УДК 004.89 DOI: 10.26102/2310-6018/2020.30.3.018

Малоранговые аппроксимации нейросетевых алгоритмов

Н.В. Шапошникова

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева», Красноярск, Российская Федерация

Резюме: На сегодняшний день искусственные нейронные сети (далее ИНС) и глубокое обучение стали практически незаменимыми в приложениях, связанных с задачами машинного зрения, машинного перевода, преобразования речи в текст, рубрикации текстов, обработки видеоданных и т.д. Однако, несмотря на наличие ряда классических теорем, обосновывающих аппроксимирующие способности нейросетевых структур, текущие успехи в области ИНС в большинстве случаев связаны с эвристическим построением архитектуры сети, применимой только для конкретной рассматриваемой задачи. С другой стороны, глубокие ИНС имеют миллионы параметров и требуют для своего функционирования мощные вычислительные устройства, что ограничивает возможности их применения, например, на мобильных устройствах. Существенный прогресс в решении данных проблем может быть получен при использовании современных мощных алгоритмов малоранговых аппроксимаций для параметров слоев ИНС, что позволит как упростить процесс разработки нейросетевой архитектуры, так и получить существенное сжатие и ускорение обучения глубоких ИНС. Рассматривая, например, ядро сверточной ИНС, как четырехмерный массив (тензор), мы можем построить для него малоранговую аппроксимацию с эффективной реализацией его свертки с вектором (прямое распространение сигнала в сети при формировании предсказания) и дифференцирования по параметрам (обратное распространение сигнала в сети при обучении). В данной работе мы рассмотрим современную парадигму машинного обучения и малоранговых тензорных аппроксимаций, и на конкретном модельном численном примере, соответствующем задаче автоматического распознавания рукописных цифр, продемонстрируем перспективы тензоризации глубоких ИНС.

Ключевые слова: машинное обучение, нейронная сеть, глубокая сверточная сеть, малоранговая аппроксимация

Для цитирования: Шапошникова Н.В. Малоранговые аппроксимации нейросетевых алгоритмов. *Моделирование, оптимизация и информационные технологии.* 2020;8(3). Доступно по: https://moit.vivt.ru/wp-content/uploads/2020/08/Shaposhnikova_3_20_1.pdf DOI: 10.26102/2310-6018/2020.30.3.018

Low rank approximations for neural networks

N.V. Shaposhnikova

Federal State Budgetary Educational Institution of Higher Education «Reshetnev Siberian State University of Science and Technology» Krasnoyarsk, Russian Federation

Abstract: Today, artificial neural networks (hereinafter ANN) and deep learning have become almost indispensable in applications related to the tasks of machine vision, machine translation, speech to text conversion, text rubrication, video processing, etc. However, despite the presence of a number of classical theorems substantiating the approximating capabilities of neural network structures, the current successes in the field of ANNs in most cases are associated with the heuristic

construction of the network architecture applicable only for the specific problem under consideration. On the other hand, deep ANNs have millions of parameters and require powerful computing devices for their functioning, which limits the possibilities of their application, for example, on mobile devices. Significant progress in solving these problems can be obtained using modern powerful algorithms of low-rank approximations for the parameters of the ANN layers, which will both simplify the process of developing a neural network architecture and will lead to significant compression and acceleration of the training of deep ANNs. Considering, for example, the core of the convolutional ANN as a four-dimensional array (tensor), we can construct a low-rank approximation for it with the effective implementation of its convolution with the vector (direct signal propagation in the network when generating the prediction) and differentiation with respect to the parameters (back signal propagation in the network when training). In this paper, we will consider the modern paradigm of machine learning and low-rank tensor approximations, and we will demonstrate the prospects for the tensorization of deep ANNs using a specific model numerical example corresponding to the task of automatic recognition of handwritten digits.

Keywords: machine learning, neural network, deep convolutional network, low rank approximation.

For citation: Shaposhnikova N.V. Low rank approximations for neural networks. *Modeling, Optimization and Information Technology*. 2020;8(3). Available from: https://moit.vivt.ru/wp-content/uploads/2020/08/Shaposhnikova_3_20_1.pdf DOI: 10.26102/2310-6018/2020.30.3.018 (In Russ).

Введение

На сегодняшний день ИНС и глубокое обучение [1] стали практически незаменимыми в приложениях, связанных с задачами машинного зрения, машинного перевода, преобразования речи в текст, рубрикации текстов, обработки видеоданных и т.д. В ближайшие годы методы ИНС начнут активно применяться в системах автономного вождения автомобилей и летательных аппаратов, автономных роботизированных системах на производстве, в автоматизированных биомедицинских системах и других робототехнических приложениях [2, 3]. Однако для практического использования нейросетевых методов в целях ускорения развития научно-технического и технологического комплекса страны необходимо довести новые алгоритмические идеи и разработки до стадии практического применения, т.е. разработать технологические модели прикладного применения новых методов.

Необходимо отметить, что несмотря на наличие ряда классических теорем, обосновывающих аппроксимирующие способности нейросетевых структур [4, 5], текущие успехи в области ИНС в большинстве случаев связаны с эвристическим построением архитектуры сети, применимой только для конкретной рассматриваемой задачи [6]. При этом у научного сообщества нет законченного понимания внутренних закономерностей функционирования сети, необходимости или избыточности тех или иных слоев сети, наилучших способов оптимального выбора гиперпараметров и т.д. Отсутствие исчерпывающих научных ответов на приведенные вопросы существенно ограничивает качественное развитие метода ИНС, таким образом существует необходимость в модификации существующих алгоритмов и разработке новых.

Существенный прогресс в решении этой проблемы был получен при установлении связи между глубокими ИНС и тензорными сетями [6, 7], что сделало возможным применение мощных методов малоранговой тензорной аппроксимации, включая каноническое разложение, разложение Таккера и др. для принципиального сжатия и ускорения обучения глубоких ИНС [8, 9].

Далее в работе мы рассмотрим основные концепты, лежащие в основе современного машинного обучения, связанного с нейросетевым подходом, и малоранговых тензорных аппроксимаций. Затем на модельном численном примере, соответствующем задаче обучения нейросетевого классификатора распознаванию оцифрованных рукописных цифр на классическом наборе данных MNIST [10], мы продемонстрируем важные преимущества, возникающие при тензоризации ИНС, связанные как с упрощением ее архитектуры, так и с существенным сжатием массива параметров сети – в рассматриваемом случае ядер сверточных слоев сети.

Современное машинное обучение

Основу большинства методов машинного обучения представляет задача восстановления функциональной зависимости $y = f(x, q)$ некоторого целевого показателя $y \in R^{n_{out}}$ исследуемого явления или процесса от набора измеримых параметров процесса $x \in R^{n_{in}}$ и некоторых скрытых параметров модели $q \in R^{n_0}$ специфичных для используемого метода. Особо отметим, что, вообще говоря, для произвольных естественных физических процессов, функция должна иметь вид $y = f(x, \zeta, q)$, где ζ – это параметры процесса, которые остались неизвестными ввиду принципиальной невозможности абсолютно точного описания физических процессов (в отличии от, например, искусственных модельных примеров, когда восстанавливается некоторая аналитическая, заданная исследователем, функция, имеющая известное фиксированное количество параметров – аргументов функции). Далее мы будем опускать переменную ζ, однако неявно учитывая приведенное соображение.

Так, например, для классической линейной регрессии предполагается линейная зависимость скалярного $(n_{out} = 1)$ целевого показателя от наблюдаемых параметров:

$$y = q_0 + q_1 x_1 + q_2 x_2 + \ldots + q_{n_{in}} x_{n_{in}}, \quad (1)$$

и в процессе обучения подбирается оптимальный набор параметров модели q $(n_0 = n_{in} + 1)$, при котором результат предсказания наиболее точно соответствует известным результатам измерений – обучающему набору данных, состоящему из пар $(x^{(k)}, y^{(k)}_{real})$ для $k = 1, 2, \ldots, M^{(trn)}$, где $M^{(trn)}$ – это полное число элементов в обучающем наборе данных (например, количество размеченных фотографий, то есть фотографий с уже известным классом в задаче классификации изображений, которые могут быть использованы для обучения классификатора).

Другой пример – это случай полносвязной ИНС, для которой векторной функцией f является композиция векторных функций отдельных слоев сети:

$$f(x_1, W_1, b_1) = \sigma(W_1 x_1 + b), \quad (2)$$

где l – это номер слоя, σ – функция активации (например, логистическая), W_l – матрица весов нейронов данного слоя, а b_l – вектор смещений нейронов. То есть для полносвязной ANN скрытые параметры – это веса и смещения нейронов всех слоев.

Обучение в контексте регрессионных и нейросетевых моделей – это итеративный процесс подстройки параметров q модели с использованием известного обучающего набора данных. Формально данный процесс можно описать как минимизацию некоторого функционала потерь $L = L(x(T))$, зависящего от степени отклонения предсказаний сети от правильных ответов (или эквивалентно – зависящего от скрытых параметров модели). Отметим, что классическим подходом для обучения является стохастический метод градиентного спуска [11] и связанный с ним метод обратного распространения ошибки [12], при которых на каждой итерации происходит эффективное вычисление градиента функционала потерь и малое изменение параметров сети в направлении противоположном этому градиенту.

ИНС за последнее десятилетие стали одним из наиболее распространенных инструментов решения задач машинного обучения и интеллектуального анализа данных [1]. Обучение современных ИНС все чаще происходит на больших и сверхбольших объемах данных (миллионы объектов) с использованием мощных вычислительных ресурсов. Подобные выборки данных во многих приложениях позволяют без риска переучивания настраивать глубокие (многослойные) сети с миллиардами параметров. Такие глубокие сети позволяют адаптивно строить отображения с очень высокой нелинейностью, что отвечает важной современной тенденции – усложнению и повышению нелинейности аппроксимируемых адаптивными методами зависимостей. Все это обусловило возрастающую популярность глубоких ИНС.

Если рассматривать ИНС как метод машинного обучения, то можно выделить следующие существенные их преимущества относительно большинства известных методов:

- учет зависимостей между признаками. Во многих задачах признаки, описывающие объект, могут быть зависимы. Примерами таких объектов могут быть изображения (каждый пиксель зависит от некоторого количества соседей). Другой пример – тексты на естественных языках, в которых важен порядок слов. Многие стандартные методы не работают с зависимостями непосредственно и вынуждены использовать различные эвристические признаки (к примеру, модель «мешка слов»). ИНС обрабатывают эти объекты более естественным образом;
- построение оптимальных признаков для описания объектов может быть сложной задачей, требующей хорошего понимания предметной области. ИНС позволяют выбирать подходящие признаки автоматически, что сокращает время на разработку и внедрение методов анализа данных на их основе;
- универсальность модели: доказано, что ИНС могут приблизить любую функцию. Таким образом если задача в принципе может быть решена по данным признакам, то она может быть решена с помощью ИНС. Из всех методов машинного обучения только ИНС обладают таким свойством, что делает исследования в этой области более актуальными.

Простейшая регрессионная модель (1) в векторной форме с точностью до обозначений может быть записана как:

$$z = w^T x + b\,, \tag{3}$$

где $z \in R$, $w \in R^{n_{in}}$, $x \in R^{n_{in}}$, $b \in R$, а символ «Т» обозначает стандартную операцию транспонирования. Если мы внесем в данную модель нелинейность, применив некоторую функцию σ к (3), то получим классическую модель искусственного нейрона:

$$a = \sigma(z) = \sigma(w^T x + b), \tag{4}$$

где $a \in R$ называют выходом нейрона (фактически данная величина является предсказанием ИНС, состоящей из одного нейрона), промежуточную величину $z \in R$ называют взвешенным выходом нейрона, $x \in R^{n_{in}}$ – входным вектором, $w \in R^{n_{in}}$ – вектором весов нейрона (чем больше абсолютное значение конкретного веса, тем большую значимость имеет соответствующая наблюдаемая переменная), $b \in R$ – смещением нейрона (чем меньше значение смещения, тем сложнее «активировать» нейрон). К введенной функции σ, называемой функцией активации, обычно предъявляют требование нелинейности и ограниченности области ее значений отрезком [0, 1]. Например, в качестве функции активации может быть выбрана классическая логистическая функция вида:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{5}$$

С использованием построенной системы обозначений, для взвешенного выхода j-ого нейрона l-ого слоя $(j = 1, 2, ..., N^{(l)}, l = 2, 3, ..., L)$ можем записать:

$$z_j^{(1)} = \sum_{i=1}^{N^{(l-1)}} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}, \tag{6}$$

при этом для первого слоя мы можем положить $z_j^{(1)} = x_j$, а в векторной форме данные формулы запишутся следующим образом:

$$z^{(1)} = x,\ \ z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)},\ \ l = 2, 3, ..., L \tag{7}$$

Выход нейронов l-ого слоя мы можем получить, подействовав функцией активации на соответствующий взвешенный выход. В векторной форме можем записать:

$$a^{(1)} = x,\ \ a^{(l)} = \sigma^{(l)}(z^{(l)}),\ \ l = 2, 3, ..., L \tag{8}$$

Таким образом, математически процесс прямого распространения сигнала в многослойной ИНС описывается следующим образом:

- на вход сети подается входной вектор x, имеющий длину, равную числу нейронов первого (входного) слоя;
- взвешенный выход и выход (итоговый) входного слоя сети задаются по первым частям формул (7) и (8);
- для каждого из последующих слоев сети (в цикле) вычисляются взвешенный выход и выход (итоговый) на основе выхода предыдущего слоя по формулам (7) и (8);
- выход последнего слоя (выходного слоя) $a^{(L)}$ является результатом работы (предсказанием) ИНС;

– если для текущего входного вектора x известен желаемый выход сети y (элемент данных обучающей выборки), то может быть также вычислена с использованием $a^{(L)}$ функция стоимости отдельного измерения C_x, которая затем используется для обучения (подстройки весов и смещений) ИНС.

Малоранговые аппроксимации и тензорные сети

Аналогично нейросетевой революции, описанной выше, в последние два десятилетия произошло также стремительное развитие численных методов работы с многомерными данными и создание принципиально новых подходов, позволяющих на порядки ускорить работу алгоритмов и в то же время на порядки снизить объемы потребляемой памяти (см., например, обзоры [7, 13]). В основе этих методов лежит идея малоранговой тензорной аппроксимации и построение соответствующего компактного (малорангового) представления (разложения) многомерного массива данных (далее мы будем использовать, общепринятый в современной прикладной математической литературе термин «тензор»).

За последнее десятилетие малоранговые тензорные аппроксимации нашли широкий спектр областей применения, при этом среди наиболее распространенных, представленных в литературе, отметим следующие (см. ссылки в вышеприведенных обзорах): приближение функции Грина для многомерных дифференциальных уравнений, решение уравнения Больцмана, решение параметрических и стохастических дифференциальных уравнений в частных производных, аппроксимация уравнений и интегралов, зависящих от параметров, аппроксимация многомерных интегралов и многомерных сверток, вычислительные задачи в области электричества, магнетизма и квантовой механики, финансовая математика, машинное обучение.

Как отмечалось выше, мы определяем тензор как простой многомерный массив данных: $Y \in R^{N_1 \times N_2 \times \ldots \times N_d}$, где d $(d = 1, 2, 3, \ldots)$ – это размерность тензора. На Рисунке 1 приведены соответствующие примеры одномерного (вектор с числом элементов N_1), двумерного (матрица с числом строк N_1 и числом столбцов N_2) и трехмерного (число элементов по соответствующим осям N_1, N_2 и N_3) тензоров. Подобный d-мерный тензор имеет всего $N_1 \times N_2 \times \ldots \times N_d \sim N_0^d$ элементов, где N_0 – это среднее число элементов по каждому из измерений, при этом конкретный элемент тензора в позиции с индексами $(n_1, n_2, \ldots, n_d)$ обозначается как $X[n_1, n_2, \ldots, n_d]$ и является вещественным числом.

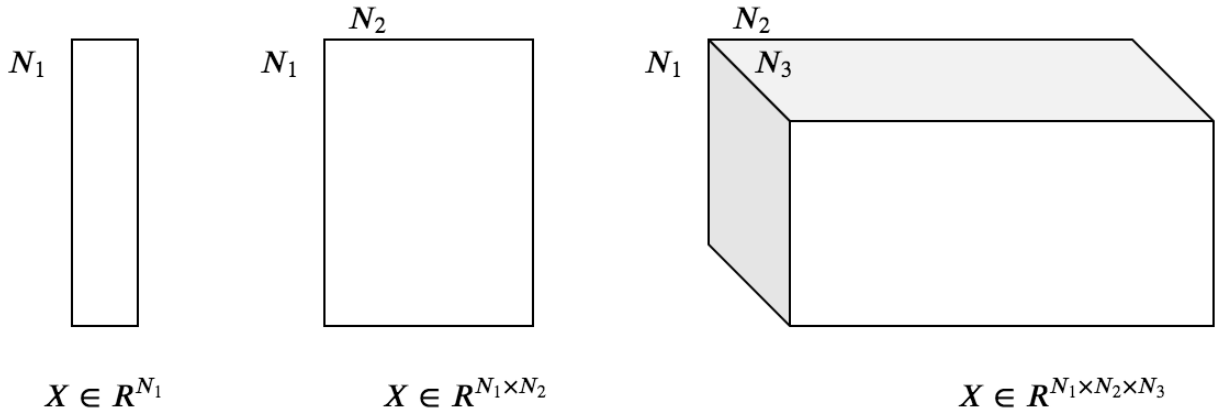

Рисунок 1 – Примеры тензоров (многомерных массивов) различных размерностей. Слева направо: одномерный тензор (вектор), двумерный тензор (матрица), трехмерный тензор

Figure 1 – Examples of tensors (multidimensional arrays) of various dimensions. From left to right: one-dimensional tensor (vector), two-dimensional tensor (matrix), three-dimensional tensor

Обычно тензоры возникают как дискретизации функций многих переменных на многомерных (тензорных) сетках. Действительно, если мы рассмотрим функцию d переменных $f(x_1, x_2, ..., x_d)$ и по каждому измерению k $(k = 1, 2, ..., d)$ введем сетку из N_k $(N_k > 0)$ точек $\left[x_{k,1}, x_{k,2}, ..., x_{k,N_k} \right]$, то соответствующий полный набор значений рассматриваемой функции в узлах сетки может быть представлен как соответствующий тензор $Y \in R^{N_1 \times N_2 \times ... \times N_d}$ с элементами:

$$Y\left[n_1, n_2, ..., n_d\right] = f(x_{1,n_1}, x_{2,n_2}, ..., x_{d,n_d}). \quad (9)$$

Для больших значений размерности тензора d возникает так называемая проблема проклятия размерности («curse of dimensionality»), заключающаяся в экспоненциальном росте числа параметров (N_0^d), требуемых для хранения тензора, и сложности вычислительных операций. Этим обусловлена необходимость приближенного представления (аппроксимации) многомерных тензоров в сжатом (малоранговом) формате.

Если для двумерных задач $(d = 2)$, по существу, единственным малоранговым представлением является сингулярное разложение матриц SVD, то для случая размерностей $d > 2$ существует большее разнообразие методов и алгоритмов.

Одной из первых предложенных тензорных аппроксимаций является каноническое разложение (употребляется также термин «CANDECOMP /PARAFAC») [14], представляющее дискретный аналог классической идеи о разделении переменных для функций многих переменных. Для произвольного элемента $Y\left[n_1, n_2, ..., n_d\right]$ заданного тензора $Y \in R^{N_1 \times N_2 \times ... \times N_d}$ данное разложение может быть записано как:

$$Y\left[n_1, n_2, ..., n_d\right] = \sum_{r=1}^{R} U_1\left[n_1, r\right] U_2\left[n_2, r\right] ... U_d\left[n_d, r\right], \quad (10)$$

где R – это тензорный (канонический) ранг, а матрицы U_k имеют размеры $N_k \times R$ $(k = 1, 2, ..., d)$ и называются каноническими факторами или факторными матрицами.

Основное привлекательное свойство канонического разложения – это отсутствие экспоненциального роста числа параметров с ростом размерности. Для хранения тензора в данном формате, как следует из формулы (10), требуется всего:

$$(N_1 + N_2 + ... + N_d)R \sim dN_0R, \quad (11)$$

параметров вместо N_0^d параметров исходного d-мерного тензора.

Вместо канонического разложения может быть также использовано популярное разложение Таккера [15], имеющее вид:

$$Y\left[n_1, n_2, ..., n_d\right] = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} ... \sum_{r_d=1}^{R_d} \quad (12)$$

$$G\left[r_1, r_2, ..., r_d\right] U_1\left[n_1, r_1\right] U_2\left[n_2, r_2\right] ... U_d\left[n_d, r_d\right], \quad (13)$$

где числа $R_1, R_2, ..., R_d$ называют Таккеровскими рангами, d -мерный тензор $G \in R^{R_1 \times R_2 \times ... \times R_d}$ – ядро разложения Таккера, а матрицы U_k имеют размеры $N_k \times R_k$ $(k = 1, 2, ..., d)$ и называются факторами Таккера.

Как можно видеть из формулы (13), в рамках разложения Таккера исходный тензор $Y \in R^{N_1 \times N_2 \times ... \times N_d}$ заменяется на ядро $G \in R^{R_1 \times R_2 \times ... \times R_d}$, которое обычно имеет существенно меньший размер по каждому из d-измерений. Соответственно разложение Таккера все же сохраняет экспоненциальный характер зависимости числа элементов тензора (а значит и сложности вычислительных операций с его участием) от размерности, поэтому разложение Таккера практически не используется в многомерных задачах (однако оно часто применяется для трехмерных задач).

Тензоризация искусственной нейронной сети

Оптимальной архитектурой ИНС для работы с графическими изображениями является совокупность последовательных сверточных слоев. Поскольку каноническое разложение (9) обеспечивает наибольшую степень сжатия, мы воспользуемся именно этим малоранговым тензорным форматом для компактного представления ядер сверточных слоев ИНС.

Для программной реализации мы будем использовать популярный фреймворк машинного обучения PyTorch [16] на языке программирования python и облачный сервис colab [17], позволяющий осуществлять обучение мощных глубоких нейросетевых архитектур с использованием современных GPU. Как отмечалось выше, мы будем проводить обучение ИНС на наборе данных MNIST [10], соответствующем размеченному набору изображений рукописных цифр (60000 обучающих изображений и 10000 проверочных изображений). Разработанный программный код и результаты тестовых расчетов (в формате colab ноутбука) находятся в открытом доступе в сети Интернет по адресу:
https://colab.research.google.com/drive/1zzX703DBGFpgqQbNEMigzSJnovHSwryQ?usp=sharing .

В качестве базовой модели для задачи распознавания рукописных цифр мы рассматриваем стандартную архитектуру (см. Рисунок 2), состоящую из двух

последовательных сверточных слоев, одного внутреннего полносвязного слоя и выходного слоя с 10 выходами, каждый из которых соответствует вероятности присутствия на изображении соответствующей цифры.

Тензорный слой реализован нами в виде самостоятельного класса LayTens (см. Рисунок 3), который может заменить соответствующий библиотечный класс стандартного сверточного слоя Conv2d во фреймворке PyTorch. В рамках описанного выше в работе подхода, мы рассматриваем ядро сверточного слоя как тензор, представленный в малоранговом каноническом формате, при этом ранг разложения (определяющий фактически степень сжатия), задается пользователем в качестве параметра слоя. При прохождении сигнала через данный слой (метод forward), мы осуществляем преобразование входного вектора в 4-х мерный тензор, а затем производим соответствующие свертки с четырьмя факторными матрицами канонического разложения.

Иллюстрация возможного использования тензорного слоя приводится на Рисунке 4. Мы создаем класс NetTens, соответствующий тензоризованной сверточной сети. Архитектура данной сети совпадает с базовой NetBase, за исключением замены обычных сверточных слоев их тензоризованной версией.

Архитектура базовой сети

```
[ ]
class NetBase(nn.Module):
    ''' Классическая сверточная нейронная сеть. '''

    def __init__(self, f_n, f_h, f_w, cl, sh, r=None):
        super(NetBase, self).__init__()
        self.f_n = f_n
        self.f_h = f_h
        self.f_w = f_w
        self.cl = cl
        self.sh = sh
        self.r = r
        self.init_()

    def init_(self):
        self.conv1 = nn.Conv2d(self.sh[1], self.f_n, kernel_size=self.f_h, bias=False)
        self.conv2 = nn.Conv2d(self.f_n, self.f_n, kernel_size=self.f_h, bias=False)
        self.full1 = nn.Linear(self.shf, 50)
        self.full2 = nn.Linear(50, self.cl)

    def forward(self, x):
        y = self.conv1(x)
        y = F.max_pool2d(y, 2)
        y = F.relu(y)
        y = self.conv2(y)
        y = F.max_pool2d(y, 2)
        y = F.relu(y)
        y = y.view(-1, self.shf)
        y = self.full1(y)
        y = self.full2(y)
        y = F.log_softmax(y, dim=1)
        return y
```

Рисунок 2 – Программная реализация базовой сверточной ИНС
Figure 2 – Software implementation of the basic convolutional INN

Архитектура тензорного слоя

```
[ ]
class LayTens(nn.Module):
    def __init__(self, f_n, f_h, f_w, ch, r):
        super(LayTens, self).__init__()
        self.f_n = f_n
        self.f_h = f_h
        self.f_w = f_w
        self.ch = ch
        self.r = r
        self.init_()

    def init_(self):
        f = np.random.randn(self.f_n, self.f_h, self.f_w, self.ch)
        f/= self.f_h * self.f_w
        fs = parafac(tensor(f), self.r).factors
        self.f0 = nn.Parameter(torch.tensor(fs[0]).to('cuda'))
        self.f1 = nn.Parameter(torch.tensor(fs[1]).to('cuda'))
        self.f2 = nn.Parameter(torch.tensor(fs[2]).to('cuda'))
        self.f3 = nn.Parameter(torch.tensor(fs[3]).to('cuda'))

    def forward(self, x):
        sh0 = x.shape
        z = x.unfold(1, self.f_h, 1).unfold(2, self.f_w, 1)
        sh = z.shape
        z = z.reshape(sh[0], sh[1] * sh[2], sh[3], sh[4])
        z = z.permute(1, 2, 3, 0).to('cuda')
        for i in range(self.r):
            q = einsum('abcd,d->abc', z.float(), self.f3[:, i].float())
            q = einsum('abc,c->ab', q, self.f2[:, i].float())
            q = einsum('ab,b->a', q, self.f1[:, i].float())
            q = einsum('a,b->ab', q, self.f0[:, i].float())
            y = q if i == 0 else y + q
        y = y.reshape((sh0[1] - 2, sh0[2] - 2, self.f_n))
        return y.permute(2, 0, 1)
```

Рисунок 3 – Программная реализация тензорного сверточного слоя
Figure 3 – Software implementation of a tensor convolutional layer

Архитектура тензорной сети

```
[ ]
class NetTens(NetBase):
    '''
    Тензоризованная сверточная нейронная сеть.
    Используется NetBase в качестве базового класса и
    LayTens в качестве тензоризованного тензорного слоя (вместо nn.Conv2d).
    '''

    def init_(self):
        self.conv1 = LayTens(self.f_n, self.f_h, self.f_w, self.sh[1], self.r)
        self.conv2 = LayTens(self.f_n, self.f_h, self.f_w, self.f_n, self.r)
        self.full1 = nn.Linear(self.shf, 50)
        self.full2 = nn.Linear(50, self.cl)
```

Рисунок 4 – Программная реализация тензоризованной сверточной ИНС
Figure 4 – Software implementation of a tensorized convolutional INN

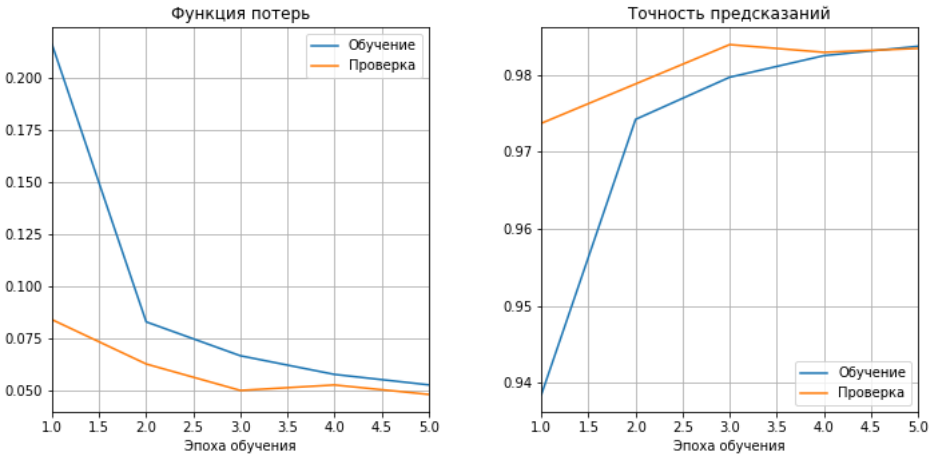

Рисунок 5 – Результат расчета для базовой сверточной ИНС
Figure 5 – The result of the calculation for the basic convolutional INN

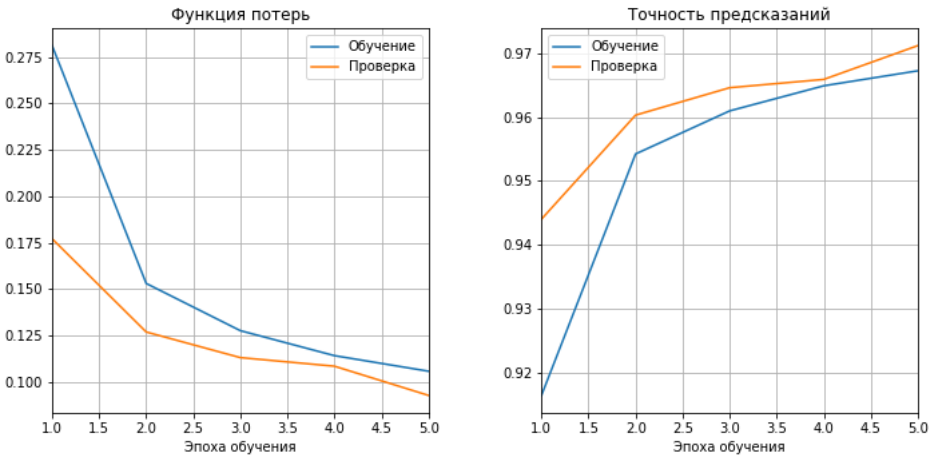

Рисунок 6 – Результат расчета для тензоризованной сверточной ИНС (тензорный ранг равен 3)
Figure 6 – Calculation result for a tensorized convolutional INN (the tensor rank is 3)

Таблица 1 – Результаты расчета в зависимости от архитектуры сети и тензорного ранга
Table 1 – Calculation results depending on the network architecture and tensor rank

Архитектура	Тензорный ранг	Точность, %	Степень сжатия
Сверточная базовая	-	98.34	1
Сверточная тензорная	3	97.12	7.7
Сверточная тензорная	6	97.95	3.8
Сверточная тензорная	9	97.67	2.6

На Рисунке 5 и Рисунке 6 мы приводим результат обучения базовой и тензоризованной сети (с тензорным рангом равным 3) соответственно для пяти эпох

обучения. Для сверточных слоев мы используем 10 фильтров с шириной окна 3x3, параметр скорости обучения выбран равным 0.0001.

В Таблице 1 мы приводим зависимость результата от выбранного ранга канонического разложения. Точность, представленная в таблице, соответствует проценту корректно распознанных изображений в проверочном наборе данных, а степень сжатия определяется как отношение числа параметров базового сверточного слоя к соответствующему числу параметров аналогичного тензоризованного слоя.

Как можно видеть, с уменьшением ранга канонического разложения степень сжатия существенно растет, при этом точность предсказаний сети уменьшается незначительно. Так при ранге равном 3, мы имеем сжатие в более чем 7 раз и соответствующее снижение точности примерно на 1 процент.

Заключение

В работе был рассмотрен перспективный подход для компактного представления нейросетевых архитектур, связанный с тензоризацией их слоев, то есть с их компактным представлением с использованием малоранговой тензорной аппроксимации. Для приведенного в работе численного примера было получено сжатие в 7.7 раз классических сверточных слоев при снижении качестве предсказания на 1 процент. Отметим, что для современных многослойных сверточных архитектур, содержащих десятки слоев, соответствующее сжатие при использовании малоранговых аппроксимаций может сделать возможным их использование на маломощных, в частности, мобильных устройствах.

ЛИТЕРАТУРА

1. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444.
2. Zhang C., Patras P., Haddadi H. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*. 2019;21(3):2224-2287.
3. Zhao Z., Zheng P., Xu S. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*. 2019;30(11):3212-3232.
4. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*. 1989;2(4):303–314.
5. Hornik K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*. 1991;4(2):251–257.
6. Cohen N., Sharir O., Shashua, A. On the expressive power of deep learning: a tensor analysis. *arXiv preprint*. 2015;arXiv:1509.05009.
7. Cichocki A. Tensor networks for dimensionality reduction and large-scale optimization. *Foundations and Trends in Machine Learning*. 2016;9.4-5.
8. Lebedev V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint*, 2014;arXiv:1412.6553.
9. Novikov A., Podoprikhin D., Osokin A., Vetrov D. Tensorizing neural networks. *In Advances in neural information processing systems*. 2015;442-450.
10. Deng L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*. 2012;29(6):141-142.
11. Bottou L. Large-scale machine learning with stochastic gradient descent. *In Proceedings of COMPSTAT'2010*. 2010;177–186.
12. Rumelhart D., Hinton G., Williams R. Learning representations by back-propagating errors. *Nature*. 1986;323(6088):533–538.

13. Grasedyck L., Kressner D., Tobler C. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*. 2013:36(1):53-78.
14. Harshman R. Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*. 1970;1–84.
15. Tucker L. Some mathematical notes on three-mode factor analysis. *Psychometrika*. 1966;31:279–311.
16. PyTorch, фреймворк машинного обучения [Электронный ресурс]. – Режим доступа: https://pytorch.org – Дата доступа: 10.06.2020
17. Colab, интерактивная облачная среда [Электронный ресурс]. – Режим доступа: https://colab.research.google.com – Дата доступа: 10.06.2020

REFERENCES

1. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444.
2. Zhang C., Patras P., Haddadi H. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials.* 2019;21(3):2224-2287.
3. Zhao Z., Zheng P., Xu S. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems.* 2019;30(11):3212-3232.
4. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*. 1989;2(4):303–314.
5. Hornik K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*. 1991;4(2):251–257.
6. Cohen N., Sharir O., Shashua, A. On the expressive power of deep learning: a tensor analysis. *arXiv preprint*. 2015;arXiv:1509.05009.
7. Cichocki A. Tensor networks for dimensionality reduction and large-scale optimization. *Foundations and Trends in Machine Learning*. 2016;9.4-5.
8. Lebedev V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint*, 2014;arXiv:1412.6553.
9. Novikov A., Podoprikhin D., Osokin A., Vetrov D. Tensorizing neural networks. *In Advances in neural information processing systems*. 2015;442-450.
10. Deng L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*. 2012;29(6):141-142.
11. Bottou L. Large-scale machine learning with stochastic gradient descent. *In Proceedings of COMPSTAT'2010*. 2010;177–186.
12. Rumelhart D., Hinton G., Williams R. Learning representations by back-propagating errors. *Nature*. 1986;323(6088):533–538.
13. Grasedyck L., Kressner D., Tobler C. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*. 2013:36(1):53-78.
14. Harshman R. Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*. 1970;1–84.
15. Tucker L. Some mathematical notes on three-mode factor analysis. *Psychometrika*. 1966;31:279–311.
16. PyTorch, machine learning framework [Electronic resource]. – Mode of access: https://pytorch.org – Date of access: 10.06.2020
17. Colab, interactive cloud framework [Electronic resource]. – Mode of access: https://colab.research.google.com – Date of access: 10.06.2020

ИНФОРМАЦИЯ ОБ АВТОРЕ / INFORMATION ABOUT THE AUTHOR

Шапошникова Нина Владимировна, аспирант, кафедра системного анализа и исследования операций, ФГБОУ ВО "Сибирский государственный университет науки и технологии имени академика М.Ф. Решетнева», Красноярск, Российская Федерация.
e-mail: shapninel@gmail.com
ORCID: 0000-0003-2792-7693

Nina V. Shaposhnikova, PhD Student, System analysis and operations research, Federal State Budgetary Educational Institution of Higher Education «Reshetnev Siberian State University of Science and Technology», Krasnoyarsk, Russian Federation