УДК 004.032.26 **DOI** 10.15622/ia.2021.20.2.8

А.Н. Голубинский, А.А. Толстых

ГИБРИДНЫЙ МЕТОД ОБУЧЕНИЯ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Голубинский А.Н., Толстых А.А. **Гибридный метод обучения сверточных нейронных сетей.**

Аннотация. Предложен гибридный метод обучения сверточных нейронных сетей. Метод заключается в объединении методов второго и первого порядка для разных элементов архитектуры сверточной нейронной сети. Гибридный метод обучения сверточных нейронных сетей позволяет добиваться значительно лучшей сходимости по сравнению с методом обучения сверточных нейронных сетей «Adam» и требует меньше вычислительных операций для реализации. Рассматриваемый метод применим для обучения сетей, на которых происходит паралич обучения при использовании методов первого порядка. Более того, предложенный метод обладает способностью подстраивать свою вычислительную сложность под аппаратные средства, на которых производится вычисление, вместе с тем гибридный метод позволяет использовать подход обучения мини-пакетов.

Приведен анализ соотношения вычислений между сверточными нейронными сетями и полносвязными искусственными нейронными сетями. Рассмотрен математический аппарат оптимизации ошибки искусственных нейронных сетей, включающий в себя метод обратного распространения ошибки, алгоритм Левенберга-Марквардта. Проанализированы основные ограничения данных методов, возникающие при обучении сверточной нейронной сети.

Проведен анализ устойчивости предлагаемого метода при изменении инициализирующих параметров. Приведены результаты применимости метода в различных задачах.

Ключевые слова: сверточные нейронные сети, методы обучения искусственных нейронных сетей, методы оптимизации

1. Введение. В задаче классификации объектов на телевизионных изображениях широко применяют сверточные нейронные сети (СНС), которые показывают в данном классе задач наилучшие результаты [1, 2]. Однако существует набор проблем, присущих этому подходу.

В настоящий момент невозможно использовать детерминистические алгоритмы оптимизации функции ошибки СНС в реальных задачах ввиду их большой вычислительной сложности, что влечет за собой трудности при обучении СНС. Основным инструментом обучения СНС являются стохастический метод и его модернизации [1, 3, 4]. Данные методы подвержены сильному воздействию начальных условий и не могут гарантировать сходимость. С другой стороны, существуют методы оптимизации второго порядка [5, 6], которые гарантируют сходимость и менее зависимы от способа инициализации. Их основной проблемой является большая вычислительная сложность и использование большего объема памяти.

СНС можно разделить на параметризатор и классификатор. Первый представляет собой набор сверточных слоев и слоев подвыборки разной конфигурации, второй – полносвязную искусственную нейронную сеть (ПИНС). Обычно классификатор имеет в своей архитектуре 3–4 полносвязных слоя, в то время как параметризатор может иметь сотни и даже тысячи слоев [7, 8]. Методы обучения сверточных нейронных сетей как первого, так и второго порядков, используют понятие градиента, через которое могут быть связанны. Таким образом, методы второго порядка целесообразно применить к классификатору, получив на входе классификатора градиент, который будет передан методам первого порядка для оптимизации параметризатора.

В литературе в явном виде отсутствуют сведения о применении методов обучения второго порядка к СНС, в связи с этим необходимо рассмотреть эквивалентность сверточных и полносвязных слоев для адаптации вышеуказанных алгоритмов к СНС. Методы обучения первого порядка имеют ряд существенных недостатков, связанных с медленной скоростью обучения, возникновением эффекта паралича сети и др., что обуславливает актуальность научно-прикладных исследований по применению методов второго порядка к СНС.

Целью работы является разработка гибридного метода обучения СНС в задачах классификации объектов на цифровых изображениях, позволяющего эффективно обучать СНС, достигая заданного критерия останова.

2. Анализ эквивалентности СНС и ПИНС. Рассмотрим подробнее архитектуру сверточных слоев для анализа их эквивалентности полносвязным слоям. На рисунке 1 приведена схема сверточного слоя (CL).

Каждая карта CL-слоя может быть связана с некоторым количеством карт предыдущего слоя. Из рис. 1 видно, что карта CL связана с двумя картами предыдущего слоя или входного слоя, которые располагаются перед ней. Каждый нейрон CL имеет рецептивное поле (РП) сразу на двух картах в идентичных позициях предыдущего слоя. На рисунке 1 РП размером 2×2 обозначены сплошной линией и штрих-пунктиром. Значит, для нейрона CL необходимо 8 настраиваемых параметров (весов) с учётом смещения (*bias*). Особенность CL заключается в том, что для всех положений РП используются одни и те же веса. Такие веса называются связанными (*sharing weights*). Пунктиром на рис. 1 обозначено РП для следующего нейрона. Эти РП пересекаются, шаг пересечения регулируется на этапе построения СНС. РП и общие веса – это встроенная априорная информация, которая является составной частью гиперпараметров, за счёт которой происходит выделение признаков. Нейроны CL имеют структуру, представленную на рисунке 2.

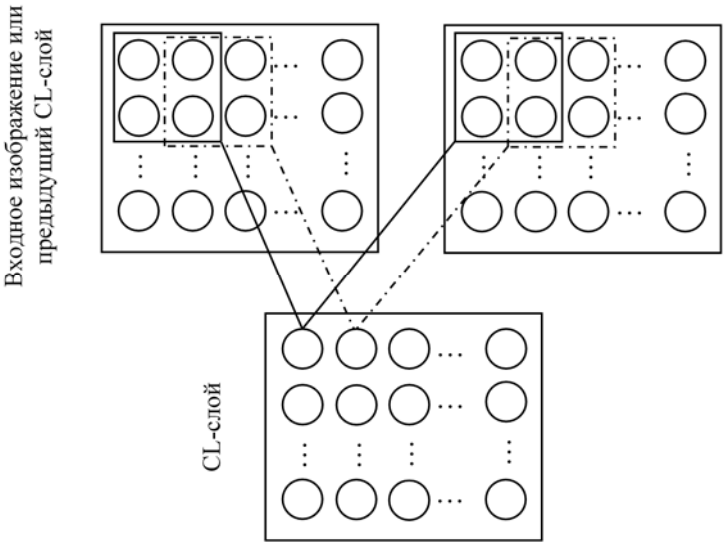

Рис. 1. Схема сверточного слоя (CL)

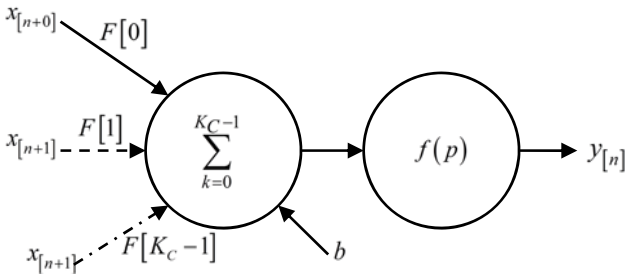

Рис. 2. Структура нейрона CL

На рисунке 2 K_C – общее количество нейронов или элементов входного вектора, входящих в РП *n*-го нейрона CL; $F[k]$ – настраиваемые веса нейрона; b – смещение *n*-го нейрона, причём b и $F[k]$ – одни и те же для всей карты CL; $x[n+k]$ – входные данные для *n*-го нейрона CL $k = 0 \ldots K_C - 1$. Входные данные $x[n+k]$ дают взвешенную сумму с настраиваемыми параметрами $F[k]$:

$$p = b + \sum_{k=0}^{K_C - 1} F[k] x_{n+k}. \quad (1)$$

Ядро выбирается исходя из размеров изображения, поступающего на вход СНС, шага сканирующего окна и заданной вычислительной сложности СНС. В общем случае ядро может быть любой прямоугольной матрицей, включая 1×1 [9]. Отклик нейрона определяет функция активации, на вход которой поступает взвешенная сумма *p*. Наиболее часто [1, 9, 7] используется семейство кусочно-линейных функций (*Rectified Linear Unit*, *ReLU*). Типичная кусочно-линейная функция выглядит следующим образом:

$$f(x) = \max(0, x), \quad (2)$$

где x – отклик нейрона; f – кусочно-линейная функция активации.

Рассмотрим одно ядро F CL. Пусть вход CL X имеет размерность $3 \times 6 \times 4$, где первое измерение обозначает количество карт признаков во входных данных (например, цветовые каналы, в случае поступления на вход изображения). Сверточное ядро положим размерностью 2×2, смещения $S = 2$, заполнение нулями $P = 0$ по каждому измерению. Для каждой ячейки выходной карты признаков необходимо произвести операцию:

$$a = f\left(\sum_{c=0}^{2} \sum_{i=0}^{2} \sum_{j=0}^{1} X_{c, i \cdot 2 + i, j \cdot 2 + j} \odot F + b \right), \quad (3)$$

где $i \cdot 2 + i, j \cdot 2 + j$ – способ выбора подматрицы X; $\odot$ – поэлементное умножение. Индексы матриц считаются с «0». Заметим, что необходимо вычислить 6 положений сканирующего окна для одного канала. Данное вычисление представлено графически на рисунке 3.

На рисунке 4 представлена схема выбора значений внутри сканирующего окна [1]: стрелками изображен способ преобразования двумерного изображения к одномерной последовательности пикселей, в рамках сканирующего окна.

Приведем схему из рисунка 3 к полносвязному виду (рис. 5), используя схему выбора значений внутри скользящего окна, приведенную выше.

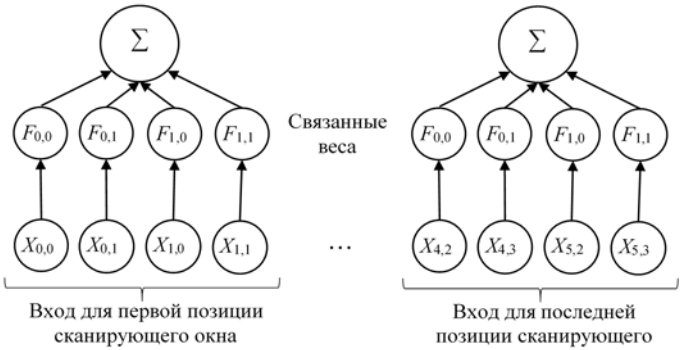

Рис. 3. Схема вычисления отклика без активации для одного сверточного ядра и одной карты признаков входных данных

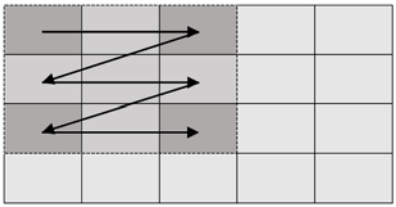

Рис. 4. Схема выбора значений внутри сканирующего окна

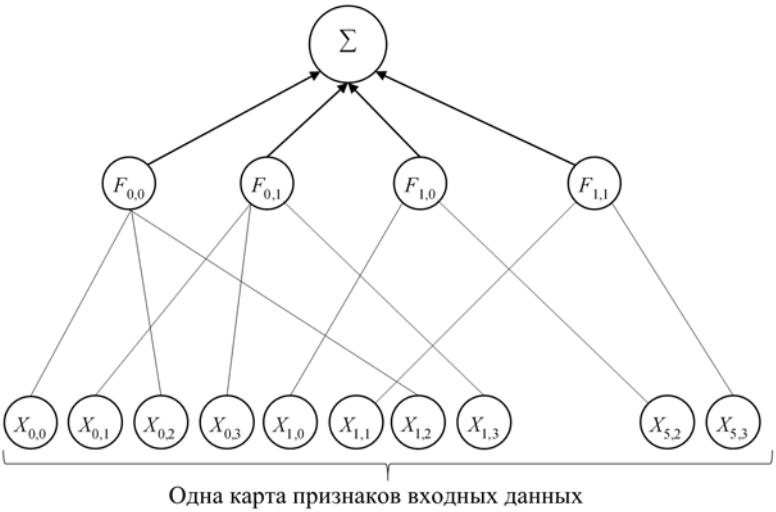

Рис. 5. Схема вычисления отклика без активации для одного сверточного ядра и одной карты признаков входных данных в виде полносвязного слоя

Как видно из рисунка 5, операцию вычисления для одного сверточного ядра и одной карты признаков входных данных можно представить как разреженный полносвязный слой. В данном случае каждый вес сверточного ядра участвует только в 6 операциях, вместо 24. Определим оператор $M(\cdot)$, преобразующий вычисление для одного сверточного ядра и одной карты признаков входных данных к вычислению разреженного полносвязного слоя:

$$M\left(CL_F^C(X,F)\right) \to MLP(X,F), \tag{4}$$

где C – конкретная карта признаков входных данных; F – ядро для данной карты. Оператор M преобразует входную карту признаков к виду вектор-строки, а также преобразует веса ядра F к виду матрицы размером $[f_1 f_2 \times s_1 s_2]$, где s_1, s_2 – количество смещений, получаемое при проходе сканирующим окном по вертикали и горизонтали соответственно; f_1, f_2 – размерность ядра по вертикали и горизонтали соответственно. Используя (3) и свойство коммутативности умножения, группа преобразований входных данных будет иметь вид:

$$X' = \sum_{i=0}^{2}\sum_{j=0}^{1} X_{i\cdot 2+i,\, j\cdot 2+j},\ X[2\times 2] \to X'[1\times 4], \tag{5}$$

или, в общем случае:

$$X' = \sum_{i=0}^{s1}\sum_{j=0}^{s2} X_{i\cdot f1+i,\, j\cdot f2+j},\ \ X' \propto F \to X'[1\times f_1 f_2], \tag{6}$$

Вторая группа преобразований в операторе M – преобразование ядра к виду вектор-строки. Данное преобразование выглядит следующим образом:

$$F'_C = F_C \to F'_C[f_1 f_2 \times 1];\ \ F'_C = IF'_C, \tag{7}$$

где I – диагональная единичная матрица размерностью $[f_1 f_2 \times f_1 f_2]$. Следует отметить, что веса вне основной диагонали не учувствуют в обучении (*frozen parameters*).

Используя принцип преобразования (см. рис. 4), получим вычисление одного сверточного ядра и одной карты признаков входных данных к вычислению разреженного полносвязного слоя:

$$M\left(CL_F^C\left(X,F\right)\right):\begin{cases} X'_C = \sum_{i=0}^{s1}\sum_{j=0}^{s2} X_{C,i\cdot f1+i,j\cdot f2+j},\ \ X'_C \propto F_C \to X'_C\left[1\times f_1 f\right. \\ F'_C = F_C \to F'_C\left[f_1 f_2 \times 1\right];\quad F'_C = IF'_C; \\ p_C = X'_C F'_C. \end{cases} \tag{8}$$

Рассмотрим операцию по вычислению свертки для всех карт признаков входных параметров, преобразуем выражение (3) с учетом (8):

$$\begin{aligned} a &= f\left(\sum_{c=0}^{2}\sum_{i=0}^{2}\sum_{j=0}^{1} X_{c,i\cdot 2+i,j\cdot 2+j} \odot F + b\right) = \\ &= f\left(\sum_{c=0}^{2} M\left(CL_F^c\left(X,F\right)\right)\right). \end{aligned} \tag{9}$$

Графически данная операция представлена на рисунке 6.

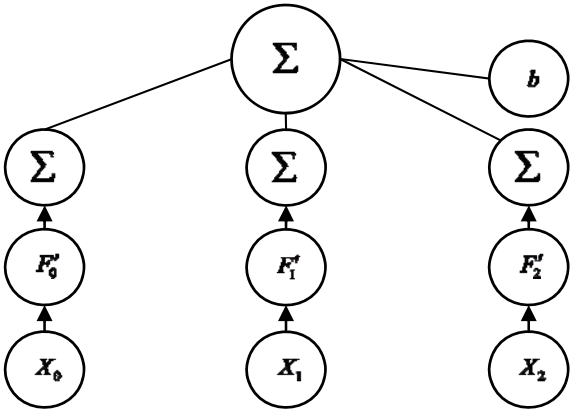

Рис. 6. Операция по вычислению свертки для всех карт признаков входных параметров

Данная операция аналогична вычислению 3х полносвязных слоев с последующим поэлементным сложением. Введем оператор $D(\cdot)$, выполняющий следующее преобразование:

$$D(X,F):\sum_{c=0}^{2} M\left(CL_F^C\left(X,F\right)\right)+b. \tag{10}$$

Таким образом, для обработки одного сверточного слоя необходимо вычислить C независимых полносвязных разряженных слоя. Обработка сверточного слоя сводится к:

$$a=f\left(\sum_{c=0}^{2}\sum_{i=0}^{2}\sum_{j=0}^{1} X_{c,i\cdot 2+i,j\cdot 2+j}\odot F_c+b\right)= \\ =f\left(D(X,F)\right). \tag{11}$$

в общем виде:

$$a=f\left(\sum_{c=0}^{C}\sum_{i=0}^{s1}\sum_{j=0}^{s2} X_{c,i\cdot s1+i,j\cdot s2+j}\odot F_c+b\right)= \\ =f\left(D(X,F)\right). \tag{12}$$

Данное преобразование позволяет более тонко настраивать свертку, вводя в диагональную матрицу весов из (8) дополнительные веса вне главной диагонали.

Рассмотрим обратное распространение ошибки через CL-слой, представленный оператором D. Данная операция аналогична классическому обратному распространению ошибки. Для экономии ресурсов целесообразно использовать маску:

$$\Delta M\left(CL_F^C\left(X,F\right)\right)=\Delta MLP\odot m. \tag{13}$$

где m – маска, формируемая по следующему принципу:

$$m_{i,j}=\begin{cases}1, & \textit{если } i=j\\ 0 & \textit{иначе}\end{cases}. \tag{14}$$

Учитывая замечание о том, что для более точной настройки свертки в (8) вводятся дополнительные веса вне главной диагонали, маска изменяется в соответствии с введенными весами. Учитывая, что матричное умножение разряженных матриц – дорогая операция, её следует заменить поэлементным умножением, градиенты в таком случае вычисляются на основе концепции вычислительного графа [10].

Таким образом, продемонстрирована эквивалентность СНС и ПИНС с разряженной матрицей весов.

3. Анализ методов обучения искусственных нейронных сетей. В настоящее время наиболее используемым методом обучения искусственных нейронных сетей (ИНС) является метод обратного распространения ошибки (его модификации, например Adam [3]). Для дальнейшего применения следует рассмотреть классический метод обратного распространения ошибки, ввиду того, что модификации не изменяют его изначальной идеи. При рассмотрении будет использоваться наиболее простая функция ошибки – среднеквадратичная ошибка. Пусть ИНС имеет множество входных нейронов $x_1,...,x_n$, множество выходных нейронов $y_1,\ldots,y_n$ и множество скрытых нейронов. Перенумеруем все узлы (включая входы и выходы) числами от 1 до N (сквозная нумерация, вне зависимости от топологии слоёв). Обозначим через $w_{i,j}$ вес, находящийся на ребре, соединяющем i-й и j-й узлы, а через r_i – выход i-го узла. Если известен обучающий пример – правильные ответы сети, такие что $y_k, k \in Y$. В данном случае функция ошибки будет иметь вид [2]:

$$E\left(\left\{w_{i,j}\right\}\right) = \frac{1}{2}\sum_{k\in Y}\left(t_k - r_k\right)^2. \tag{15}$$

Чтобы достигнуть минимума ошибки, необходимо двигаться в сторону, противоположную градиенту, то есть, на основании каждой группы правильных ответов, добавлять к каждому весу $w_{i,j}$:

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}. \tag{16}$$

Производная функции ошибки считается следующим образом. Пусть сначала $j \in Y$, то есть вес входит в нейрон последнего слоя. Следует отметить, что $w_{i,j}$ влияет на выход сети только как часть суммы:

$$S_j = \sum_i w_{i,j} x_i, \tag{17}$$

где сумма берется по входам j-го узла. Из этого следует:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j}. \tag{18}$$

Аналогично, S_j влияет на общую ошибку только в рамках выхода j-го узла r_j. Поэтому [2]:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial r_j} \frac{\partial r_j}{\partial S_j} = \left(\frac{\partial}{\partial r_j} \frac{1}{2} \sum_{k \in Y} (t_k - r_k)^2 \right) \left(\frac{\partial f(\mathrm{S})}{\partial S} \Big|_{S=S_j} \right) = \\ = -2\alpha r_j (1 - r_j)(t_j - r_j), \tag{19}$$

где $f(S)$ – функция активации (в рассматриваемом случае экспоненциальная сигмоида). Если же j-й узел не на последнем слое, то у него есть выходы. Обозначим их через $C(j)$. В этом случае:

$$\frac{\partial E}{\partial S_j} = \sum_{k \in C(j)} \frac{\partial E}{\partial S_k} \frac{\partial S_k}{\partial S_j}, \tag{20}$$

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial r_j} \frac{\partial r_j}{\partial S_j} = w_{j,k} \frac{\partial r_j}{\partial S_j} = 2\alpha w_{j,k} r_j (1 - r_j). \tag{21}$$

Описанный подход (17-21) имеет ряд недостатков. Во-первых, явление паралича сети. В процессе обучения сети значения весов могут, в результате коррекции, стать очень большими величинами. Данный факт может привести к тому, что все или большинство нейронов будут функционировать при очень больших выходных значениях в области, где производная функции активации мала. Так как посылаемая обратно в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может практически замереть. В теоретическом отношении эта проблема плохо изучена. Обычно этого избегают уменьшением размера шага η, но это увеличивает время обучения. Различные эвристики [1, 2, 11] позволяют избежать паралич или восстановить процесс обучения после него, но пока что они могут рассматриваться лишь

как экспериментальные. Во-вторых, алгоритм чувствителен к локальным минимумам. Обратное распространение использует разновидность градиентного спуска, то есть осуществляет спуск вниз по гиперповерхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Гиперповерхность ошибки сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум, когда рядом имеется гораздо более глубокий минимум. В точке локального минимума все направления ведут вверх, и сеть неспособна из него выбраться. Основную трудность при обучении нейронных сетей составляют как раз методы выхода из локальных минимумов: каждый раз выходя из локального минимума снова ищется следующий локальный минимум тем же методом обратного распространения ошибки до тех пор, пока найти из него выход уже не удаётся.

Методы второго порядка получили свое название благодаря использованию второй производной функции ошибки, а также аппроксимированного гессиана ИНС. В общем случае сущность данных методов можно выразить с помощью следующего соотношения [12]:

$$W_n = W_{n-1} - \left(H^{-1} \cdot g\right), \tag{22}$$

где W – веса ИНС; n – текущая эпоха обучения; H – гессиан функции ошибки; g – вектор градиента функции ошибки. Рассмотрим подробнее каждую составляющую данного выражения. Веса ИНС W представляют собой одномерный вектор, в который последовательно включены веса и смещения всех слоев ИНС. Градиент g представляет собой вектор, содержащий производную ошибки по каждому из весов и смещений [5]:

$$g = \nabla E = \begin{bmatrix} \dfrac{\partial E}{\partial w_1} \\ \vdots \\ \dfrac{\partial E}{\partial w_k} \end{bmatrix}, \quad w \in W. \tag{23}$$

Гессиан по определению [12] является матрицей вторых производных функции ошибки E:

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_k \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_k \partial w_k} \end{bmatrix}. \quad (24)$$

Вычисление гессиана является крайне затратной вычислительной процедурой. Так как необходимо вычислить k^2 вторых производных многомерной функции, где k –количество весов и смещений в ИНС.

Наиболее эффективным методом из данного класса является метод Левенберга-Марквардта, заключающийся в аппроксимации гессиана с помощью якобиана. Гессиан аппроксимируется следующим образом [5]:

$$H \approx J^T J + \mu I, \quad (25)$$

где H – гессиан; J – якобиан; μ – параметр регуляризации, изменяющийся в процессе выполнения алгоритма; I – единичная матрица размерностью $J^T J$, в некоторых источниках [5] предлагается использование диагональных элементов якобиана вместо единичной матрицы. Однако при проведении экспериментов значительные отличия применения единиц или элементов якобиана на диагонали I обнаружены не были.

Градиент (23) в данном методе вычисляется следующим образом:

$$g = \nabla E = J^T e;$$
$$e = d - r = \begin{bmatrix} e_{11} \\ \vdots \\ e_{M1} \\ e_{12} \\ \vdots \\ e_{MP} \end{bmatrix}, \quad (26)$$

где e –векторизованная ошибка; d – истинное значение выхода ИНС; r – отклик ИНС; M – количество выходов; P – количество объектов в обучающей выборке.

Рассмотрим составление матрицы якобиана. Так как якобиан – это матрица частных производных функции ошибки по всем настраиваемым параметрам модели, в случае ИНС – весам и смещениям, на

всей обучающей выборке [13], то размер данной матрицы составит $(P \cdot M) \times T$, где T – количество весов и смещений ИНС. В общем случае на каждой строке якобиана содержатся частные производные функции ошибки по всем весам и смещениям модели для конкретного выхода и объекта из обучающей выборки. Формально это выглядит следующим образом:

$$J = \begin{bmatrix} \frac{\partial e_1(P_1)}{\partial W_1} & \cdots & \frac{\partial e_1(P_1)}{\partial W_T} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_{MP}(P_P)}{\partial W_1} & \cdots & \frac{\partial e_{MP}(P_P)}{\partial W_T} \end{bmatrix}. \quad (27)$$

Следует отметить, что для применения метода Левенберга-Марквардта удобнее представлять смещения как дополнительный вес нейрона, на вход которого всегда подается единица, подробнее данный подход к вычислению смещений рассмотрен в [2]. Для вычисления отдельных элементов якобиана необходимо модифицировать метод обратного распространения ошибки. Введем в (17) обозначение слоя и произведем замену входных данных на отклик предыдущего слоя для общности изложения:

$$s_j^l(p) = \sum_i w_{ji}^l y_i^{l-1}(p), \quad (28)$$

где y – отклик предыдущего слоя, p – текущий объект из обучающей выборки P; l – текущий слой ИНС; j – текущий нейрон в j-ом слое; i – вес j-го нейрона. Отклик текущего слоя выражается следующим образом:

$$y_j^l(p) = f(s_j^l(p)), \quad (29)$$

где $f(\cdot)$ – функция активации. Используя (27) и применяя правило дифференцирования по частям [11], элемент якобиана будет иметь вид:

$$\frac{\partial e_m(p)}{\partial w_{ji}^l} = -\frac{\partial r_m(p)}{\partial y_j^l(p)} \cdot \frac{\partial y_j^l(p)}{\partial s_j^l(p)} \cdot \frac{\partial s_j^l(p)}{\partial w_{ji}^l}, \quad (30)$$

где m – текущий выход из набора M.

После вычисления матрицы якобиана для изменения весов ИНС необходимо выполнить процедуру, аналогичную обратному распространению ошибки [5]. Для проведения операции обратного распространения ошибки необходимо вычислить производную функции, описывающей преобразование сигнала между выходом нейрона *j* слоя *l* и выходом сети *m* на объекте из обучающей выборки *p*:

$$\delta_{mj}^{l} = -\partial F_{mj}^{l}(p) \cdot \partial f_{j}^{l}(p), \tag{31}$$

где $\partial f_{j}^{l}(p)$ – значение производной функции активации для нейрона *j* слоя *l* на объекте из обучающей выборки *p*. Для нейрона *j* выходного слоя $l = L$, выхода сети *m* и объекта обучающей выборки *p* значение производной описываемой функции определяется следующим образом [5]:

$$\partial f_{mj}^{l=L}(p) = \begin{cases} -\partial f_{j}^{l=L}(p), & \textit{если} \quad m = j \\ 0, & \textit{иначе} \end{cases}. \tag{32}$$

Для полного описания производной функции, описывающей преобразование сигнала между выходом нейрона *j* выходного слоя $l = L$ и выходом сети *m* на всех объектах обучающей выборки *P*, необходимо *P* диагональных матриц $M \times M$, где *M* – количество выходов ИНС. После вычисления производной передаточной функции для выходного слоя последовательно вычисляются передаточные функции для всех остальных слоев:

$$\delta_{mk}^{l}(p) = \sum_{i(l+1)} \left(w_{ki}^{l+1} \cdot \delta_{mk}^{l}(p) \right) \cdot \partial f_{k}^{l}(p), \tag{33}$$

где *k* – нейрон скрытого слоя *l*, *i* – скрытый нейрон последующего слоя *l*+1. Выражение (33) может быть записано в матричной форме (для каждого объекта *p* из обучающей выборки) [5]:

$$\delta^{l}(p) = \delta^{l+1}(p) \cdot W^{l} \odot \left(\partial f^{l}(p) \right)^{T}, \tag{34}$$

где $\odot$ – поэлементное умножение матриц. Обобщенная схема алгоритма обучения Левенберга-Марквардта представлена на рисунке 7.

Алгоритм Левенберга-Марквардта наиболее устойчив к изменению начальной инициализации параметров ИНС, а также наиболее быстро сходится [14]. Недостатком данного алгоритма является высокая вычислительная сложность.

Для корректного сравнения алгоритмов обучения ИНС следует сравнить их основные характеристики: вычислительную сложность и используемое количество ячеек памяти.

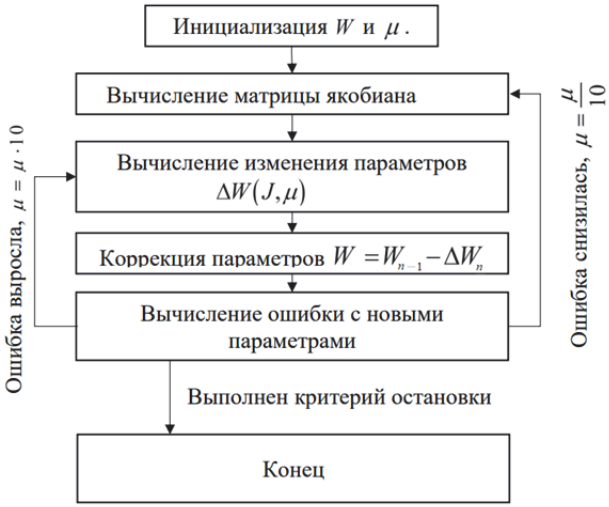

Рис. 7. Обобщенная схема алгоритма обучения Левенберга-Марквардта

4. Гибридный алгоритм обучения ИНС. Предложенная модификация заключается в гибридизации двух методов – Левенберга-Марквардта [5] и Adam [3]. Сочетание методов разных порядков возможно благодаря тому факту, что в якобиане и значении ошибки в косвенном виде присутствуют значения градиентов (26). Таким образом, вычисляется градиент на классификаторе СНС, а далее информация о градиентах передается в метод первого порядка Adam для вычисления весов сверточной части сети (параметризатора). Данная гибридизация методов обучения позволяет увеличивать эффективность шага изменения параметров в полносвязной части СНС, а также избегать излишней вычислительной сложности при вычислении изменения параметров сверточной части. Схема гибридного алгоритма обучения ИНС приведена на рисунке 8.

Рис. 8. Схема гибридного алгоритма обучения ИНС

На первом этапе, как и во всех алгоритмах обучения ИНС [1, 2], инициализируются параметры СНС – веса W и смещения b. При реализации данного алгоритма использовалась инициализация весов, описанная в [8]. Далее выполняются вычисления откликов СНС и ошибки

классификации. Затем выполняется цикл вычисления приращений параметров *W, b* классификатора с помощью (22), (25)-(34).

Градиенты, вычисленные на данном шаге, сохраняются в памяти для последующего использования. После вычисления приращений параметров *W, b* классификатора градиенты *g* передаются в параметризатор и вычисляются приращения параметров *W, b* параметризатора с помощью метода Adam. В конце итерации гибридный алгоритм вычисляет ошибку классификации при измененных весах и сравнивает её с критерием останова. В реализации использовались критерии [14]:

$$10\log\left(\frac{1}{4Qp}\right), \tag{35}$$

где Q – количество объектов в обучающей выборке; p – количество выходов СНС. Данный критерий выбирался в связи с тем, что при инициализации сети ошибки распознавания составляли порядка 100% [14]. Следует отметить, что при использовании мини-пакетов (mini-batches) критерий останова рассчитывается от количества всех объектов в обучающей выборке, а не от количества объектов в мини-пакете. Ошибка обучения данного метода приводилась к показателю в децибелах:

$$E^{dB} = 10\log\left(\frac{\sum\limits_{k\in p}\left(y_k^0 - u_k\right)^2}{\sum\limits_{k\in p}\left(y_k^n - u_k\right)^2}\right), \tag{36}$$

где n – текущая эпоха обучения; u – отклик при инициализации сети.

Предложенный алгоритм позволяет производить обучение СНС как в режиме обработки всех обучающих данных одновременно, так и в режиме мини-пакетов. Для использования режима мини-пакетов необходимо арифметически усреднить приращения параметров W, b по всем мини-пакетам.

Для оценки сложности рассмотренных алгоритмов введем следующие обозначения [15]: α – количество операций сложения, вычитания и сравнения; β – количество операций умножения и деления; γ – количество нелинейных операций логарифмирования, вычисления

экспоненты. В таблице 1 приведены полученные аналитические оценки на основе учета количества требуемых тактов для каждой операции сложности составных частей для алгоритмов обучения СНС. Степень соответствия полученных оценок подтверждается экспериментальным сравнением приведенных оценок с реализацией ИНС (с использованием библиотеки тензорных вычислений TensorFlow). Результаты сравнения приведены на рисунке 9.

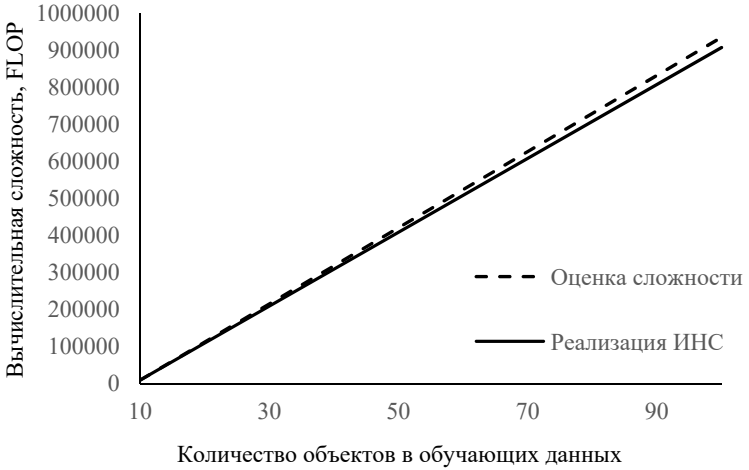

Рис. 9. Сравнение полученных оценок вычислительной сложности с реализацией ИНС

На рисунке 10 приведен график зависимости возрастания вычислительной сложности рассмотренных алгоритмов в зависимости от размера батча и количество скрытых нейронов в последнем скрытом слое для СНС с архитектурой:

– входные данные: тензор $15 \times 10 \times 1$;

– 3 сверточных ядра 5×5 со смещением 5 в первом CL;

– слой подвыборки 3×2 с единичным смещением в MP слое;

– 2 нейрона в первом скрытом слое;

– 1 выходной нейрон.

Из рисунка 10 видно, что данная модификация алгоритма позволяет обучать СНС имущие любую архитектуру в то время, как классический метод Левенберга-Марквардта не позволяет работать с относительно большими СНС – требуется производить операции

с большими матрицами (размерностью количество параметров × количество обучающих объектов).

Таблица 1. Вычислительная сложность слоев СНС

Слои СНС	Аналитическая оценка сложности
Прямое вычисление СНС	
Полносвязный слой	$b\left[n_i n_{i-1}(\alpha+\beta)+n_i(\beta+\gamma)\right]$ b_i – количество объектов в батче; n_i – количество нейронов на i слое.
Слой подвыборки	$bCWHm_w m_h(\alpha)$; $W=\frac{M_w-F_w}{S_w}+1$; $H=\frac{M_h-F_h}{S_h}+1$. C – количество каналов в предыдущем слое; M – входной тензор; h, w – высота и ширина соответственно; S – смещение; m – окно подвыборки.
Сверточный слой	$b\left[KCWH(\alpha+\gamma)+KCWHf_w f_h(\beta)\right]$; $W=\frac{M_w-F_w+2P_w}{S_w}+1$; $H=\frac{M_h-F_h+2P_h}{S_h}+1$. K – количество ядер; f – матрица ядра; S – смещение; P – заполнение нулями.
Adam	
Выходной полносвязный слой	$\delta=(d-y)N(1-N)$; $b\left[nn_{i-1}(10\alpha+13\beta+2\gamma+2t\beta)\right]$, δ – составляющая градиента без умножения на входной вектор; N – отклик сети.
Полносвязный слой	$\delta=\delta_{i+1}W$; $n_i n_{i-1}(10\alpha+15\beta+2\gamma+n_i n_{i+1}\alpha+n_i n_{i+1}\beta+2t\beta)$, n_i – количество нейронов в слое i, W – веса предыдущего слоя.
Слой подвыборки	Нет настраиваемых параметров

Продолжение таблицы 1

Слои СНС	Аналитическая оценка сложности
Сверточный слой	$\delta = \delta_{i+1} w_{i+1} \max(0, CL);$ $Kf_w f_h \left[7\alpha + 9\beta + 2t\beta + KCWH(\alpha + \gamma) + KCWHf_w f_h(\beta) \right];$ $W = \frac{M_w - F_w + 2P_w}{S_w} + 1;$ $H = \frac{M_h - F_h + 2P_h}{S_h} + 1,$ CL – отклик сверточного слоя; w – веса ядра сверточного слоя.
Метод Левенберга-Марквардта	
Построение Якобиана	Выходной слой: $J_{вых} = bp \cdot [n_{i-1}\beta];$ слой i: $J_i = bp\left[2n_i n_{i-1}^2 \left[\sum_{k=0}^{i-1} k n_{k-1}\beta + \beta \right] \right],$ J_i – якобиан для i-го слоя; p – количество объектов в батче.
Аппроксимация якобиана	$H \approx J^T J + \mu I : l^{3/2} \log l + l^{2.37} + l\beta + l\alpha;\ l = \max\left(b \cdot p, w^{общ}\right.$ H – якобиан; l – количество параметров СНС. Оценка $l^{3/2}\log(l)$ является оценкой транспонирования, оценка $l^{2.37}$ – оценка матричного умножения [16]: $w^{общ}$ – общее количество настраиваемых параметров СНС.
Общая вычислительная сложность	$K_{СЛМ} = \sum_{i=0}^{L} \left(bp \cdot [n_{i-1}\beta] + (L-1) bp \left[2n_i n_{i-1}^2 \left[\sum_{k=0}^{i-1} k n_{k-1}\beta + \beta \right] \right] \right.$ $+ \left(l^{3/2} \log l + l^{2.37} + l\beta + l\alpha \right) + M\left(l\beta + l\alpha + F_B + E + \beta \right) +$ $+ pmba;$ $E = (pb\alpha)^{2.37} + 2\beta + pb\beta,$ где L – общее количество слоев СНС, E –вектор ошибки.

Продолжение таблицы 1

Слои СНС	Аналитическая оценка сложности
Предложенный гибридный метод	
Общая вычислительная сложность	$K_{СЛМ} = \sum_{i=0}^{O}\left(bp\cdot[n_{i-1}\beta] + (O-1)bp\left[2n_i n_{i-1}^2 \left[\sum_{k=0}^{O-1} kn_{k-1}\beta + \beta \right] \right] \right) + + \left(l_O^{3/2}\log l_O + l_O^{2.37} + l_O\beta + l\alpha \right) + M\left(l_O\beta + l_O\alpha + F_B + E + \beta \right) + + pmba + Adam[L-O],$ O – количество слоев обучаемых методом Левенберга-Марквардта; L – общее количество слоев СНС; $Adam$ – вычислительная сложность для метода Adam.

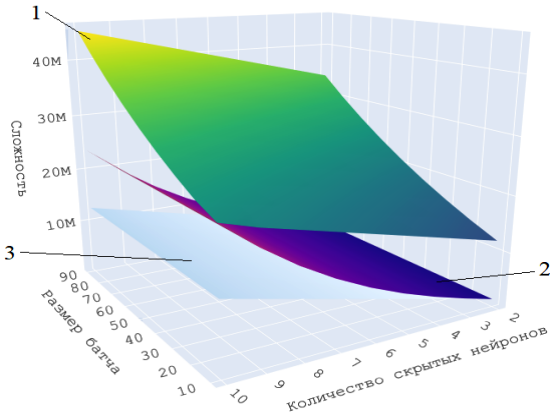

Рис. 10. Зависимость вычислительной сложности от размера батча и количества скрытых нейронов: 1 – сложность метода Левенберга-Марквардта; 2 – сложность предложенного гибридного метода; 3 – сложность метода Adam

5. Анализ эффективности гибридного алгоритма обучения ИНС. Для анализа предложенного гибридного алгоритма была разработана программа на языке python. Использовалась библиотека тензорных вычислений TensorFlow [10]. В качестве задач использовались следующие наборы данных: Iris [1] – набор данных 4 значимых признака, 150 объектов в обучающей выборке, 3 выхода ИНС; MNIST [17] – 784 значимых признака; 60000 объектов в обучающей выборке, 10 выходов ИНС. Результаты ошибки обучения для набора Iris приведены на рисунке 11. Архитектура для решения задачи:

– 4 сверточных ядра 2×2 в первом CL;

– 4 нейрона в первом скрытом слое;

– 3 выходных нейрона.

На рисунке 11 изображены только 100 эпох обучения, так как обучение ИНС методом Adam не улучшало ошибку ниже –23 дБ. Следует отметить, что предложенный метод эффективен: обладает высокой сходимостью и скоростью (по сравнению с Adam), а также мало отличается по сходимости (не более чем на 10 дБ на протяжении всего обучения) от метода Левенберга-Марквардта. Вычислительно предложенный алгоритм достаточно сложен, однако его сложность возможно контролировать путем изменения количества слоев, обучаемых методом Левенберга-Марквардта, например, если классификатор имеет в своей архитектуре более 1000 нейронов, целесообразно обучать методом Левенберга-Марквардта только два-три выходных слоя классификатора. В предельном случае, когда число параметров во всей СНС менее 2000, предложенный гибридный алгоритм сводится к методу Левенберга-Марквардта. Подобная гибкость позволяет применять предложенный гибридный алгоритм как на мощных вычислительных серверах, так и на конечных устройствах, изменяя границу перехода между методом первого и второго порядка в зависимости от производительности вычислительных средств. В таблице 2 приведены полученные аналитические оценки сложности вычисления алгоритмов обучения СНС для рассмотренных случаев.

За основу расчета количества тактов процессора на выполнение операций были выбраны следующие усредненные значения: сложение и вычитание (α) – 4 такта; умножение и деление (β) – 20 тактов; нелинейные операции (γ) – 70 тактов. Оценка времени производилась на основе технических параметров процессора Intel Core i7-4930K (Ivy Bridge), 3,4 ГГц, 6 ядер (2013 г.) с пиковой производительностью Z=163 ГФлопс/с. Для оценки времени использовалась следующая формула:

$$T = \frac{K_{вс^*}}{Z \cdot V}, \tag{38}$$

где V – загруженность процессора конкретной задачей; Z – пиковая производительность. В расчетах V=0.75.

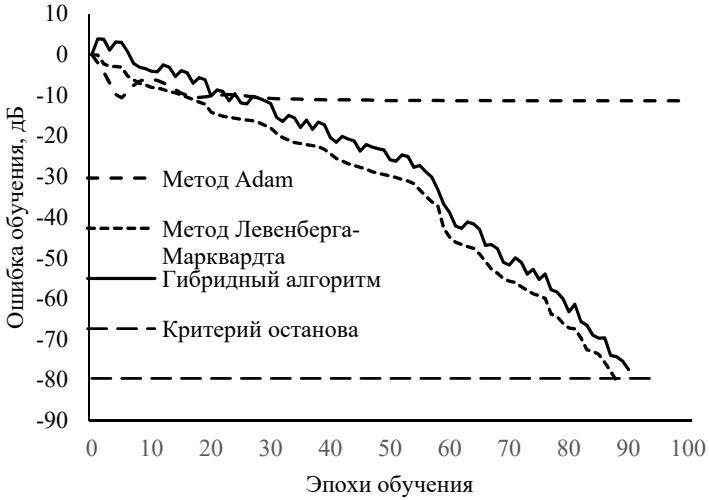

Рис. 11. Ошибка обучения ИНС различными методами на наборе данных iris

Таблица 2. Вычислительная сложность распознавания и обучения СНС

Операция	Такты (усредненное значение)	Время выполнения
Распознавание	2172	$2.6 \cdot 10^{-8}$ с.
Обучение Adam (1 эпоха)	1890400	$2.31 \cdot 10^{-5}$ с.
Обучение Левенберг-Марквардт (1 эпоха)	14658396	$1.79 \cdot 10^{-4}$ с.
Обучение гибридный алгоритм (1 эпоха)	12252313	$1.20 \cdot 10^{-4}$ с.

Рассмотрим более сложную задачу классификации изображений – набор данных MNIST. Это набор рукописных цифр, включающий в себя 60000 образцов. Гибридный метод обучения применялся в режиме мини-пакетов, результаты приведены на рисунке 12. Архитектура для решения задачи: 4 сверточных ядра 3×3 со смещением 2 в первом CL; подвыборка 2×2 со смещением 2; 5 сверточных ядер 3×3 со смещением 2 во втором CL; подвыборка 2×2 со смещением 2; 12 нейронов в первом скрытом слое; 11 нейронов во втором скрытом слое; 10 выходных нейронов.

Подобную СНС затруднительно обучить, используя метод Левенберга-Марквардта ввиду того, что необходимо оперировать с матрицами размерностью более чем $5 \cdot 10^3$, однако предложенный гибридный алгоритм позволяет производить обучение. Следует отметить, что предложенный метод обеспечивает лучшую сходимость за эквивалентное время. Под эквивалентным временим понимается время, затраченное на вычисление обоих алгоритмов. В приведенном эксперименте эквивалентным временем выступает время 4000 эпох обучения методом Adam, соответствующее 20 эпохам обучения гибридного метода. При этом для обучения алгоритмом Adam наблюдается эффект паралича сети (сеть не обучилась), а гибридный метод обеспечил обучение до критерия останова –8 дБ, что соответствует 95% точности классификации. Аналогичные результаты наблюдались и на выборках большей размерности, например, для данных размерностью $224 \times 224 \times 3$ (цветные изображения военной техники) точность классификации составила 93.5% при эквивалентном времени расчета.

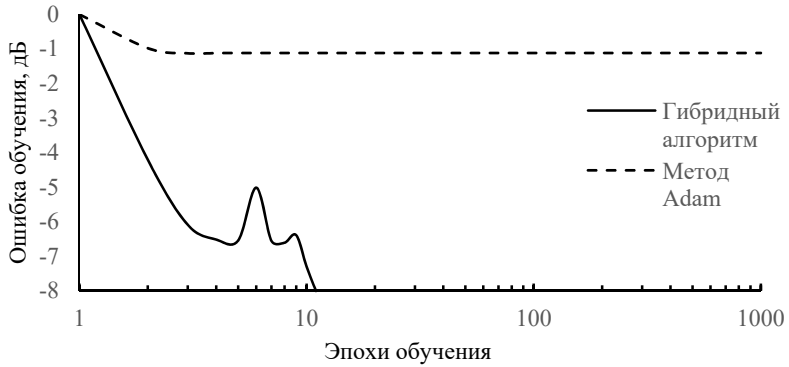

Рис. 12. Ошибка обучения ИНС различными методами на наборе данных MNIST (логарифмический масштаб по оси эпох)

Предложенный гибридный алгоритм обучения СНС эффективен при выполнении следующих условий: ограниченное время обучения на заданной архитектуре и достаточное количество ресурсов для проведения вычислений значений гессиана.

6. Заключение. В работе продемонстрирована эквивалентность СНС и ПИНС, предложен способ внесения дополнительных весов в структуру CL-слоев.

Предложен гибридный метод обучения СНС, совмещающий в себе метод Левенберга-Марквардта и Adam. Гибридный метод обучения СНС позволяет добиваться значительно лучшей сходимости по сравнению с Adam и требует меньше вычислительных операций для реализации. Используя предложенный метод возможно обучать сети, на которых происходит паралич обучения при использовании методов первого порядка. Более того, предложенный метод обладает способностью подстраивать свою вычислительную сложность под аппаратные средства, на которых производится вычисление, вместе с тем гибридный метод позволяет использовать подход обучения мини-пакетов.

Также в работе приведены результаты вычислительных экспериментов на различных наборах данных для оценки эффективности гибридного алгоритма обучения СНС. Показано, что предложенный алгоритм обучения позволяет достичь ошибки, отличающейся не более чем на 10 дБ от ошибки, полученной методом Левенберга-Марквардта.

Литература

1. *Гудфеллоу Я., Бенджио И., Курвилль А.* Глубокое обучение // М.: ДМК Пресс. 2017. 652 с.
2. *Хайкин С.* Нейронные сети. Полный курс // М.: Вильямс. 2006. 1104 с.
3. *Kingma D.P., Ba J.A.* A Method for Stochastic Optimization // CoRR. T. abs/1412.6980. 2014.
4. *Bo Y.H., WeiL., I-Chen W.* Stochastic Gradient Descent with Hyperbolic-Tangent Decay // CoRR. T. abs/1806.01593. 2018. pp. 1–10.
5. *Wilamowski B.M., Irwin D.J.* Intelligent systems // CRC Press. 2011. 568 p.
6. *Smith J.S., Wu B., Wilamowski B.M.* Neural Network Training With Levenberg-Marquardt and Adaptable Weight Compression // IEEE Transactions on Neural Networks and Learning Systems. 2018. pp. 1–8.
7. *Szegedy C., Ioffe S., Vanhoucke V., Alemi A.A.* Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning // International Conference on Learning Representations (ICLR) Workshop. 2016. pp. 375–387.
8. *He K., Zhang X., Ren S., Sun J.* Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification // arXiv.org e-Print archive. 2015. URL: https://arxiv.org/abs/1502.01852 (дата обращения: 12.11.2020).
9. *Szegedy C. et al.* Going deeper with convolutions // IEEE Conference on Computer Vision and Pattern Recognition. 2014. pp. 1–9.
10. *Zaccone G., Karim R., Menshawy A.* Deep Learning with TensorFlow: Explore neural networks with Python // Packt Publishing. 2017. 320 p.
11. *Вьюгин В.В.* Математические основы теории машинного обучения и прогнозирования // М.: МЦНМО. 2013. 387 с.
12. *Shepherd A.J.* Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons // Springer. 1997. 160 p.
13. *Nocedal J., Wright S.J.* Numerical Optimization // Springer. 2006. 664 p.
14. *Голубинский А.Н.* О построении архитектур и оценке параметров искусственных нейронных сетей // Теория и техника радиосвязи. 2020. № 1. С. 72–87.
15. *Максимушкин В.В., Арзамасцев А.А.* Сравнительная оценка вычислительной сложности обучения искусственной нейронной сети с жестким ядром и сети с

классической структурой // Вестник российских университетов. Математика., 2006. № 2. С. 190–197.
16. *Абрамов С.А.* Лекции о сложности алгоритмов // М.: МЦНМО. 2012. 248 с.
17. Yann LeCun's Home Page. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. 2012. URL: http://yann.lecun.com/exdb/mnist/ (дата обращения: 23.05.2020).
18. *Smith L.N.* No More Pesky Learning Rate Guessing Games // CoRR. 2015. pp. 1–10.
19. *Salimans T., Kingma D.P.* Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks // Advances in Neural Information Processing Systems 29. 2016. pp. 901–909.
20. *Matuszyk P., Castillo R.T., Kottke D., Spiliopoulou M.* A Comparative Study on Hyperparameter Optimization for Recommender Systems // Workshop on Recommender Systems and Big Data Analytics (RS-BDA'16). 2016.
21. *Hu G. et al.* Frankenstein: Learning Deep Face Representations using Small Data // IEEE Transactions on Image Processing. 2017. vol. 27. no. 1. pp. 293–303.
22. *Ioffe S., Szegedy C.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // International conference on machine learning. 2015. pp. 448–456.
23. *Lewkowycz A. et al.* The large learning rate phase of deep learning: the catapult mechanism. 2020.
24. *Liao Q., Kawaguchi K., Poggio T.A.* Streaming Normalization: Towards Simpler and More Biologically-plausible Normalizations for Online and Recurrent Learning // CoRR. T. abs/1610.06160. 2016.
25. *Mahajan D. et al.* Exploring the Limits of Weakly Supervised Pretraining // CoRR. T. abs/1805.00932. 2018.
26. *Petroski S.F. et al.* Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning // ArXiv e-prints. 2017.
27. *Su H., Zhu X., Gong S.* Deep Learning Logo Detection with Data Expansion by Synthesising Context // 2017 IEEE winter conference on applications of computer vision (WACV). 2017. pp. 530–539.
28. *Xu C., Qin T., Wang G., Liu T.Y.* An Actor-Critic Algorithm For Learning Rate 2017. pp. 1–12.
29. *Xu C., Qin T., Wang G., Liu T.Y.* Reinforcement Learning for Learning Rate Control // arXiv preprint arXiv:1705.11159. 2017.

Голубинский Андрей Николаевич – д-р техн. наук, доцент, заместитель научного руководителя, АО «Концерн «Созвездие». Область научных интересов: математическое моделирование систем с элементами искусственного интеллекта. Число научных публикаций – 205. annikgol@mail.ru; Московский пр., 92, 394068, Воронеж, Россия; р.т.: +79103436537.

Толстых Андрей Андреевич – преподаватель кафедры, кафедра специальных информационных технологий, Московский университет МВД России им. В.Я. Кикотя. Область научных интересов: искусственные нейронные сети, машинное обучение. Число научных публикаций – 52. tolstykh.aa@yandex.ru; Коптевская, 63, 125239, Москва, Россия; р.т.: +79102427955.

DOI 10.15622/ia.2021.20.2.8

A. GOLUBINSIY, A. TOLSTYKH

HYBRID METHOD OF CONVENTIONAL NEURAL NETWORK TRAINING

Golubinsiy A., Tolstykh A. **Hybrid Method of Conventional Neural Network Training.**

Abstract. The paper proposes a hybrid method for training convolutional neural networks. The method consists in combining second and first order methods for different elements of the architecture of a convolutional neural network. The hybrid convolution neural network training method allows to achieve significantly better convergence compared to Adam and requires fewer computational operations to implement. Using the proposed method, it is possible to train networks on which learning paralysis occurs when using first-order methods. Moreover, the proposed method could adjust its computational complexity to the hardware on which the computation is performed; at the same time, the hybrid method allows using the batch learning approach.

The analysis of the ratio of computations between convolutional neural networks and fully connected artificial neural networks is presented. The mathematical apparatus of error optimization of artificial neural networks is considered, including the method of back propagation of the error, the Levenberg-Marquardt algorithm. The main limitations of these methods that arise when training a convolutional neural network are analyzed.

The analysis of the stability of the proposed method when the initialization parameters are changed. The results of the applicability of the method in various problems are presented.

Keywords: Convolutional Neural Networks, Training Methods for Artificial Neural Networks, Optimization Methods

Golubinskiy Andrey – Ph.D., Dr.Sci., Associate Professor, Head of Scientific Director, JSC "Concern "Sozvezdie". Research interests: mathematical modeling of systems with elements of artificial intelligence. The number of publications – 205. annikgol@mail.ru; 92, Moskovsky prospect, 394068, Voronezh, Russia; office phone: +79103436537.

Tolstykh Andrey – Lecturer of the Department, Department of Special Information Technologies, Moscow University of the Ministry of Internal Affairs of Russia. Research interests: artificial neural networks, machine learning. The number of publications – 52. tolstykh.aa@yandex.ru; 63, Koptevskaya, 125239, Moscow, Russia; office phone: +79102427955.

References

1. Gudfellou Ia., Bendjıo I., Kýrvıll A. *Glubokoe obuchenie* [Deep learning]. M.: DMK Press. 2017. 652 p. (In Russ.).
2. Haıkın S. *Nejronnye seti. Polnyj kurs* [Neural networks. Complete course]. M.: Vil'yams2006. 1104 p. (In Russ.).
3. Kingma D.P., Ba J.A. A Method for Stochastic Optimization. CoRR. T. abs/1412.6980. 2014.
4. Bo Y.H., WeiL., I-Chen W. Stochastic Gradient Descent with Hyperbolic-Tangent Decay. CoRR. T. abs/1806.01593. 2018. pp. 1–10.
5. Wilamowski B.M., Irwin D.J. Intelligent systems. CRC Press. 2011. 568 p.
6. Smith J.S., Wu B., Wilamowski B.M. Neural Network Training With Levenberg-Marquardt and Adaptable Weight Compression. *IEEE Transactions on Neural Networks and Learning Systems.* 2018. pp. 1–8.
7. Szegedy C., Ioffe S., Vanhoucke V., Alemi A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. International Conference on Learning Representations (ICLR) Workshop. 2016. pp. 375–387.

8. He K., Zhang X., Ren S., Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv.org e-Print archive. 2015. Available at: https://arxiv.org/abs/1502.01852 (accessed: 12.11.2020).
9. Szegedy C. et al. Going deeper with convolutions. IEEE Conference on Computer Vision and Pattern Recognition. 2014. pp. 1–9.
10. Zaccone G., Karim R., Menshawy A. Deep Learning with TensorFlow: Explore neural networks with Python. Packt Publishing. 2017. 320 p.
11. Shepherd A.J. Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons. New York: Springer, 1997. 160 pp.
12. Shepherd A.J. Second-Order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-Layer Perceptrons. Springer. 1997. 160 p.
13. Nocedal J., Wright S.J. Numerical Optimization. Springer. 2006. 664 p.
14. Golubinskij A.N. [On the construction of architectures and the assessment of the parameters of artificial neural networks]. *Teoriya i tekhnika radiosvyazi – Theory and technology of radio communication.* 2020. vol. 1. pp. 72–87. (In Russ.).
15. Maksimushkın V.V., Arzamastsev A.A. [Comparative evaluation of the computational complexity of training an artificial neural network with a hard core and a network with a classical structure]. *Vestnik rossiiskih universitetov. Matematika – Russian Universities Reports. Mathematics.* 2006. vol. 2. pp. 190–197. (In Russ.).
16. Abramov S.A. *Lekcii o slozhnosti algoritmov* [Lectures on the complexity of algorithms]. M.: MCNMO. 2012. 248 p. (In Russ.).
17. Yann LeCun's Home Page. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. 2012. Available at: http://yann.lecun.com/exdb/mnist/ (accessed: 23.05.2020).
18. Smith L.N. No More Pesky Learning Rate Guessing Games. CoRR. 2015. pp. 1–10.
19. Salimans T., Kingma D.P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *Advances in Neural Information Processing Systems 29.* 2016. pp. 901–909.
20. Matuszyk P., Castillo R.T., Kottke D., Spiliopoulou M. A Comparative Study on Hyperparameter Optimization for Recommender Systems. Workshop on Recommender Systems and Big Data Analytics (RS-BDA'16). 2016.
21. Hu G. et al. Frankenstein: Learning Deep Face Representations using Small Data. *IEEE Transactions on Image Processing.* 2017. vol. 27. no. 1. pp. 293–303.
22. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. International conference on machine learning. 2015. pp. 448–456.
23. Lewkowycz A. et al. The large learning rate phase of deep learning: the catapult mechanism. 2020.
24. Liao Q., Kawaguchi K., Poggio T.A. Streaming Normalization: Towards Simpler and More Biologically-plausible Normalizations for Online and Recurrent Learning. CoRR. T. abs/1610.06160. 2016.
25. Mahajan D. et al. Exploring the Limits of Weakly Supervised Pretraining. CoRR. T. abs/1805.00932. 2018.
26. Petroski S.F. et al. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. ArXiv e-prints. 2017.
27. Su H., Zhu X., Gong S. Deep Learning Logo Detection with Data Expansion by Synthesising Context. 2017 IEEE winter conference on applications of computer vision (WACV). 2017. pp. 530–539.
28. Xu C., Qin T., Wang G., Liu T.Y. An Actor-Critic Algorithm for Learning Rate 2017. pp. 1–12.
29. Xu C., Qin T., Wang G., Liu T.Y. Reinforcement Learning for Learning Rate Control. arXiv preprint arXiv:1705.11159. 2017.