$lr = \frac{2}{\sqrt{l_1}} \quad (3)$$

In contrast to the usual multilayer perceptron, the constructed network was trained as a deep NN.

II. Algorithm of loading heterogeneous computing devices

Color images were represented by a set of independent copies, formed in a separate package S. For compression, n computational (heterogeneous) U_i, $i = \overline{1, n}$ power devices were used.

It was necessary to construct such a partition of S into n parts, which would minimize the processing time of the entire package.

We denote by S_i the set of images processed on the i-th device and T_i – the time of its processing. Then,

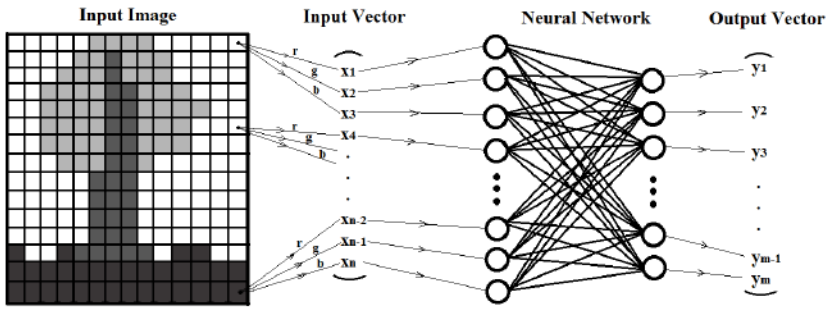

Figure 1. Image compressing

formally, the partitioning problem can be written as follows:

$$\begin{cases} S_i \cap S_j = \varnothing \\ \bigcup\limits_{i=\overline{1,n}} S_i = S \\ \max\limits_{i=\overline{1,n}} T_i \to \min \end{cases} \tag{4}$$

Note. In the problem, it is important to find the values of $|S_i|$ that satisfy (4).

Denote by P – the amount of computation required to process a single image from S. It is known that the processing time of input signals in forward propagation NN is constant.

To achieve the minimum processing time, two conditions must be met. The computing devices must work (a) simultaneously and (b) the same amount of time.

With this in mind, the task can be rewritten as:

$$\begin{cases} \frac{P|S_i|}{U_i} = \frac{P|S_n|}{U_n}, \forall i = \overline{1, n-1} \\ \sum\limits_{i=1}^{n} |S_i| = |S| \end{cases} \tag{5}$$

in the formula, the restriction on the coverage of all images of the package is omitted, because it is obvious. Fix the parameters U_i, $i = \overline{1,n}$, and solve the system (5):

$$\begin{cases} |S_n| = \frac{|S|}{\sum\limits_{i=1}^{n-1} \frac{U_i}{U_n} + 1} \\ |S_i| = \frac{U_i}{U_n} |S_n|, \forall i = \overline{1, n-1} \end{cases} \tag{6}$$

When calculating $|S_i|$, $\forall i = \overline{1,n}$ unknown in (6) remain the ratio of power. There are two ways to get them:

1) you can use the table values specified by the manufacturer. This is not suitable in the case of video cards. The same video card, depending on the specifics of the task, may have different power;
2) it is possible to experimentally calculate the power ratio of computing devices.

The following method for estimating power is proposed.

Let x be the subset of images needed to process the package S, and τ_i is the i-th device operation time on this subset. Then we get the system of equations:

$$\tau_i = \frac{P|x|}{U_i}, \forall i = \overline{1,n} \tag{7}$$

It is not difficult to notice that the time of the work of computing device at a fixed number of operations is inversely proportional to its power.

In order to obtain the power ratio, the last (n-th) equation we divide into $n-1$ previous ones.

$$\frac{U_i}{U_n} = \frac{\tau_n}{\tau_i}, \forall i = \overline{1, n-1} \tag{8}$$

The expression (8) means that the power ratio is inversely proportional to the ratio of time spent on processing a fixed number of operations.

The resulting expression accurately reflects the power ratio. It takes into account the actual conditions in which the calculations are made.

An approach to the efficient use of resources based on data parallelization technology is proposed. Formally, it can be described as the following algorithm.

Step 1. From the set of images S select (small in power) a subset necessary for processing the batch S.

Step 2. Process this subset (with time measurement for their execution) on each of the devices.

Step 3. According to the formula (8) determine the power ratio.

Step 4. By the formula (6) calculate the optimal load for each computing device. For this:

– the set of images S is divided into $|S|$ subsets of P_j, $j = \overline{1, |S|}$,
– for each i-th device $i = \overline{1,n}$, a subset of S_i is formed by join P_j, i.e. we have:

$$\begin{cases} S_i = \bigcup_{h=\overline{1,|S_i|}} P_{j_{ih}} \\ \bigcup\limits_{i=\overline{1,n}} S_i = S \\ S_i \cap S_j = \varnothing, \forall i \neq j \end{cases} \tag{9}$$

We assume that the devices are commensurate with each other in power. This condition is satisfied if:

$$\max_{i=\overline{1,n}} \cup_i \leq c \min_{j=\overline{1,n}} \cup_j \tag{10}$$

where c is a constant equal to two.

2) The package of images S must be large enough. In this case, it makes sense to parallelize their processing.

3) After calculating by formula (6) some values of the powers $|S_i|$, $i = \overline{1,n}$ may turn out to be fractional numbers. In this case, it is necessary to round them to a integer, taking into account the restriction (5). If $|S_i|$, $i = \overline{1,n}$ is substantially greater than n, then rounding will not practically slow down the computation time. Otherwise, it is necessary to round up to an integer (according to standard mathematical rules). In the case of "shortage", "free" images of the package S must be transferred to computing devices in descending order of their power. In the case of "busting" – remove one image from computing devices in order of increasing power.

The proposed approach has several advantages:

1) When solving applied problems, one can efficiently use the computing resources of a standard computer. They usually contain a multi-core processor and a video card.
2) High accuracy estimates of the optimal $|S_i|$, $i = \overline{1,n}$ in (6) is guaranteed.
3) In the process of solving the problem, no additional calculations are required, but simply to solve the system (6).

III. Experiments

The STL-10 data from the Stanford University repository [8] was used as the source data. The dataset contains one hundred thousand unmarked color images measuring 96x96 pixels. Each image is described by 27648 integers (in the range from 0 to 255) specifying the content of red, green, and blue colors [9]. Based on the obtained characteristics (the sample is set to about 2.8 * 109 numbers, contains descriptions of arbitrary, unmarked objects), we can conclude that the process of compressing the images of this sample with low losses is quite a challenge.

To illustrate the nature of the data used in the experiment, we give examples of some instances of images from the STL-10 dataset. (see Figure 2).

For data processing, a standard computer with an 8-core processor and a video card was used:

Video card nvidia – 1050 2gb; processor – amd ryzen 7 1700 3.0 GHz; RAM: 2x8 Gb 3200 MHz; Hard drive – samsung 850 pro 256 Gb; operation system – Lubuntu 16.04.3.

Computer graphics card is about 60% more powerful than the processor. Power of devices are commensurate with each other. Therefore, the configuration of the computing heterogeneous device corresponds to the conditions noted above.

The nvcc 7.5 compiler was used as software (libaries CUDA (Version 9.1 [10]) and OpenMP) with options:

```
nvcc -D_FORCE_INLINES -O2 –machine 64 -lgomp
-Xcompiler -fopenmp program_code.cu
```

Measurement of time of operations was carried out using the function "gettimeofday".

Note. The compiler option "–machine 64" shows that the application is 64 bit, therefore, it is not compatible with 32 bit systems.

When training a neural network, batch mode was used. The batch size was fixed and equal to 6561 images. The latter was due to the following considerations:

The size of the training dataset is one hundred thousand images. This number is not divisible by three, therefore, the degree of the triple can be chosen as the batch size to maximize the LCM and, therefore, as the repetition period of the batches. The batch size is three to the eighth degree, hence the repetition period of the batches is 656100000 images.

As part of the computational experiment, two different cases were considered:

– image batch processing is carried out only on a video card,
– simultaneously on the processor and video card (using the developed approach).

The general scheme of the experiments included the following main steps:

1) Read input data.
2) Preliminary image processing.
3) Initialization of the initial values of the weights of the neural network.
4) "Endless" learning cycle (each batch includes 6561 images).
    a) The learning process without a teacher.
    b) Measuring training time for 15 packets and checking the conditions for exit from the cycle.

The criterion for exiting the cycle: «the total root-mean-square error on 15 image packets increased». In the process of implementing the experiment (in accordance with the described approach), the following actions were performed:

– on the first iteration of the «infinite» cycle on the video card, the first half of the image batch is processed and the processing time τ_{gpu} is measured;
– the second half of the packet is processed on the processor and the time τ_{cpu} is measured;
– using formulas (8) and (6), the number of images that are fed to the processor and video card, respectively, is calculated;
– adjustment of neural network weights is carried out;
– the number of images of the batch, which is calculated on the first iteration, is processed on the

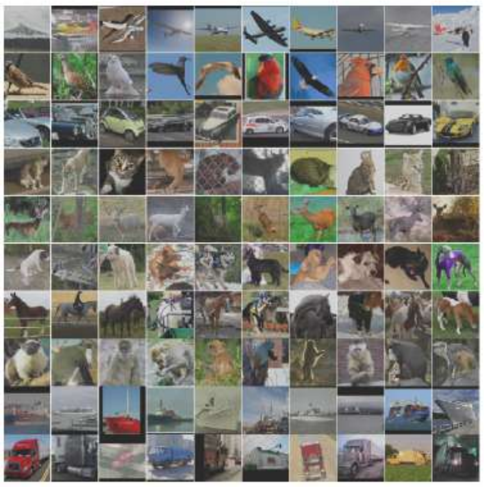
Figure 2. STL-10 pictures examples [9]

processor and video card simultanioulsy (in the loop);
- the neural network weights are adjusted again.

According to the results of the experiments, the following results were obtained (see table 1):

Table I
PROBLEM SOLVING TIME

Used computing devices	Videocard	Processorand video card
Timeof processing (in sec.)	24	15

It is easy to see that the use of the algorithm has reduced the processing time by 60%.

Thus, due to the rational loading of heterogeneous computing devices, it is possible to reduce the total processing time. This is important when solving applied problems in real-time conditions.

IV. CONCLUSION

The paper presents an algorithm for calculating the loading of heterogeneous devices, which reduced the processing time for image compression.

The efficiency of the algorithm is demonstrated on a computer with a multi-core processor and a video card.

The ideas described in the paper may be useful in processing large amounts of data on heterogeneous cluster calculators that are being actively developed at the present time.

REFERENCES

[1] Pavan Balaji Programming models for parallel computing (Scientific and engineering computation) / Pavan Balaji. – The MIT press, 2015 - 488p. ISBN – 978-0262528818

[2] Voevodin, V. V. Parallel computations [Text]: book. / V. V. Voevodin – Piter: BXV Saint Petersberg, 2004. – 608p. – ISBN 5-94157-160-7.

[3] Gregory R. Andrews Foundations of multithreaded, parallel and distributed programming [Text]: book – 688p. – ISBN 978-0201357523

[4] Cavaro-Menard C. Compression of biomedical images and signals / C. Cavaro-Menard – Wiley, 2013. –288p. – ISBN 978-1-84821-028-8

[5] Burger W. Principles of digital image processing / W. Burger. – Springer, 2013. – 369p. – ISBN 978-1-84882-919-0

[6] Marz N., Warren J. Big Data: Principles and Best Practices of Scalable Real-time Data Systems / N. Marz, J. Warren – Manning Publications, 2015 – 328p. – ISBN 978-1617290343

[7] Simon Haykin Neural Networks and Learining machines (third edition) / Simon Haykin. – Pearson Prentice Hall, 2009. – 936p. – ISBN 978-0-13-147139-9

[8] STL-10 dataset [electronic resource]. – link : academictorrents.com/details/a799a2845ac29a66c07cf74e2a2838b6c5698a6a – Access date: 25.02.2018.

[9] STL-10 dataset description [electronic resource]. – link : stanford.edu/ acoates//stl10/ – Access date: 24.02.2018.

[10] CUDA toolkit [electronic resource]: – link : developer.nvidia.com/cuda-downloads – Access date: 23.02.2018.

АЛГОРИТМ БЫСТРОГО СЖАТИЯ ИЗОБРАЖЕНИЙ НА ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВАХ

Мацкевич В.В., Краснопрошин В.В.

В работе рассмотрена проблема организации эффективной обработки данных на гетерогенных вычислительных устройствах. Предложен один из возможных подходов к решению проблемы с использованием технологии распараллеливания данных. Показано, что в общем случае проблема представляется нетривиальной математической задачей. Для одного из частных случаев предложен алгоритм решения. Эффективность подхода демонстрируется на примере решения задачи сжатия цветных изображений с использованием нейронной сети прямого распространения. Описанные в работе идеи могут оказаться полезными при обработке больших объемов данных на гетерогенных кластерных вычислителях.

Ключевые слова: Параллельные вычисления, вычислительные устройства, производительность системы, нейронная сеть, обучение, выборка.

Received 28.12.18