Neural Networks Interpretation Improvement

Aliaksandr Kroshchanka
Brest State Technical University
Brest, Belarus
kroschenko@gmail.com

Vladimir Golovko
Brest State Technical University
Brest, Belarus
vladimir.golovko@gmail.com

***Abstract*— The paper is devoted to studying the issues of interpretability of neural network models. Particular attention is paid to the training of heavy models with a large number of parameters. A generalized approach for pretraining deep models is proposed, which allows achieving better performance in final accuracy and interpreting the model output and can be used when training on small datasets. The effectiveness of the proposed approach is demonstrated on examples of training deep neural network models using the MNIST dataset. The obtained results can be used to train fully connected type of layers and other types of layers after applying of flatting operation.**

Keywords— deep neural network, pretraining, Explainable AI, Restricted Boltzmann Machine.

I. INTRODUCTION

Neural network models are used in various areas of human life, often associated with critical infrastructure and healthcare. Under such conditions, the ability to explain the operation of the model determines the level of confidence in the results obtained by it. A model that can "tell" about what particular features of the input data were used to obtain the final results is easier to control for human experts, who, if necessary, can cross-check the results of the neural network model.

On the other hand, modern neural network models have a large number of parameters that are adjusted during training. Unfortunately, the conventional training of such models on large datasets is in most cases inaccessible to most researchers, and the use of small datasets leads to the effect of overfitting the model and, as a result, to unsatisfactory results obtained during testing.

A possible way to overcome this problem is to use pretraining. It is necessary to distinguish between pretraining as a stage of preparing a neural network model on one training dataset and pretraining as a process of training a model on a large dataset and its additional finetuning for solving other problems. In the first case, training can be carried out on small datasets and, in general, is not resource-intensive [1]. In the second case, it is necessary to use special hardware to complete the training within the specified time limits and dataset size.

Combining the interpretability of the model with the ability to train it on a small dataset in reasonable time limits opens up opportunities for a wider application of neural networks in real-time applications in which fast decision making with the ability to easily explain the results is critical. These qualities make it possible to obtain easily interpretable large models in the presence of severe limitations of the hardware capabilities of the systems used to prepare such neural networks.

On the other hand, interpretability opens door to using methods from different paradigm of AI (for example, semantic webs). New intelligent systems that will be developed in the next few years will most likely be developed using hybrid methods [2]. These systems will be more flexible and will get new possibilities and advantages from other approaches, which exist in artificial intelligence science.

This article is organized as follows. Part II provides an overview of the main methods for interpreting neural networks. Part III discusses the pretraining of deep neural network models. Part IV describes the proposed approach. Part V describes the numerical experiments and the obtained results. The article ends with a conclusion.

II. NEURAL NETWORKS INTERPRETATION

The issue of interpretability in neural network models has been effectively addressed by Explainable AI (XAI) methods [3]. In XAI approaches like LIME [4] and SHAP [5], the analysis relies solely on the model's input and output. These model-agnostic methods can be applied to neural network models with varying architectures.

The SHAP method aims to explain changes in model predictions resulting from alterations in input features. It quantifies the contribution of each feature to the model's prediction. Grounded in game theory, SHAP employs Shapley values as key metrics to assess the contribution of individual features to the overall model output. Features are considered players, where their presence indicates a specific value in the example *x*, while absence denotes an undefined value. Together, these features form a coalition of players.

Let $f: X \to Y$ represent the studied model, $x \in X$ denote the selected test example for which the model's output is interpreted, $X \subset R^N$ represent the feature space, N be the number of players (features), and Y be the output space of the model. Assuming that some features in x are known while others remain undefined, we obtain a vector x_S containing the known feature values.

The Shapley values for each player are calculated using the following formula:

$$\phi(i) = \sum_{S \in 1,2,\ldots,N} \frac{|S|!(|N|-|S|-1)!}{N!} \Delta(i,S)$$

Here, S represents the coalition of players, and $\Delta(i,S)$ is the efficiency gained by adding player i to the coalition S:

$$\Delta(i,S) = \nu(S \cup i) - \nu(S)$$

The efficiency $\Delta(i,S)$ is determined by a characteristic function ν that assigns a numerical value to each coalition of players. In the SHAP method, the characteristic function for the feature set S of example x is given by the conditional expectation:

$$\nu(S) = E[f(x) \mid x_S]$$

In practice, simplifications are often applied when calculating the characteristic function, such as the Kernel

SHAP modification. This method builds upon LIME and Shapley values.

III. Pretraining of Deep Neural Networks

Pretraining can be conditionally classified into two main types – type I – this is pretraining on small datasets using a special algorithm that allows you to get a good initialization of the parameters of the neural network model and type II – pretraining on a large dataset using the backpropagation method.

Until the mid-2000s, training large neural networks was not seen as a promising area of research. The reasons for this were, on the one hand, the lack of special approaches to training "heavy" models and the impossibility of obtaining good results due to the problems of vanishing and exploding gradients, and on the other hand, insufficient technical capabilities for training such models. Thanks to the results obtained by G. Hinton and Y. Bengio, significant progress has been made in training "heavy" models. Also, since the beginning of the 2010s, there has been significant progress in the use of technical means for working with such models, in particular, video cards and video accelerators have become more widely used, which made it possible to speed up the training process, reducing it by several times compared to training only using a processor.

At the moment, the simplest strategy for applying neural network models has become the use of libraries of pretrained networks. This approach simplifies the preparation of models for custom data, saving time for additional finetuning on user datasets. However, it also has a number of disadvantages. One of them is the need to use a given network configuration without the possibility of its significant change. In other words, when retraining a model, researchers add or remove the last layers of the model, but they cannot globally influence its structure anymore (for example, by increasing or decreasing the number of neural network units in layers), because this will negatively affect the distribution of model parameter values and lead to poor initialization.

These problems are solved by using pretraining of type I, which is based on the use of special methods of layer-wise unsupervised learning. One of such methods is an approach for which a network consisting of fully connected layers is considered as a sequence of independent restricted Boltzmann machines that are trained sequentially in accordance with a "greedy" algorithm.

IV. Proposed Approach

The approach we propose is based on the use of type I pretraining and generalizes the method of layer-wise greedy training proposed by G. Hinton earlier [1].

The layers of deep neural network can be trained in a sequential mode, if each layer is represented as the Restricted Boltzmann machine – RBM [6-9] (Fig. 1). RBM is a stochastic neural network with two layers – visible (*X*) and hidden (*Y*). The input data for each RBM model is the data obtained by passing through the previous pretrained layers.

For the first RBM, the initial data is taken from the training dataset, for the second, the data obtained after applying the first pretrained layer, and so on. After performing layer-wise pretraining, the model is finetuned by training it by the method of error back propagation.

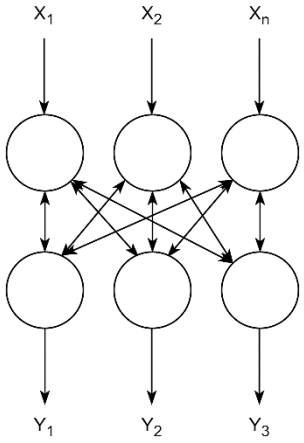

Fig. 1. Architecture of Restricted Boltzmann Machine

Classical rules for RBM training can be obtained by maximization of logarithm of likelihood function:

$$P(x) = \sum_{y} P(x, y),$$

where $P(x, y)$ is the probability for neurons state (x, y).

$P(x, y)$ can be obtained as $P(x, y) = \frac{e^{-E(x,y)}}{Z}$, where $Z = \sum_{x,y} e^{-E(x,y)}$ – probability normalization parameter, E – the energy of the system in given state (x, y).

Classical rules to train such network have next view (batch learning case):

$$w_{ij}(t+1) = w_{ij}(t) + \frac{\alpha}{L}\left(\sum_{l=1}^{L}(x_i^l(0)y_j^l(0) - x_i^l(1)y_j^l(1))\right)$$

$$T_i(t+1) = T_i(t) + \frac{\alpha}{L}\left(\sum_{l=1}^{L}(x_i^l(0) - x_i^l(1))\right)$$

$$T_j(t+1) = T_j(t) + \frac{\alpha}{L}\left(\sum_{l=1}^{L}(y_j^l(0) - y_j^l(1))\right)$$

In contrast to the classical method, we used a simple function representing the mean square error (MSE) of data reconstruction on the visible and hidden layers of the restricted Boltzmann machine [10-11]:

$$E_s = \frac{1}{2L}\sum_{l=1}^{L}\sum_{j=1}^{m}\left(\Delta y_j^l(1)\right)^2 + \sum_{l=1}^{L}\sum_{i=1}^{n}\left(\Delta x_i^l(1)\right)^2,$$

where $\Delta y_j^l(1) = y_j^l(1) - y_j^l(0)$, $\Delta x_i^l(1) = x_i^l(1) - x_i^l(0)$, $x_i^l(0)$, $y_j^l(0)$ – original data on visible and hidden layers of RBM, $x_i^l(1)$, $y_j^l(1)$ – reconstructed data on visible and hidden layers of RBM, L – size of the training dataset.

Based on the idea of minimizing this function, the following rules can be obtained (batch learning case):

$$w_{ij}(t+1) = w_{ij}(t) - \frac{\alpha}{L}\left(\begin{array}{l}\sum_{l=1}^{L}\Delta y_j^l(1)x_i^l(1)F'(S_j^l(1)) + \\ \Delta x_i^l(1)y_j^l(0)F'(S_i^l(1))\end{array}\right)$$

$$T_i(t+1) = T_i(t) - \frac{\alpha}{L}\sum_{l=1}^{L}\Delta x_i^l(1)F'(S_i^l(1))$$

$$T_j(t+1) = T_j(t) - \frac{\alpha}{L}\sum_{l=1}^{L}\Delta y_j^l(1)F'(S_j^l(1))$$

where $w_{ij}(t+1)$ - next step weights, $T_i(t+1)$ - next step thresholds for visible layer, $T_j(t+1)$ - next step thresholds for hidden layer, α - learning rate, $S_i^l(1)$ - weighted sums for visible layer, $S_j^l(1)$ - weighted sums for hidden layer, F' - derivative of activation function.

It can also theoretically be proven that maximizing the likelihood function of data distribution $P(x)$ is equivalent to minimizing the MSE function in the context of using RBM with linear neural units. These results make it possible to expand the classical approach by adding alternative learning rules for non-linear types of neural units, which can also be successfully applied to rectified linear units.

V. Experiments and Main Results

The experiments were carried out on well-known computer vision dataset MNIST [12].

The main parameters for all datasets are shown in Table I. We varied parameters such as mini-batch size and learning rate, fixing the best resulting results, shown in Table II. The following image (Fig. 2 and 3) shows the evolution of the error when training with three different methods - 1. Classical pretraining + BP finetuning (PBP), 2. Proposed pretraining + BP finetuning (PPBP), 3. No pretraining, only using the backpropagation method (BP) and 4. Hybrid training, based on the use of a combination of classical and proposed methods + BP finetuning (HPBP). For such combination we used 7 epochs of classic method and 3 epochs of proposed method.

TABLE I. Main parameters

Pretraining phase		**Training phase**	
Parameter	***Value***	***Parameter***	***Value***
Epochs count	10	Epochs count	90
Momentum	0.5…0.9	Momentum	0.9
Learning rate	0.01…0.04	Learning rate	0.1
Batch size	128	Batch size	128

TABLE II. Results Of Models Training

Network architecture	**Activation functions**	**Accuracy on test dataset, %**			
		PBP	***PPBP***	***BP***	***HPBP***
784-800-800-10	ReLU	98.62	**98.81**	98.64	98.51
784-1600-1600-800-800-10	ReLU	98.58	**98.7**	98.56	98.68
784-800-800-10	Sigmoid	**98.47**	98.36	98.18	98.43
784-1600-1600-800-800-10	Sigmoid	**98.54**	98.47	98.0	98.46

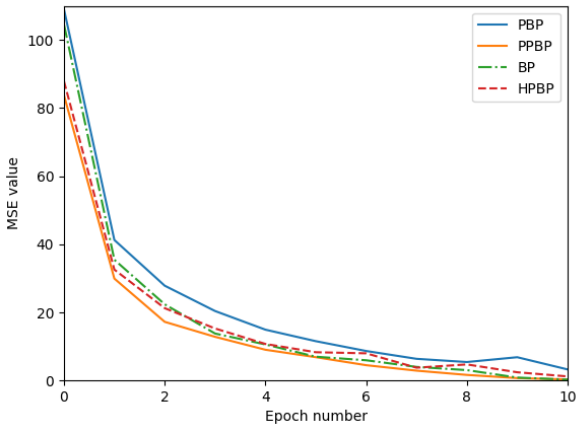

Fig. 2. Evolution of MSE for tested methods (ReLU activation function, architecture 784-800-800-10, first 10 epochs)

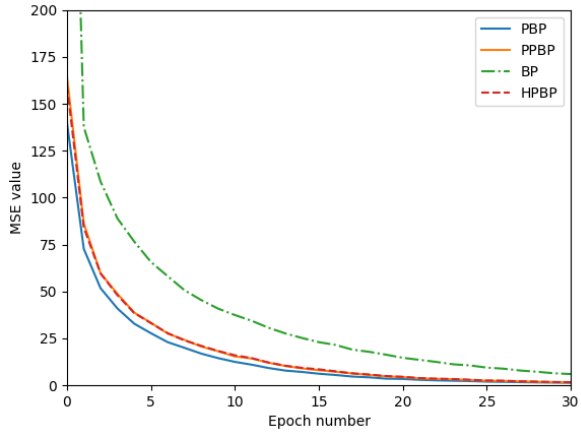

Fig. 3. Evolution of MSE for tested methods (sigmoid activation function, architecture 784-800-800-10, first 30 epochs)

From the obtained data, the following conclusions can be drawn:

1. Increasing the network depth does not always lead to better results with the same experiment parameters
2. The results obtained for different pretraining methods can vary greatly depending on the activation functions used.
3. Actually in some cases pure backpropagation gets better results in compare with approach with pretraining phase (for example, if used ReLU activation function).

After training the models, we used the XAI method to interpret instances from the original datasets (from their test parts). To do this, we used the models with the best final results.

The interpretation results are shown in the image (Fig. 4).

On this image we can seen extraction of main features, which are used by neural network model to define which class this number has. The greater the SHAP-value, the stronger the impact on the result has a given feature.

VI. Conclusion

In this paper, we propose a approach for layer-wise pretraining of deep neural networks based on the use of a sequence of restricted Boltzmann machines. The practical

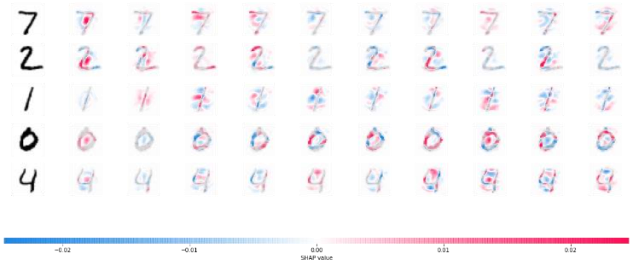

Fig.4. Model input interpretation

results obtained by us confirm the effectiveness of the proposed training method. This approach allows using small dataset to pretrain “heavy” neural networks models. The trained model was used as basis to perform the interpretation of the input data.

The authors see the expansion of the theoretical and experimental part of the research to the area of convolutional and recurrent neural networks, as well as the interpretation of models that have layers of these types in their architectures, as the main direction of further work.

This work was supported by the Belarusian Republican Foundation for Basic Research BRFBR, project F22KI-046.

REFERENCES

[1] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006b). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1554.

[2] Kovalev, M. Convergence and integration of artificial neural networks with knowledge bases in next-generation intelligent computer systems / M. Kovalev, A. Kroshchanka, V. Golovko // Open Semantic Technologies for Intelligent Systems (OSTIS 2022). – BSUIR, 2022. – P. 173-186.

[3] A. Thampi, Interpretable AI: Building Explainable Machine Learning Systems. Manning, 2022. [Online]. Available: https://books.google.by/books?id=ePN0zgEACAAJ

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why Should I Trust You?: Explaining the Predictions of Any Classifier,” 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[5] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[6] Hinton, G. and Sejnowski, T. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, pages 282–317. MIT Press, Cambridge, MA.

[7] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8):1771–1800, 2002

[8] Hinton, G. E. A practical guide to training restricted Boltzmann machines: Tech. Rep. 2010-000 / G. E. Hinton: University of Toronto, 2010.

[9] Liao, Renjie & Kornblith, Simon & Ren, Mengye & Fleet, David & Hinton, Geoffrey. (2022). Gaussian-Bernoulli RBMs Without Tears. 10.48550/arXiv.2210.10318.

[10] V. Golovko, A. Kroshchanka, and E. Mikhno, “Deep Neural Networks: Selected Aspects of Learning and Application,” in Pattern Recognition and Image Analysis. Cham: Springer International Publishing, 2021, pp. 132–143.

[11] Kroshchanka, A. Deep neural networks application in next-generation intelligent computer systems / A. Kroshchanka // Open Semantic Technologies for Intelligent Systems (OSTIS 2022). – BSUIR, 2022. – P. 187-194.

[12] LeCun, Y. The mnist database of handwritten digits. — http://yann.lecun.com/exdb/mnist/. — Accessed: 2023-08-01.