УДК 007.519.7
doi:10.31799/1684-8853-2020-2-20-30

Научные статьи
Articles

Алгоритм вычисления значений весов синапсов первого слоя нейронной сети на основе метрических методов распознавания. Часть 1

П. Ш. Гейдаров[a], *канд. техн. наук, доцент, orcid.org/0000-0002-3881-0629, plbaku2010@gmail.com*
[a]*Институт систем управления Национальной академии наук Азербайджана, Бахтияр Вагабзаде ул., 9, Баку, Az 1141, Азербайджан*

Введение: *метрические методы распознавания позволяют предварительно и строго определять структуры нейронных сетей прямого распространения, а именно количество нейронов, слоев и связей на основе начальных параметров задачи распознавания. Они также дают возможность на основе метрических выражений близости аналитически вычислять веса синапсов нейронов сети. Процедура настройки для данных сетей включает в себя последовательное аналитическое вычисление значения каждого веса синапса в таблице весов для нейронов нулевого или первого слоя, что позволяет уже на начальном этапе без применения алгоритмов обучения получить работоспособную нейронную сеть прямого распространения. Затем нейронные сети прямого распространения могут дообучаться известными алгоритмами обучения, что в целом ускоряет процедуры их создания и обучения.* ***Цель:*** *определить, сколько времени требует процесс вычисления значений весов и, соответственно, насколько является оправданным предварительное вычисление значений весов нейронной сети прямого распространения.* ***Результаты:*** *предложен и реализован алгоритм автоматизированного вычисления всех значений таблиц весов синапсов для нулевого и первого слоя применительно к задаче распознавания черно-белых однотонных изображений символов. Описание предлагаемого алгоритма приведено в программной среде Builder C++. Рассмотрена возможность оптимизировать процесс вычисления весов синапсов в целях ускорения всего алгоритма. Выполнена оценка затрачиваемого времени на вычисление этих весов для разных конфигураций нейронных сетей на основе метрических методов распознавания. Приведены примеры создания таблиц весов синапсов согласно рассмотренному алгоритму. Результаты вычисления таблиц показывают, что на процедуру аналитического вычисления весов нейронной сети потрачены считанные секунды, минуты, что никак не сравнимо со временем, необходимым для обучения нейронной сети.* ***Практическая значимость:*** *аналитическое вычисление значений весов нейронной сети позволяет существенно ускорить процедуру создания и обучения нейронной сети прямого распространения. На основе предложенного алгоритма может быть также реализован и алгоритм вычисления трехмерных таблиц весов для более сложных, черно-белых и цветных полутонных, изображений.*

Ключевые слова *— нейронные сети, весовые и пороговые значения, нейрокомпьютер, алгоритмы обучения, программирование нейронных сетей.*

Для цитирования: Гейдаров П. Ш. Алгоритм вычисления значений весов синапсов первого слоя нейронной сети на основе метрических методов распознавания. Ч. 1. *Информационно-управляющие системы*, 2020, № 2, с. 20–30. doi:10.31799/1684-8853-2020-2-20-30

For citation: Geidarov P. Sh. Algorithm for calculating synapse weights of the first layer of a neural network on the base of metric recognition methods. Part 1. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 2, pp. 20–30 (In Russian). doi:10.31799/1684-8853-2020-2-20-30

Введение

Искусственные нейронные сети (НС) широко используются в различных задачах [1–24]. Известно, что реализация классических архитектур НС [25] является непрозрачной, трудоемкой, а зачастую и непредсказуемой на практике работой. Процедуры обучения НС требуют наличия значительных по объему обучающих выборок по решаемой задаче. Из-за возможных проблем, возникающих в процессе обучения, таких как попадание в локальный минимум, паралич сети, может потребоваться повторное обучение с другими настройками алгоритмов обучения и с другими параметрами структуры сети. Выбор структуры классических НС также реализуется в большей степени на основе неточных рекомендаций, сформированных на эмпирических данных. Поэтому создание новых архитектур НС, позволяющих быстро и понятно строить НС для решения различных задач, остается актуальной целью.

В работах [26, 27] рассмотрены архитектуры НС, реализуемые метрическими методами распознавания [28]. Показано, что, в отличие от классических подходов, для рассматриваемых НС количество нейронов, связей и слоев определяется строгими выражениями и значениями исходя из начальных условий задачи, таких как количество используемых эталонов и распознаваемых образов. Весовые значения связей для этих сетей вычисляются аналитически на основе используемых выражений, определяющих меру близости объектов к образу. Такой подход позволяет получить работающую НС без использования классических алгоритмов обучения и без наличия обучающих выборок, что дает возможность

ускорить процедуру создания НС по сравнению с классическими решениями, для которых значения весов синапсов настраиваются длительными алгоритмами обучения. При этом для предлагаемых сетей могут дополнительно быть применены классические алгоритмы обучения НС в целях дообучения [27]. Таким образом, предлагаемые сети объединяют преимущества метрических методов распознавания и классических НС.

Напомним, что в качестве метрической меры близости могут использоваться различные формулы, среди которых и такие, как разность квадратов евклидовых расстояний:

$$w_{c,r}^{(1)} = d_1^2 - d_2^2 = \left(\left(c_1 - c_p\right)^2 + \left(r_1 - r_p\right)^2\right) - \left(\left(c_2 - c_p\right)^2 + \left(r_2 - r_p\right)^2\right), \quad (1)$$

где (c_1, r_1) и (c_2, r_2) являются координатами точек (или ячеек таблицы весов) до ближайшей точки (или ячейки) изображения эталона с координатами (c_p, r_p), как показано на рис. 1. Могут также использоваться и другие, более простые или сложные, выражения, например, выражение, определяющее среднеквадратичную разность значений, выполняемую только по оси *Y*:

$$w_{c,r}^{1} = (r_1 - r_p)^2 - (r_2 - r_p)^2. \quad (2)$$

Выражение (2) может использоваться, например, для задачи распознавания кривых.

Еще один пример характеристики близости представляет выражение

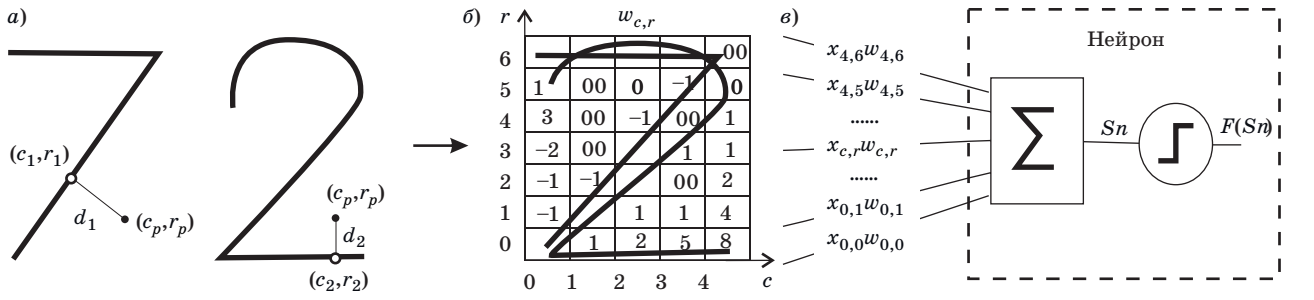
$$w_{c,r}^{(2)} = \frac{M}{1 + d_1^2} + \frac{M}{1 + d_2^2} = \frac{M}{1 + \left(\left(c_1 - c_p\right)^2 + \left(r_1 - r_p\right)^2\right)} + \frac{M}{1 + \left(\left(c_2 - c_p\right)^2 + \left(r_2 - r_p\right)^2\right)}, \quad (3)$$

где M — выбранная постоянная величина. Полученные при помощи выражений (1)–(3) таблицы весов синапсов свойственны для архитектур НС без нулевого слоя (рис. 2). В этом случае понадобится $N(N - 1)$ таких таблиц [26], определяемых отдельно для каждого нейрона первого слоя, где N — количество используемых эталонов.

Нейронная сеть с добавленным нулевым слоем, в котором количество вычисляемых таблиц весов синапсов уменьшается до количества используемых эталонов N, представлена на рис. 3. В этом случае в качестве выражений вычисления значения весов синапсов используются те, которые составляют одно из слагаемых значений в выражениях (1)–(3), например, для (1) это будет

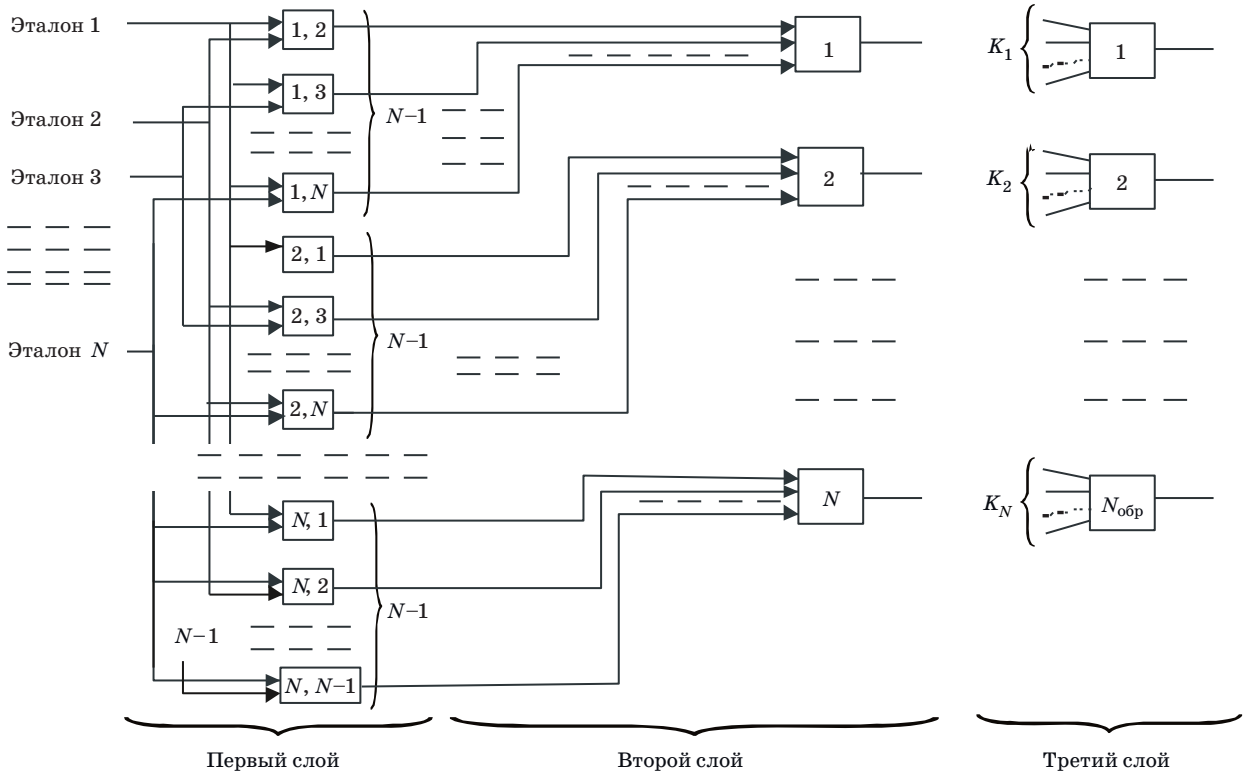
$$w_{c,r}^{(0)} = d_1^2 = ((c_1 - c_p)^2 + (r_1 - r_p)^2). \quad (4)$$

Для НС на основе метрических методов распознавания процедура вычисления значений весов синапсов нулевого или первого слоя выполняется последовательно для каждой ячейки таблицы весов, что требует определенного времени. При этом при создании НС на основе метрических методов распознавания наибольшее время занимают вычисления значений таблиц весов синапсов нулевого или первого слоя нейронов. И понятно, что остается открытым вопрос: «Сколько времени занимает процесс вычисления значений весов, и, соответственно, насколько является оправданным предварительное вычисление значений весов НС?», — на который автор ищет ответ. При этом, если в предыдущих работах [26, 27] приводятся аналитические выражения вычисления значений весов нулевого и первого слоя, то в данной работе ставится цель разработать и описать

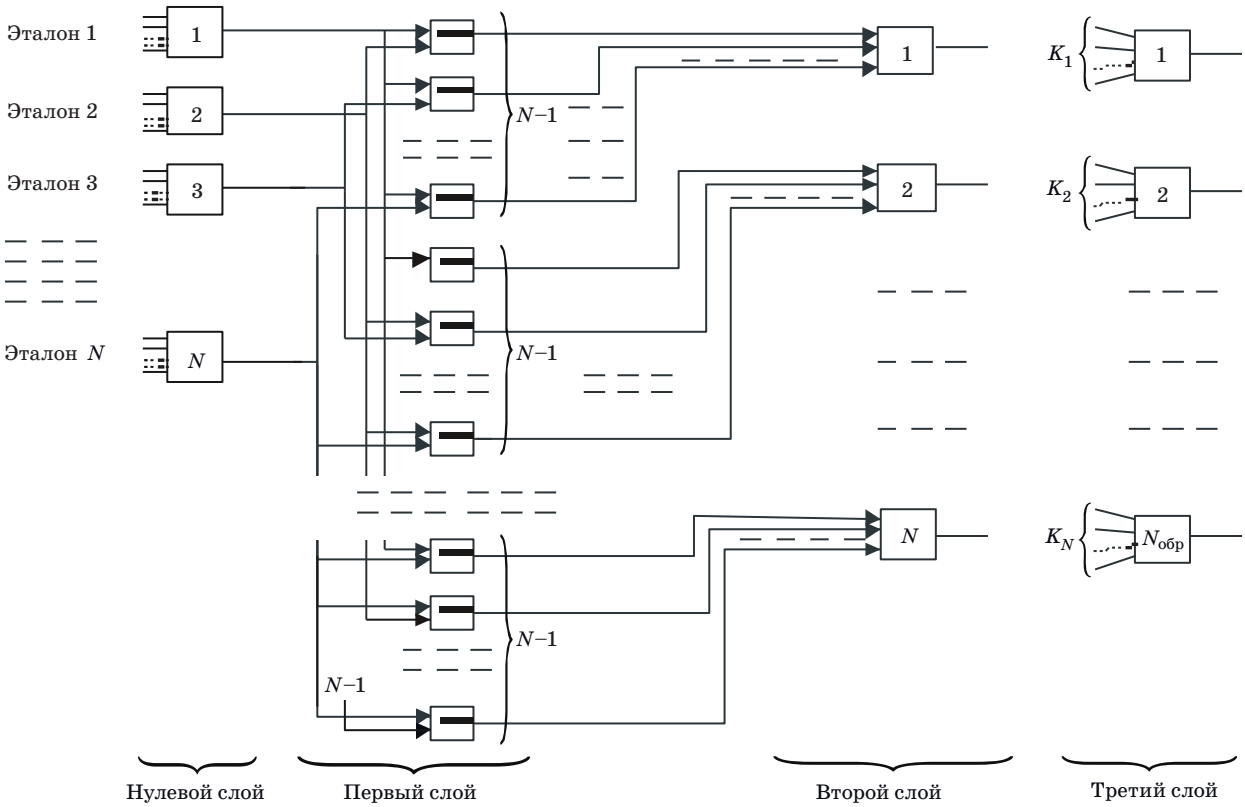


■ ***Рис. 1.*** Расстояния d_1 и d_2 для точки (c_p, r_p) (*а*); таблица весов для эталонов «2» и «7» (*б*); нейрон с пороговой функцией активации (*в*)

■ ***Fig. 1.*** Distances d_1 и d_2 for point (c_p, r_p) (*а*); table of weights for standards "2" and "7" (*б*); neuron with threshold activation function (*в*)



■ ***Рис. 2.*** Общая схема НС на основе метрического метода распознавания без нулевого слоя
■ ***Fig. 2.*** A general diagram of a neural network based on the metric recognition method without a zero layer



■ ***Рис. 3.*** Общая схема НС на основе метрического метода распознавания с нулевым слоем
■ ***Fig. 3.*** A general diagram of a neural network based on the metric recognition method with a zero layer

полный автоматизированный алгоритм вычисления значений всех весов нулевого и первого слоя, а также оценить время вычисления таблиц весов для разных конфигураций НС.

Алгоритм и программа вычисления таблиц весов

Вычисление таблицы весов синапсов реализовано в программном модуле с использованием объектно-ориентированной среды Boland C++. Основная часть программного кода реализована в четырех функциях, наименованных как

```
void __fastcall serch_point();
int __fastcall scan();
voidк __fastcall NaytiMin(int x, int y);
void __fastcall Izmeneniye_X0_Y0();
```

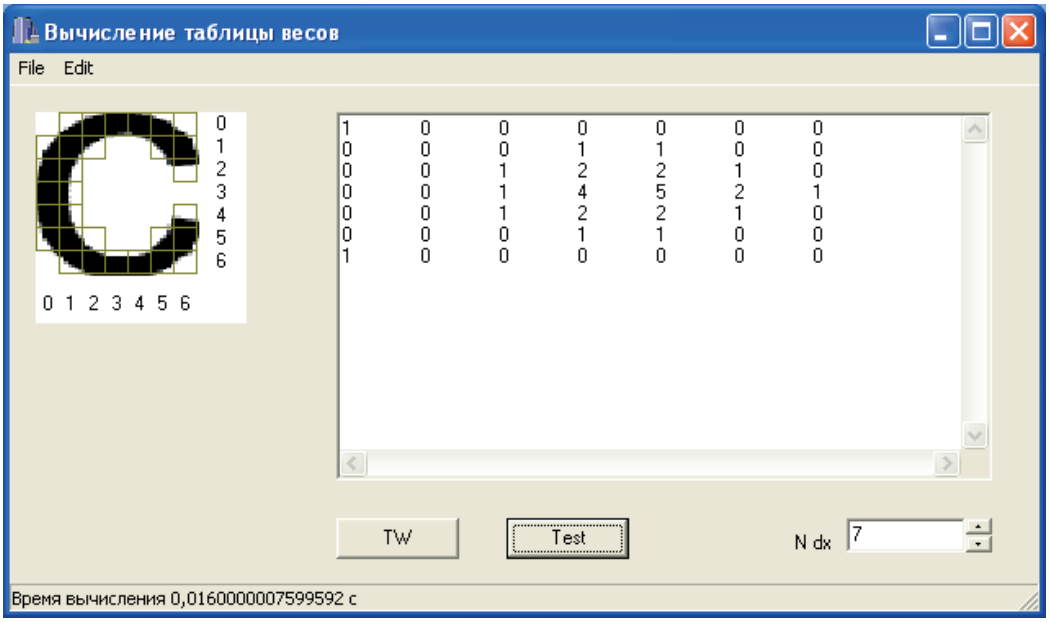
Функция *serch_point()* выполняет просмотр ячеек таблицы, при каждой найденной новой ячейке выполняет вызов функции *scan()*, которая проверяет, является ячейка активной или не активной. Здесь в качестве понятия активности понимается наличие в данной ячейке части изображения эталона. А в качестве текущей ячейки понимается ячейка таблицы весов, для которой вычисляется весовое значение синапса. Если ячейка активна, то в функцию *serch_point()* возвращается значение 1, в противном случае возвращается 0. Функция *NaytiMin* получает координаты активной ячейки из функции *serch_point()* и определяет значение веса синапса для текущей ячейки. В функции *Izmeneniye_X0_Y0* последовательно меняется текущая ячейка и определяются значения весов синапсов для всех ячеек таблицы весов нейрона нулевого или первого слоя.

Для вычисления таблицы весов выбранного изображения (рис. 4) предварительно выбирается размерность таблицы весов — количество колонок и строк (текстовое поле *N dx*). На основе этого значения определяется высота (*dy*) и ширина (*dx*) ячейки таблицы весов в пикселях путем разделения ширины изображения в пикселях на выбранное количество строк в текстовом поле *N dx*:

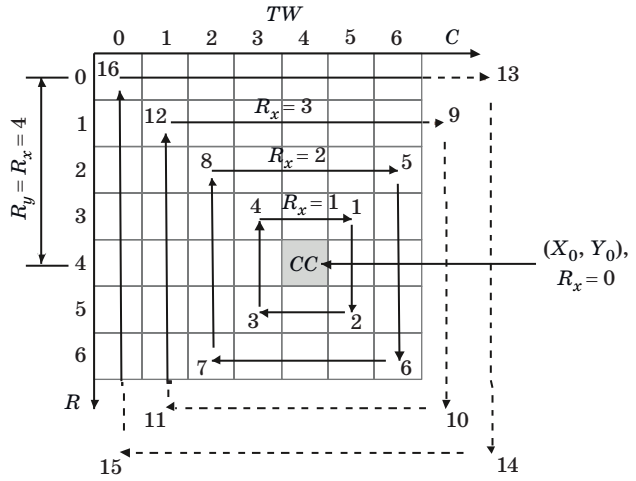
$$dx = pBitmap->Width/StrToInt(Edit1->Text);$$

В данной реализации значения *dx* и *dy* принимаются равными (*dx* = *dy*).

Процедура просмотра ячеек таблицы весов *TW* относительно текущей ячейки [на рис. 5 текущая ячейка *CC* с координатами (4, 4)] выполняется в функции *serch_point()* и выглядит следующим образом. Сначала проверяется текущая ячейка. Далее выполняется просмотр ближайших ячеек таблицы весов, располагаемых вокруг текущей ячейки. Движение происходит по замкнутому контуру. Контур движения в таблице представляет из себя квадрат. Сама процедура просмотра одного контура реализована в четырех последовательных циклах, где на каждом цикле выполняется просмотр ячеек по одной стороне квадрата: первый цикл — правая сторона квадрата, выполняется движение сверху вниз из точки *1* в точку *2*; второй цикл — нижняя сторона квадрата,



■ ***Рис. 4.*** Программный модуль вычисления таблицы весов синапсов нулевого слоя
■ ***Fig. 4.*** Software module for calculating the table of synapse weights of the zero layer



■ ***Рис. 5.*** Поиск по контурам активной ячейки относительно текущей ячейки *CC* в таблице весов *TW*

■ ***Fig. 5.*** Search the contours of the active cell relative to the current cell (*CC*) in the weight table (*TW*)

движение справа налево из точки *2* в точку *3*; третий цикл — левая сторона квадрата, движение снизу вверх из точки *3* в точку *4*; четвертый цикл — верхняя сторона квадрата, движение слева направо из точки *4* в точку *1*. Ниже приведен листинг четырех циклов, в которых реализуется движение по квадратному контуру:

```
M1: x = X0 + Rx;
for (y = Y0 − Ry; y < Y0 + Ry; y++)     // Цикл 1
{......}
y = Y0 + Ry;
for (x = X0 + Rx; x > X0 − Rx; x−−)  // Цикл 2
{......}
x = X0 − Rx;
for (y = Y0 + Ry; y > Y0 − Ry; y−−)    // Цикл 3
{......}
y = Y0 − Ry;
for (x = X0 − Rx; x < X0 + Rx; x++)   // Цикл 4
{.......}
```

Здесь *X0*, *Y0* — координаты текущей ячейки (на рис. 5 $X_0 = 4$, $Y_0 = 4$); *Rx*, *Ry* — радиус контура по оси *X* и *Y* соответственно относительно текущей ячейки. Начальные значения R_x, R_y принимаются равными единице. После просмотра одного контура выполняется увеличение значений R_x и R_y на единицу, после чего повторяется просмотр ячеек по замкнутому контуру, только теперь уже с новыми значениями $R_x = R_y$ (см. рис. 5). Таким образом, создается еще один цикл, который реализуется меткой *M1* и командой перехода *goto*:

```
Rx = Ry = Ry + 1;
goto M1;
```

Процесс выполняется до тех пор, пока значения R_x и R_y не выйдут за пределы всех сторон изображения, что проверяется следующим условием:

```
if ((X0 + Rx)*dx > W + dx && (X0 − Rx)*dx
<−dx && (Y0 + Ry)*dx > H + dx && (Y0 − Ry)*dx
<−dx) return;
```

где *W* и *H* — ширина и высота изображения, определяемые строками

```
W = pBitmap−>Width;
H = pBitmap−>Height;
```

Для случаев, когда по ходу движения по квадратному контуру происходит выход за пределы выбранной размерности таблицы весов (но происходит это не со всех сторон изображения; на рис. 5 контуры с точками *9*, *10*, *11*, *12*, *13*, *14*, *15*, *16*), ячейки, оказавшиеся за пределами таблицы весов синапсов, пропускаются в циклах при помощи следующей строки:

```
if (X1 <−dx || Y1 <−dx || X1 > = W + dx || Y1 > =
= H + dx) continue;
```

где *X1*, *Y1* — значения координат изображения в пикселях, полученные от перебираемых в циклах координат ячеек таблиц весов *x*, *y* следующим образом:

```
Y1 = y*dx; X1 = x*dx;
```

Для каждой ячейки таблицы весов синапсов, встречаемой на пути движения вдоль замкнутого контура, выполняется проверка активности данной ячейки таблицы в функции *scan*. Функция *scan*, используя координаты выбранной ячейки в изображении *X1*, *Y1*, посредством двух встроенных циклов и метода *ScanLine* объекта изображения *pBitmap* сканирует все пиксели в данной ячейке, значение которых не выше 50, на наличие затемненных участков:

```
for (x = X1 + 2; x < X2 − 1; x++)
for (y = Y1 + 2; y < Y2 − 1 ; y++)
      {
      ptr = (Byte *)pBitmap−>ScanLine[y];
      pt = ptr[x];
      if (ptr[x] < 50) return 1;
      }
return 0;
```

Здесь если *X1*, *Y1* — координаты левого верхнего угла ячейки, то *X2*, *Y2* — координаты правого нижнего угла ячейки, которые определяются как

```
X2 = X1 + dx; Y2 = Y1 + dx;
```

Сканирование выполняется последовательно для всех пикселей выбранной ячейки таблицы весов. Активность пикселя определяется, если его значение меньше 50, что соответствует затемненным оттенкам черно-белого изображения, где тональность пикселя измеряется в диапазоне от 0 до 256. При выполнении условия затененности пикселей в ячейке таблицы процесс прерывается, и функция *scan* возвращает значение 1 в функцию *serch_point*, что соответствует активности ячейки. Если после сканирования всей ячейки не определяются активные пиксели, то функция *scan* возвращает значение 0, что соответствует неактивности данной ячейки. В листинге строки необходимы также строки, исключающие выход процедуры сканирования за пределы изображения:

```
if (y <= 0 || y >= pBitmap->Height) continue;
if (x <= 0 || x >= pBitmap->Width) continue;
```

Функция *serch_point* получает выходное значение функции *scan*, и если оно активно, вызывает функцию *NaytiMin*:

```
Aktiv = scan();
if(Aktiv == 1) NaytiMin(x, y);
```

В функции *NaytiMin* касательно найденной активной ячейки с координатами x и y выполняется вычисление значения веса синапса для текущей ячейки с использованием выбранного метрического выражения близости, например квадрата евклидова расстояния между ячейками:

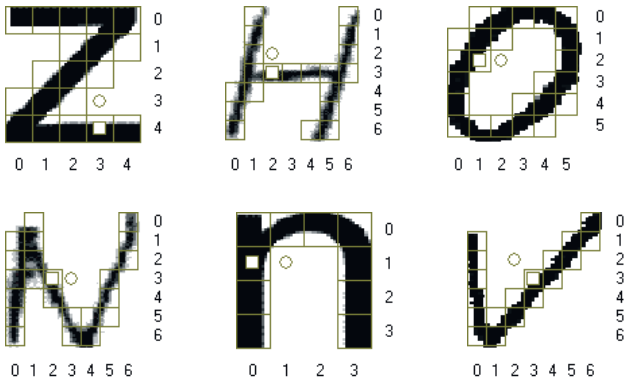
$$W = pow(x - X0, 2) + pow(y - Y0, 2),$$

которое далее сравнивается с предыдущим определенным минимальным значением веса синапса *minW* для данной текущей ячейки. При этом, если новое значение веса W меньше предыдущего минимального значения, то данное полученное значение W запоминается как минимальное:

```
if(W < minW) minW = W;
```

Таким образом, в функции *serch_point* при просмотре всех контуров относительно текущей ячейки с координатами *(X0, Y0)* (на рис. 6 отмечена кружком) определяется минимальное значение веса синапса активной ячейки (на рис. 6 ближайшая активная ячейка отмечена белым квадратом), которое и принимается за значение веса синапса текущей ячейки *(X0, Y0)* таблицы весов.

Если в качестве меры близости используются формулы, которые выражают зависимость от всех активных ячеек изображения, то в этом случае при просмотре ячеек таблицы должно определяться уже не минимальное, а суммарное значение веса синапса, которое формируется в текущей ячейке *(X0, Y0)* под влиянием каждой активной ячейки изображения и вычисляется по ходу выявления активных ячеек.



■ ***Рис. 6.*** Результаты определения ближайшей активной ячейки (отмечена белым квадратом) относительно текущей ячейки (отмечена белым кругом)

■ ***Fig. 6.*** Results of determining the nearest active cell (marked with a white square) relative to the current cell (marked with a white circle)

В итоге, таким образом, вычисляется значение веса синапса для одной ячейки таблицы весов синапсов. Для вычисления значений весов синапсов всех остальных ячеек таблицы весов выполняется последовательное изменение номера текущей ячейки *(X0, Y0)*, которое реализуется в двух встроенных циклах функции *Izmeneniye_X0_Y0()*, от ячейки *(0, 0)* до ячейки *(Nx, Ny)*. После каждого изменения текущей ячейки в цикле запускается функция *serch_point*, которая определяет значение веса синапса для этой выбранной текущей ячейки:

```
for (Y0 = 0; Y0 < Ny; Y0++)
{
for (X0 = 0; X0 < Nx; X0++)
{
serch_point();
if(X0 == 0) str = IntToStr(minW);
else str = str + «\t» + IntToStr(minW);
}
Form1->RichEdit1->Lines->Add(str); str = «»;
}
```

Здесь Nx, Ny — количество строк и столбцов таблицы весов (значение текстового поля *N dx* на рис. 4). Таким образом, вычисляется таблица весов, значения которых выводятся на экран при помощи текстового элемента *RichEdit*.

Ускорение работы алгоритма

В приведенном выше алгоритме в функции *Izmeneniye_X0_Y0* выполняется перебор всех

ячеек таблицы выбранной размерности. Как выше показано, для текущей ячейки таблицы выполняется поиск активных ячеек по контурам также путем перебора всех ячеек таблицы весов. При этом время вычисления таблиц весов может быть значительно меньше, в случае если в предложенном алгоритме исключить рассмотрение ненужных контуров и, соответственно, ненужных ячеек таблиц весов. В частности, если сама текущая ячейка с координатами таблицы *(X0, Y0)* является активной, то нет необходимости просматривать все остальные ячейки. Программа в этом случае при помощи следующего кода вычисляет значение веса синапса для текущей ячейки и прерывает просмотр всех остальных контуров и ячеек изображения:

```
X1 = X0*dx; Y1 = Y0*dx;
if (X1 <–dx || Y1 <–dx || X1 >= W + dx || Y1 >=
= H + dx) goto M1;
Aktiv = scan();
if(Aktiv = = 1) {NaytiMin(X0, Y0); return;}
```

Эта проверка реализуется до начала четырех циклов, выполняющих поиск активной ячейки по контурам таблицы весов. Кроме того, если минимальное значение веса синапса определяется на каком-то контуре, то нет также необходимости рассматривать все последующие контуры и, соответственно, ячейки таблицы в этих контурах. В этом случае требуется определить минимально достаточное количество просматриваемых контуров x, необходимых для просмотра последующих контуров (рис. 7). Поскольку максимальное удаление активной ячейки *AC* при текущем значении R_x от центра квадрата (текущей ячейки *CC*) будет в вершинах квадрата, и если при этом для вычисления значения веса синапса используется метрическая мера квадрата евклидова расстояния, то для выявления значения x необходимо решить следующее неравенство:

$$(R_x + x)^2 < R_x^2 + R_x^2, \quad (5)$$

$$2R_x x + x^2 < R_x^2, \quad (6)$$

$$x^2 + 2R_x x - R_x^2 < 0. \quad (7)$$

Для решения неравенства определяем корни квадратного уравнения

$$x^2 + 2R_x x - R_x^2 = 0, \quad (8)$$

для чего определяем дискриминантное значение D и значения x_1, x_2:

$$D = \pm\sqrt{4R_x^2 - 4\left(-R_x^2\right)} = \pm\sqrt{8R_x^2} = \pm 2R_x\sqrt{2}; \quad (9)$$

$$x_1 = \left(-2R_x\sqrt{2} - 2R_x\right)/2 = -R_x\left(\sqrt{2}+1\right); \quad (10)$$

$$x_2 = \left(2R_x\sqrt{2} - 2R_x\right)/2 = R_x\left(\sqrt{2}-1\right). \quad (11)$$

Решением неравенств (5) и (7) является множество значений

$$-R_x\left(\sqrt{2}+1\right) < x < R_x\left(\sqrt{2}-1\right), \quad (12)$$

а максимальное необходимое значение

$$x_{\max} = R_x\left(\sqrt{2}-1\right). \quad (13)$$

Отсюда определяем необходимое максимальное значение R'_x (см. рис. 7), которое может быть не больше $R_x\sqrt{2}$, что следует из выражения

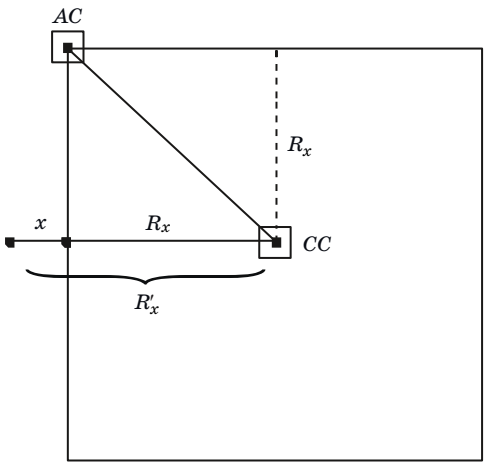
$$R'_x < x + R_x = R_x\left(\sqrt{2}-1\right) + R_x = \\ = R_x\sqrt{2} \approx R_x \times 1{,}414. \quad (14)$$

Поскольку значение $\sqrt{2} < 1{,}5 = 3/2$, то для упрощения вычисления можно определить условие проверки как

$$R'_x < 3R_x / 2. \quad (15)$$

При этом для исключения рассмотрения ненужных контуров необходимо запоминать текущее значение R_x при определении каждого нового минимального значения веса синапса *minW* в процессе рассмотрения активных ячеек таблицы весов, что выполняется в функции *NaytiMin* с помощью следующего условия:

```
if(W < minW) {minW = W; RxMin = Rx;},
```



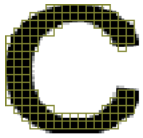
■ ***Рис. 7.*** Определение минимально достаточного количества просматриваемых контуров x после выявления активной ячейки *AC*

■ ***Fig. 7.*** Determination of x — the minimum sufficient number of contours viewed, after identifying the active cell *AC*

а)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	29	20	13	8	4	1	0	0	0	0	1	4	8	13	20	29
1	26	17	10	5	2	1	0	0	0	0	1	2	5	10	17	25
2	25	16	9	4	1	0	0	0	0	0	0	1	4	8	13	20
3	20	13	8	4	1	0	0	0	0	0	0	1	2	5	10	17
4	17	10	5	2	1	0	0	0	1	0	0	0	1	4	9	16
5	13	8	4	1	0	0	0	1	2	1	0	0	1	4	8	13
6	10	5	2	1	0	0	1	2	4	1	0	0	1	2	5	10
7	9	4	1	0	0	0	1	4	4	1	0	0	0	1	4	8
8	8	4	1	0	0	0	1	4	4	1	0	0	0	1	2	5
9	5	2	1	0	0	0	1	1	1	1	1	0	0	0	1	4
10	4	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2
11	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1
13	1	0	0	0	1	4	4	4	4	4	4	1	0	0	0	1
14	1	0	0	1	2	5	9	9	9	9	5	2	1	0	0	0
15	0	0	0	1	4	8	13	16	16	13	8	5	2	1	0	0

б)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	8	5	2	1	1	0	0	0	0	0	0	0	0	1	2	4
1	5	2	1	0	0	0	0	0	0	0	0	0	0	0	1	1
2	2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
3	1	0	0	0	0	0	1	2	4	4	2	1	0	0	0	0
4	0	0	0	0	1	1	2	5	8	8	5	2	1	0	0	0
5	0	0	0	0	1	4	5	8	13	13	8	5	2	1	0	1
6	0	0	0	1	2	5	10	13	18	18	13	8	5	2	1	2
7	0	0	0	1	4	8	13	20	25	25	18	13	8	5	4	5
8	0	0	0	1	4	9	16	25	34	32	25	18	13	10	9	9
9	0	0	0	1	4	8	13	20	25	25	18	13	8	5	4	4
10	0	0	0	1	2	5	10	13	16	16	13	8	5	2	1	1
11	0	0	0	0	1	4	5	8	9	9	8	5	2	1	0	0
12	0	0	0	0	1	1	2	4	4	4	4	2	1	0	0	0
13	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
14	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	5	2	1	0	0	0	0	0	0	0	0	0	0	0	1	2

■ ***Рис. 8.*** Таблицы весов для печатных символов «А» (*а*) и «С» (*б*) с размерностью 16×16 для нейронов нулевого слоя

■ ***Fig. 8.*** Tables of scales for printed characters “A” (*а*) and “C” (*б*) with a dimension of 16×16 for neurons of the zero layer

а в функции *serch_point* после увеличения значения R_x выполняется проверка условия

$Rx = Ry = Ry + 1;$
$if(Rx > RxMin*3/2)\ return;$

Если это условие выполняется, то выполняется выход из функции *serch_point*, а в функции *Izmeneniye_X0_Y0* выполняется выбор следующей текущей ячейки.

Нужно сказать, что если используется другая метрическая мера близости, то выражение (5) будет иным, и, соответственно, значения x будут определяться по-иному.

На рис. 8, *а* и *б* приведены таблицы весов для эталонных печатных символов «А» и «С» с размерностью 16 строк на 16 столбцов. Данные таблицы соответствуют таблицам весов синапсов нейронов нулевого слоя. Из рисунка можно также видеть, что активные ячейки таблиц имеют значения весов, равные 0. Данные таблицы весов были вычислены с использованием метрической меры близости — квадрата евклидова расстояния. Если используются другие меры близости, то значения весов в таблицах будут другими.

Окончание следует

Литература

1. **Азаров И. С., Петровский А. А.** Формирование персональной модели голоса диктора с универсальным фонетическим пространством признаков на основе искусственной нейронной сети. *Труды СПИИРАН*, 2014, № 5, с. 12–15. https://doi.org/10.15622/sp.36.8
2. **Будко Р. Ю., Старченко И. Б.** Создание классификатора мимических движений на основе анализа электромиограммы. *Труды СПИИРАН*, 2016, № 3 (46), с. 76–89. https://doi.org/10.15622/sp.46.7
3. **Осипов В. Ю., Никифоров В. В.** Возможности рекуррентных нейронных сетей с управляемыми элементами по восстановлению потоков кадров. *Информационно-управляющие системы*, 2019, № 5, с. 10–17. doi.org/10.31799/1684-8853-2019-5-10-17
4. **Ле Т. Ч.** Сравнение нейронной сети CMAC и многослойной нейронной сети в задаче обнаружения DoS-атак. *Нейрокомпьютеры: разработка, применение*, 2016, № 7, с. 65–69.
5. **Катасев А. С., Катасева Д. В. , Кирпичников А. П.** Распознавание рукописных символов на базе искусственной нейронной сети. *Вестник технологического университета,* 2018, т. 18, № 11, с. 173–176.
6. **Дрокин И. С.** Об одном алгоритме последовательной инициализации весов глубоких нейронных сетей и обучении ансамбля нейронных сетей. *Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления*, 2016, № 4, с. 66–74.
7. **Лещенко Ю. Е.** Ассоциативно-вербальные сети и искусственные нейронные сети как метод моделирования ментального лексикона индивида. *Глобальный научный потенциал*, 2014, № 10 (43), с. 154–156.

8. **Гафуров Д. О., Гафуров О. М., Конторович В. А.** Интерпретация данных геофизических исследований Талаканского нефтегазоконденсатного месторождения обучаемыми нейронными сетями, прогноз строения Осинского горизонта. *Технологии сейсморазведки*, 2014, № 4, с. 85–92.
9. **Владимирова Д. Б., Кокшарова А. А.** Прогнозирование финансовых рынков искусственными нейронными сетями. *Наука и бизнес: пути развития*, 2014, № 3 (33), с. 42–46.
10. **Синчук О. Н., Бойко С. Н.** Нейронные сети и управление процессом управления электроснабжением объектов от комбинированных электрических сетей. *Технічна електродинаміка*, 2014, № 5, с. 53–55.
11. **Шведов В. А., Ильин Е. С.** Распознавания дорожных знаков сверточными нейронными сетями различной структуры, с применением графических процессоров CUDA. *Транспортная инфраструктура Сибирского региона: материалы Шестой международной научно-практической конференции*, Иркутск, 30 сентября–03 октября 2015 г., 2015, т. 2, с. 282–286.
12. **Туровский Я. А., Кургалин С. Д., Адаменко А. А.** Сравнительный анализ программных пакетов для работы с искусственными нейронными сетями. *Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии*, 2016, № 1, с. 161–168.
13. **Лукина А. С., Некрасов М. В., Пакман Д. Н.** Обработка телеметрической информации космического аппарата нейронными сетями на основе теории фильтров Калмана. *Тенденции развития науки и образования*, 2016, № 13-1, с. 43–45.
14. **Стасюк В. В.** Прогнозирование чувствительности к удару взрывчатых веществ нейронными сетями с предварительной группировкой данных. *Фундаментальные исследования*, 2015, № 12–6, с. 1139–1143.
15. **Хусаинов А. Т.** Оценка прогнозируемости системы поддержания пластового давления нейронными сетями на нефтяных месторождениях. *Академический журнал Западной Сибири*, 2016, т. 12, № 3 (64), с. 48.
16. **Бондарко В. М., Бондарко Д. В., Солнушкин С. Д., Чихман В. Н.** Моделирование результатов психофизических экспериментов нейронными сетями. *Нейрокомпьютеры: разработка, применение*, 2018, № 5, с. 31–33.
17. **Sedov V. A., Sedova N. A.** Modelling collision avoidance actions in closest approach zones by means of neural networks. *Asia-Pacific Journal of Marine Science & Education*, 2014, vol. 4, no. 2, pp. 104–111.
18. **Peng Shi, Fanbiao Li, Ligang Wu.** Neural network-based passive filtering for delayed neutral-type semi-Markovian jump systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2017, vol. 28 , iss. 9, pp. 2101–2114.
19. **Chenggang Yan, Hongtao Xie, Dongbao Yang, Jian Yin, Yongdong Zhang, Qionghai Dai.** Supervised hash coding with deep neural network for environment perception of intelligent vehicles. *EEE Transactions on Intelligent Transportation Systems*, 2018, vol. 19, iss. 1, pp. 284–295.
20. **Anusha N., Gregory K., Ronald S. F., Sergey Levine.** Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018, pp. 156–162.
21. **Wei He, Yuhao Chen, Zhao Yin.** Adaptive neural network control of an uncertain robot with full-state constraints. *IEEE Transactions on Cybernetics*, 2016, vol. 46, iss. 3, pp. 620–629.
22. **Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. H., William J.** Efficient inference engine on compressed deep neural network. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA*), Seoul, South Korea, 2016, pp. 304–309.
23. **William Chan, Navdeep Jaitly, Quoc Le, Oriol Vinyals.** Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 107–115.
24. **Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, Sebastian Thrun.** Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017, vol. 542, pp. 115–118.
25. **Круглов В. В., Борисов В. В.** *Искусственные нейронные сети. Теория и практика*. М., Горячая линия–Телеком, 2001. 328 с.
26. **Geidarov P. Sh.** Neural networks on the basis of the sample method. *Automatic Control and Computer Sciences*, 2009, vol. 43, pp. 203–210. https://doi.org/10.3103/S0146411609040063
27. **Гейдаров П. Ш.** Алгоритм реализации метода ближайшего соседа в многослойном персептроне. *Труды СПИИРАН*, 2017, т. 51, с. 123–151. https://doi.org/10.15622/sp.51.6
28. **Биргер И. А.** *Техническая диагностика*. М., Машиностроение, 1978. 240 с.

UDC 007.519.7
doi:10.31799/1684-8853-2020-2-20-30

Algorithm for calculating synapse weights of the first layer of a neural network on the base of metric recognition methods. Part 1

P. Sh. Geidarov[a], PhD, Tech., Associate Professor, orcid.org/0000-0002-3881-0629, plbaku2010@gmail.com
[a]Azerbaijan National Academy of Sciences Institute of Control Systems, 9, Bahtijar Vagabzade St., Az 1141, Baku, Azerbaijan

Introduction: Metric recognition methods make it possible to preliminarily and strictly determine the structures of feed-forward neural networks, namely, the number of neurons, layers, and connections based on the initial parameters of the recognition problem. They also make it possible to analytically calculate the synapse weights of network neurons based on metric expressions. The setup procedure for these networks includes a sequential analytical calculation of the values of each synapse weight in the weight table for neurons of the zero or first layer, which allows us to obtain a working feed-forward neural network at the initial stage without the use of training algorithms. Then feed-forward neural networks can be trained by well-known learning algorithms, which generally speeds up the process of their creation and training. **Purpose:** To determine how much time the process of calculating the values of weights requires and, accordingly, how reasonable it is to preliminarily calculate the weights of a feed-forward neural network. **Results:** An algorithm is proposed and implemented for the automated calculation of all values of synapse weight tables for the zero and first layers as applied to the task of recognizing black-and-white monochrome symbol images. The proposed algorithm is described in the Builder C++ software environment. The possibility of optimizing the process of calculating the weights of synapses in order to accelerate the entire algorithm is considered. The time spent on calculating these weights for different configurations of neural networks based on metric recognition methods is estimated. Examples of creating and calculating synapse weight tables according to the considered algorithm are given. According to them, the analytical calculation of the weights of a neural network takes just seconds or minutes, being in no way comparable to the time necessary for training a neural network. **Practical relevance:** Analytical calculation of the weights of a neural network can significantly accelerate the process of creating and training a feed-forward neural network. Based on the proposed algorithm, we can implement one for calculating three-dimensional weight tables for more complex images, either black-and-white or color grayscale ones.

Keywords — neural networks, weight and threshold values, neurocomputer, learning algorithms, programming of neural networks.

For citation: Geidarov P. Sh. Algorithm for calculating synapse weights of the first layer of a neural network on the base of metric recognition methods. Part 1. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 2, pp. 20–30 (In Russian). doi:10.31799/1684-8853-2020-2-20-30

References

1. Azarov I. S., Petrovskij A. A. Training personal voice model of a speaker with unified phonetic space of features using artificial neural network. *SPIIRAS Proceedings*, 2014, no. 5, pp. 128–15 (In Russian). https://doi.org/10.15622/sp.36.8
2. Budko R. Yu., Starchenko I. B. Creation of the facial gestures dlassifier based on the electromyogram analysis. *SPIIRAS Proceedings*, 2016, no. 3 (46), pp. 76–89 (In Russian). https://doi.org/10.15622/sp.46.7
3. Osipov V. Yu., Nikiforov V. V. Recurrent neural networks with controlled elements in restoring frame flows. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 5, pp. 10–17 (In Russian). https://doi.org/10.31799/1684-8853-2019-5-10-17
4. Le T. Ch. The comparison of neural network CMAC and multilayer neural network in the task of detection of DoS attacks. *Neurokomputers*, 2016, no. 7, pp. 65–69 (In Russian).
5. Katasev A. S., Kataseva D. V., Kirpichnikov A. P. Handwritten character recognition based on artificial neural network. *Vestnik tehnologicheskogo universiteta*, 2018, vol. 18, no. 11, pp. 173–176 (In Russian).
6. Drokin I. S. About an algorithm for consistent weights initialization of deep neural networks and neural networks ensemble learning. *Vestnik of Saint Petersburg University Applied Mathematics. Computer Science. Control Processes*, 2016, no. 4, pp. 66–74 (In Russian).
7. Leshhenko Ju. E. Associative-verbal networks and artificial neural networks as a method of mental lexicon modeling. *Global Scientific Potential*, 2014, no. 10 (43), pp. 154–156 (In Russian).
8. Gafurov D. O., Gafurov O. M., Kontorovich V. A. Interpretation of data from geophysical studies of the Talakan oil and gas condensate field by trained neural networks, forecast of the structure of the Osinsky horizon. *Seismic Technologies*, 2014, no. 4, pp. 85–92 (In Russian).
9. Vladimirova D. B., Koksharova A. A. Financial Markets Forecasting Using Artificial Neural Networks. Science and Business: Ways of Development, 2014, no. 3 (33), pp. 42–46 (In Russian).
10. Sinchuk O. N., Bojko S. N. Neural networks and process control of power supply of objects from combined electric networks. *Tehnichna elektrodinamika*, 2014, no. 5, pp. 53–55 (In Russian).
11. Shvedov V. A., Il'in E. S. Recognition of road signs by convolutional neural networks of various structures, using cuda GPUs. *Materialy Shestoy mezhdunarodnoy nauchno-prakticheskoy konferentsii "Transportnaja infrastruktura Sibirskogo regiona"* [Proc. 6th Int. Sump. "Transport infrastructure of the Siberian region"], 2015, vol. 2, pp. 282–286 (In Russian).
12. Turovskij Ya. A., Kurgalin S. D., Adamenko A. A. Comparative analysis of software packages for working with artificial neural networks. *Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Sistemnyj analiz i informacionnye tekhnologii*, 2016, no. 1, pp. 161–168 (In Russian).
13. Lukina A. S., Nekrasov M. V., Pakman D. N. Processing telemetric information of a spacecraft by neural networks based on the theory of Kalman filters. *Tendencii razvitiya nauki i obrazovaniya*, 2016, no. 13-1, pp. 43–45 (In Russian).
14. Stasjuk V. V. Prediction of the explosives impact sensitivity by neural networks with preliminary grouping data. *Fundamental Research*, 2015, no. 12-6, pp. 1139–1143 (In Russian).
15. Husainov A. T. Predictability assessment of a reservoir pressure maintenance system by neural networks in oil fields. *Academic Journal of West Siberia*, 2016, vol. 12, no. 3 (64), p. 48 (In Russian).
16. Bondarko V. M., Bondarko D. V., Solnushkin S. D., Chihman V. N. Modeling the results of psychophysical experiments by neural networks. *Neurokomputers*, 2018, no. 5, pp. 31–33 (In Russian).
17. Sedov V. A., Sedova N. A. Modelling collision avoidance actions in closest approach zones by means of neural networks. *Asia-Pacific Journal of Marine Science & Education*, 2014, vol. 4, no. 2, pp. 104–111.
18. Peng Shi, Fanbiao Li, Ligang Wu. Neural network-based passive filtering for delayed neutral-type semi-Markovian jump systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2017, vol. 28, iss. 9, pp. 2101–2114.
19. Chenggang Yan, Hongtao Xie, Dongbao Yang, Jian Yin, Yongdong Zhang, Qionghai Dai. Supervised hash coding with deep neural network for environment perception of intelligent vehicles. *EEE Transactions on Intelligent Transportation Systems*, 2018, vol. 19, iss. 1, pp. 284–295.

20. Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018, pp. 156–162.
21. Wei He, Yuhao Chen, Zhao Yin. Adaptive neural network control of an uncertain robot with full-state constraints. *IEEE Transactions on Cybernetics*, 2016, vol. 46, iss. 3, pp. 620–629.
22. Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, William J. Efficient inference engine on compressed deep neural network. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Seoul, South Korea, 2016, pp. 304–309.
23. William Chan, Navdeep Jaitl, Quoc Le, Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, 2016, pp. 107–115.
24. Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017, vol. 542, pp. 115–118.
25. Kruglov V. V., Borisov V. V. *Iskusstvennye nejronnye seti. Teoriya i praktika* [Artificial neural networks. Theory and practice]. Moscow, Goriachaia liniia-Telekom Publ., 2001. 328 p. (In Russian).
26. Geidarov P. Sh. Neural networks on the basis of the sample method. *Automatic Control and Computer Sciences*, 2009, vol. 43, pp. 203–210. https://doi.org/10.3103/S0146411609040063
27. Geidarov P. Sh. An algorithm implementing the method of the nearest neighbor in a multi-layer perceptron. *SPIIRAS Proceedings*, 2017, vol. 51, pp. 123–151 (In Russian).
28. Birger I. A. *Tekhnicheskaya diagnostika* [Technical diagnostics]. Moscow, Mashinostroenie Publ., 1978. 240 p. (In Russian).