UDC 004.42:681.324
DOI: 10.15587/2706-5448.2021.239784
Article type «Original research»

**Yaroslav Sokolovskyy,
Denys Manokhin,
Yaroslav Kaplunsky,
Olha Mokrytska**

DEVELOPMENT OF SOFTWARE AND ALGORITHMS OF PARALLEL LEARNING OF ARTIFICIAL NEURAL NETWORKS USING CUDA TECHNOLOGIES

The object of research is to parallelize the learning process of artificial neural networks to automate the procedure of medical image analysis using the Python programming language, PyTorch framework and Compute Unified Device Architecture (CUDA) technology. The operation of this framework is based on the Define-by-Run model. The analysis of the available cloud technologies for realization of the task and the analysis of algorithms of learning of artificial neural networks is carried out. A modified U-Net architecture from the MedicalTorch library was used. The purpose of its application was the need for a network that can effectively learn with small data sets, as in the field of medicine one of the most problematic places is the availability of large datasets, due to the requirements for data confidentiality of this nature. The resulting information system is able to implement the tasks set before it, contains the most user-friendly interface and all the necessary tools to simplify and automate the process of visualization and analysis of data. The efficiency of neural network learning with the help of the central processor (CPU) and with the help of the graphic processor (GPU) with the use of CUDA technologies is compared. Cloud technology was used in the study. Google Colab and Microsoft Azure were considered among cloud services. Colab was first used to build a prototype. Therefore, the Azure service was used to effectively teach the finished architecture of the artificial neural network. Measurements were performed using cloud technologies in both services. The Adam optimizer was used to learn the model. CPU duration measurements were also measured to assess the acceleration of CUDA technology. An estimate of the acceleration obtained through the use of GPU computing and cloud technologies was implemented. CPU duration measurements were also measured to assess the acceleration of CUDA technology. The model developed during the research showed satisfactory results according to the metrics of Jaccard and Dyce in solving the problem. A key factor in the success of this study was cloud computing services.

Keywords: *software, artificial neural networks, Python, PyTorch framework, CUDA, modified U-Net architecture.*

Received date: 30.04.2021
Accepted date: 07.06.2021
Published date: 30.09.2021

How to cite
Sokolovskyy, Y., Manokhin, D., Kaplunsky, Y., Mokrytska, O. (2021). Development of software and algorithms of parallel learning of artificial neural networks using CUDA technologies. Technology Audit and Production Reserves, 5 (2 (61)), 21–25. doi: http://doi.org/10.15587/2706-5448.2021.239784

1. Introduction

Artificial intelligence (AI) methods are increasingly used to analyze source texts and understand their meaning, requirements management, specification development, design, code generation, verification, testing, quality assessment, reusability identification, problem solving on parallel systems, etc. medical systems, advise doctors in emergency situations, robotic manipulators to perform precise actions during surgical operations. However, due to the complexity of calculations and large amounts of data, it was possible to effectively learn artificial neural networks (ANNs) only with the help of computers with high power. The situation has changed only recently, in particular with the advent of Compute Unified Device Architecture (CUDA) technology. This made it possible to learn ANNs using the relatively cheap and powerful graphics processing units (GPUs) found in most personal computers today. And the development of frameworks based on the Python programming language has greatly simplified the development process. Together, this lowered the barriers to entry into the deep learning industry and ensured the popularity it has today.

2. The object of research and its technological audit

The object of research is the parallelization of the learning process of artificial neural networks to automate the procedure for analyzing medical images using the Python programming language, PyTorch framework and Compute Unified Device Architecture (CUDA) technology.

The subject of research is the learning of artificial neural networks.

One of the biggest bottlenecks is the use of the network, can learn effectively from small datasets. Since in the

ISSN 2664-9969

field of medicine, there are often problems with the presence of large datasets, due to data confidentiality requirements of this nature.

3. The aim and objectives of research

The aim of research is to develop software and algorithmic support for parallelizing the learning process of artificial neural networks using CUDA technologies, which will automate the procedure for analyzing medical images.

To achieve this aim, the following main objectives have been identified:

1. To analyze learning algorithms for artificial neural networks, their software.

2. To analyze the parameters of parallelization of learning algorithms

4. Research of existing solutions to the problem

In works [1, 2], the features of the software implementation of the learning procedure for artificial neural networks using a graphics processor were considered and an acceleration was obtained in comparison with learning using a central processor. Since the goal was to study the learning process itself, only classical datasets and simple ANN models were used. Therefore, as a continuation of this topic, it was decided to apply the knowledge gained to detect intracranial bleeding from images obtained using computed tomography (CT). Hemorrhage due to acute traumatic brain injury (TBI) is the third most common cause of death. And among young patients, even ahead of diseases of the cardiovascular system and oncology, taking first place. In addition, there is a high risk of losing the capacity of a person who has suffered intracranial hemorrhage [3]. Timely and accurate detection of hemorrhage is very important for successful treatment. Since the analysis of CT scans is done manually, it can take a lot of precious time to save the patient's life, and the correct diagnosis depends entirely on the qualifications of the doctor. Therefore, it is important to develop software for deciding on suspicious areas of the image. Computed tomography was invented by Sir Hounsfield, for which he was awarded the Nobel Prize in 1979. Together with him were awarded another scientist, Alan Cormack, who independently of Hounsfield mathematically described the theoretical prototype of the CT scanner [4].

The authors of [5] described the process of reconstruction of images obtained after CT scan for analysis by a radiologist. In particular, these images are obtained as different shades of gray, depending on the passage of the beams of rays through various tissues of the patient's body. The works [6, 7] describe the methods and tools necessary for artificial learning based on a neural network, and also covers the mathematical and conceptual prerequisites for their implementation. Works [8, 9] are devoted to the learning of neural networks, in particular in the field of medical image segmentation. Moreover, the authors of the study [9] proposed the U-Net network as specially developed for the tasks of segmentation of medical images.

Such programs can be based both on classical algorithms of computer vision and image processing, and on methods of machine learning. New ANN architectures have proven themselves very well in solving image segmentation problems. The problem is that ANN learning requires the selection of millions of parameters and processing of gigabytes of information, and therefore takes a very long time. CUDA technology allows using the GPU for computing, significantly reducing the time required for learning. However, the power of the GPU in the middle price segment is still not enough for solving modern problems. Therefore, cloud services are used.

Since it is easier to mark only the very presence and type of hemorrhage in a picture than to indicate its location, datasets for the task of classifying CT scans are more common. The authors of [10, 11] presented a dataset with a large number of images of hemorrhages, which makes it possible to solve the problems of classifying CT scans of the brain for their presence. But the question of researching this problem remains.

5. Methods of research

Having measurements for different angles of arrival of X-rays, the computer program can reconstruct the values of the absorption coefficients μ at each point of the scanned section. The results are presented as a grayscale image, with more absorption coefficients in light shades and less in dark shades. The coefficient μ shows how much the beam of rays will be weakened after passing through the tissues of the patient's body [5]. Before the reconstructed image is ready for analysis by a radiologist, the value of each pixel is converted to Hounsfield units. The Hounsfield unit (denoted HU from the Hounsfield Unit) is a linear transformation of the absorption coefficient μ:

$$HU = 1{,}000 \cdot \frac{\mu - \mu_W}{\mu_W - \mu_A}, \quad (1)$$

where μ_W – water absorption coefficient; μ_A – air absorption coefficient.

For the analysis of CT scans, a narrow range window is selected from the image, which covers only the part of the Hounsfield scale that interests us. The window is characterized by the position of the center on the Hounsfield scale (WL from window level) and width (WW from window width). The minimum and maximum values ($I_{\min}$ and $I_{\max}$, respectively) that fall into the given window can be found using the following formulas:

$$I_{\min} = WL - (WW \div 2), \quad (2)$$

$$I_{\max} = WL + (WW \div 2). \quad (3)$$

In the resulting image, all values less than $I_{\min}$ will be black, and those larger than $I_{\max}$ will be white (Fig. 1).

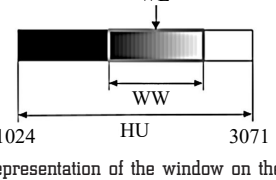

Fig. 1. Schematic representation of the window on the Hounsfield scale

Let's consider the algorithmic support of ANN learning using the example of one of the simplest ANN models – a multilayer perceptron [7].

Image segmentation using ANN. Image segmentation is one of the key tasks of computer vision. There are many segmentation algorithms with different levels of complexity. Most of them boil down to the fact that each pixel is attributed to the corresponding class based on certain characteristics common to this class, such as brightness, color or texture. However, often, the objects that interest us in the picture are not so easy to select, since they can be very heterogeneous. Each task requires an individual approach. This problem is especially acute in the segmentation of medical images. ANN can help solve these problems. Therefore, in the field of medical image segmentation, the following way of working is now common practice. Professional doctors for a large data set manually select the areas that need to be learned to segment automatically, and the developer's task is to learn the ANN using this data. Taking into account the effectiveness of the application of star neural networks (NN) to classification problems, this type of networks was also used for segmentation [8]. The U-Net, proposed in [9], is a fully convolutional network developed specifically for the tasks of segmentation of medical images.

The need for its creation lies in effective learning on small datasets, since in the field of medicine there are often problems with the presence of large datasets, in particular, due to data confidentiality requirements of this nature. The network consists of two symmetrical parts, which are also often called an encoder and a decoder. The encoder works like a classical neural network, where each layer decreases the resolution of the input image and increases the number of feature maps at the output. The decoder, on the other hand, restores the full-size image from the feature maps received at the input. To assess the segmentation accuracy, popular metrics are the Jaccard and Dice coefficients [9].

Since it is easier to mark only the very presence and type of hemorrhage in a picture than to indicate its location, datasets for the task of classifying CT scans are more common. So, a popular dataset for validating the classifier is CQ500, a set containing 491 CT scans of the head, for each of which the corresponding type of hemorrhage or its absence is indicated. For it, results were obtained with about 95 % recognition accuracy for each class [10]. A dataset with over 25,000 images has also been released [11]. As it is possible to see, the availability of a large amount of publicly available data has made it possible to significantly advance in solving the problem of classifying CT scans of the brain for the presence of one or another type of hemorrhage. And due to the lack of a standard dataset for segmentation, problems arise in the study of this problem. An open dataset was also created for learning and validation of intracranial hemorrhage segmentation models [12, 13].

The data for creating the dataset was obtained in [14]. If to consider individual layers, let's obtain 318 images of the existing bleeding and 2491 without. Images are presented in the NIfTI format, with an extension of 512×512 and a variable number of layers for each patient (usually about 30). In addition, each image has a corresponding mask, where, in the presence of hemorrhage, the corresponding pixels are highlighted in white. Also, a csv file is provided to the data for each layer of each image. In the aforementioned work [14], the authors also described the process of learning the U-Net network on this dataset and gave estimates of the accuracy of deriving the learned model in various metrics. For learning, separate layers were used, extracted from volumetric scans with a window (WL 40, WW 120), and both samples of patients with hemorrhage and without were used. Two experiments were carried out, both with full images and with images divided into 160x160 blocks. More attention was paid to the second method. The workout was conducted over 150 eras using an Nvidia GeForce RTX2080 GPU with 11 GB of memory. This was spent about 5:00. The Adam optimizer with a learning factor of 10–5 was used to learn the model. Cross entropy was used as a loss function. The workout was done with a portion size of 32.

6. Research results

Two experiments were performed, both with full images and with images divided into 160×160 blocks. More attention was paid to the second method. The learning lasted for 150 epochs using the Nvidia GeForce RTX2080 GPU with 11 GB of memory. It took about 5 hours. The Adam optimizer with a learning factor of 10-5 was used to learn the model. Cross entropy was used as a function of losses. The learning took place with a portion size of 32. The test results of the learned network are given in Table 1.

Table 1

Results of testing a learned artificial neural network

Level	Jaccard coefficient	Dice coefficient	Sensitivity (%)	Specificity (%)
Min	0.00	0.00	50	0
Max	0.528	0.677	100	100
STD	0.163	0.211	9.9	29.9
Average	0.218	0.315	97.28	50.4

Using the results of [11], the NIfTI format images were divided into separate layers, to which a window was applied (WL 40, WW: 120). The resulting images were saved in 8-bit PNG format in two directories, scans – in image, and the corresponding masks with the same file name – in label. The division into learning and test samples will be carried out immediately before learning.

Since, in addition to developing a model capable of segmenting intracranial hemorrhages, the purpose of this work is also to study the acceleration of the learning process using CUDA technology, it was decided to simplify the first task. Therefore, the problem of binary segmentation was considered, where each pixel must be associated with a label of one class or another. Moreover, only images where hemorrhage is definitely present will be considered. There are only 318 of those, as previously mentioned, in the dataset.

CUDA technology can significantly speed up the learning process using the PyTorch framework. (Python) for working with ANN. It is based on the Define-by-Run model [15], the main idea of which is that the computation graph is dynamically built right under the learning duration. During the forward pass, each function, except for calculating a value, saves the history of the calculations along with a link to the previous node in the graph. Since the structure of the graph depends on the path

ISSN 2664-9969

of the program execution, it became possible to use the syntactic constructions of the programming language, such as conditional statements and loops. In addition, now it is possible to use the debugger for a more detailed study of the learning process [16].

To implement the ANN architecture for working with medical data and some auxiliary functions for their learning [17], the MedicalTorch library was used. Jupyter Notebook was chosen as the programming environment, because the interactive mode of program execution, it offers convenient for research. In addition, it allows to conveniently visualize data, and the ipynb files that are created in it are supported for execution by many cloud services. Among the cloud services, Google Colab and Microsoft Azure were considered. First, Colab was used to build a prototype. Azure was used to effectively learn the finished ANN architecture. But for comparison, measurements were taken on two cloud services.

Features of software implementation using the PyTorch framework. To learn the ANN, the following components are needed [18]:

– Model – the ANN itself, represented by a class object, follows torch.nn.Module and implements the forward method for traversing the network;
– Dataset – a class object, followed by torch.utils.data. Dataset and implements methods for getting the number of elements and an arbitrary element of the dataset;
– DataLoader – an object used to efficiently load dataset objects;
– Optimizer – an object used for the learning itself, that is, updating the network parameters to minimize the loss function.

First, the ICHDataset class was created. The constructor for this class takes a path to a directory containing two subdirectories, image and label, and an optional transform parameter. Each element of the dataset is a dictionary, where the «input» key corresponds to the image itself, and the «gt» key is a properly segmented mask. This data format requires converting images from the MedicalTorch library. After that, the corresponding object was created, the dataset was divided into learning and test samples in such a way that 80 % of randomly selected images were selected for learning, and the rest for testing. Corresponding data loaders were also created.

A newer modification of the U-Net architecture from the MedicalTorch library was used as a network. In it, before the convolutions, frames are added to the input data in order to get an image of the same size at the output as at the input. Also, bilinear interpolation is performed instead of inverse convolution.

In addition to the model, the Adam optimizer is also created, and the Dice coefficient with a negative sign is used as the loss function. One of the main parts is also the learning algorithm. PyTorch requires hand-writing this function. On the one hand, this often forces to write a lot of the same code, but on the other hand, it provides flexibility, because different tasks may require a different learning algorithm.

It is also possible to upload an existing laptop to the Google Drive cloud storage, and then simply open it as a regular file, by default it will be loaded into Colab. If it is necessary a GPU for calculations, it is necessary to go to Runtime->Change runtime type and change the current runtime from CPU to GPU. The GPU is automatically selected by the load balancer [19]. At the time of the research, an Nvidia Tesla T4 with 16 GB of video memory was used.

Unlike Colab, Azure can be tricky to set up as it is a professional service with many options available. A schematic of the infrastructure for providing a solution to the problems of machine learning Microsoft Azure can be seen in [19]. The centerpiece of this system is the work area, which integrates and effectively manages all other components.

For experiments in this work, a new workspace course-project-ws was created, and a virtual machine was added to it. For this, a compute instance of the Standard_NC6 type was created (Table 2).

Table 2

Characteristics of the Standard_NC6 virtual machine

Number of CPU cores	6
RAM	56 Gb
SSD storage	340 Gb
GPU	Nvidia Tesla K80 12 Gb

Since there is not much data, it was loaded directly into the storage of the virtual machine. An ipynb file was also uploaded, and the ability to edit jupyter files of the Microsoft Azure Machine Learning environment was used. Therefore, all research was carried out exclusively using cloud services. As for the ANN itself, the learning factor was 0.001, and the chunk size was 4. The duration of the learning of such a network was measured in both cloud services. And also for it, it was analyzed how well it coped with the assigned applied task.

7. SWOT analysis of research results

Strengths. The strengths of the results of these studies are that the studied features of the software implementation of the ANN learning procedure using GPU and cloud services and the advantages and disadvantages of this approach are assessed.

Weaknesses. It seems expedient to analyze the results of indicators of the parallelization of ANN learning for large datasets.

Opportunities. The developed software and algorithmic support for parallelizing the learning process of ANN using CUDA technologies and cloud services allows the automation of the procedure for analyzing medical data. Such a series of studies can be useful for improving the quality and organization of data processing processes.

Among the possibilities for further development of the topic under consideration, there are two main ways of research. The first is to improve the model to improve the segmentation accuracy. For example, a variant with a combination of several ANNs is possible. The second is the lower-level work with CUDA technologies. For example, optimization of the execution time of some operations by creating your own CUDA kernels.

Threats. The research results proposed in the work contain only theoretical character. But they can be used for analysis and further application in practical implementation.

8. Conclusions

1. The paper analyzes the available technologies and algorithms for parallelization by teaching artificial neural networks. The search for input data was carried out, their analysis was carried out, and the functionality of their reading and parsing was implemented. The necessary software was developed using the Python programming language, PyTorch framework and Compute Unified Device Architecture (CUDA) technology. The framework is based on the Define-by-Run model. A modified U-Net architecture from the MedicalTorch library was used. Among the cloud services, Google Colab and Microsoft Azure were used.

2. The developed model has shown satisfactory results in terms of the Jaccard and Dice metrics in solving the problem. Cloud computing services were a key factor in the success of this study. Since, as has been shown, even with the use of CUDA technology, the capacity of a medium-price PC was not enough to learn the selected ANN.

There was also a comparatively long learning session with and without parallelization using CUDA technology. Acceleration was achieved despite being used from the dataset.

References

1. Sokolovskyy, Y. I., Shymanskyi, V. M., Mokrytska, O. V., Kharko, Y. V. (2019). Neural network model for identification of material creep curves using CUDA technologies. *Ukrainian Journal of Information Technology, 1 (1),* 11–16. doi: http://doi.org/10.23939/ujit2019.01.011
2. Manokhin, D. (2021) Prohramno alhorytmichne zabezpechennia rozparalelennia protsesu navchannia shtuchnykh neironnykh merezh z vykorystanniam tekhnolohii CUDA. *Mizhnarodna studentska naukova konferentsiia z pytan prykladnoi matematyky ta kompiuternykh nauk (MSNKPMK-2021).* Lviv. Available at: https://ami.lnu.edu.ua/wp-content/uploads/2021/05/Ministerstvo-osvity-i-nauky-Ukrainy.docx
3. Ambros, R., Waltham, R. et. al. (2021). *Godfrey Hounsfield.* Available at: https://radiopaedia.org/articles/godfrey-hounsfield?lang=us
4. Bell, D. J., Mirjan, Pr., Nadrljanski, M. et. al. (2021). *Computed tomography.* Available at: https://radiopaedia.org/articles/computed-tomography
5. Bell, D. J., Greenway, K. et. al. (2021). *Hounsfield unit.* Available at: https://radiopaedia.org/articles/hounsfield-unit
6. Goodfellow, I., Bengio, Yo., Courville, A. (2016). *Deep Learning.* MIT Press, 781.
7. Ciresan, D. C., Gambardella, L. M., Giusti, A. (2012). *Deep neural networks segment neuronal membranes in electron microscopy images.* NIPS, 2852–2860.
8. Ronnenbergerm, O., Fischerm, P., Broxm, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention (MICCAI).* Springer, LNCS, 9351, 234–241. doi: http://doi.org/10.1007/978-3-319-24574-4_28
9. Chilamkurthy, S., Ghosh, R., Tanamala, S., Biviji, M., Campeau, N. G., Venugopal, V. K., Warier, P. (2018). *Development and validation of deep learning algorithms for detection of critical findings in head CT scans.* arXiv preprint. Available at: https://arxiv.org/abs/1803.05854
10. RSNA Intracranial Hemorrhage Detection (2019). *Radiological Society of North Ameriaca.* Available at: https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/overview
11. Hssayeni, M. D., Croock, M. S., Salman, A. D., Al-khafaji Hassan Falah, Yahya, Z. A., Ghoraani, B. (2020). Intracranial Hemorrhage Segmentation Using a Deep Convolutional Model. *Data, 5 (1),* 14. doi: http://doi.org/10.3390/data5010014
12. Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G. et. al. (2000). PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation, 101 (23),* E215–E220. doi: http://doi.org/10.1161/01.cir.101.23.e215
13. Perone, C. S., Cclauss, Saravia, E., Ballester, P. L., Tare, M. (2018). *Perone/medicaltorch: Release v0.2 (v0.2).* doi: https://doi.org/10.5281/zenodo.1495335
14. Hssayeni, M. (2020). *Computed Tomography Images for Intracranial Hemorrhage Detection and Segmentation. PhysioNet.* 1.3.1. doi: https://doi.org/10.13026/4nae-zg36
15. Tokui, S., Oono, K. (2015). *Chainer: a Next-Generation Open Source Framework for Deep Learning.* Available at: http://learningsys.org/papers/LearningSys_2015_paper_33.pdf
16. Perone, C. S., Cclauss, Saravia, E., Ballester, P. L., Tare, M. (2018). *Perone/medicaltorch: Release v0.2 (v0.2).* doi: https://doi.org/10.5281/zenodo.1495335
17. *PyTorch Documentation* (2021) Available at: https://pytorch.org/docs/stable/index.html
18. *Colaboratory Frequently Asked Questions* (2021). Available at: https://research.google.com/colaboratory/faq.html
19. *How Azure Machine Learning works: Architecture and concepts* (2020). Available at: https://docs.microsoft.com/en-us/azure/machine-learning/concept-azure-machine-learning-architecture

✉ ***Yaroslav Sokolovskyy,*** *Doctor of Technical Sciences, Professor, Department of Computer-Aided Design, Lviv Polytechnic National University, Lviv, Ukraine, e-mail: sokolowskyyyar@yahoo.com, ORCID: https://orcid.org/0000-0003-4866-2575*

Denys Manokhin, *Department of Information Systems, Ivan Franko National University of Lviv, Lviv, Ukraine, ORCID: https://orcid.org/0000-0002-8590-7626*

Yaroslav Kaplunsky, *Postgraduate Student, Department of Information Technology, Ukrainian National Forestry University, Lviv, Ukraine, ORCID: https://orcid.org/0000-0002-7550-8357*

Olha Mokrytska, *PhD, Associate Professor, Department of Information Technology, Ukrainian National Forestry University, Lviv, Ukraine, ORCID: https://orcid.org/0000-0002-2887-9585*

✉ *Corresponding author*