UDC 004.8:519.816
DOI: 10.20535/SRIT.2308-8893.2026.1.11

PHYSICAL-INFORMED NEURAL NETWORK IN SIGNAL PROCESSING AND NETWORK TRAFFIC COMMUNICATIONS

O.V. ZOLOTUKHIN, M.S. KUDRYAVTSEVA, Y.V. BODYANSKIY, V.O. FILATOV, A.V. ANTILIKATOROV, D.V. KALININ

Abstract. Problem of signal processing and network traffic optimization is solved at the hardware level and is interesting, modern and relevant from the point of view of the application level. It is necessary to propose an approach that combines machine learning methods with network bandwidth tasks and traffic over the network. To solve this problem, it is proposed to use concept of Informed Machine Learning (IML), that is the Taxonomy of IML, the principles of constructing deep machine learning systems based on information about the physical properties of the data transmission network under study. The platform for developing is deep machine learning models using PINN neural networks. The PINN represents the class of deep learning algorithms that can integrate data with or without physical processes description. As an algorithm it is proposed to use the popular algorithm in the field of deep learning – ADAM (Adaptive Moment Estimation) for optimizing network traffic. Using the PINN trained with the ADAM algorithm to transmit data the efficiency has increased. Thanks to this method, it is possible to obtain a low noise signal in practice, due to which network traffic is optimized.

Keywords: Adaptive Moment Estimation, artificial intelligence, artificial neural network, Data Visualization, dynamic neuron, gradient decent, Informed Machine Learning, learning algorithm, network bandwidth utilization, network traffic optimization, neuro-fuzzy logic, Python.

INTRODUCTION

The main task of optimizing information networks is to maintain the required level of performance characteristics and network capacity under conditions of load changes.

To assess the quality of any network, characteristics such as data transfer speed, reliability of transmitted information, bandwidth and reliability of the communication channel are used.

The main characteristic of information transmission channels is their bandwidth, that is the amount of data that is going threw a network with a given speed depending on the channel capacity.

Currently, the load on the data transmission channel can vary by an order of magnitude depending on the time of day, and the nature of the transmitted data is determined by the user's field of activity (for example, large videos, project files, high quality images), therefore, optimization problems must be solved in real time.

First of all, it is necessary to formulate criteria for the effectiveness of the information network. Most often, these criteria are performance and reliability, which in turn require the selection of specific evaluation indicators. For example, the performance of an information network is determined by the response time (the time interval between the occurrence of a user request for any network service and

the receipt of a response to this request), and the reliability of the network is determined by the readiness ratio of the network equipment to perform it's functions at an arbitrary point in time.

In addition, it is necessary to determine many variable network parameters that directly or indirectly affect the performance criteria. Settings can be devices, protocols, or transmission technologies.

Thus, to optimize data transmission over a network, it is advisable to consider three successive stages of network management.

1. Bringing the network into working condition, which usually includes:
- searching for faulty network elements at the physical or data link levels;
- checking the compatibility of equipment and software (at the network level);
- selection of correct values for key parameters of programs and devices that ensure data transfer between network nodes — addresses of networks and nodes, protocols used, types of frames, packets (at the transport level).

In this case, optimization comes down to diagnosing faults and bringing the network into an operational state.

2. Primary configuration — selection of parameters that significantly affect the characteristics of the network. If the network is efficient, but communication is very slow due to unacceptable latency or communication sessions are frequently interrupted, then you need to look for the key factors that degrade the network. Typically, the reasons for a noticeable decrease in network performance or unstable network operation are found in an incorrectly operating element or an incorrectly set parameter, but due to the large number of such parameters, solving this problem may require long-term monitoring of network operation, collecting statistics, and searching through options. At this stage, a certain threshold value of the efficiency indicator is also set and it is required to find a network variant for which this value would be no worse than the threshold.

3. The final configuration of network parameters is directly optimizing the network operation. In the case of a normally operating network, further improving it's quality, as a rule, requires finding some optimal combination of values for a large number of parameters.

During the final network setup, in which the parameters of the operating network, for example, the frame size or the size of the window of unacknowledged packets, can be varied in order to improve performance (for example, the average response time) by at least a few percent.

As a rule, network optimization is understood as some kind of compromise approach. It is necessary to select values of network parameters such that it's efficiency indicators are at least not lower than the maximum permissible values specified when choosing the global Quality of Service level.

In real conditions, it is enough to find a solution close to the optimal one, i.e. it is necessary to find some rational version of the structure and parameters of the network.

MATERIALS AND METHODS

The business of geographically distributed companies with an extensive branch network largely depends on the speed and stability of information exchange between departments. The WANs (Wide Area Networks) used for this usually do not always meet the requirements for the productive operation of such critical

applications as VoIP, video conferencing, Enterprise Resource Planning systems and other software.

WAN optimization solutions help companies increase data transfer speeds without expanding the channel, and network administrators can improve application performance, avoid data transfer delays and packet losses, and ensure guaranteed service quality.

At present, when companies are actively centralizing IT infrastructures to improve security and reduce management costs, and end users are increasingly working in mobile office mode, the most pressing task for network administrators is increasing data transfer speeds.

The modern network architecture is presented on Fig. 1. The distributed network architecture is presented in the form of a chain, which consists of: the user's local network, the user's provider, the global Internet, the recipient's provider and the recipient's local network.

Based on this complex hierarchical architecture, the problem of packet data transmission becomes clear — the amount of data transmitted exceeds the amount of data received due to network noise.

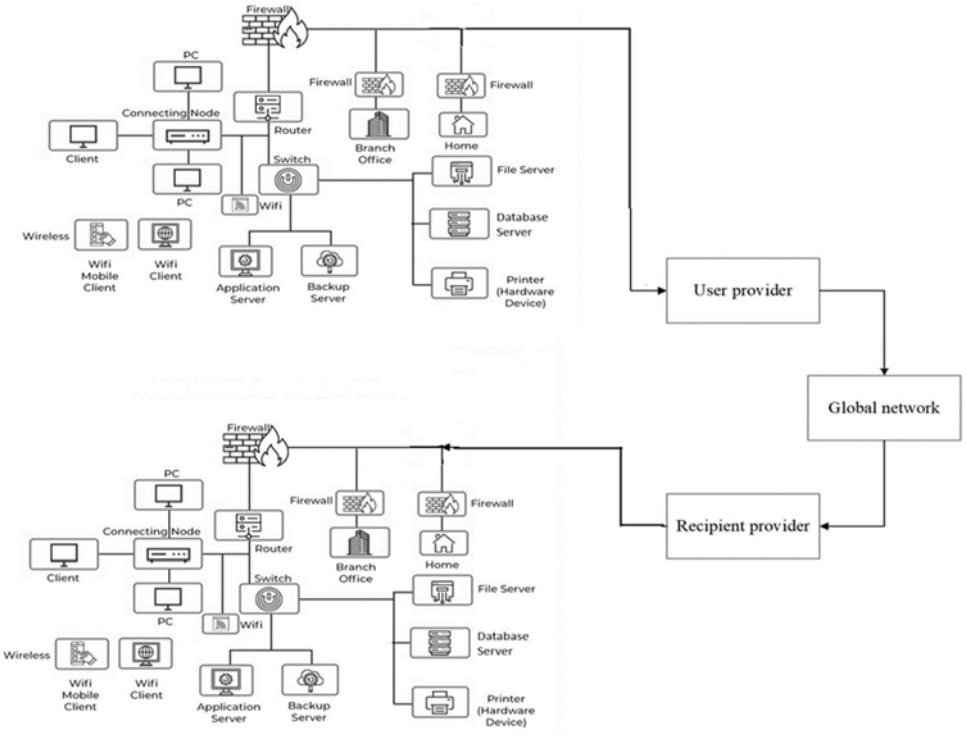

Fig. 1. The distributed network architecture

So, today problem of signal processing and network traffic optimization is solved at the hardware (or protocol level) and is interesting, modern and relevant from the point of view of the application level. Representation of the protocol and user data transfer levels you can see on the Fig. 2.

It is necessary to propose an approach that combines machine learning methods, for example, neural networks with network bandwidth tasks and traffic over the network.

This will allow us to present not only a theoretical, but also a practical solution to the problem.

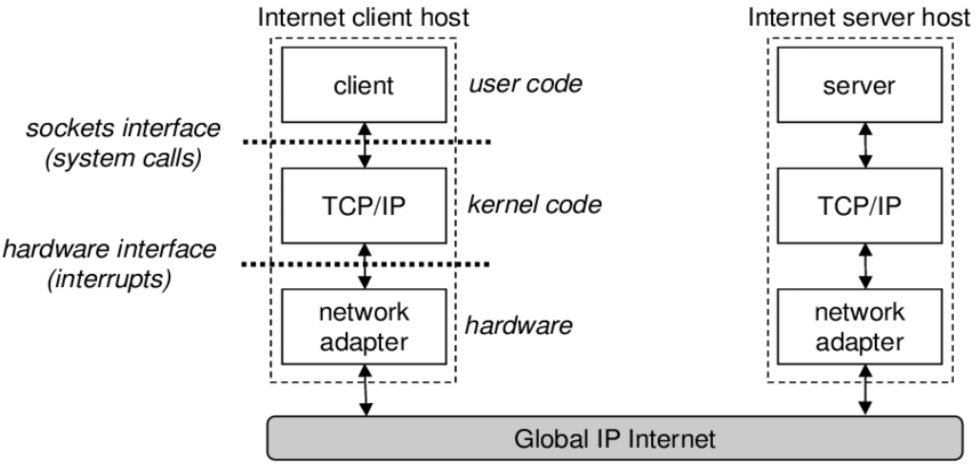

Fig. 2. Representation of the protocol and user data transfer levels

One way to optimize of data traffic at the hardware level is solved by configuring protocol parameters for packet transmission: packet size and number of packets. The packet transmission rate is determined by the speed of light and depends on the number of nodes from point A to point B of network.

For example, we established a connection with a remote computer, sent 16 packets, received confirmation that 12 packets were received, and in some packets the checksum does not match. We transmit the following packets, to them we re-add packets in which the correct checksum was not received. This is already an optimization compared to the fact that it is possible to transmit data in one packet and wait for confirmation each time, and in case of incorrect transmission of the packet, resend it.

To solve this problem in [1, 2] works it is proposed to use concept of Informed Machine Learning (IML). IML describes learning from a hybrid information source that consists of data and prior knowledge. The prior knowledge comes from an independent source, is given by formal representations, and is explicitly integrated into the machine learning pipeline.

Physics-informed machine learning integrates seamlessly data and mathematical physics models, even in partially understood, uncertain and high-dimensional contexts [1].

Taxonomy of Informed Machine Learning is represented on Fig. 3. This taxonomy serves as a classification framework for informed machine learning and structures approaches according to the three above analysis questions about the knowledge source, knowledge representation and knowledge integration. Authors identified for each dimension a set of elements that represent a spectrum of different approaches [2].

This paper proposes to use the Taxonomy of Informed Machine Learning, the principles of constructing deep machine learning systems based on information about the physical properties of the information data transmission network under study. The platform for developing intelligent tools is deep machine learning models using PINN neural networks.

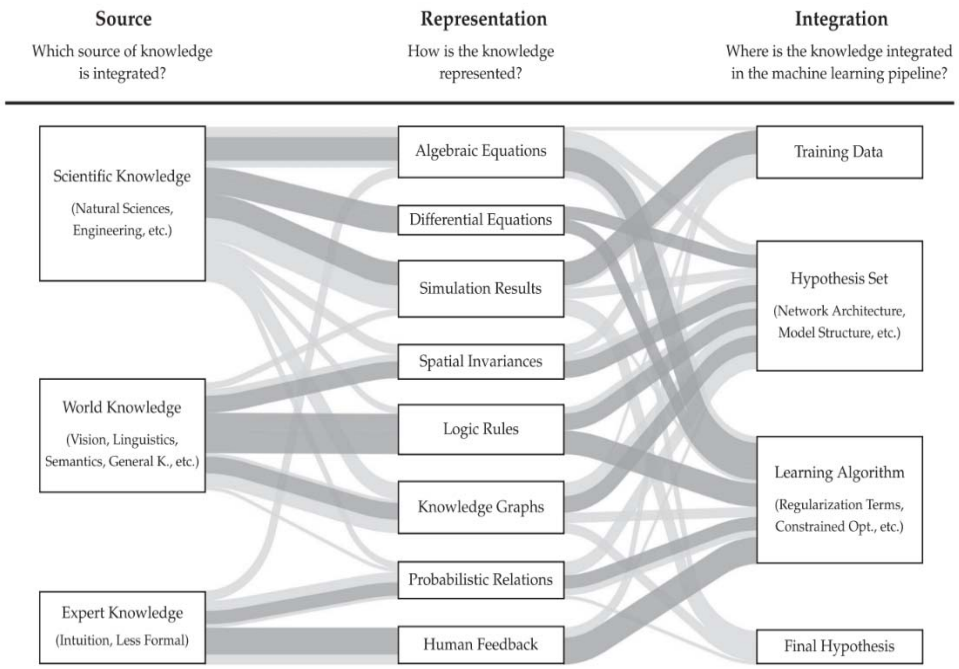

Fig. 3. Taxonomy of Informed Machine Learning

ARCHITECTURE OF PHYSICS-INFORMED NEURAL NETWORK

Thus, it is necessary to explore the gray box model in state space. A gray box represents a model that learns from data, guided by information about the applied physical properties or laws. Such models can be further used for adaptive control and self-organization. Peculiarity PINN is to initially take into account the underlying description of the physical interpretation of partial or ordinary differential equations, that is, the physics of the problem, rather than trying to derive a solution based solely on the data, that is, by approximating a set of state-value pairs with a neural network. Therefore, the use of state space models is considered.

So, the PINNs represent the class of deep learning algorithms that can seamlessly integrate data and abstract mathematical operators, including partial differential equations (PDE) with or without missing physics.

According to [1, 2] to design the PINN network for signal processing and network traffing optimisation it is necessary to implement the chain: Scientific Knowledge-Algebraic or Differential Equations-Learning algorithms (Fig. 4).

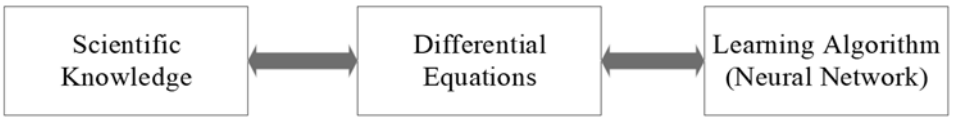

Fig. 4. Stages of the approach based on Physics-Informed Neural Networks

Let’s explore this method from the point of view of using a simple neural network architecture.

A regular Neural Network approximates various functions well (Fig. 5).

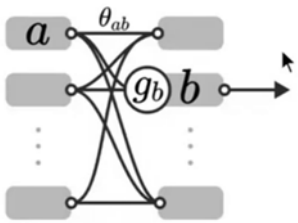

Fig. 5. A regular Neural Network

According to the Tsybenko theorem [3], an artificial neural network of feed-forward with one hidden layer can approximate any continious functionof many variables using the input vector and the weights of each vector with any accuracy, i.e.

$$b = g_b(\sum_a \theta_{ab} \cdot a)$$

Neural networks provide access to the computational graph (Fig. 6), that is access to all elementary operations on numbers that enter or exit the neural network [4].

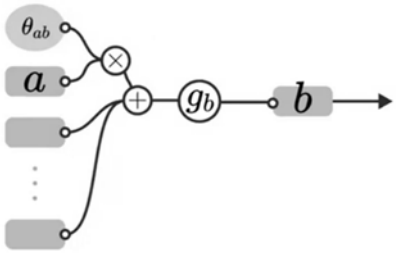

Fig. 6. The computational graph

The graph gives the property of auto differentiation. This allows to find the derivative of a neural network based on its parameters. This is used to update the parameters using the backpropagation algorithm (Fig. 7) through the derivative of the error and update the weights Θ. This is how a neural network is trained.

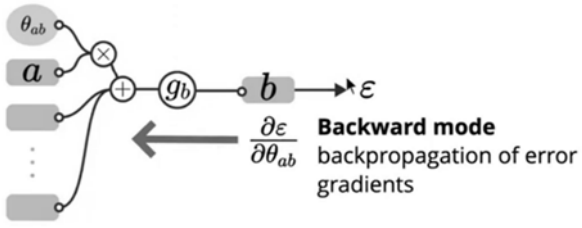

Fig. 7. The computational graph with the backpropagation algorithm

With the help of the computational graph, it is possible to differentiate the neural network with respect to intermediate results of the calculation of any input

parameter a, $\frac{dF}{da}$, or with respect to the input parameters X, for example, with respect to some physical parameter X [5].

You can write an equation in the form of partial derivatives $PDE\ (F, \frac{dF}{dx}, \ldots)$(Fig. 8), which can be used in physics (these can be coordinates or time of a process or other physical parameters).

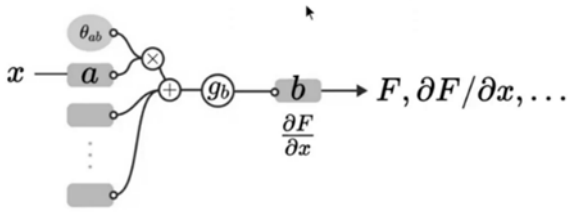

Fig. 8. The computational graph with partial derivatives

Let's consider the learning methods of a neural network. There are three categories of methods to train a neural network:

- observational bias involves supervisor training, submiting a lot of data on the physical parameters of the system;
- inductive bias involves constructing a neural network so that it complies to physical laws, for example using a convolutional neural network;
- learning bias involves to reduce the error through the derivatives of the parameters.

Let's consider the Physics-Informed Neural Network architecture for network traffic optimization.

The PINN for network traffic optimization represents a function of variables: the time of data transfer, the volume of data transfer in packets, the number of nodes (Fig. 9).

So, to solve the problem of digital signal processing and network traffic optimization, you can use dynamic neurons that are described by differential or difference equations. The behavior of such dynamic neurons is significantly determined by their prehistory.

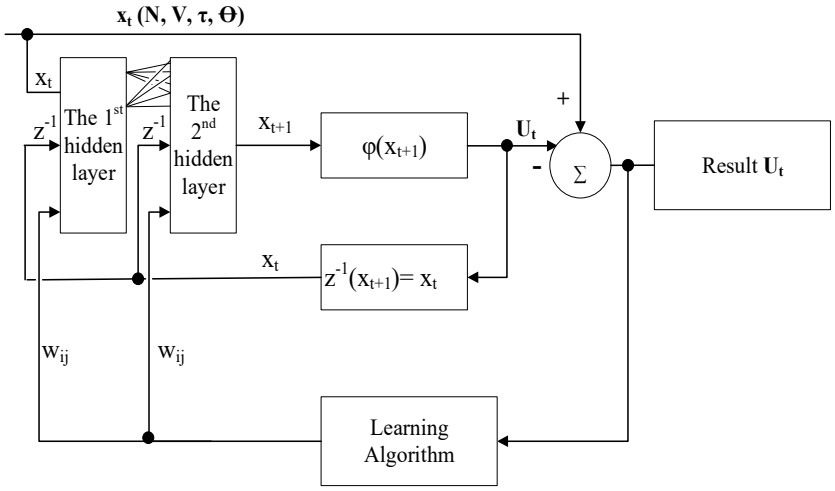

Fig. 9. The PINN neural network for network traffic optimization

The input parameters of the neural network are: the time — τ; the amount of transmitted data in packets – V; the number of nodes — N; the output parameter is the network bandwidth utilization – U. The time includes: time of protocol connection and confirmation of two devices, $\tau\ con$; packet transferring time, $\tau\ pass$; processing time, τ proc: $\tau = 2* \tau_{con} + \tau_{pass} + \tau_{proc}$.

What is network bandwidth utilization? In the simplest terms, network bandwidth utilization is the rate at which data can flow through the network. Traditionally measured in Mbits per second (Mbps), higher bandwidth allows more traffic flow from one device to another. Utilization is the percentage of a network's bandwidth that is currently being consumed by network traffic.

Let's consider neural network. Neural Network has two hidden layers (determined experimentally). The Input vector of neural network is

$$x_t\ (N,\ V,\ \tau,\ \Theta)$$

where $\tau = 0,1..n$; $\Theta = \{w_{ij}\}$, $i,j \in [1..k]$; t – timestep (discrete moments in time); the output vector of neural network is Ut.

The neuron of the neural network is a model of a nonlinear dynamic system in state space, that is, the time factor influences the behavior of the neuron, and the output signal U_t is determined by the input signals x_{t+1} and depends on the past states of the system x_t.

In this neural network, a delay element is included in the feedback circuit, which implements the backward shift operation $z^{-1}(x_{t+1}) = x_t$ and provides the neuron with the necessary dynamic properties over time. This can be seen in the inner circuit of the neural network. Thereby, the neural network allows to calculate the data network utilization parameter dynamically, at times t, $t+1$, and so on.

A dynamic neuron is described by the recurrent equation

$$x_{t+1} = \varphi(\sum w_{ij} * x_t + Q_j)$$

where $t = 0.1..$; $j = 1,2..n$; $i = 1,2..n$.

We will update the parameters of the neural network with learning algorithm until the error does not satisfy the result. If the error is satisfactory, then we will assume that we have found a solution to the system, that is, we have determined the network bandwidth utilization.

As a training algorithm, consider the classic gradient descent, which is an optimization algorithm used to minimize errors in a machine learning model.

Despite the fact that the gradient method of training a neural network and its modified version in the form of backpropagation are the most common algorithms in machine learning, the task of optimizing neural networks remains a rather difficult problem today.

The gradient descent method has a number of problems for training deep neural networks [6–8]:

- the gradient method gets stuck in a fairly deep local minimum (Fig. 10). There are solutions to sometimes work around such problems, such as momentum, which can carry optimizers through large lifts, or batch normalization, which smooths out the error space. But the root cause of many branching problems in neural networks is still a local minimum;

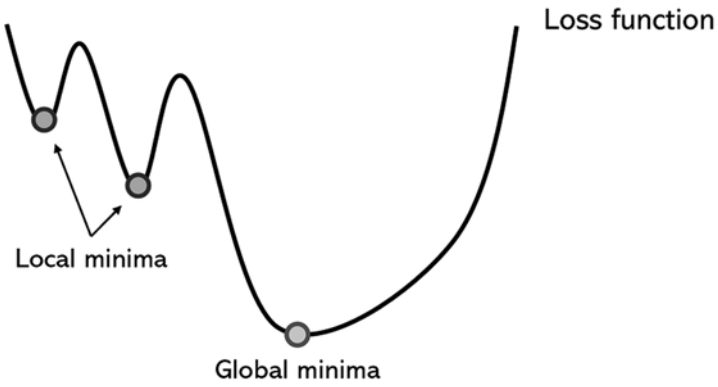

Fig. 10. Using the gradient descent method in the problem of searching for a global extremum

- high time spent on training. Gradient descent, due to its low convergence rate, is typically a time-consuming method, even with adaptation options for large data sets such as batch gradient descent.
- the gradient method is highly sensitive to optimizer initialization. For example, performance may be significantly better if the optimizer is initialized near the second local minimum rather than the first, but this is all arbitrary.
- the pace of learning determines the degree of confidence and riskiness of the optimizer. Setting the learning rate too high may result in missing the global minimum, while setting the learning rate too low will increase execution time. To solve this problem, the learning rate is reduced gradually, but choosing the rate of decrease, taking into account many other variables that determine it, is quite difficult.
- gradient descent requires gradients. This means that it is vulnerable to inherent problems like damping or exploding gradients, in addition to its inability to handle non-differentiable functions.

Based on the analysis of gradient descent problems, it is proposed to use a complex optimizer to train a neural network.

An optimizer is an algorithm for achieving the best, most accurate results while increasing the learning rate. In other words, it is an algorithm used to change parameters such as weights of neural network and learning rates slightly so that the model is adequate and can produce accurate results quickly. To do this, the optimizer algorithm adjusts the neural network connection weights [9,10].

We proposed to use the popular optimization algorithm in the field of deep learning – optimizer ADAM. The name is derived from Adaptive Moment Estimation.

The algorithm has the following advantages compared to other optimizers:

- the algorithm works effectively with online and streaming data, works well with non-stationary data, for example, it allows to optimize noise [11,12];
- the algorithm allows to very accurately approximate a set of points with a linear (Fig. 11) or nonlinear function (Fig. 12);
- the algorithm has a simple implementation and computational efficiency, provides good accuracy compared to other optimizers, and is not demanding on computer resources [13,14];

- an important advantage of ADAM is that updating the w_t parameter is completely invariant to gradient scaling, the algorithm converges even if the objective function changes over the time or the objective function is nonlinear and contains several local extrema, which is relevant for the data transmission over a network (Fig. 13).

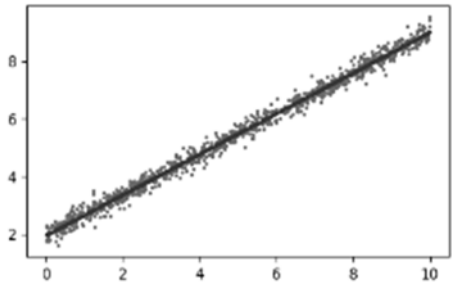

Fig. 11. Linear approximation function

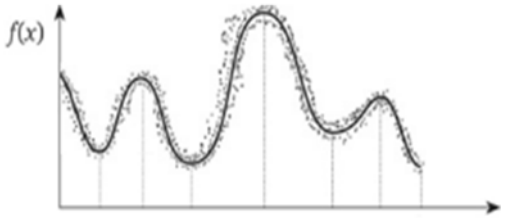

Fig. 12. Nonlinear approximation function

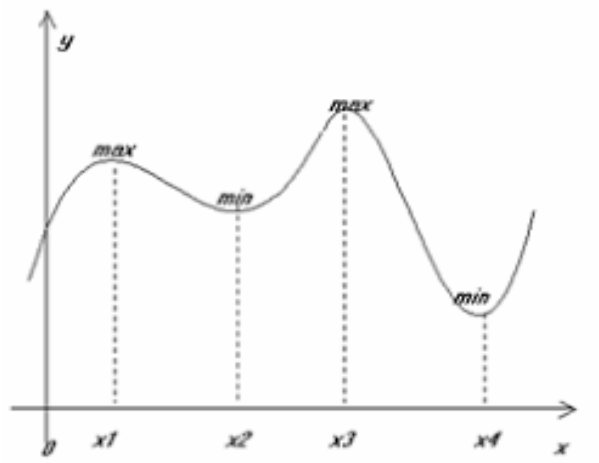

Fig. 13. Function with several local extrema

Let's consider the ADAM algorithm for optimizing network traffic.

The optimizer is called Adaptive Moment Estimation because it uses estimation of the first moment of the gradient, m_1 (the mean), and estimation of the second moment of the gradient, m_2 (the uncentered variance) to adapt the learning rate for each weight of the neural network.

The algorithm calculates the exponential moving average of the gradient p_t and the squared gradient q_t, and the parameters m_1 and m_2 control the decay rate of these moving averages.

The initial value of the moving averages p_t, $_{qt}$ and values of m_1 and m_2 close to 1 (which are recommended for this algorithm, since the algorithm is stable to the initial values of the learning rate and damping coefficient), you can see this on command initialize.

This leads to a shift in the moment estimates $\widehat{p_t}\widehat{q_t}$, towards zero. This displacement is overcome by correcting the 1st and 2nd order moments. Next, the weight parameter w_t is updated.

The algorithmic model can be presented in the following form.

Initialize t=0, m_1=0.9, m_2=0.999, ϵ=10-8, α=0.001

Step 1: while w_t do not converges

do{

t=t+1

Step 2: Calculate gradient $g_t = \frac{\partial f(x,w)}{\partial w}$

Step 3: Calculate $p_t = m_1 \cdot p_{t-1} + (1 - m_1) \cdot g_t$

Step 4: Calculate $q_t = m_2 \cdot q_{t-1} + (1 - m_2) \cdot g_t^2$

Step 5: Calculate $\widehat{p_t} = p_t / (1 - m_1^t)$

Step 6: Calculate $\widehat{q_t} = q_t / (1 - m_2^t)$

Step 7: Update the parameter $w_t = w_{t-1} - \alpha \cdot \widehat{p_t} / \left(\sqrt{\widehat{q_t}} + \epsilon\right)$

}

Step 8: return w_t

where *f(w)* — nonlinear stochastic objective function with parameter *w* containing local extremes; *g(t)* — gradient at time t along w, allows to get the direction to move towards local extrema; *t* — timestep, t=1, 2..; m_1 — the first moment vector (mean), the proposed value for ADAM is m_1=0.9; m_2 — the second moment vector (uncertained variance), the proposed value for ADAM is m_2=0.999; p_t — bias of the first order moment; q_t — bias of the second order moment; $\widehat{p_t}$ — bias correction of the first order moment; $\widehat{q_t}$ — bias correction of the second order moment; α — initial learning rate, the proposed value for ADAM is α=0.001; ϵ — parameter preventing division by zero, does not affect learning, the proposed value for ADAM is $\epsilon = 10^{-8}$.

The implementation of this algorithm allows adaptive adjustment of the optimal learning speed of each model parameter, which leads to highly accurate results, but storing the model parameters consumes memory twice as much as the model itself, which is an obstacle when training large models. In practice, to support such an algorithm with high memory consumption, it is necessary to use unloading to the CPU, which increases the delay and slows down the learning process.

RESULTS OF COMPUTATIONAL EXPERIMENTS

This section presents a description of the experimental learning and obtained results. A practical implementation of PINN learning with the ADAM algorithm using Python you can see on the Figs. 14–16.

Fig. 14 shows an example of a fragment of a network signal.

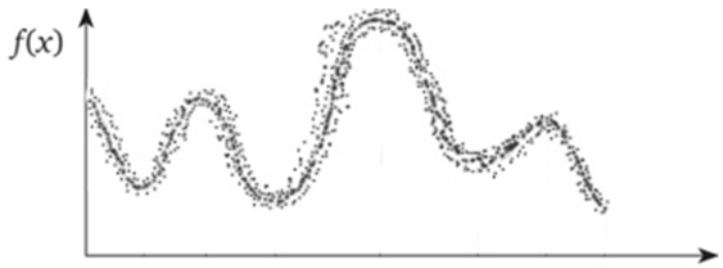

Fig. 14. An example of a fragment of a network signal

The Fig. 15 shows the restored at these points the initial nonlinear function *f*, in our case this is the optimized network signal. From Fig. 15 it follows that we quite accurately approximated the set of points with a nonlinear function. This was the purpose of learning — finding unknown parameters $(\widehat{p_t}, \widehat{q_t})$ to minimize a given loss function.

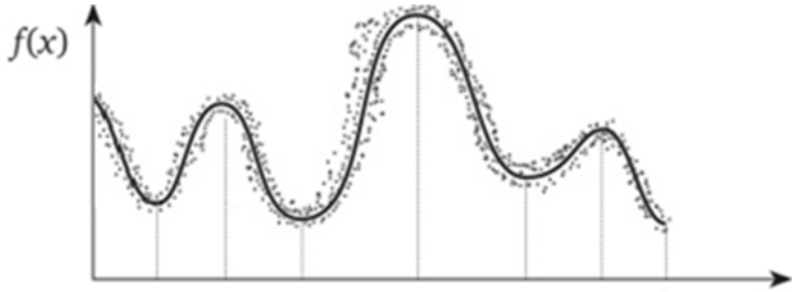

Fig. 15. The restored at points the initial nonlinear function *f*

Fig. 16 shows the network signal function directly without noise.

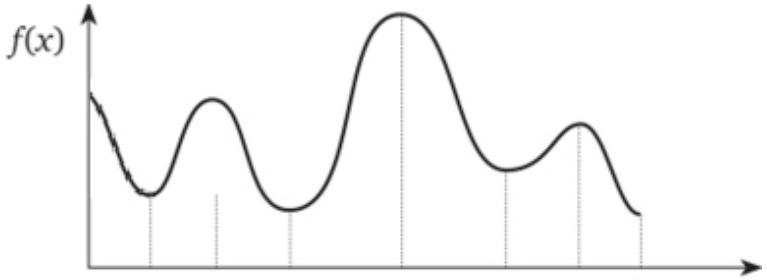

Fig. 16. The network signal function directly without noise

Let's consider the practical example. For example, we tried to send packages to Tfl.gov.uk from our network in Kharkiv, Ukraine. Let's complete the following steps:

- trace using the command tracert in Windows Operation system to determine the number of nodes;
- execute a command ping to send packets without using PINN;
- calculate network bandwidth utilization for data transfer without using PINN;
- execute a command ping to send packets using PINN;
- calculate network bandwidth utilization for data transfer using PINN.

Calculate the result of sending 4 packet of 4096 B to Tfl.gov.uk without using PINN.

$$U1 = \frac{4 \times 4096\ B}{(27 + 193 + 61 + 49)\ ms} \times 100\% = \frac{0{,}016384\ MB}{0{,}327\ s} \times 100\% =$$

$$= 0{,}05 \frac{MB}{s} \times 100\% = 5{,}0\%$$

And testing data transmission using the PINN trained with the ADAM algorithm we obtained the following results:

$$U2 = \frac{4 \times 4096\ B}{(17 + 12 + 15 + 35)\ ms} \times 100\% = \frac{0{,}016384\ MB}{0{,}079\ s} \times 100\% =$$

$$= 0{,}207 \frac{MB}{s} \times 100\% = 20{,}7\%$$

So, using the PINN trained with the ADAM algorithm to trasnsmit data the efficiency has increased. Thanks to this method, it is possible to obtain a low noise signal in practice, due to which network traffic is optimized.

DISCUSSION

Based on the research results presented in this article, the following main conclusions can be drawn.

Network traffic optimization can be improved at the protocol level and at the application level.

At the protocol level you can increase the number of transmitted packets. The more packets are transmitted, the greater efficiency of data transmission will be, but taking into account the channel capacity and packet loss in the network.

At the application level you can use Physics-Informed Neural Network for optimization of network bandwidth utilization. While high network utilization indicates the network is busy, low network utilization indicates the network is idle. Using the PINN trained with the ADAM algorithm to trasnsmit data the efficiency is 15.7 %.

CONCLUSION

At the application level it is proposed to use also another neural network architecture – Learning Vector Quantization (LVQ) with Encoder-Decoder architecture, which is suitable for optimizing of network noises, texts and images over the network.

ACKNOWLEDGMENTS

This paper is part of the DIOR project that has received funding from the European Union's MSCA RISE programme under grant agreement No. 10100828.

REFERENCES

1. C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017. doi: https://doi.org/ 10.1109/MWC.2016.1500356WC
2. X. Zhang, T. Wang, "Elastic and reliable bandwidth reservation based on distributed traffic monitoring and control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4563–4580, Aug. 2022. doi: https://doi.org/10.1109/TPDS.2022.3196840
3. V. Paxson, F. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, Jun. 1995. doi: https://doi.org/10.1109/90.392383
4. S. Huang, P. Wei, B. Hualaitu, "Bandwidth optimization of information application system under fine integral method of fuzzy fractional order ordinary differential equations," *Alexandria Engineering Journal*, vol. 59, issue 4, pp. 2793–2801, 2020. doi: https://doi.org/10.1016/j.aej.2020.06.015

5. G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021. doi: https://doi.org/10.1038/s42254-021-00314-5
6. L. Von Rueden et al., "Informed machine learning – A taxonomy and survey of integrating prior knowledge into learning systems," *IEEE Transactions on Knowledge and Data Engineering*, 2023, pp. 614–633. doi: https:// doi.org/10.1109/TKDE.2021.3079836
7. E. Bodyanskyi, O. Rudenko, *Artificial neural networks: Architecture, training, application.* Kharkiv: Teletech, 2004, 369 p.
8. O. Rudenko, E. Bodiansky, *Artificial neural networks.* Kharkiv: Smith Company, 2006, 390 p.
9. V. Filatov, O. Zolotukhin, A. Yerokhin, M. Kudryavtseva, "The methods for the prediction of climate control indicators in the Internet of Things systems," *CEUR Workshop Proceedings*, pp. 391–400, 2021. doi: https://doi.org/10.5281/zenodo.14526027
10. I.E. Lagaris, A. Likas, D.I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sept. 1998. doi: https://doi.org/ 10.1109/72.712178
11. Y. Bodyanskiy, O. Zolotukhin, A. Yerokhin, M. Kudryavtseva, M. Yerokhin, "Fast stacking neuro-neo-fuzzy system for inverse modeling in online mode," *International Journal of Computing*, vol. 24, no. 4, pp. 661–667, 2025. doi: https://doi.org/10.47839/ijc.24.4.4330
12. O. Zolotukhin, V. Filatov, A. Yerokhin, M. Kudryavtseva, and V. Semenets, "An approach to the selection of behavior patterns autonomous intelligent mobile systems," in *IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, *Kharkiv, Ukraine, 2021*, pp. 349–352. doi: https://doi.org/10.1109/PICST54195.2021.9772110
13. V. Filatov, O. Zolotukhin, M. Kudryavtseva, "Intellectual data analysis in relational information and analytical systems," *Innovative Technologies and Scientific Solutions for Industries*, no. 4, pp. 101–111, 2025. doi: https://doi.org/10.30837/2522-9818.2025.4.101
14. J. Uduagbomen, S. Lakshminarayana, M. Leeson, T. Xu, "Physics-informed neural network modeling of solution pulses in optical communication systems," *2022 IEEE Photonics Society Summer Topicals Meeting Series (SUM), Cabo San Lucas, Mexico, 2022*, pp. 1–2. doi: https://doi.org/10.1109/ SUM53465.2022.9858244

Received 16.01.2025

INFORMATION ON THE ARTICLE

Oleg V. Zolotukhin, ORCID: 0000-0002-0152-7600, Kharkiv National University of Radio Electronics, Ukraine, e-mail: oleg.zolotukhin@nure.ua
Maryna S. Kudryavtseva, ORCID: 0000-0003-0524-5528, Kharkiv National University of Radio Electronics, Ukraine, e-mail: maryna.kudryavtseva@nure.ua
Yevgeniy V. Bodyanskiy, ORCID: 0000-0001-5418-2143, Kharkiv National University of Radio Electronics, Ukraine, e-mail: yevgeniy.bodyanskiy@nure.ua
Valentin O. Filatov, ORCID: 0000-0002-3718-2077, Kharkiv National University of Radio Electronics, Ukraine, e-mail: valentin.filatov@nure.ua
Andriy V. Antilikatorov, Kharkiv National University of Radio Electronics, Ukraine, e-mail: andrii.antilikatorov@nure.ua
Denys V. Kalinin, Kharkiv National University of Radio Electronics, Ukraine, e-mail: denis.kalinin@teamdev.com

НЕЙРОННА МЕРЕЖА З ФІЗИЧНОЮ ІНФОРМАЦІЄЮ ДЛЯ ОБРОБЛЕННЯ СИГНАЛІВ ТА ЗВ'ЯЗКУ МЕРЕЖЕВОГО ТРАФІКУ/ О.В. Золотухін, М.С. Кудрявцева, Є.В. Бодянський, В.О. Філатов, А.В. Антілікаторов, Д.В. Калінін

Анотація. Проблема оброблення сигналів та оптимізації мережевого трафіку вирішується на апаратному рівні і є цікавою, сучасною та актуальною з точки зору прикладного рівня. Необхідно запропонувати підхід, який поєднує методи машинного навчання із завданнями пропускної здатності мережі та трафіком по мережі. Для вирішення цієї задачі запропоновано використовувати концепцію інформованого машинного навчання (IML), а саме таксономію IML, принципи побудови систем глибокого машинного навчання на основі інформації про фізичні властивості досліджуваної мережі передавання даних. Платформою для розроблення є моделі глибокого машинного навчання з використанням нейронних мереж з фізичною інформацією (PINN). Нейронна мережа представляє клас алгоритмів глибокого навчання, які можуть інтегрувати дані з описом фізичних процесів або без них. Як алгоритм запропоновано використовувати популярний алгоритм у галузі глибокого навчання – ADAM для оптимізації мережевого трафіку. Використання PINN, навченого за алгоритмом ADAM, для передавання даних підвищило ефективність. Завдяки такому методу на практиці вдається отримати малошумний сигнал, внаслідок чого оптимізується мережевий трафік.

Ключові слова: оцінка адаптивного моменту, штучний інтелект, штучна нейронна мережа, візуалізація даних, динамічний нейрон, градієнтне зниження, інформоване машинне навчання, алгоритм навчання, використання пропускної здатності мережі, оптимізація мережевого трафіку, нейро-нечітка логіка, Python.