ISSN 0027-1349, Moscow University Physics Bulletin, 2024, Vol. 79, Suppl. 2, pp. S828–S843. © Allerton Press, Inc., 2024.

A Transformer Architecture for Risk Analysis of Group Effects of Food Nutrients

A. N. Balandina[1]*, B. V. Gruzdev[1], N. A. Savelev[2], Y. S. Budakyan[1], S. I. Kisil[3], A. R. Bogdanov[4], and E. A. Grachev[1]

[1] *Faculty of Physics, Lomonosov Moscow State University, Moscow, Russia*
[2] *PJSC VimpelCom, Moscow, Russia*
[3] *Institute of Russian Language and Culture, Lomonosov Moscow State University, Moscow, Russia*
[4] *13th Moscow City Hospital, Moscow, Russia*

Received October 1, 2024; revised October 10, 2024; accepted October 20, 2024

Abstract—In medicine, context is crucial for accurate patient diagnosis, as the same indicator can have different implications based on its setting. Transformer architecture models have not yet been applied to analyze nutritional data in patient histories. These models offer significant advantages for biomedical analysis, such as considering global context, interpreting attention weights, and generating informative input vectors. The attention mechanism's ability to uncover multifactorial relationships can help physicians save time and concentrate on specific patterns identified by the neural network. This study adapted the encoder transformer for tabular data, applying it to classify metabolic disorders in patient history. Research into applying transformer architecture to tabular and dietary data shows great promise, yielding results that align with established medical findings while introducing innovative methods for utilizing attention and vectorizing this data.

Keywords: actual nutrition, machine learning, metabolic syndrome, nutrients, obesity, insulin resistance, nutritional epidemiology, data analysis, transformer

DOI: 10.3103/S0027134924702291

1. INTRODUCTION

The number of scientific papers devoted to machine learning and artificial intelligence is growing every year. The relevance of research in this direction is confirmed by the mass of various applications of machine learning methods, including automation of pattern recognition processes, adaptive control, approximation of functionals, forecasting, creation of expert systems, organization of associative memory and many other applications. Modern machine learning methods make it possible to perform some tasks that previously could only be performed by humans. In addition to achieving high quality performance of some tasks, it turns out that some machine learning methods are well-interpretable. Due to such methods it is possible to identify important factors and patterns that were not detected before. This makes this area of research very interesting, because "understanding" artificial neural network of the device of some objects can hepl see once unseen things. One example of such discoveries is the emergence of the AlphaZero neural network algorithm and the new chess tactics [1] discovered using it.

Introduced in 2017, the transformer [2] architecture has achieved tremendous results in the field of machine translation, and subsequently in other areas not even related to text analysis [3]. The basis of this architecture is the attention [4] layer, which, in addition to demonstrating impressive success in sequential data, also makes it possible to interpret certain decisions of algorithms built on its basis [5].

In this paper, developed neural network based on transformer techniques is used to analyze biomedical data.

2. MATERIALS AND METHODS

2.1. Motivation

Metabolic syndrome is a medical condition characterized by visceral obesity. The main cause of

*E-mail: annikk007@yandex.ru

metabolic syndrome is an unhealthy lifestyle. Metabolic syndrome leads to cardiovascular and other diseases. Early detection and treatment of metabolic syndrome can reduce the risk of developing serious diseases in the future. That means, early diagnosis of abnormalities becomes one of the key tasks in the evaluation of patients with suspected metabolic syndrome.

Given statements above, it is important to understand which combinations of features leads to development of metabolic syndrome. So, the problem is to find consistent relationships between nutrients that lead to an increased risk of metabolic syndrome development.

We chose to utilize a neural network based on the transformer architecture to identify nutrient groups that correlate with the risk of metabolic syndrome. This approach allowed us to examine the attention mechanism within the trained network and determine which nutrient groups the network focuses on when predicting the presence of metabolic syndrome in a patient.

2.2. Data Description

A dataset of patients obtained in 13th Moscow City Hospital was analysed. The features include both quantitative (anthropometry, actual nutrition, laboratory parameters and others) and categorical (exercise frequency and others). Clinicians have also identified patients whose diagnoses fall into the category of the metabolic syndrome.

Actual nutrition at home was assessed by the method of frequency-quantitative analysis using the "Digital Dietetics" service (PROFIT-M LLC—IPI FGBU "NMIC TPM" of the Ministry of Health of Russia [6]). Energy intake with food was analyzed, as well as the following macro- and micronutrients: protein (g), fats (g), saturated fats (g), monounsaturated fats (g), polyunsaturated fats (g), oleic acid (g), polyunsaturated fatty acids of omega-3 family (g), polyunsaturated fatty acids of omega-6 family (g), gamma-linolenic acid (mg), carbohydrates (g), refined carbohydrates (g), dietary fiber (g), starch (g), fructose (g), lactose (g), galactose (g), cholesterol (mg), sodium Na (mg), potassium K (mg), calcium Ca (mg), silicon Si (mg), sulfur S (mg), magnesium Mg (mg), phosphorus P (mg), iron Fe (mg), boron B (μg), vanadium V (μg), iodine I (μg), cobalt Co (μg), manganese Mn (mg), copper Cu (mg), molybdenum Mo (μg), nickel Ni (μg), selenium Se (μg), fluorine F (μg), chromium Cr (μg), zinc Zn (mg), vitamin A (μg), β-carotene (μg), retinol equivalent, thiamine (μg), riboflavin (μg), cyancobalamin (μg), folic acid (μg), pyridoxine (μg), pantothenic acid (μg), choline (μg), ascorbic acid (μg), vitamin D (μg), α-tocopherol (μg), biotin (μg), niacin (μg).

The method allows assessing nutrition over a selected period of time and is based on the assessment of the frequency of consumption of certain food products and dishes specified in the questionnaire. The respondent selects a suitable category of consumption frequency ("once a month," "once a week," etc.) in the questionnaire. The method is designed to obtain qualitative and quantitative descriptive information about the habitual pattern of eating. In questionnaires, the frequency of consumption of average portions is determined; in quantitative assessment, the respondent chooses the portion size from a list or specifies it himself. In this method, the average amount of food consumed is estimated using a conversion to the volume of the average portion of the product or dish indicated in the questionnaire near each food item and a coefficient of the frequency of consumption per day. The questionnaire used in this study had 50 questions that included foods typical for consumption in Russia [7].

Assessment of anthropometry and body composition was performed by bioimpedanceometry using the InBody520 analyzer (In Body, Korea) and included the following indicators: height, body weight, body mass index, body mass index, amount of fat mass, amount of muscle mass, lean body mass, total fluid, extracellular fluid, intracellular fluid, visceral fat area, distribution of lean mass and fluid by sectors.

Laboratory parameters of blood plasma were evaluated using a biochemical analyzer "Konelab 30i" from "Thermo Clinical Labsystems" (Finland). The following parameters were analyzed: glucose, total cholesterol, triglycerides, high-density lipoproteins, low-density lipoproteins, uric acid, alanine aminotransferase (ALT), aspartate aminotransferase (AST), total protein and fractions, vitamin status.

Data preparation for modeling included data cleaning, marking up the target variable, and working with traits. Duplicate patients (by full name and date of birth) were excluded from the sample. The final sample size was 2885 patients, of which 70% were taken in the train sample and 30% in the test sample. The number of attributes after pretraining was 85, of which 67 described the nutrient composition of the actual diet and 18 were related to anthropometry and other patient data. Diagnoses associated with the presence of metabolic syndrome were labeled as 1 (positive class), people with no diagnosis or with a record of "examination" were labeled as 0 (negative class). The class balance in the dataset (ratio of the number of healthy patients to those who had a diagnosis associated with metabolic syndrome present) was 187/1013 (65%/35%).

2.3. Methods

2.3.1. Attention. In neural networks, attention is a technique that mimics cognitive attention. The effect enhances some parts of the input data while reducing other parts, suggesting that the network should pay more attention to that small but important part of the data. Learning which part of the data are more important than others is context dependent and is trained using gradient descent. The Attention mechanism is used in a wide variety of architectures and tasks (translation, text generation, image annotation, etc.).

The attention-mechanism diagram is shown below in Fig. 1.

Below is shown the principle of operation of attention on the example of self-attention:

1. Given a sequence of vectors $S = \{x_1, x_2, ..., x_n\}$, $x \in \mathbb{R}^m$ from which a matrix $X \in \mathbb{R}^{n*m}$ is obtained, where the ith row contains vector x_i.

2. The metrics $Q = W_Q X$, $K = W_K X$, $V = W_V X$ are obtained, where W_Q, W_K, W_V are the training weights.

3. $X_{\text{new}} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, where d_k is the dimensionality of the vectors of Q and K.

2.3.2. Transformer. Transformer is a deep learning neural network, that utilizes attention mechanism by differently evaluating the importance of each part of the input data. This algorithm was developed in 2017 for translating texts from one language to another. Similar to recurrent neural networks (RNN) [9], transformers are designed to process sequential input data such as natural language. However, unlike RNNs, transformers process the entire input at once. The attention mechanism provides context for any position in the input sequence. For example, if the input data are natural language sentences, the transformer does not process one word at a time. This provides more parallelization than RNNs and hence reduces the training time.

In the original paper [2] in which transformer was presented, the authors showed that this neural network shows better results in the task of machine translation of texts. So far, the transformer architecture and neural networks based on it have shown the best results in natural language analysis tasks such as text generation, text understanding, and text comprehension. In addition to text analysis, transformer-based models have achieved state-of-the-art results in image analysis tasks [3, 10, 11].

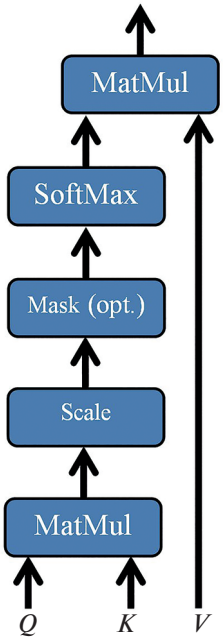

Fig. 1. Attention operation diagram.

Transformer operation diagram is shown in Fig. 2.

Transformer generates elements of the output row sequentially. Generation of the kth element formally looks as follows:

1. The transformer inputs a sequence of n input elements (initially text token numbers) $S = \{x_1, ..., x_n\}$, and a sequence of $k-1$ (technically k at once, where the kth vector is special) already generated objects $U = \{y_1, ..., y_{k-1}\}$.

2. These sequences are fed into a mechanism that converts them into sequences of vectors of dimension d_{model}, which are then encoded positively.

3. The elements of the input sequence go to the transformer's encoder, and the output sequence goes to the decoder.

4. The elements of the input sequence passing through the encoder and the output sequence passing through the first part of the Decoder fall into the attention mechanism, where Q is a matrix composed of $k-1$ members of the output sequence, and K and V are matrices composed of n vectors of the input sequence. It is worth considering that the matrices Q, K, V are not composed from the input vectors directly. First, the input vectors get into the fully connected neural layers, which prepare them for Q, K, V.

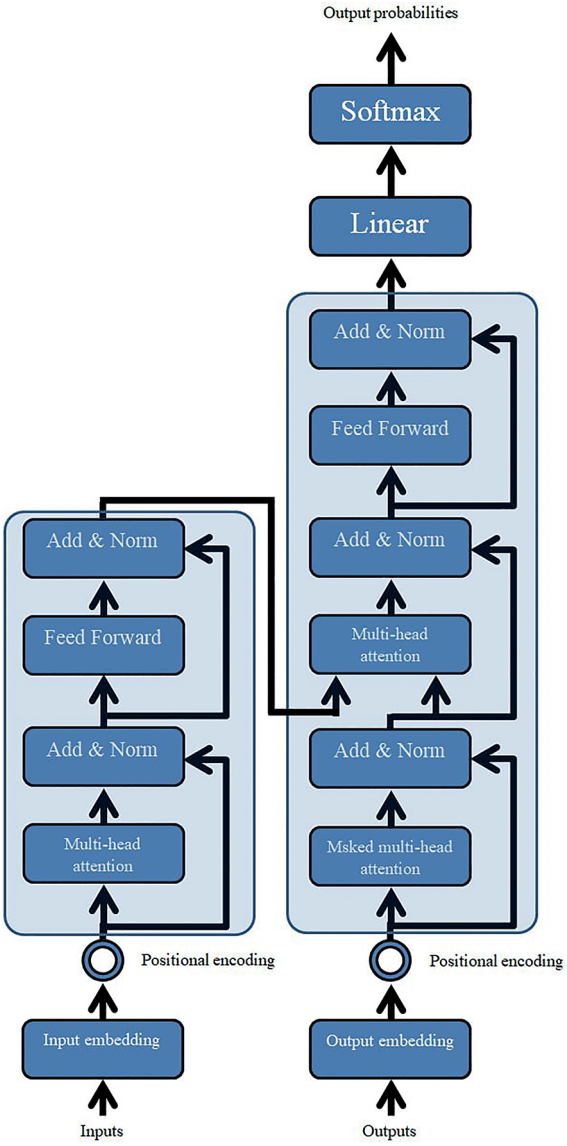

Fig. 2. Transformer operation diagram.

5. The $k-1$ vectors received and slightly processed generate the kth member of the sequence.

6. If the generated element is a stop element, the generation ends, otherwise the generation continues.

2.3.2.1. Position coding of input vectors. This neural network receives as input a sequence of elements $S = \{x_1, x_2, ..., x_n\}$. Since each element of a given neural network architecture interacts with a particular element of the sequence at a point in time—it is necessary to convey information on which position in the sequence it is located.

The authors used the following positional encoding [12] for the vector x_{pos}:

$$PE_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right),$$

$$PE_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right).$$

This encoding allows us to understand at which position the input vector is located due to the linear transformation.

2.3.2.2. Encoder. The encoder consists of N consecutive identical layers. Each layer has two sublayers. The first is a self-attention mechanism with several "heads," and the second is a simple fully-connected neural network. The authors use feedback [13] on each sublayer. That is, the output of each sublayer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function realized by the sublayer. $LayerNorm(x)^j = \frac{x^j - \mu_x}{\sigma_x + \epsilon}$, where μ_x and σ_x are the sample mat expectation and standard deviation on x, respectively.

2.3.2.3. Decoder. The decoder also consists of a stack of N identical layers. In addition to the two sublevels in each layer of encoder, decoder adds a third sublevel that performs multihead attention over the output of the encoder stack. as in encoder, the authors use feedback on each of the sublayers followed by layer normalization. They also modify the self-attention sublayer in the decoder stack to prevent the possibility of looking ahead during decoding. Attention now looks like $X_{\text{new}} = softmax\left(\frac{QK^T}{\sqrt{d_k}} + MASK\right)V$, where:

$$MASK_{i,j} = \begin{cases} -\infty, & \text{if} \quad i > j \\ 0, & \text{otherwise.} \end{cases}$$

This masking, combined with the fact that the output vectors are offset by one position, ensures that predictions for position i can only depend on known outputs at positions less than i.

2.3.3. BERT. BERT (bidirectional encoder representations from transformers) is a neural network published relatively recently by Google AI Language researchers. It has caused a stir in the machine learning community by providing state-of-the-art results on a wide range of NLP tasks, including question answering (SQuAD v1.1), natural language inference (MNLI), and others.

A key technical innovation of BERT is the application of transformer bidirectional learning to language modeling. This contrasts with previous attempts that have considered text either from left to right or a combination of left to right and right to left. The results of the paper show that a language model with

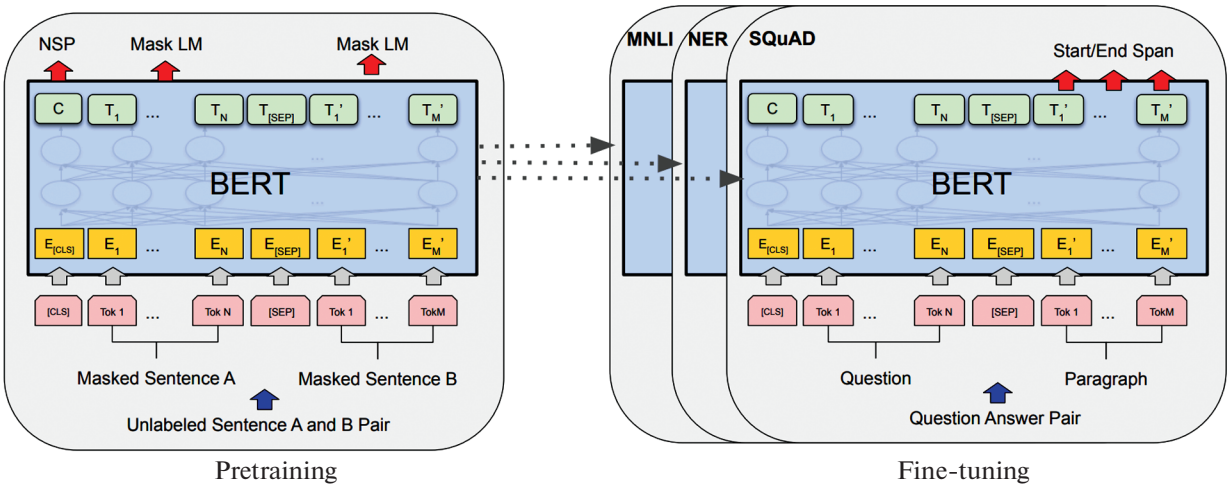

Fig. 3. BERT diagram.

bidirectional learning can have a deeper understanding of linguistic context than unidirectional language models. In the [14] paper, the researchers detail a new technique called masked LM (MLM) that allows bidirectional learning in neural networks where it was previously impossible. In fact, BERT is a transformer coder with self-learning and learning methods developed by the authors of the article. As well as recommendations for performing certain tasks and applying entities at a level higher than text tokens (segments).

This work is extremely important for machine learning in general, because in the field of computer vision researchers have repeatedly demonstrated the value of transfer learning—pretraining a neural network model for a known task, such as ImageNet, and then performing fine-tuning (fine-tuning or pretraining)—using the trained neural network as the basis for a new target model. After BERT, this method of self-training became applicable in the field of text analysis. And it is worth noting that in the case of BERT, self-training is used as a pretraining task, which makes it possible to compose training corpora from billions and trillions of objects, since there are countless texts on the Internet.

Below in Fig. 3 is a diagram of BERT's self-training (pretraining) and fine-tuning.

2.3.3.1. Self-learning. The authors proposed two methods that they recommended to be applied consistently in self-training. The first is called masked LM and the second is called next sentence prediction.

The principle of masked LM:

1. 15 percent of the tokens are selected from the input sequence;
2. Each of the selected tokens is replaced with a special token [MASK] with probability 0.8, replaced with a random token with probability 0.1, and left unchanged with probability 0.1;
3. By passing through BERT, the output vectors are classified and then the error (cross-entropy) for each token is calculated;
4. A gradient descent step is performed.

Due to this type of learning, the vectors of each token at the output of the pretrained BERT acquire unique $\mathbb{R}^n$ values due to the global context, due to which they can be further used to find the proximity of phrases in terms of meaning and words in a sentence [15]. Principle of next sentence prediction:

1. With probability 0.5, consecutive sentences A and B are taken in the text, and with probability 0.5, random sentences are taken.
2. The token [CLS] is inserted at the beginning of the first sentence, and the token [SEP] is inserted at the end of each of the two (A and B) input sentences.
3. A segment (sentence) vector indicating sentence A or sentence B is added to each token. Proposal vectors are similar in concept to token vectors.
4. This sequence of tokens is run through BERT.
5. The token [CLS] is fed into a simple binary classifier.
6. The error(binary cross entropy) is calculated and a gradient descent step is performed.

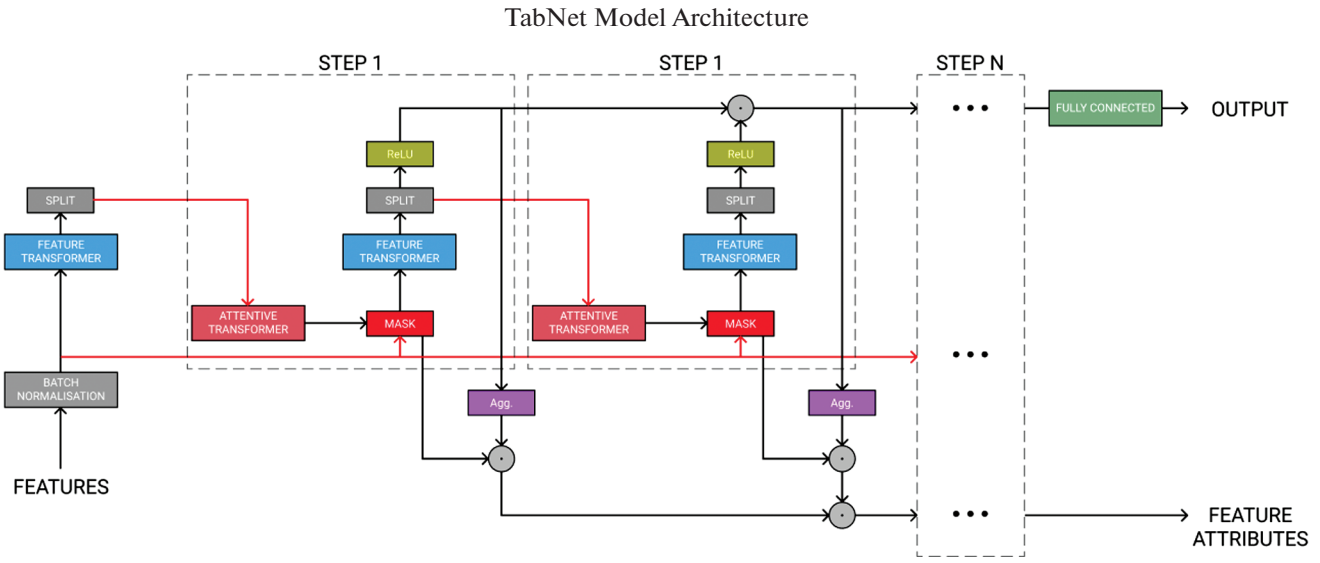

Fig. 4. TabNet architecture.

2.3.3.2. Training. When pretrained for an applied problem, the authors offer their solutions for classification and regression. The main idea is that the neural network is some backbone neural network whose output is fed to a rather simple classifier or regressor. The authors propose to use [CLS] token and solve classification and regression problems on it. In addition, the authors provide their recipes for additional training for specific tasks of text analysis and comprehension. These recipes can be found in the original article [14].

The authors first perform masked LM training and then NSP, giving each step an equal number of iterations.

2.4. Related Works

2.4.1. TabNet.

2.4.1.1. General description. Gradient boosting algorithms such as XGBoost [16], CatBoost [17], and LightGBM [18] have long been considered the best for tabular data tasks. Even with rapid advances in text analysis and computer vision, still tree-based algorithms usually outperform neural networks on tabular data.

According to the TabNet [19] article, this neural network was able to outperform leading tree-based algorithms on a variety of criteria. Not only that, it is much more explainable than advanced tree architectures because it has built-in explainability.

The main features of TabNet neural network:

1. TabNet uses sequential attention to select features at each stage of the decision making process, providing interpretability and better learning as learningopportunities are utilized for the most useful features.
2. The importance of features is not constant and depends on the x instance.
3. Design options allow TabNet to provide two types of interpretability: local interpretability, which visualizes the importance of features and how they combine for a single row, and global interpretability, which quantifies the contribution of each feature to the trained model across the entire dataset.

The architecture image below shows the various components of TabNet (Fig. 4).

TabNet is a recurrent neural network [9], especially resembling (by the block action) the LSTM [20] network. Each step of the recurrent network is shown in Fig. 1, and the number of such steps is a hyperparameter of the network (not a trainable parameter).

2.4.1.2. Self-learning. In TabNet, the authors have provided the ability to train a neural network unsupervised. The principle of such training is presented in Fig. 5.

The table shown in Fig. 5 summarizes the actual anonymized biomedical data of the patients.

The principle of self-learning is that some data fed to the input of the network are masked, and then, passing through the network itself and a decoder, are restored. Masking refers to the process by which a neural network receives not real data, but a label that tells it that the data has been masked. The probability of a feature being masked is a hyperparameter of the network during the self-learning phase. Arbitrary classifiers or regressors can be used as the decoder of the network depending on the type of data being

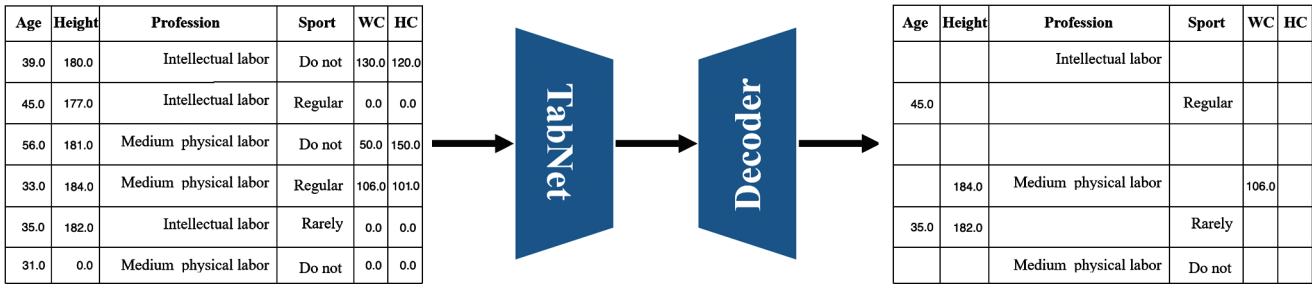

Fig. 5. TabNet self-learning.

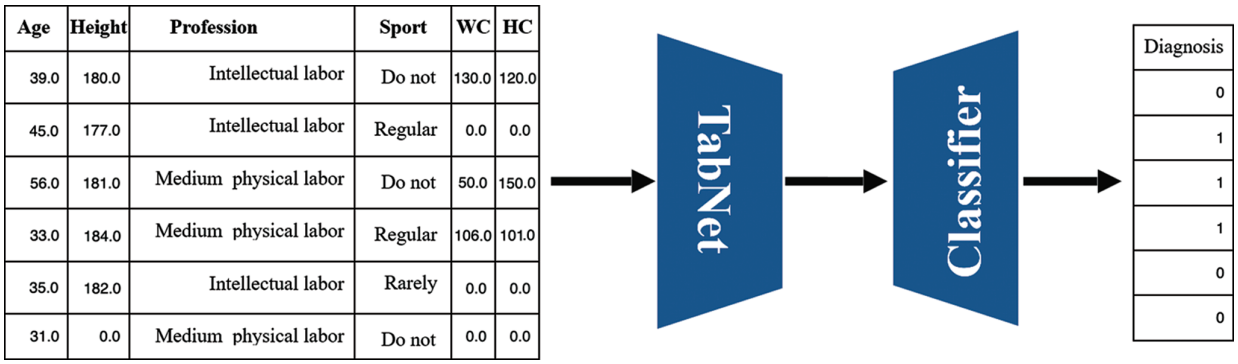

Fig. 6. TabNet training. Example of categorizing patient history for metabolic disorders.

masked. The mathematical formulation of the problem of TabNet self-training on patient anamnesis data will be the example from paragraph 2.2, where

$$weight_j^i = \begin{cases} 1, & \text{if } x_i^j \text{ is masked} \\ 0, & \text{otherwise.} \end{cases}$$

The main advantage of self-learning is that it is not necessary to know the diagnoses of patients in a particular task and the answers of a classification or regression task in general. The principle of TabNet self-learning allows a neural network to understand the general context, i.e., to evaluate features through their neighbors. Such a network, as the results of the study of the authors of the [19] show, is subsequently better trained for applied tasks. Also, this type of self-learning allows recovering missing values, although from the point of view of the nature of the data—it is a bit incorrect, because masking is made artificially, and the values in the table with data are missed in reality.

2.4.1.3. Training. TabNet is designed as a backbone neural network for regression and classification tasks. This means that this network is some kind of feature converter, allowing another neural network to perform classification or regression much easier. The training scheme of the TabNet network is shown below in Fig. 6.

During such training, both networks are trained simultaneously. It is common to use simple classifiers, because in this way the base network is forced to share more informative vectors in the output.

The same scheme is used for TabNet training in the case of regression, only a classifier there is replaced with a regressor.

2.4.1.4. Interpretability. Feature selection masks (not the same with the masking process in self-learning) in TabNet can be used to retrieve information about the selected features at each step. This feature would not be possible for conventional neural networks with fully connected layers, since the hidden units of each successive layer would jointly process all features without a sparsity-driven selection mechanism. For feature selection masks, if $M_{b,j}[i] = 0$, where i refers to the layer number, then the jth feature of the bth feature must have zero contribution to the overall solution. If $f[i]$ were a linear function, then the coefficient $M_{b,j}[i]$ would correspond to the importance of the feature $f_{b,j}$. Although each decision step uses nonlinear processing, their outputs are later combined in a linear fash-

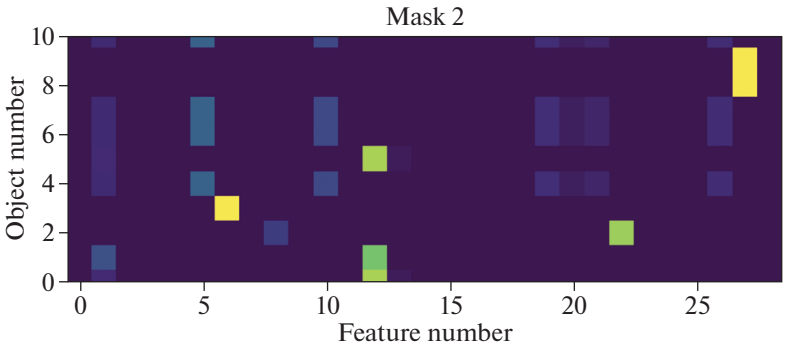

Fig. 7. Trait importance for a random sample of 10 patient history items on the TabNet mask of Step 2.

ion. An example of the constructed mask is shown in Fig. 7.

The approaches used in TabNet do not allow for multivariate analysis to the extent that it is performed in algorithms that use a full-fledged attention architecture. For example, TabNet can find the importance of features at a certain step, but it cannot find the relationships of features between each other.

2.4.2. BioBert. BioBERT [8] is a pretrained model of the original BERT [14], trained for medical text processing. It can be applied to named entity recognition (NER), semantic relation extraction (RE), question answering (QA), information retrieval, and other NLP tasks. The pretraining was performed on a large set of annotated biomedical texts. The architecture of the neural network and its training and self-training methods were not changed.

BioBert training and self-learning diagrams can be observed in the artide [20].

This neural network shows excellent results on applied tasks of analyzing biomedical texts. It also allows finding dependencies between text tokens in these texts. The main problem with this network for current work is that it is not designed for analyzing tabular data and cannot perform it, unlike the TabNet network.

2.5. Markov Clustering

Markov clustering (MCL) is a well-known algorithms for obtaining clusters in stochastic graph [21]. MCL is an algorithm to identify the strongly connected nodes and group them into clusters [22]. Based on simulating flow in a stochastic graph, this algorithm is well-suited for our purpose. Below are shown sequential steps of the algorithm (Algorithm 1):

Algorithm 1. Markov clustering algorithm

1. $t = 0$
2. while $t < N$ do:
3. $M_t = Expand(M_{t-1})$
4. $M_t = Inflate(M_t)$
5. $M_t = Prune(M_t)$
6. output: M_n as a clustering,

where:

$Expand : M_t = M_t * M_0$. where M_0 is a flow graph obtained from the initial graph;

$Inflate : M_t(m,n) = \frac{M_t(m,n)}{\sum_l M_t(l,n)}$;

$Prune : M(m,n) = 0$ if $M(m,n) < threshold$.

In this paper we used library markov_clustering [23].

3. DESIGNED NEURAL NETWORK ARCHITECTURE

We state a problem as follows:

1. Given the dataset, solve the binary classification task.
2. Train a neural network to estimate the probability of having metabolic syndrome based on features.
3. Evaluate the accuracy of the predictions.
4. Use the trained model to find groups of features that influence the outcome.

3.1. Outline

To solve the problem of searching for clusters of features that have a combined effect on the result of binary classification of the neural network, it was decided to combine the principles of BioBERT, TabNet, as well as other modern neural network approaches (transformer, ERNIE [24], etc.). This neural network, as well as transformer, is based on the attention mechanism, which allows estimating the interrelations between the elements of the input sequence.

Table 1. Clusters

Unhealthy	Healthy
Body weight, cholesterol, saturated fat, magnesium	Age, height, total fluid
Height, waist circumference, galactose	Age, height, fat mass
Waist circumference, fat mass, total fluid, isoleucine, GLA	Gender, height, waist circumference, cholesterol
Thigh circumference, fat mass, muscle mass, sport	Body weight, saturated fat, sugars, magnesium
Waist circumference, fat mass, iodine	Saturated fats, sugars, carbohydrates, folate
Fat mass, muscle mass, total fluid, isoleucine, nickel	Saturated fats, sugars, oleic acid
Fat mass, muscle mass, total fluid, isoleucine, fluoride	Saturated acids, sugars, leucine
Fat mass, muscle mass, copper	Folate, omega-3, biotin
Fat mass, muscle mass, molybdenum	Folate, omega-3, phenylalanine+tyrosine
Fat mass, muscle mass, selenium	β-carotene, phosphorus, methylalonine, no sports, BMI
Fat mass, muscle mass, chromium	
Fat mass, muscle mass, zinc	
Fat mass, muscle mass, intellectual work	
Fat mass, muscle mass, light physical work	
Total fluid, saturated fat, isoleucine, oleic acid	
Total fluid, alcohol, isoleucine	
Total fluid, isoleucine, fructose	
Total fluid, isoleucine, pyridoxine	
Total fluid, isoleucine, folate	
Total Fluid, isoleucine, cobalamin	
Total fluid, isoleucine, choline	
Total fluid, isoleucine, leucine	
Total fluid, isoleucine, α-tocopherol	
Histidine, β-carotene, BMI	
Fats, β-carotene, BMI	
β-carotene, phosphorus, BMI	
Waist circumference, monounsaturated fat, β-carotene	

Neural network operation on the real data example is shown in Fig. 8.

The developed neural network is able to work with data without preprocessing, and there are several strategies to work with different types of data. The vector of each element of the input sequence at the output of the pretrained network is unique and is conditioned by its context (values in the columns of Table 1).

This architecture fulfills all the above requirements and consists of the following parts:

1. Embedding layer;
2. Encoder transformer, built on masked self-attention.

The neural network has the ability to pretrain unsupervised and to pretrain for a specific application.

3.2. Embedding

In this sector, the embedding layer developed during the research will be discussed. The general scheme of embedding layer operation is presented below in Fig. 9.

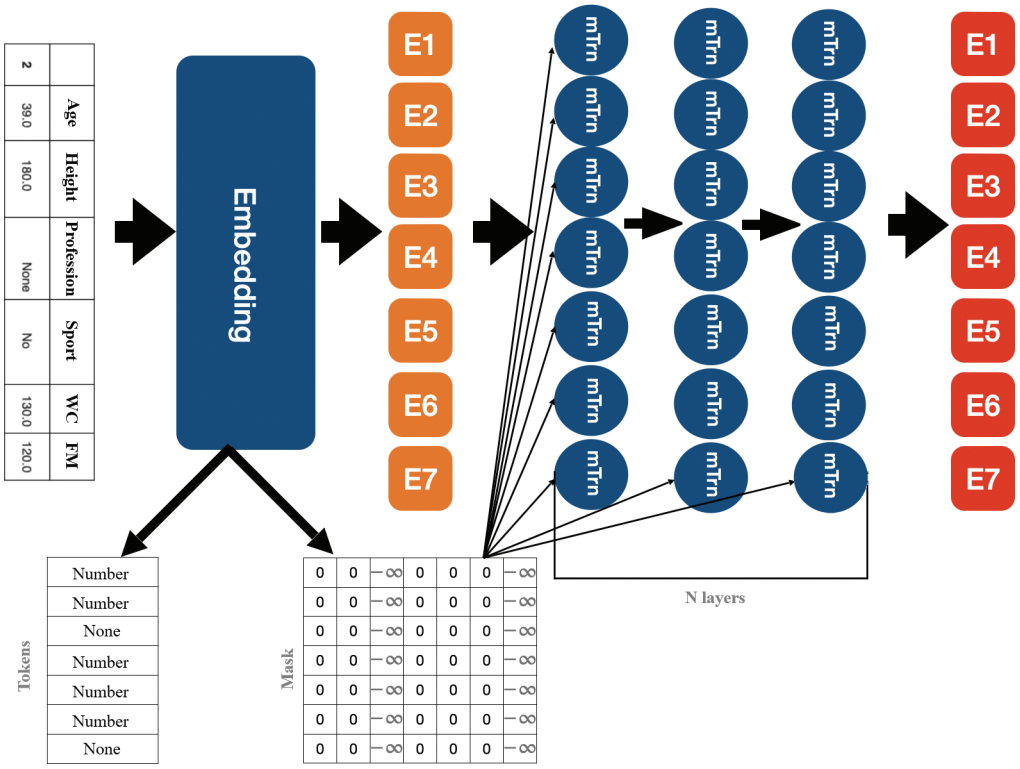

Fig. 8. Diagram of neural network operation on the example of real data.

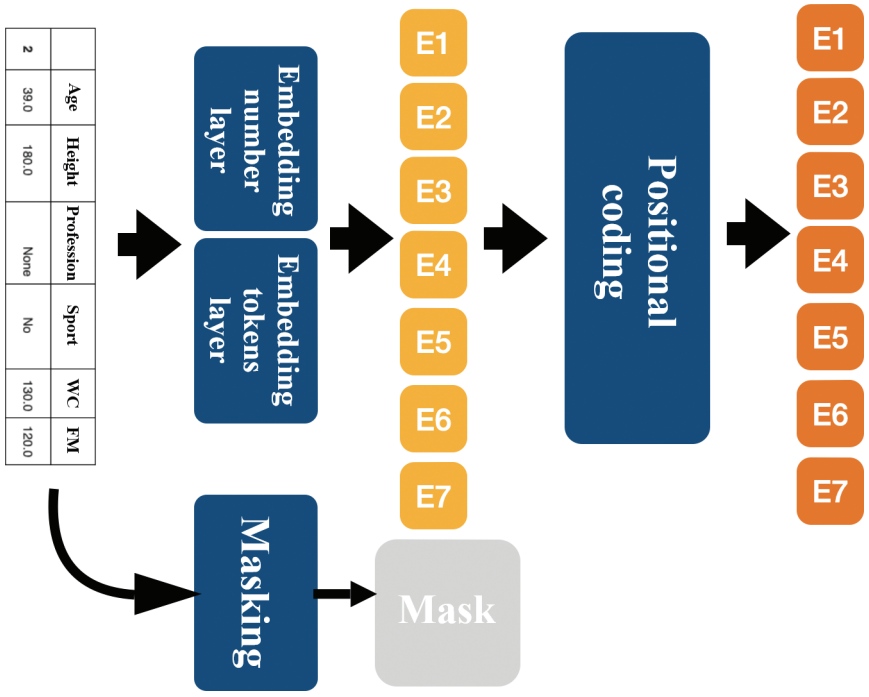

Fig. 9. Embedding operation diagram.

3.2.1. Vectorization of input sequence elements. A sequence of vectors must be fed to the encoder transformer input. This is necessary due to the principle of classical attention. This chapter describes how these elements appear.

There are different variants of tokenization of table values. There are works in which feature values are combined. Such combinations are considered as input elements [25], works in which a text query is composed from the table values and then tokeniza-

tion of this query [26] takes place. In my work, we consider the table values going in the order of their corresponding columns as input elements.

The basis vectors of elements can be obtained by various methods, some of them are summarized below:

1. If each element of the input sequence is an integer and the set of all possible such elements is bounded by $S = \{x_1, ..., x_m\}, \forall x_i \in X \subset \mathbb{N}, |X| < \infty$. The elements of a given sequence are called tokens. Each of these tokens corresponds to a trainable vector of size d_{model}. In transformer-based models, it is common to use this method of obtaining initial vectors.

2. In the case when the members of the input sequence are arbitrary numbers from $\mathbb{R}$, there are no established approaches to obtain their vectors. However, there are many such approaches and one of them will be described below.

During the course of the study, it was decided to use both of these approaches. This decision is due to the fact that table elements can take both real and categorical values.

3.2.1.1. Processing of elements accepting real values. The algorithm for obtaining vectors for real numbers is based on the PAF (periodic activation functions) method of the recent paper [27]. In the above article, the authors showed that their approach performs well on a number of applications (including different transformer architectures).

The algorithm is as follows:

1. For each column of the table, a vector $z = [z_1, z_2, ..., z_{\frac{d_{\text{model}}}{2}}]$ is generated, where z_i is initialized from $\mathcal{N}(0, \sigma)$. It is necessary that d_{model} is divisible by two, and σ is a hyperparameter of the model (it takes the value 1 in all measurements). The vectors z are trainable.

2. If element x_{col} from column "col" has a real value, the vector corresponding to it will be calculated as follows: $\text{vec}(x_{\text{col}}) = [\sin(v_{x_{\text{col}}}), \cos(v_{x_{\text{col}}}))]$, where $v_{x_{\text{col}}} = [2\pi z_i x_{\text{col}}, ..., 2\pi z_{\frac{d_{\text{model}}}{2}} x_{\text{col}}]$.

This approach showed good quality in the task of biomedical data analysis, but it was refined. Since it was customary to process not only real features but also categorical features in tables, we can consider that if an element $x_{\text{col}} \in \mathbb{R}$, then it belongs to the category of real numbers. From the above, the final formula for calculating the vector for x_{col} is as follows:

$$\text{vec}(x_{\text{col}}) = [\sin(v_{x_{\text{col}}}), \cos(v_{x_{\text{col}}}))] + m_{\mathbb{R}},$$

where $m \in \mathbb{R}^{d_{\text{model}}}$ is the trainable vector and is the same for all real-valued elements. This method of handling real-valued elements improved the performance of the model.

3.2.1.2. Processing of elements accepting categorical values. In case x_{col} takes the value of some category "cat," the method for obtaining its vector is fairly standard for models based on the transformer architecture and is as follows:

$$\text{vec}(x_{\text{col}}) = m_{\text{cat}}, \quad \text{where} \quad m_{\text{cat}} \in \mathbb{R}^{d_{\text{model}}}$$

is trainable and singular for a given category.

There were too many categorical data in the table with patient history data and for simplicity, 4 categories were taken in this experiment:

1. {None} Category—this category is assigned to x_{col} when it takes the value None and is not masked.

2. {not None} Category—this category is assigned to x_{col} when it takes the value None, [CLS] and is not masked.

3. {MASK} Category—this category is assigned to x_{col} when it is decided to be masked.

4. {CLS} Category—this given category is assigned to x_{col} when it is equal to [CLS].

Categorical data are transformed using the label encoding method [28].

3.2.1.3. Position encoding. The neural network receives as input a sequence of elements that are the values of a row of the table $X = \{x_{\text{col}_1}, .., x_{\text{col}_n}\}$ Since each element of a given neural network architecture interacts with a particular element of the sequence at a given moment—it is necessary to convey information about what position in the sequence it is at, in order for the network to understand what that element is. For example, the category "often" can be either in the "sports" column or in the "overeating" column (the data may not match the values and column names of real anamneses and are given as an example only).

It was decided to use the following positional encoding [12] for element $x_{\text{col}_{\text{pos}}}$:

$$PE_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right),$$

$$PE_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right).$$

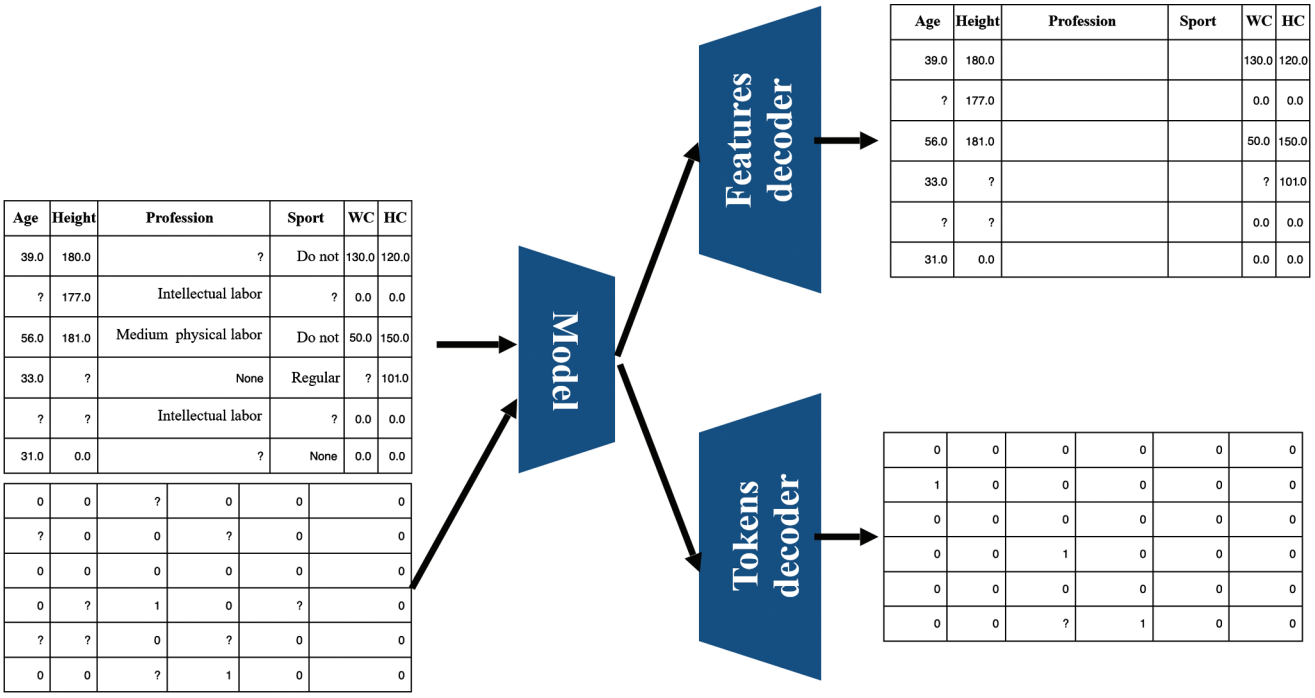

Fig. 10. Self-learning diagram of the developed neural network.

This vector is then summed with the other vector $x_{\mathrm{col}_{\mathrm{pos}}}$ described above.

This encoding allows us to understand at which position the input vector is located due to linear transformation. Also this type of encoding is used in a large number of transformer based models.

3.2.1.4. Masking. Sometimes it is useful to make certain elements of a sequence not "see" others when attention is applied. This becomes useful when, for example, there are too many gaps in a patient's history and we do not want the only values we have to be blurred. This can happen if the patient histories that were provided to the neural network for training were "almost complete," which is possible because they were given after the diagnosis had been made.

To prevent the above situations from leading to incorrect model performance, we can use masked self-attention with the formula $X_{\mathrm{new}} = softmax\left(\frac{QK^T}{\sqrt{d_k}} + MASK\right)V$, where

$$MASK_{i,j} = \begin{cases} -\infty, & x_{\mathrm{col}_j} = \mathrm{None} \\ 0, & \text{otherwise.} \end{cases}$$

3.2.1.5. Encoder transformer based on masked self-attention. The idea of using encoder transformer is based on the idea of BERT. By means of elaborate self-training it is possible to make a bidirectional model that understands the context well. In contrast to the BERT approach, it was decided to use masked self-attention rather than the usual self-attention. The MASK generated for an element on the embedding layer is passed to the attention of each encoder layer.

The self-learning method was developed from the self-learning principles of TabNet and BioBERT. Self-learning diagram is shown in Fig. 10.

This principle is as follows:

1. Among the input elements $X = \{x_1, x_2, ..., x_n\}$ with a given probability p(is a hyperparameter of the model), the elements that are masked for the model are selected. Masking means replacing the value of an element with [MASK]. Elements with [MASK] value form a separate category and in fact the original values of their vectors are replaced by $m_{[\mathrm{mask}]} + pos(x_i)$. Elements with the value [CLS] are not masked.

2. After the above "masking" operation, the obtained sequence $X = \{x_1, x_2, ..., x_n\}$ is fed to the model input.

3. The sequence transformed by the model enters both the feature decoder and the token decoder. The feature decoder is a simple fully connected neural network with one output, and the token decoder is also a simple network with a number of outputs equal to the number of categories (not counting the MASK and CLS categories).

Age	Height	Profession	Sport	WC	HC
39.0	180.0	Intellectual labor	Do not	130.0	120.0
45.0	177.0	Intellectual labor	Regular	0.0	0.0
56.0	181.0	Medium physical labor	Do not	50.0	150.0
33.0	184.0	Medium physical labor	Regular	106.0	101.0
35.0	182.0	Intellectual labor	Rarely	0.0	0.0
31.0	0.0	Medium physical labor	Do not	0.0	0.0

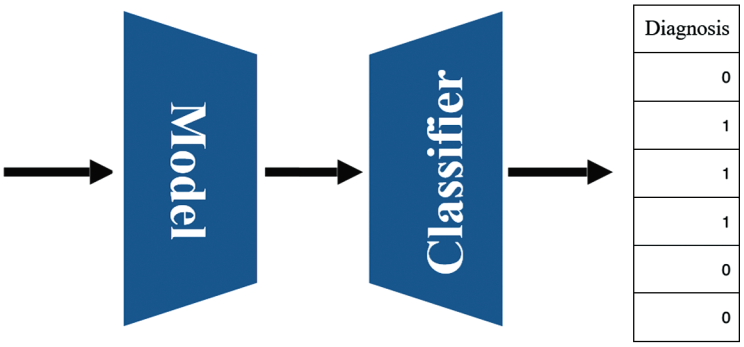

Fig. 11. Training diagram of the developed neural network.

Only those elements whose values were real before masking get into the token decoder.

4. The outputs from the decoders are $MSELoss$ and $LogLoss$, by which $Loss = aMSELoss + (1-a)LogLoss$ is calculated, where $a \in [0,1]$ is a hyperparameter of the network.

5. This error is used to calculate the gradient along which the parameters of the neural network are "descended."

6. If the stopping criterion is not reached, steps (1)–(5) are repeated. The stopping criterion can be the number of iterations or restrictions on the value of validation metrics.

3.2.1.6. Training. Training diagram of the developed neural network is shown in Fig. 11.

The pretraining of the model corresponds to the pretraining of TabNet and BioBERT neural networks. It takes place according to the following principle:

1. The input sequence $X = \{x_1, x_2, ..., x_n\}$ enters the model, which outputs a sequence of vectors of dimension $d_{\text{model}}, E = \{e_1, e_2, ..., ..., e_n\}$.

2. The vectors of the output sequence are used to calculate the total vector of the sequence. If an element with the value [CLS] is added at the beginning, the total vector is $e_{[CLS]}$, otherwise the total vector is $e_{[CLS]}$ averaged over all e_i vectors. In this problem, the quality of the model was higher in the case with averaging of output vectors.

3. The total vector falls into a simple fully connected network.

4. The error is calculated from the output value based on the problem (LogLoss for classification and MSELoss for regression).

5. This error is used to calculate the gradient along which the parameters of the neural network (model and subsequent fully connected network) are "descended."

6. If the stopping criterion is not reached, steps (1)–(5) are repeated. The stopping criterion can be the number of iterations or restrictions on the value of validation metrics.

4. RESULTS

Due to light class imbalance we were able to use accuracy metric, calculated as follows:

1. Accuracy $= \frac{TP+TN}{TP+TN+TN+FP+FN}$.

2. TP—the number of correct diagnoses with metabolic syndrome.

3. FP—number of times the trained model classified there was a syndrome, but there was not.

4. TN—the number of correctly found diagnoses with no metabolic syndrome.

5. FN—number of times the trained model thought there was no syndrome, but there was.

Designed neural network for solving binary classification problem was successfully trained and showed following metric on test set:

$Accuracy = 0.72$;
$Precision = 0.73$;
$Recall = 0.69$.

The convergence plots of the model for pretraining and the classification task are shown in Figs. 12 and 13.

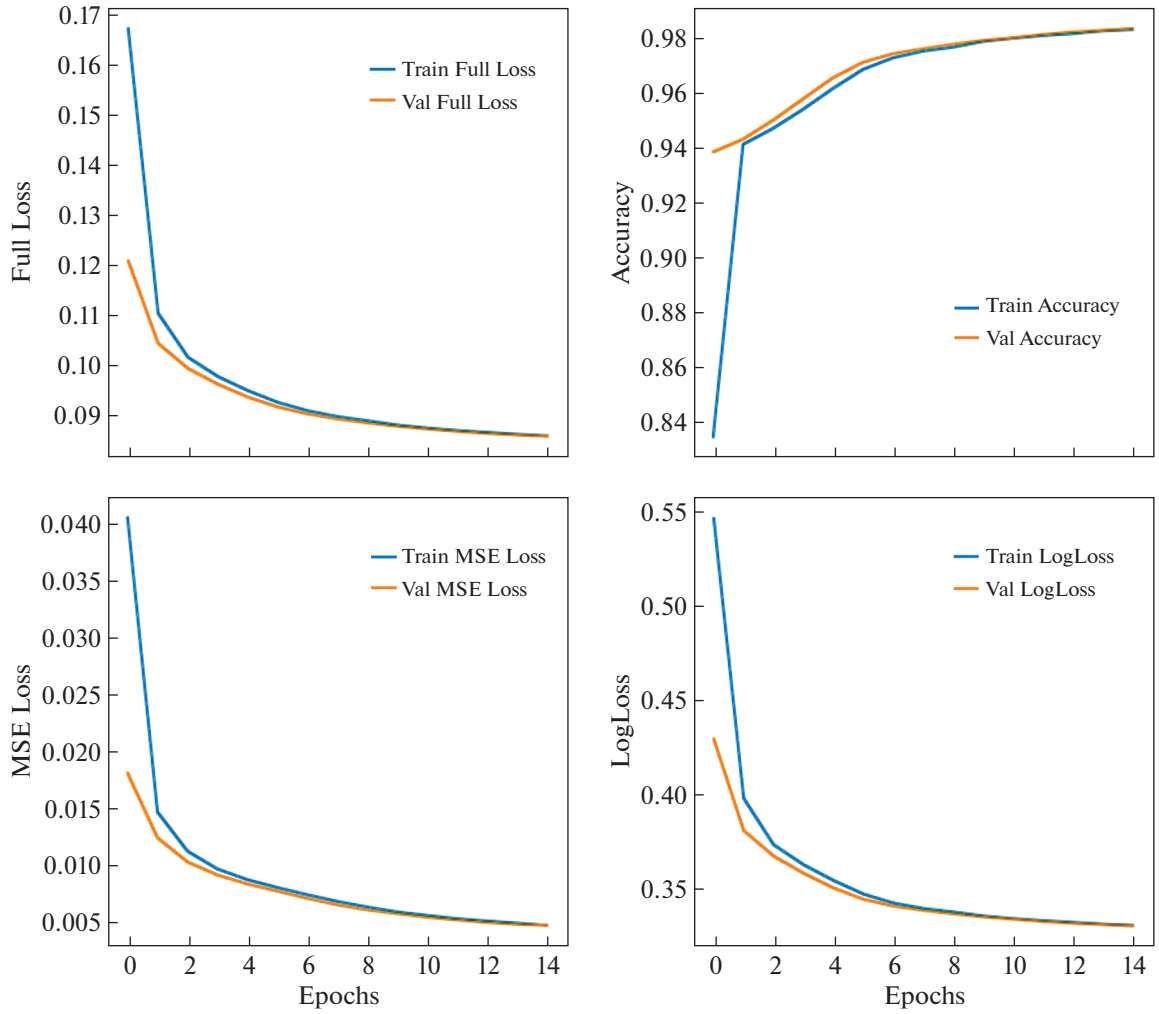

Fig. 12. Retraining convergence plot.

After model training and validating we get attention-layer matrix as follows:

for each example from training set we run the model and get the first calculated attention-layer from the model. Each attention layer is a 85×85 matrix, where 85 is a number of features. In the next step each value in the matrix was averaged and on the gained matrix markov clustering algorithm was run.

Obtained from attention-layer clusters of features are listed in Table 1.

5. DISCUSSION

Biomedical analysis of the obtained results allows us to outline some new directions in risk stratification of MS development in patients, which require further verification studies.

Thus, the results of high significance of the amount of fat mass and body fluid in combination with age and height look extremely interesting. Age is a known independent risk factor for metabolic disorders, while height is a key component of body surface area calculation. The amount of fat mass related to height and age can be studied in terms of a new prognostic index, an increase in which possibly more accurately determines the risk of metabolic syndrome than the degree of obesity or BMI. No less interesting looks the significance of the "total fluid—height—age cluster," the increase of which may indicate the negative prognostic significance of this index in relation to MS.

It is known that such a widely discussed nutritional risk factor for MS as dietary cholesterol intake demonstrates a very unbalanced value in different studies and patient samples—from minimal to highly significant correlation with MS. Our findings suggest that cholesterol intake may be most effectively assessed not independently, but in combination with sex, height, and waist circumference. Thus, increased cholesterol intake may be most important in men with abdominal obesity (waist circumference to height).

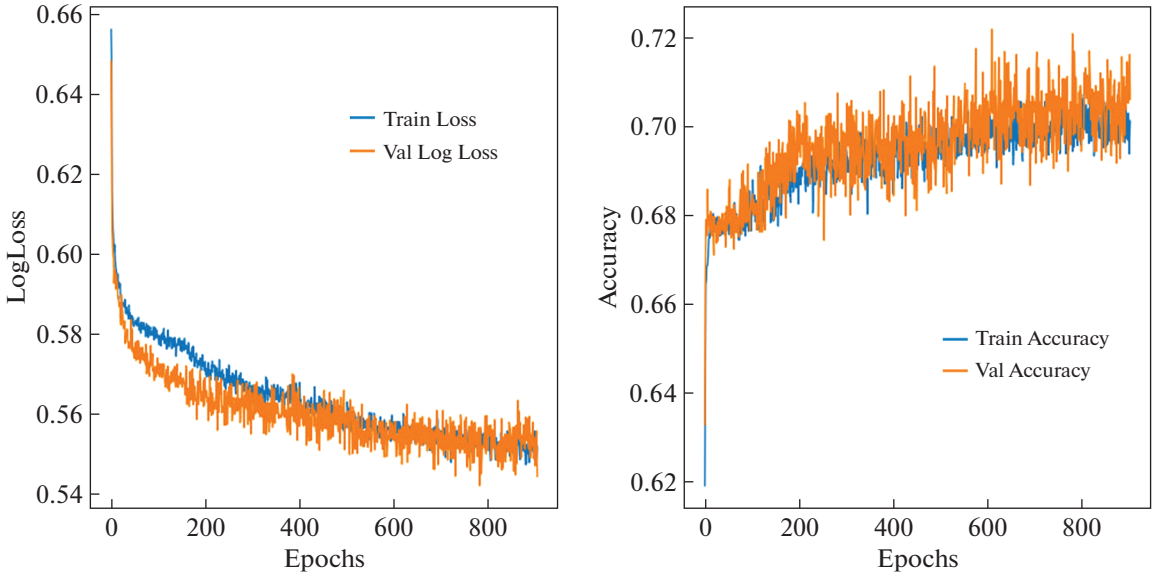

Fig. 13. Classification task convergence plot.

The results of the study also showed that excessive intake of saturated fats and sugar combined with a lack of magnesium intake in overweight individuals has an independent prognostic value.

Finally, the study allowed us to identify several significant patterns of consumption of individual nutrients: "LDL, sugar, carbohydrates, folic acid," "LDL, sugar, leucine," "folic acid, omega 3, biotin," and "folic acid, omega 3, phenylalanine + tyrosine." The importance of each of the presented nutrients in relation to MS is known to some extent, so the combination of their joint influence requires additional study. The significance of beta-carotene, phosphorus and methylalanine intake in individuals with low physical activity and increased BMI looks quite interesting. The identified patterns can be proposed to identify directions for further research in this area.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. Y. Tian, J. Ma, Q. Gong, S. Sengupta, Z. Chen, J. Pinkerton, and L. Zitnick, Proceedings of Machine Learning Research **97**, 6244 (2019). https://proceedings.mlr.press/v97/tian19a.html.
2. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, arXiv Preprint (2023). https://doi.org/10.48550/arXiv.1706.03762
3. N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, Proceedings of Machine Learning Research **80**, 4055 (2018). https://proceedings.mlr.press/v80/parmar18a.html.
4. D. Bahdanau, K. Cho, and Y. Bengio, arXiv Preprint (2014). https://doi.org/10.48550/arXiv.1409.0473
5. S. Vashishth, S. Upadhyay, G. Singh Tomar, and M. Faruqui, arXiv Preprint (2019). https://doi.org/10.48550/arXiv.1909.11218
6. Questionnaire testing the balance of nutrition. https://nutribalance24.ru/startqr.html?organisation_key=9536b0e4-218f-4f81-9fb7-cd1c9388da6d.
7. Nutrilogic. https://go.nutrilogic.ru/.
8. J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, Ch. H. So, and J. Kang, Bioinformatics **36**, 1234 (2019). https://doi.org/10.1093/bioinformatics/btz682
9. J. Schmidhuber, Neural Networks **61**, 85 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
10. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, arXiv Preprint (2020). https://doi.org/10.48550/arXiv.2010.11929
11. Z. Liuand, Y. Lin, H. Cao, Y. Hu, Z. Wei, S. Zhang, B. Lin, and B. Guo, arXiv Preprint (2021). https://doi.org/10.48550/arXiv.2103.14030

12. J. Gehring, M. Auli, D. Grangier, D. Yarats, and Ya. N. Dauphin, Proceedings of Machine Learning Research **70**, 1243 (2017). https://proceedings.mlr.press/v70/gehring17a.html.

13. K. He, X. Zhang, Sh. Ren, and J. Sun, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016* (IEEE, 2016), p. 770. https://doi.org/10.1109/cvpr.2016.90

14. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, 2019*, Ed. by J. Burstein, Ch. Doran, and T. Solorio (Association for Computational Linguistics, 2019), Vol. 1, p. 4171. https://doi.org/10.18653/v1/N19-1423

15. N. Reimers and I. Gurevych, arXiv Preprint (2019). https://doi.org/10.48550/arXiv.1908.10084

16. T. Chen and C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 2016* (Association for Computing Machinery, New York, 2016), p. 785. https://doi.org/10.1145/2939672.2939785

17. A. V. Dorogush, A. Gulin, G. Gusev, N. Kazeev, L. Ostroumova-Prokhorenkova, and A. Vorobev, in *Advances in Neural Information Processing Systems*, Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, 2018), Vol. 31. https://proceedings.neurips.cc/paper_files/paper/2018/file/ 14491b756b3a51daac41c2486328-5549-Paper.pdf.

18. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, in *Advances in Neural Information Processing Systems*, Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, 2017), Vol. 30, p. 3149. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

19. S. Ö. Arik and T. Pfister, Proceedings of the AAAI Conference on Artificial Intelligence **35**, 6679 (2019). https://doi.org/10.1609/aaai.v35i8.16826

20. S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

21. S. van Dongen, SIAM J. Matrix Anal. Appl. **30**, 121 (2008). https://doi.org/10.1137/040608635

22. L. Zichuan, G. Lin, S. Yang, J. Feng, W. Lin, and W. L. Goh, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018* (IEEE, 2018), p. 6936. https://doi.org/10.1109/CVPR.2018.00725

23. Markov clustering in python. https://github.com/GuyAllard/markov_clustering.

24. Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, 2019, Ed. by A. Korhonen, D. Traum, and L. Màrquez, Association for Computational Linguistics, 2019, pp. 1441–1451. https://doi.org/10.18653/v1/P19-1139

25. P. Yin, G. Neubig, W.-T. Yih, and S. Riedel, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault (Association for Computational Linguistics, 2020), p. 8413. https://doi.org/10.18653/v1/2020.acl-main.745

26. Z. Chen, M. Trabelsi, J. Heflin, Y. Xu, and B. D. Davison, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, New York, 2020), p. 589. https://doi.org/10.1145/3397271.3401044

27. Y. Gorishniy, I. Rubachev, and A. Babenko, arXiv Preprint (2023). https://doi.org/10.48550/arXiv.2203.05556

28. L. Yu, R. Zhou, R. Chen, and K. K. Lai, Emerging Markets Finance and Trade **58**, 472 (2022). https://doi.org/10.1080/1540496x.2020.1825935