Научная статья

DOI: 10.15593/2224-9397/2022.1.06
УДК 004.89

В.В. Бахтин

Пермский национальный исследовательский политехнический университет,
Пермь, Россия

АЛГОРИТМ РАЗДЕЛЕНИЯ МОНОЛИТНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ РЕАЛИЗАЦИИ ТУМАННЫХ ВЫЧИСЛЕНИЙ В УСТРОЙСТВАХ НА ПРОГРАММИРУЕМОЙ ЛОГИКЕ

В проектах, где нет возможности использовать нейронную сеть в рамках одного устройства, целесообразно применять метод синтеза каскада из устройств, который бы позволял реализовать распределенную (блочную) нейронную сеть. **Целью исследования** является разработка метода синтеза устройств реализации искусственных нейронных сетей на программируемой логике, ориентированных на туманные вычисления. Основой для создания рассматриваемых устройств выступает искусственная нейронная сеть, которую требуется разделить на несколько блоков. Каждый из этих вычислительных блоков исполняется на отдельном физическом устройстве. Связь между блоками осуществляется с помощью стандартных каналов и протоколов. **Методика исследования** базируется на математическом моделировании нейронной сети, которая будет пригодной для работы в режиме туманных вычислений, методах алгоритмизации и программирования, которые позволят реализовать необходимые алгоритмические структуры. В **результате исследования** формируется алгоритм разделения монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений в устройствах на программируемой логике, который позволит реализовать метод синтеза устройств нейросетевого распознавания. В статье рассмотрена математическая модель, ставшая отправной точкой для алгоритмов. Сформулирован и реализован алгоритм: разделения монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений в устройствах на программируемой логике и вычисления результатов работы нейронной сети в новом формате, адаптированный для туманных вычислений. Предложен оригинальный формат хранения нейронной сети, созданный для предлагаемых алгоритмов.

Ключевые слова: алгоритм, искусственная нейронная сеть, программируемая логика, туманные вычисления, принятие решений, устройство нейросетевого распознавания, метод синтеза, математическая модель.

V.V. Bakhtin

Perm National Research Polytechnic University, Perm, Russian Federation

ALGORITHM FOR DECOMPOSITION OF A MONOLITHIC NEURAL NETWORK FOR IMPLEMENTED FOG COMPUTING IN DEVICES BASED ON PROGRAMMABLE LOGIC

In projects where it is not possible to use a neural network within a single device, a method of synthesizing a cascade of devices that would implement a distributed (block) neural network would be useful. The aim of the study is to develop a method for synthesizing devices for implementing artificial neural networks based on programmable logic, focused on fog computing. The basis for the creation of the devices in question will be an artificial neural network, which will need to be divided into several blocks. Each of these computing units will be executed on a separate physical device, communication between them will be carried out using standard channels and protocols. The research methodology is based on mathematical modelling of a neural network that will be suitable for operation in the mode of foggy calculations, algorithmization and programming methods that will allow implementing the necessary algorithmic structures. As a result of the research, it is planned to obtain an algorithm for dividing a monolithic neural network into a cascade of neural network blocks adapted for fog computing in devices based on programmable logic. This algorithm will allow the implementation of a method for synthesizing neural network recognition devices. The article considers a mathematical model as the starting point for algorithms. The following algorithms are formulated and implemented: the separation of a monolithic neural network into a cascade of neural network blocks adapted for fog computing in devices based on programmable logic and the calculation of the neural network results in a new format adapted for fog computing. The original neural network storage format created for the proposed algorithms is proposed.

Keywords: algorithm; artificial neural network; FPGA; microcontrollers; fog computing; decision making; neural network recognition device; synthesis method, programmable logic, mathematical model.

Введение

Исследования в области создания нейронных сетей, ориентированных на туманные вычисления, сегодня имеют высокую актуальность, в различных научных коллективах мира ведутся разработки в данном направлении. Проблемы создания распределенных НС исследуются в работах Bradley McDanel, Surat Teerapittayanon, H.T. Kung [1], Andrzej Sobecki, Julian Szymanski, David Gil [2], Yuchen Liang, W.D. Li, Zhi-Cong Chen [3] и других современных ученых. Однако на текущий момент по-прежнему не решена задача использования нейронных сетей на нескольких физически распределенных устройствах в ситуации небольших вычислительных ресурсов каждого из одиночных устройств. Это создает предпосылки для создания нового поколения туманных вычислений, которые поддерживают искусственный интеллект и используют архитектуру, подходящую для интеллектуальных реше-

ний, с малыми требованиями к вычислительной мощности отдельных устройств. В **результате исследования** планируется получить метод декомпозиции нейронной сети на блоки, которые будут работать на оконечных устройствах, также будет проведено испытание данного метода на тестовом каскаде вычислительных устройств. Данный подход важен тем, что позволяет отойти от оптимизации сети в пользу разделения вычислительной нагрузки между устройствами [4].

У проблемы высокой ресурсоемкости нейросетевых вычислений могут быть различные решения [5], которые должны учитывать также влияние на окружающую среду [6]. Возможным решением данной проблемы, которое мы хотели бы предложить в данной работе, является разделение монолитной нейронной сети на каскад блоков, последовательно выполняемых на связанных между собой вычислителях [7]. Раскроем предполагаемое решение в терминах предметной области.

Нейронную сеть, все слои которой производят свои вычисления на одном и том же вычислительном устройстве, назовем **монолитной нейронной сетью**. Если перед нами стоит задача получения нейронной сети, которая может проводить свои вычисления на нескольких связанных между собой устройствах, то нам потребуется разделить слои этой нейронной сети на блоки последовательных, идущих друг за другом слоев. Каждый из этих блоков будет выполнен на отдельном вычислительном устройстве, а промежуточные результаты будут переданы по сети между ними. Нейронную сеть, разбитую на набор подобных блоков, назовем **блочной нейронной сетью**. Именно метод преобразования монолитной нейронной сети к виду блочной нейронной сети, адаптированной для выполнения туманных вычислений [8], и является ключом к созданию каскадов из нескольких не очень мощных вычислительных устройств на программируемой логике, которые будут сопоставимы с более производительными вычислительными устройствами и будут иметь достаточную производительность для работы нейронных сетей.

Подробное описание сущности и методов функционирования облачных и туманных вычислений представлены в работах [9–11]. Туманные вычисления – это метод распределения вычислительных задач в виде небольших блоков на небольшие устройства, которые обрабатывают информацию в процессе ее передачи от отправителя к получателю [12]. Именно на такие блоки предлагается разбивать монолитную

нейронную сеть, чтобы выполнение ее вычислений стало возможным на небольших промежуточных устройствах. Это позволит осуществлять нейросетевые вычисления на нескольких вычислительных устройствах небольшой мощности, а значит, использовать нейронные сети в системах управления или принятия решений, которые ограничены в вычислительных мощностях [13].

Исходя из представленных предположений, можно сформулировать **цель представленной работы**: разработка метода синтеза устройств реализации искусственных нейронных сетей на ПЛИС и микроконтроллерах, ориентированных на туманные вычисления. В предыдущей статье была сформулирована математическая модель искусственной нейронной сети для устройств на плис и микроконтроллерах, ориентированных на туманные вычисления [1], т.е. была решена первая задача исследования. В данной статье будут сформулированы два алгоритма: алгоритм разделения монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений в устройствах на программируемой логике, главными особенностями которого являются учет входных параметров: мощности устройств, оптимального объема передаваемых между устройствами данных, пропорциональности по слоям или по нейронам. А также алгоритм вычисления результатов работы нейронной сети в новом формате, адаптированный для туманных вычислений в устройствах на программируемой логике, главными особенностями которого являются возможность выполнения алгоритма как на программируемой логике, так и на процессорных системах и встроенный словарь функций активации нейрона, который может быть расширен новыми функциями, если этого потребует задача. Предлагаемые алгоритмы являются следующим шагом к созданию оригинального метода синтеза устройств нейросетевого распознавания.

1. Математическая модель искусственной нейронной сети, ориентированной на туманные вычисления

В данной статье описывается работа с уже обученными нейронными сетями, т.е. к моменту разделения на блочные нейронные сети веса синапсов уже будут известны и зафиксированы [14]. Как было отмечено ранее, при построении предсказательных моделей исходные данные обычно разбиваются на обучающую и контрольную выборки.

Обучающая выборка используется для обучения модели, тогда как контрольная выборка служит для получения оценки прогнозных свойств модели на новых данных [15].

В описываемом исследовании важно, чтобы к моменту синтеза устройства нейросетевого распознавания обучение было завершено и веса синапсов стабилизированы, это ограничение временное и его можно преодолеть в будущем. Также в приведенном исследовании рассматриваются именно многослойные нейронные сети, так как в полных нейронных сетях слишком много связей и их сложно разделить на независимые кластеры, которые могли бы выполняться на различных физических устройствах. В данной статье будет рассматриваться математическая модель разделения нейронной сети без обратных связей, в будущем в рамках исследования возможна доработка, которая позволит разделять сети с обратными связями.

Дано: монолитная многослойная нейронная сеть X, результат работы которой – последовательность сигналов $\left\{y_0^K,...,y_{H_K}^K\right\}$.

Найти: последовательность нейронных сетей $\left\{\bar{X}_0,...,\bar{X}_{D-1}\right\}$, где результат вычисления $\bar{X}_1 \rightleftarrows \bar{X}_2 \rightleftarrows ... \rightleftarrows \bar{X}_{D-1}$ совпадает с результатом работы сети X.

Процесс преобразования монолитной ИНС в каскад блочных ИНС, результат вычисления которого совпадает с результатами монолитной нейронной сети, назовем **декомпозицией** искусственной нейронной сети. Начнем создание математической модели искусственной нейронной сети, ориентированной на туманные вычисления, с того, что воспользуемся общепринятыми определениями функционирования нейронных сетей. Это и станет отправной точкой наших изысканий (рисунок).

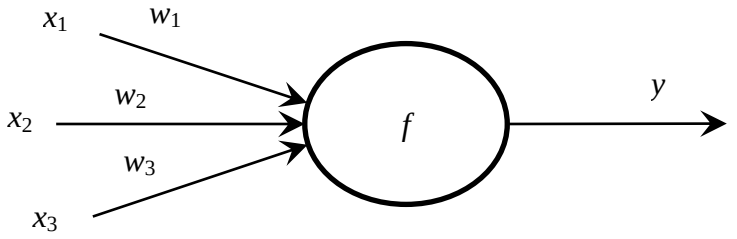

Рис. Математический нейрон

$$y_j = \sum_{i=1}^{N} x_i w_{ij}\,, \qquad (1)$$

где x_i – вход синапса, y_j – выход синапса, w_{ij} – вес синапса [16].

Тогда функция работы ИНС на примере многослойного персептрона:

$$\begin{cases} y_i^{(k)} = f_i^{(k)}\left(\sum_{j=0}^{H_{k-1}} w_{ij}^{(k)} \cdot y_i^{(k-1)}\right), \\ y_i^{(0)} = x_i^{(0)}, \end{cases} \tag{2}$$

где $x_i^{(0)}$ – входные ИНС, $y_i^{(k)}$ – выход i-го нейрона k-го слоя, H_{k-1} – число нейронов на данные слое $k-1$, $f_i^{(k)}$ – функция активации нейрона [17].

Разделение на подсети может осуществляться различными способами. Рассмотрим простой пример деления монолитной нейронной сети на блочные нейронные сети, когда в каждой блочной нейронной сети содержится одинаковое количество слоев.

Ответ: общая рекуррентная формула описания каждой из полученных блочных нейронных сетей, где массив $\{L_0,...,L_{D-1}\}$ – количество слоев нейронной сети, которые должны быть переданы на устройство с номером N:

$$\bar{X}_N : \begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right), N = D-1, \\ y_i^{\left(\sum_{k=0}^{N} L_k\right)} = f_i^{\left(\sum_{k=0}^{N} L_k\right)}\left(\sum_{j=0}^{H_{\sum_{k=0}^{N} L_k - 1}} w_{ij}^{\left(\sum_{k=0}^{N} L_k\right)} \cdot y_i^{\left(\sum_{k=0}^{N} L_k - 1\right)}\right), N < D-1, \end{cases} \\ \dots \\ \begin{cases} y_i^{(L_{N-1}+1)} = y_i^{(L_{N-1})}, N > 0, \\ y_i^{(0)} = x_i^{(0)}, N = 0, \end{cases} \end{cases} \tag{3}$$

где D – число устройств, на которые размещаются блоки, K – общее число слоев монолитной нейронной сети.

Существуют различные варианты входных параметров, в зависимости от которых будут реализовываться конкретные способы разделения монолитной ИНС на части:

1) Разделение на блоки с равным количеством слоев на узле.

Для самого частого случая, когда K не кратно D, количество слоев L_i, которые должны быть переданы на устройство D_i, вычисляется по формуле:

$$L_i = \left| \frac{K}{D} \right|. \quad (4)$$

Поскольку применяется округление в меньшую сторону для получения целого числа, количество слоев нейронной сети, которое будет на последнем вычислителе с номером $D - 1$, вычисляется по формуле:

$$L_{D-1} = K - (D-1)L_0. \quad (5)$$

2) Разделение пропорционально производительности устройств (слои). Для разделения пропорционально производительности (по количеству слоев нейронной сети) нужно сделать массив мощностей $\{P_0,...,P_{D-1}\}$, выраженных в некоторых абсолютных единицах (результатах тестов производительности, тактовых частотах, объемах оперативной памяти), и распределить число слоев пропорционально значениям массива.

3) Разделение на блоки с равным количеством нейронов на узле. Берем сумму всех нейронов на всех слоях монолитной нейронной сети. На каждом слое всего H_b нейронов, тогда желаемое число нейронов в каждой из блочных ИНС будет равно сумме всех нейронов на слоях, деленной на число устройств.

4) Разделение пропорционально производительности устройств (нейроны) осуществляется модификацией метода выше для разделения по числу нейронов, а не слоев нейронной сети.

5) Разделение с условием минимизации передаваемых по сети данных осуществляется с учетом нахождения минимальных по размеру векторов данных, которые потребуется передать между двумя блоками нейронной сети.

2. Формат файла, описывающего нейронную сеть ANN

Для работы с нейронными сетями в унифицированном виде на устройствах с программируемой логикой потребовалось создать несложный формат хранения нейронной сети в файле, чтобы иметь возможность читать и записывать нейронные сети при разделении на блоки из постоянной памяти вычислительного устройства. Формат файлов

получил расширение .ann – сокращенное от artificial neural network. Нейронная сеть в этом файле хранится в следующем формате:

– в первой строке единственное целое число – количество слоев в нейронной сети K;

– далее в каждой строке описывается отдельный слой нейронной сети. В описании слоя части разделяются литерой «,». Сначала идут два целых числа – порядковый номер слоя и число нейронов на слое, затем через разделитель располагаются описания каждого из нейронов;

– в описании нейрона части разделяются литерой «;». Описание состоит из пяти частей. Два целых числа вначале обозначают номер нейрона на слое и номер функции активации данного нейрона в справочнике функций. Далее идут три массива: массив констант для вычислений, массив весов связей и массив номеров нейронов предыдущего слоя, с которыми связан данный нейрон. Для разделения элементов массива используется символ пробела.

Описание небольшой нейронной сети в предлагаемом в статье формате представлено в примере 1.

```
3
0,3,0;12;1;1;0,1;12;1;1;1,2;12;1;1;2
1,3,0;10;1;1 1;0 1,1;10;1;1 1;1 2,2;10;1;1 1;0 2
2,1,0;11;0;1 1 1;0 1 2
```

Пример 1. Нейронная сеть, использованная при классификации терминов в программном продукте TSBuilder.

Это исходное состояние нейронной сети, которая была использована в наших предыдущих исследованиях [18]. Преимуществом такого способа хранения является его компактность, недостаток выходит из преимущества – данное представление, даже в случае небольшой нейронной сети, сложно читается человеком.

Перейдем непосредственно к рассмотрению алгоритма разделения монолитной нейронной сети на каскад блочных нейронных сетей для туманных вычислений на устройствах с программируемой логикой.

Массив констант хранится в справочнике, в нашем случае – в файле NeuronFunctions.java. Сейчас он представлен 13 функциями. Этот список функций может быть расширен. В данном файле инкапсулируется выбор функции для вычисления в конкретном нейроне.

```
public enum NeuronFunctions {
   None(999),
   Sum (0),
   Max (1),
   Sigmoida (2),
   Linear (3),
   Threshold (4),
   Or(5),
   And(6),
   Tanh (7),
   ReLu (8),
   MaxCounter(9),
   Ntwo(10),
   Nthree(11),
   Equals(12); }
```

Здесь представлены следующие функции: сумма входных параметров, максимум среди входных параметров, сигмоида, линейная функция активации нейрона, пороговая функция, булева функция «ИЛИ», булева функция «И», гиперболический тангенс, функция подсчета максимальной частоты входных параметров, эквивалентность и авторские функции активации, которые использовались нами на начальных этапах исследования в нейронной сети продукта TSBuilder [18–23].

3. Алгоритм разделения МНС на блоки БНС

Алгоритм был реализован на языке программирования Java. Возможны различные сочетания входных параметров алгоритма разделения, мы рассмотрим вариант, когда монолитная нейронная сеть, имеющая в своем составе K слоев, разделяется на D различных блочных нейронных сетей. В представленном варианте алгоритма мы ориентируемся на равномерное распределение нагрузки на все вычислительные узлы, если это возможно, т.е. мы пытаемся распределить на каждое устройство одинаковое число слоев из исходной нейронной сети. Для того чтобы алгоритм учитывал различные вычислительные мощности вычислительных устройств, достаточно поменять методику расчёта количества слоев для каждого устройства с равномерной на пропорциональную. Рассмотрим предлагаемый алгоритм разделения по шагам.

1. Создается объект класса NNetwork. Объекты этого класса описывают как монолитные, так и блочные нейронные сети. Такие классы называются унифицированными классами.

2. В созданный объект считывается исходная монолитная нейронная сеть из файла.

```
NNetwork monoNetwork =
NNetwork.ReadNNFromFile("/TSNetwork.ann");
```

3. Создается объект класса Separator, который отвечает непосредственно за разделение нейронной сети.

Separator separator = new Separator();

4. Вызывается метод splitOnEqualNumberOfLayers объекта separator, реализующий разделение на блочные нейронные сети с равномерным распределением слоев исходной нейронной сети. На вход методу передаются нейронная сеть monoNetwork и число устройств D, на которые требуется разделить нейронную сеть.

```
ArrayList<NNetwork> blockNetworks = separa-
tor.splitOnEqualNumberOfLayers(monoNetwork, D);
```

5. Дальнейшие операции производятся внутри метода splitOnEqualNumberOfLayers. Инициализируется список для хранения создаваемых блочных нейронных сетей blockNetworks.

```
ArrayList<NNetwork> blockNetworks = new
ArrayList();
```

6. Подсчитывается число слоев, которое будет передано на каждую блочную нейронную сеть, кроме последней. Для этого осуществим целочисленное деление числа слоев *K* на число устройств *D*.

```
int numberOfLayers = network.K / D;
```

7. В цикле с итератором i от 0 до D-2: создается массив слоев, которые будут переданы в очередную блочную нейронную сеть.

```
            for (int i = 0; i< D-1; i++) {
                ArrayList<Layer> layers = new
ArrayList();
                for (int j = i*numberOfLayers; j <
(i+1)*numberOfLayers; j++) {
                    lay-
ers.add(network.LayersList.get(j));
                }
                blockNetworks.add(new
NNetwork(numberOfLayers, layers));
            }
```

8. В цикле с итератором j добавляются слои, которые будут переданы в блочную сеть, номера слоев считаются, начиная с i*numberOfLayers. Окончание цикла по итератору j.

```
                for (int j = i*numberOfLayers; j < 
(i+1)*numberOfLayers; j++) {
                    lay-
ers.add(network.LayersList.get(j));
                }
```

9. Создается новая блочная нейронная сеть, ей передается число слоев и непосредственно слои монолитной нейронной сети layers. Нейронная сеть добавляется в список blockNetworks.

```
    blockNetworks.add(new NNetwork(numberOfLayers, lay-
ers));
```

10. Окончание цикла по итератору i. Подсчитывается число слоев в последней блочной нейронной сети. Оно может быть отличным от numberOfLayers и получается вычитанием числа слоев во всех предыдущих слоях из общего числа слоев K.

```
    int numberOfLastLayers = network.K - (D-
1)*numberOfLayers;
```

11. Копируются все оставшиеся в монолитной нейронной сети слои, создается последняя нейронная сеть. Ей передаются слои lastLayers, и сеть добавляется в общий список blockNetworks.

```
            ArrayList<Layer> lastLayers = new 
ArrayList();
            for (int j = (D-1)*numberOfLayers; j < net-
work.K; j++) {

lastLayers.add(network.LayersList.get(j));
            }
            blockNetworks.add(new NNetwork(network.K -
(D-1)*numberOfLayers, lastLayers));
```

12. Блочные нейронные сети полностью готовы, они сохраняются в файлы. На вход функции сохранения подаются список нейронных сетей blockNetworks и предпочитаемое название файлов.

```
    NNetwork.SaveNNToFile (blockNetworks, 
"TSNetworks").
```

Каждая нейронная сеть сохраняется на диск в отдельном файле. Название файла формируется из переданной в функцию строки, но́мера нейронной сети в каскаде и расширения .ann. Например, TSNetworks1.ann для первой блочной нейронной сети.

4. Алгоритм запуска и работы распределенной нейронной сети

Также был реализован алгоритм вычисления результатов работы нейронной сети в предложенном формате. Для начала рассмотрим алгоритм запуска и работы нейронной сети, которая получена из одного конкретного *.ann файла:

1. Получен входной вектор данных float[] nums.

2. Создается объект класса NNetwork, объекты этого класса описывают как монолитные, так и блочные нейронные сети. Такие классы называются унифицированными классами.

3. В этот объект считывается нейронная сеть из файла.

```
    NNetwork network =
NNetwork.ReadNNFromFile("/TSNetwork.ann");
```

4. Вызвать метод объекта NNetwork для вычисления результата работы нейронной сети computing.

```
    float[] result = network.computing(nums);
```

5. Дальнейшие операции производятся внутри метода computing (NNetwork). Инициализируем векторы для хранения результатов работы текущего слоя actualVector и результатов работы предыдущего слоя lastVector, изначально инициализируем его входным вектором nums, полученным на вход нейронной сети.

```
        float[] lastVector = nums;
        float[] actualVector;
```

6. В цикле, который проходит итератором по всем элементам списка слоев нейронной сети LayersList, запускаем соответствующий метод вычисления результатов слоя computing для каждого слоя layer.

```
            for (Layer layer: LayersList) {
                actualVector = lay-
er.computing(lastVector);
                lastVector = actualVector;
            }
```

7. Дальнейшие операции производятся внутри метода computing (Layer). Инициализируем результирующий вектор resultVector, каждое из значений в котором будет результатом вычисления конкретного нейрона, поэтому его размерность NeuronsCount (число нейронов на слое).

```
float[] resultVector = new float[NeuronsCount];
```

8. В цикле с итератором i от 0 до NeuronsCount-1 получаем из списка нейронов соответствующий нейрон get(i) и запускаем для нейрона метод computing.

```
            for (int i = 0; i < NeuronsCount; i++) {
                resultVector[i] =
NeuronsList.get(i).computing(lastVector);
            }
```

9. Дальнейшие операции производятся внутри метода computing (Neuron). Вызывается метод вычисления результатов для класса Functions, отвечающего за работу всех функций активации нейронов. На вход подается номер функции активации f, константа для функции c, вектор весов синапсов weightsList, вектор номеров нейронов предыдущего слоя, с которыми связан данный нейрон, sinapses и вектор результатов работы предыдущего слоя lastVector.

```
return Functions.computing(f, c, weightsList,
sinapses, lastVector);
```

10. Дальнейшие операции производятся внутри метода computing (Functions). Инициализируем переменную для результата result и вектор, который пойдет на вход функции активации нейрона. В цикле с итератором i от 0 до weightsList.length-1 умножаем вес синапса weightsList[i] на входной сигнал, пришедший по соответствующему синапсу lastVector[sinapses[i]].

```
            float result = 0;
            float[] inputVector = new
float[weightsList.length];
         for (int i = 0; i < weightsList.length; i++) {
                inputVector[i] =
weightsList[i]*lastVector[sinapses[i]];
            }
```

11. Вызываем функцию calculate, передав ей номер функции, константу и входной вектор для функции активации inputVector.

```
result = calculate(f, c, inputVector);
```

12. В функции calculate выбирается по номеру функции соответствующая реализация функции активации. Вызывается функция программного кода, которая реализует данную функцию активации с нужными ей параметрами.

```
        private static float calculate(NeuronFunctions
f, float c, float[] inputVector) {
                if (f == NeuronFunctions.Equals) {
                return inputVector[0];
            }
            if (f == NeuronFunctions.Ntwo) {
                return (float) Func-
tions.Ntwo(inputVector[0], inputVector[1]);
            }
            if (f == NeuronFunctions.Nthree) {
                return (float) Func-
tions.Nthree(inputVector[0], inputVector[1],
inputVector[2]);
            }
    …
```

13. Возвращаем result из класса Functions, далее возвращаем этот же результат из класса Neuron.

14. Затем в классе Layer этот результат попадает в результирующий вектор с соответствующим номером resultVector[i], и этот вектор возвращается в класс NNetwork.

15. После прохождения итератора по всему списку LayersList последнее значение в векторе lastVector и является результатом работы нейронной сети. Возвращаем его на уровень выше и осуществляем дальнейшую обработку полученного результата.

Описанный алгоритм работает для распределенного случая на каждом из вычислителей с тем дополнением, что между вычислителями необходимо наладить каналы связи по unicast socket, по которым будет передаваться результат работы каждого из блоков блочной нейронной сети.

Результирующий вектор первого блока станет вектором входных данных для второго блока и так далее, пока последний блок не выдаст результат работы всего каскада устройств.

5. Результаты и предлагаемые дальнейшие шаги исследования

В результате проделанной работы был сформулирован и реализован алгоритм разделения монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений в устройствах на программируемой логике. Главными особенностями этого алгоритма являются: учет входных параметров, т.е. его можно настраивать под различные задачи, а также то, что исходная и блочные нейронные сети хранятся в одинаковом формате *.ann. Данный алгоритм описан на языке программирования высокого уровня.

Вторым важным результатом является алгоритм вычисления результатов работы нейронной сети в новом формате, адаптированный для туманных вычислений в устройствах на программируемой логике. Главными особенностями данного алгоритма являются: возможность его выполнения, как на программируемой логике, так и на процессорных системах, а также встроенный словарь функций активации нейрона, который может быть расширен новыми функциями.

По итогам оценки сложности предлагаемых алгоритмов были получены следующие результаты:

– алгоритм декомпозиции МНС:

$$O(K * \max(\{H_0, \dots, H_{K-1}\}));$$

– алгоритм вычисления результатов работы НС в формате *.ann:

$$O(K * \max(\{H_0, \dots, H_{K-1}\} * \max(\{H_0, \dots, H_{K-1}\})).$$

Порядок сложности вычислений результатов блока на устройстве будет рассчитываться аналогично, с поправкой на то, что в блок войдет лишь часть слоев нейронной сети.

Поэтому оценим сложность вычислений, проводимых одним вычислительным устройством на программируемой логике:

$$O(L_i * \max(\{H_0, \dots, H_{L_i-1}\} * \max(\{H_0, \dots, H_{L_i-1}\})).$$

Первая **задача исследования** уже была решена: математическая модель, подходящая для создания метода синтеза устройств нейросетевого распознавания на программируемой логике, представлена в предыдущей статье [1]. Вторая **задача исследования** – реализация необходимых для метода синтеза алгоритмов также успешно решена, полученные алгоритмы представлены выше. Из полученных вариантов

входных параметров для алгоритма декомпозиции для непосредственного моделирования и реализации на физических устройствах была выбрана разновидность алгоритма, которая осуществляет разделение на блоки с равным количеством слоев на узле. Выбрано разделение на блоки с такими входными ограничениями по той причине, что вычислители, которые будут использованы в дальнейших экспериментах, будут иметь одинаковые параметры вычислительной мощности.

Существуют архитектуры ИНС и варианты использования блоков, которые совпадут в каскадах различных монолитных ИНС после декомпозиции, в которых последовательность блочных ИНС будет превращаться в граф вычислений [24], потенциально возможны даже асинхронные реализации подобных графов нейронных сетей [25].

Следующим шагом станет формализация метода синтеза устройств. Уже реализованы и зарегистрированы в Роспатенте программные продукты NNSplitter и NNImplementer, которые реализовывают алгоритм декомпозиции и алгоритм работы распределенной нейронной сети соответственно.

Заключение

Целью исследования являлась разработка метода синтеза устройств реализации искусственных нейронных сетей на программируемой логике, ориентированных на туманные вычисления. В **результате исследования** создан алгоритм декомпозиции монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений в устройствах на программируемой логике. Также сформулирован алгоритм вычисления результатов работы нейронной сети в новом формате, адаптированный для туманных вычислений в устройствах на программируемой логике. Оба предложенных алгоритма реализованы на языке программирования JAVA.

Научная новизна представленных результатов состоит том, что разработанные алгоритмы помогают сформировать распределенные блочные нейронные сети для использования их в устройствах на программируемой логике, а также запускать блоки нейронной сети в каскаде на физически распределенных устройствах. Дальнейшие шаги исследования завершатся формулированием полного метода синтеза устройств нейросетевого распознавания для реализации туманных вычислений и моделированием работы подобных каскадов устройств в САПР Proteus.

Определены параметры, с которыми будут осуществляться декомпозиция и тестирование работы распределенной блочной нейронной сети как при виртуальном, так и при физическом моделировании. Осуществление подобного решения задачи блочного синтеза устройств нейросетевого распознавания позволит использовать нейронные сети для решения различных задач в системах управления или диагностики в тех случаях, где использование нейронной сети в рамках одного устройства недоступно в силу низкой вычислительной мощности устройств.

Библиографический список

1. Teerapittayanon S., McDanel B., Kung H.T. Distributed deep neural networks over the cloud, the edge and end devices // IEEE 37th International Conference on Distributed Computing Systems (ICDCS). – 2017. – P. 328–339. DOI: 10.1109/ICDCS.2017.226

2. Deep learning in the fog / A. Sobecki, J. Szymański, D. Gil, H. Mora // International Journal of Distributed Sensor Networks. – 2019. – Vol. 15, iss. 8. DOI: 10.1177/1550147719867072

3. An efficient binary convolutional neural network with numerous skip connections for fog computing / L. Wu, X. Lin, Z. Chen, J. Huang, H. Liu, Y. Yang // IEEE Internet of Things Journal. – 2021. – Vol. 8, № 14. – P. 11357–11367. DOI: 10.1109/JIOT.2021.3052105

4. Бахтин В.В. Математическая модель искусственной нейронной сети для устройств на плис и микроконтроллерах, ориентированных на туманные вычисления // Вестник Пермского национального исследовательского политехнического университета. Электротехника. Информационные технологии, системы управления. – 2021. – № 40. – С. 109–129.

5. Akhmetzyanov K., Yuzhakov A. Waste sorting neural network architecture optimization // International Russian Automation Conference (RusAutoCon). – 2019. – P. 1–5. DOI: 10.1109/RUSAUTOCON.2019.8867749

6. Akhmetzyanov K.R., Yuzhakov A.A., Kokoulin A.N. Neural network development based on knowledge about environmental influence // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2020. – P. 218–221. DOI: 10.1109/EIConRus49466.2020.9039226

7. Бахтин В.В. Модификация алгоритма идентификации и категоризации научных терминов с использованием нейронной сети // Нейрокомпьютеры: разработка, применение. – 2019. – Т. 21, № 3. – С. 14–19.

8. Aazam Mohammad, Zeadally Sherali, Harras Khaled. Fog computing architecture, evaluation, and future research directions // IEEE Communications Magazine. – 2018. – № 56. – P. 46–52. DOI: 10.1109/MCOM.2018.1700707

9. Surbiryala J., Rong C. Cloud Computing: History and Overview // IEEE Cloud Summit. – 2019. – P. 1–7. DOI: 10.1109/CloudSummit47114.2019.00007

10. Zhang W., Xiao K. Communication mode of computer cluster network in cloud environment based on neural computing // 5th International Conference on Computing Methodologies and Communication (ICCMC). – 2021. – P. 48–52. DOI: 10.1109/ICCMC51019.2021.9418350

11. An application placement technique for concurrent iot applications in edge and fog computing environments / M. Goudarzi, H. Wu, M. Palaniswami, R. Buyya // IEEE Transactions on Mobile Computing. – 2021. – Vol. 20, № 4. – P. 1298–1311. DOI: 10.1109/TMC.2020.2967041

12. Fog computing: a comprehensive architectural survey / P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, A. Leon-Garcia // IEEE Access. – 2020. – Vol. 8. – P. 69105–69133. DOI: 10.1109/ACCESS.2020.2983253

13. Priyabhashana H.M.B., Jayasena K.P.N. Data analytics with deep neural networks in fog computing using tensorflow and Google cloud platform // 14th Conference on Industrial and Information Systems (ICIIS). – 2019. – P. 34–39. DOI: 10.1109/ICIIS47346.2019.9063284

14. Гафаров Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие. – Казань: Изд-во Казан. ун-та, 2018. – С. 121.

15. Fast deep neural networks with knowledge guided training and predicted regions of interests for real-time video object detection / W. Cao, J. Yuan, Z. He, Z. Zhang, Z. He // IEEE Access. – 2018. – Vol. 6. – P. 8990–8999. DOI: 10.1109/ACCESS.2018.2795798

16. Yasnitsky L.N., Yasnitsky V.L. Technique of design of integrated economic and mathematical model of mass appraisal of real estate property

by the example of Yekaterinburg housing market // Journal of Applied Economic Sciences. – 2016. – Vol. 11, № 8. – P. 1519–1530.

17. Ясницкий Л.Н. Интеллектуальные системы. – М.: Лаборатория знаний, 2016.

18. Bakhtin V., Isaeva E. Developing an algorithm for identification and categorization of scientific terms in natural language text through the elements of artificial intelligence // 14th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE) – 44894. Proceedings. – Novosibirsk, 2018. – P. 384–390.

19. Isaeva E., Bakhtin V., Tararkov A. Collecting the database for the neural network deep learning implementation // Digital Science. DSIC18 2018. Advances in Intelligent Systems and Computing / T. Antipova, A. Rocha (eds). – 2019. – Vol. 850. – P. 12–18. Springer, Cham. DOI: 10.1007/978-3-030-02351-5_2

20. Bakhtin V.V., Isaeva E.V. New TSBuilder: shifting towards cognition // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2019. – P. 179–181. DOI: 10.1109/EIConRus.2019.8656917

21. Bakhtin V.V., Isaeva E.V., Tararkov A.V. TSBuilder 2.0: improving the identification accuracy due to synonymy // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2020. – P. 225–228. DOI: 10.1109/EIConRus49466.2020.9039207

22. Bakhtin V.V., Isaeva E.V., Tararkov A.V. TSMiner: from TSBuilder to ecosystem // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – 2021. – P. 221–224. DOI: 10.1109/ElConRus51938.2021.9396569

23. Isaeva E., Bakhtin V., Tararkov A. Formal cross-domain ontologization of human knowledge // Information Technology and Systems. ICITS 2020. Advances in Intelligent Systems and Computing / Á. Rocha, C. Ferrás, C. Montenegro Marin, V. Medina García (eds). – 2020. – Vol. 1137. – P. 94–103. Springer, Cham. DOI: 10.1007/978-3-030-40690-5_10

24. Бахтин В.В., Подлесных И.А. Алгоритм построения графа совместной работы каскадов устройств нейросетевого распознавания, реализующих блочные нейронные сети // Сб. матер. IX Междунар. науч. конф., посв. 85-лет. проф. В.И. Потапова. – Омск, 2021. – С. 277–278.

25. Каменских А.Н., Тюрин С.Ф. Методика комбинированного резервирования асинхронных нейронных сетей // Нейрокомпьютеры: разработка, применение. – 2016. – № 8. – С. 36–40.

References

1. Teerapittayanon S., McDanel B., Kung H.T. Distributed deep neural networks over the cloud, the edge and end devices. *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 3280339. DOI: 10.1109/ICDCS.2017.226

2. Sobecki A., Szymański J., Gil D., Mora H. Deep learning in the fog. *International Journal of Distributed Sensor Networks*, 2019, vol. 15, iss. 8. DOI: 10.1177/1550147719867072

3. Wu L., Lin X., Chen Z., Huang J., Liu H., Yang Y. An Efficient Binary Convolutional Neural Network with Numerous Skip Connections for Fog Computing. *IEEE Internet of Things Journal*, 2021, vol. 8, no. 14, pp. 11357-11367. DOI: 10.1109/JIOT.2021.3052105

4. Bakhtin V.V. Matematicheskaia model' iskusstvennoi neironnoi seti dlia ustroistv na plis i mikrokontrollerakh, orientirovannykh na tumannye vychisleniia [Mathematical model of an artificial neural network for fpga devices and microcontrollers focused on fog computing]. *Vestnik Permskogo natsional'nogo issledovatel'skogo politekhnicheskogo universiteta. Elektrotekhnika. Informatsionnye tekhnologii, sistemy upravleniia*, 2021, no. 40, pp. 109-129.

5. Akhmetzyanov K., Yuzhakov A. Waste Sorting Neural Network Architecture Optimization. *International Russian Automation Conference* (RusAutoCon), 2019, pp. 1-5. DOI: 10.1109/RUSAUTOCON.2019.8867749

6. Akhmetzyanov K.R., Yuzhakov A.A., Kokoulin A.N. Neural network development based on knowledge about environmental influence. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2020, pp. 218-221. DOI: 10.1109/EIConRus49466.2020.9039226

7. Bakhtin V.V. Modifikatsiia algoritma identifikatsii i kategorizatsii nauchnykh terminov s ispol'zovaniem neironnoi seti [Modification of the algorithm for identification and categorization of scientific terms using a neural network]. *Neirokomp'iutery: razrabotka, primenenie*, 2019, vol. 21, no. 3, pp. 14-19.

8. Aazam Mohammad, Zeadally Sherali, Harras Khaled. Fog Computing Architecture, Evaluation, and Future Research Directions. *IEEE Communications Magazine*, 2018, no. 56, pp. 46-52. DOI: 10.1109/MCOM.2018.1700707

9. Surbiryala J., Rong C. Cloud Computing: History and Overview. *IEEE Cloud Summit*, 2019, pp. 1-7. DOI: 10.1109/CloudSummit47114.2019.00007

10. Zhang W., Xiao K. Communication Mode of Computer Cluster Network in Cloud Environment based on Neural Computing. *5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, pp. 48-52. DOI: 10.1109/ICCMC51019.2021.9418350

11. Goudarzi M., Wu H., Palaniswami M., Buyya R. An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments. *IEEE Transactions on Mobile Computing*, 2021, vol. 20, no. 4, pp. 1298-1311. DOI: 10.1109/TMC.2020.2967041

12. Habibi P., Farhoudi M., Kazemian S., Khorsandi S., Leon-Garcia A. Fog Computing: A Comprehensive Architectural Survey. *IEEE Access*, 2020, vol. 8, pp. 69105-69133. DOI: 10.1109/ACCESS.2020.2983253

13. Priyabhashana H.M.B., Jayasena K.P.N. Data Analytics with Deep Neural Networks in Fog Computing Using TensorFlow and Google Cloud Platform. *14th Conference on Industrial and Information Systems (ICIIS)*, 2019, pp. 34-39. DOI: 10.1109/ICIIS47346.2019.9063284

14. Gafarov F.M., Galimianov A.F. Iskusstvennye neironnye seti i prilozheniia [Artificial neural networks and applications]. Kazan': Kazanskii universitet, 2018, 121 p.

15. Cao W., Yuan J., He Z., Zhang Z., He Z. Fast Deep Neural Networks With Knowledge Guided Training and Predicted Regions of Interests for Real-Time Video Object Detection. *IEEE Access*, 2018, vol. 6, pp. 8990-8999. DOI: 10.1109/ACCESS.2018.2795798

16. Yasnitsky L.N., Yasnitsky V.L. Technique of design of integrated economic and mathematical model of mass appraisal of real estate property by the example of Yekaterinburg housing market. *Journal of Applied Economic Sciences*, 2016, vol. 11, no. 8, pp. 1519-1530.

17. Iasnitskii L.N. Intellektual'nye sistemy [Intelligent systems]. Moscow: Laboratoriia znanii, 2016.

18. Bakhtin V., Isaeva E. Developing an Algorithm for Identification and Categorization of Scientific Terms in Natural Language Text through

the Elements of Artificial Intelligence. *14th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE) - 44894. Proceedings*. Novosibirsk, 2018, pp. 384-390.

19. Isaeva E., Bakhtin V., Tararkov A. Collecting the Database for the Neural Network Deep Learning Implementation. *Digital Science. DSIC18 2018. Advances in Intelligent Systems and Computing*. T. Antipova, A. Rocha (eds), 2019, vol. 850, pp. 12-18. Springer, Cham. DOI: 10.1007/978-3-030-02351-5_2

20. Bakhtin V.V., Isaeva E.V. New TSBuilder: Shifting towards Cognition. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2019, pp. 179-181. DOI: 10.1109/EIConRus.2019.8656917

21. Bakhtin V.V., Isaeva E.V., Tararkov A.V. TSBuilder 2.0: Improving the Identification Accuracy Due to Synonymy. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2020, pp. 225-228. DOI: 10.1109/EIConRus49466.2020.9039207

22. Bakhtin V.V., Isaeva E.V., Tararkov A.V. TSMiner: from TSBuilder to Ecosystem. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 2021, pp. 221-224. DOI: 10.1109/ElConRus51938.2021.9396569

23. Isaeva E., Bakhtin V., Tararkov A. Formal Cross-Domain Ontologization of Human Knowledge. *Information Technology and Systems. ICITS 2020. Advances in Intelligent Systems and Computing.* Rocha Á., Ferrás C., Montenegro Marin C., Medina García V. (eds), 2020, vol. 1137, pp. 94-103. Springer, Cham. DOI: 10.1007/978-3-030-40690-5_10

24. Bakhtin V.V., Podlesnykh I.A. Algoritm postroeniia grafa sovmestnoi raboty kaskadov ustroistv neirosetevogo raspoznavaniia, realizuiushchikh blochnye neironnye seti [Algorithm for constructing a graph of collaboration of cascades of neural network recognition devices implementing block neural networks]. *Sbornik materialov IX Mezhdunarodnoi nauchnoi konferentsii, posviashchennoi 85-letiiu professora V.I. Potapova*. Omsk, 2021, pp. 277-278.

25. Kamenskikh A.N., Tiurin S.F. Metodika kombinirovannogo rezervirovaniia asinkhronnykh neironnykh setei [A technique for combined redundancy of asynchronous neural networks]. *Neirokomp'iutery: razrabotka, primenenie*, 2016, no. 8, pp. 36-40.

Сведения об авторах

Бахтин Вадим Вячеславович (Пермь, Россия) – аспирант, младший научный сотрудник кафедры «Автоматика и телемеханика» Пермского национального исследовательского политехнического университета (614990, Пермь, Комсомольский пр., 29, e-mail: bakhtin_94@bk.ru).

About the authors

Vadim V. Bakhtin (Perm, Russian Federation) – Graduate Student, junior researcher of the Department of Automation and Telemechanics Perm National Research Polytechnic University (614990, Perm, 29, Komsomolsky pr., e-mail: bakhtin_94@bk.ru).

Поступила 25.02.2022
Одобрена 20.03.2022
Принята к публикации 20.06.2022

Финансирование. Исследование проводится при поддержке РФФИ на средства гранта № 20-37-90036 Аспиранты «Метод синтеза устройств нейросетевого распознавания для реализации режима Fog computing».

Конфликт интересов. Конфликт интересов по отношению к статье отсутствует.

Вклад автора. 100 %.