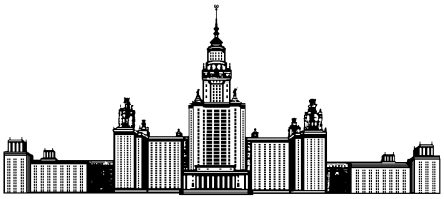

Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Гарипов Тимур Исмагилевич

Тензоризованные нейронные сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф-м.н.
Ветров Дмитрий Петрович

Москва, 2017

Содержание

1 Введение

Алгоритмы машинного обучения, основанные на свёрточных нейронных сетях, в настоящее время демонстрируют передовые результаты в задачах компьютерного зрения [1, 2] , обработки естественных языков [3] и в других областях. Но для применения таких алгоритмов требуются значительные вычислительные ресурсы. Современные архитектуры нейронных сетей включают миллионы настраиваемых параметров, для хранения которых требуются сотни мегабайт [4]. Это обстоятельство не позволяет оптимально использовать энергоэффективные запоминающие устройства при работе со свёрточными нейронными сетями, а следовательно, ограничивает их применение на мобильных устройствах.

В работе [5] был предложен метод сжатия матрицы весов полносвязных слоёв нейронной сети, использующий разложение тензорного поезда (Tensor-Train, TT) [6]. TT-разложение — способ представления тензора, который позволяет компактно хранить тензор в памяти компьютера и эффективно выполнять операци над тензорами, представленными в таком формате. Метод, предложенный авторами работы [5], позволяет значительно уменьшить число параметров полносвязных слоёв без значительной потери в качестве работы алгоритма.

В данной работе рассматривается задача уменьшения числа параметров свёрточных слоёв, мотивированная следующими обсотятельствами. При сжатии полносвязных слоёв (например, с помощью метода, упомянутого выше) снижается доля параметров полносвязных слоёв и основная часть потребляемой памяти расходуется на хранение параметров свёрточных слоёв. Помимо того, в последнее время появляется все больше успешных архитектур свёрточных сетей, в которых число параметров полносвязных слоёв незначительно по сравнению с числом параметров свёрточных слоёв.

В данной работе предложен метод сжатия свёрточных слоёв нейронных сетей, основанный на специальном способе представления параметров свёрточного слоя в виде многомерного тензора и уменьшения числа параметров, необходимого для хранения

полученного тензора с помощью TT-разложения. Основной вклад данной работы заключается в следующем:

- разработан и реализован метод сжатия тензора параметров свёрточного слоя;
- проведено экспериментальное исследование предложенного метода;
- исследована возможность комбинированного сжатия полносвязных и свёрточных слоёв.

Настоящая работа организована следующим образом. В разделе 2 формально описана задача обучения по прецедентам и алгоритмы обучения, основанные на свёрточных нейронных сетях, кроме того формулируется задача снижения числа параметров алгоритма классификации. В разделе 3 приводится обзор существующих подходов к уменьшению числа параметров нейронных сетей. Раздел 4 посвящен TT-разложению и его свойствам, важным для данной работы. В разделе 5 описан метод сжатия полносвязных слоёв, предложенный в работе [5]. В разделе 6 описан метод сжатия свёрточных слоёв нейронных сетей, разработанный в рамках данной работы. Экспериментальное исследование предложенного метода описано в разделе 7.

2 Обозначения и постановка задачи

2.1 Многомерные массивы

В данной работе будут рассматриваться многомерные массивы вещественных чисел. Будем называть одномерные массивы *векторами* и использовать для их обозначения жирные строчные символы ($\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$). Для обозначения двумерных массивов будем использовать термин *матрица* и жирные заглавные символы ($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$). Массивы больших размерностей будем называть *тензорами* и обозначать жирными заглавными каллиграфическими символами ($\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \ldots$).

Для обозначения элементов массивов будем рассматривать массивы как функции от индексов: $a(i) = a_i$, $A(i, j)$, $\mathcal{A}(i_1, i_2, \ldots, i_d)$, где d — размерность тензора $\boldsymbol{\mathcal{A}}$.

2.2 Формальная постановка задачи обучения по прецедентам

Задача обучения по прецедентам [7] заключается в восстановлении зависимости между скрытыми и наблюдаемыми переменными по имеющейся выборке прецедентов, для которых известны значения как наблюдаемых так и скрытых переменных.

Формально, пусть X — множество объектов (наблюдений), Y — множество допустимых значений скрытых переменных. Предполагается, существование функциональной зависимости $y^* : X \to Y$ между наблюдениями и скрытыми переменными. Само отображение y^* является неизвестным, однако имеется конечное множество объектов $\{x^1, x^2, \ldots, x^\ell\} \subseteq X$, для которых известны соответствующие значения $y^1, y^2, \ldots, y^\ell$: $y^i = y^*(x^i)$. *Признаком объекта* называется отображение $f : X \to D_f$, где D_f — множество допустимых значений признака. Объекты x^i представлены своими *признаковыми описаниями* $(f_1(x^i), f_2(x^i), \ldots, f_k(x^i))$, где $f_1, f_2, \ldots, f_k$ — заданные признаки. Набор $X^\ell = \{(x^i, y^i)\}_{i=1}^\ell$ называется *обучающей выборкой*. Требуется, используя обучающую выборку X^ℓ, построить *решающую функцию* (*решающий алгоритм*): отображение $a : X \to Y$, приближающее y^* на всём множестве X. Часто функцию a выбирают из некоторого фиксированного семейства параметрических отображений.

Одним из частных случаев задачи обучения по прецедентам является задача классификации. В этом случае множество $Y = \{1, 2, \ldots, M\}$ является конечным множеством из M элементов. В этом случае говорят, что каждый объект $x \in X$ относится к одному (и только одному) из M классов.

В данной работе рассматривается задача классификации изображений, то есть наблюдения x являются изображениями. Типичным признаковым описанием для изображения является набор интенсивностей всех пикселей изображения в фиксированной цветовой модели (например RGB).

Оценка качества классификации

Рассмотрим следующий способ оценки качества метода классификации. Пусть набор предоставленных объектов и соответствующих им значений скрытых переменных разбит на две непересекающиеся части: X_{train} и $X_{validation}$, где X_{train} — обу-

чающая выборка,$X_{validation}$ —*валидационная выборка*. Зачастую исходные данные изначально разбиты на указанные наборы. Если такого разбиения не предоставлено, то производится случайное разбиение с заранее заданным соотношением размеров выборок. Пусть с помощью выбранного метода обучения по выборке X_{train} был построен классифицирующий алгоритм a, без использования объектов из выборки $X_{validation}$. Качество решения задачи характеризуется двумя значениями $Q(a, X_{train})$ и $Q(a, X_{validation})$, где $Q(a, X^\ell)$ — *точность классификации* (accuracy) объектов выборки X^ℓ алгоритмом a. Указанная величина вычисляется следующим образом:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \left[y^i = a(x^i) \right],$$

где $[\cdot]$ — нотация Айверсона.

Точность классификации обучающей выборки характеризует способность выбранного метода обучения к настройке на обучающие данные. Однако, важно обращать внимание также и на точность классификации объектов из валидационной выборки, которые не были доступны на этапе обучения классификатора. Следует отметить, что контроль за обеими указанными величинами позволяет предотвратить эффект переобучения. Явление *переобучения* заключается в том, что построенный решающий алгоритм практически идеально предсказывает значения скрытых переменных для объектов обучающей выборки, но крайне неудачно обобщается на остальные допустимые объекты $x \in X$, которые не были использованы при обучении.

2.3 Нейронные сети

Как было сказано выше, часто решающее правило a выбирают из некоторого параметрического семейства функций. Одним из широко используемых способов параметризации функции a является представление её с помощью нейронной сети. При таком подходе сложная функция a представляется как композиция достаточно простых преобразований.

Рассмотрим формальное описание алгоритма классификации, основанного на нейронной сети. Пусть требуется решить задачу классификации объектов на K клас-

сов. Будем считать, что каждый объект задаётся своим признаковым описанием $\boldsymbol{x}_i \in \mathbb{R}^d$. Построим решающий алгоритм a следующим образом. Рассмотрим векторную функцию $\boldsymbol{p} : \mathbb{R}^d \to \mathbb{R}^K$ такую, что $p_i(\boldsymbol{x}) \geqslant 0$, $\sum\limits_{i=1}^{K} p_i(\boldsymbol{x}) = 1$. Элементы вектора $\boldsymbol{p}(\boldsymbol{x})$ будем интерпретировать как степени уверенности (вероятности) в принадлежности объекта $\boldsymbol{x}$ к каждому из K классов. Теперь решающее правило a можно определить следующим образом: $a(\boldsymbol{x}) = \arg\max\limits_{i=1,2,\ldots,K} p_i(\boldsymbol{x})$. Для полного описания правила a осталось задать конкретный вид функции $\boldsymbol{p}(\boldsymbol{x})$.

Представим функцию $\boldsymbol{p}(\boldsymbol{x})$ с помощью нейронной сети. В простейшем случае нейронные сети позволяют строить композиции функций следующего вида:

$$\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{s}(\boldsymbol{h}^n(\boldsymbol{h}^{n-1}(\ldots \boldsymbol{h}^2(\boldsymbol{h}^1(\boldsymbol{x}))\ldots))),$$

где

$$\boldsymbol{h}^i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}, \quad i = 1, 2, \ldots, n; \quad d_0 = d, \quad d_n = K, \quad \boldsymbol{s} : \mathbb{R}^K \to \mathbb{R}^K.$$

Нейронная сеть представляет собой последовательность из n слоёв, в которой i-ый слой h^i выполняет преобразование входного вектора размерности d_{i-1} в выходной вектор размерности d_i. Можно считать, что первый слой выполняет переход от исходного признакового описания объекта $\boldsymbol{x}$ к скрытому признаковому описанию $\boldsymbol{h}^1(\boldsymbol{x})$, а последующие слои извлекают признаки более высокого уровня, основанные на признаках предыдущего уровня. Функция $\boldsymbol{s}$ предназначена для перехода от признаков последнего уровня к вектору $\boldsymbol{p}$, компоненты которого имеют смысл вероятностей. Типичным вариантом функции $\boldsymbol{s}(\boldsymbol{z})$ является softmax преобразование:

$$s_i(\boldsymbol{z}) = \frac{e^{z_i}}{\sum\limits_{j=1}^{K} e^{z_j}}.$$

Стоит отметить, что каждый слой сети имеет свой вектор параметров $\boldsymbol{\theta}^i$ и формально осуществляет параметрическое преобразование $\boldsymbol{h}^i(\boldsymbol{h}^{i-1}, \boldsymbol{\theta}^i)$. То есть, нейронная сеть задаёт параметрическое семейство функций. Для обучения нейронной сети параметры всех слоёв $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^n)$ следует настроить так, чтобы алгоритм $a_{\boldsymbol{\theta}}(\boldsymbol{x})$, соответствующий выбранным значениям параметров $\boldsymbol{\theta}$, наилучшим образом

решал задачу классификации на обучающей выборке. Более подробное описание алгоритма обучения приведено ниже.

Конкретизируем вид преобразований $\boldsymbol{h}^i$, осуществляемых слоями нейронной сети. Чаще всего используется следующее представление для данных преобразований:

$$\boldsymbol{h}^i(\boldsymbol{h}^{i-1}) = \boldsymbol{f}^i(\boldsymbol{g}^i(\boldsymbol{h}^{i-1})),$$

где $\boldsymbol{f}^i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$ — поэлементное нелинейное преобразование: $f^i_j(\boldsymbol{z}) = f(z_j)$, $f : \mathbb{R} \to \mathbb{R}$; а $\boldsymbol{g}^i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ — линейное преобразование. Функцию $f(z)$ называют *функцией активации*, ниже приведены примеры широко используемых функций активации:

$$\begin{cases} f(z) = \dfrac{1}{1+e^{-z}} & \text{— сигмоида,} \\ f(z) = \tanh(z) & \text{— гиперболический тангенс,} \\ f(z) = \max(0, z) & \text{— ReLU (REctified Linear Unit).} \end{cases}$$

Обычно на всех слоях кроме последнего используют одну и ту же функцию активации, а на последнем слое используют тождественную функцию $f(z) = z$.

Линейное преобразование $\boldsymbol{g}^i$ в простейшем случае имеет вид

$$\boldsymbol{g}^i(\boldsymbol{z}) = \boldsymbol{W}^i\boldsymbol{z} + \boldsymbol{b}^i,$$

где матрица $\boldsymbol{W}^i \in \mathbb{R}^{d_i \times d_{i-1}}$ и вектор $\boldsymbol{b}^i \in \mathbb{R}^{d_i}$ — параметры i-го слоя. Слой с линейным преобразованием, имеющим описанный выше вид, называется полносвязным.

Описанное выше представление слоёв сети имеет некоторые аналогии с процессами, происходящими в биологических нейронных сетей (например, в головном мозге животных). Выпишем выражение для j-ой компоненты вектора $\boldsymbol{h}^i$:

$$h^i_j(\boldsymbol{h}^{i-1}) = f(\sum_{k=1}^{d_{i-1}} W^i_{jk} h^{i-1}_k + b^i_j).$$

Можно считать, что i-ый слой нейронной сети это набор из d_i нейронов. Каждый нейрон принимает сигнал от всех нейронов предыдущего слоя и передает свой сигнал нейронам следующего слоя. Сигнал, который передает j-ый нейрон в i-ом слое вычисляется следующим образом. Сначала вычисляется линейная комбинация

входных сигналов с весами W^i_{jk}, затем сигнал рассматриваемого нейрона рассчитывается как значение функции активации в точке, соответствующей значению линейной комбинации входов с добавленным значением параметра сдвига b^i_j. Обычно функция активации является неубывающей и принимает малые значения на отрицательной полуоси и большие на положительной полуоси. Тем самым, активация рассматриваемого нейрона зависит от соотношения между линейной комбинацией входных сигналов и числом b^i_j. Если $\sum\limits_{k=1}^{d_{i-1}} W^i_{jk} h^{i-1}_k + b^i_j < 0$, нейрон неактивен и либо не передает сигнал, либо передает сигнал малой величины, иначе нейрон активируется и передает сигнал нейронам следующего слоя. Название полносвязного слоя нейронной сети объясняется тем, что в таком слое на активацию каждого нейрона i-го слоя оказывают влияние все нейроны предыдущего слоя, таким образом между каждой парой нейронов из $(i-1)$-го и i-го слоя есть связь. На рисунке 1 изображена схема полносвязной нейронной сети.

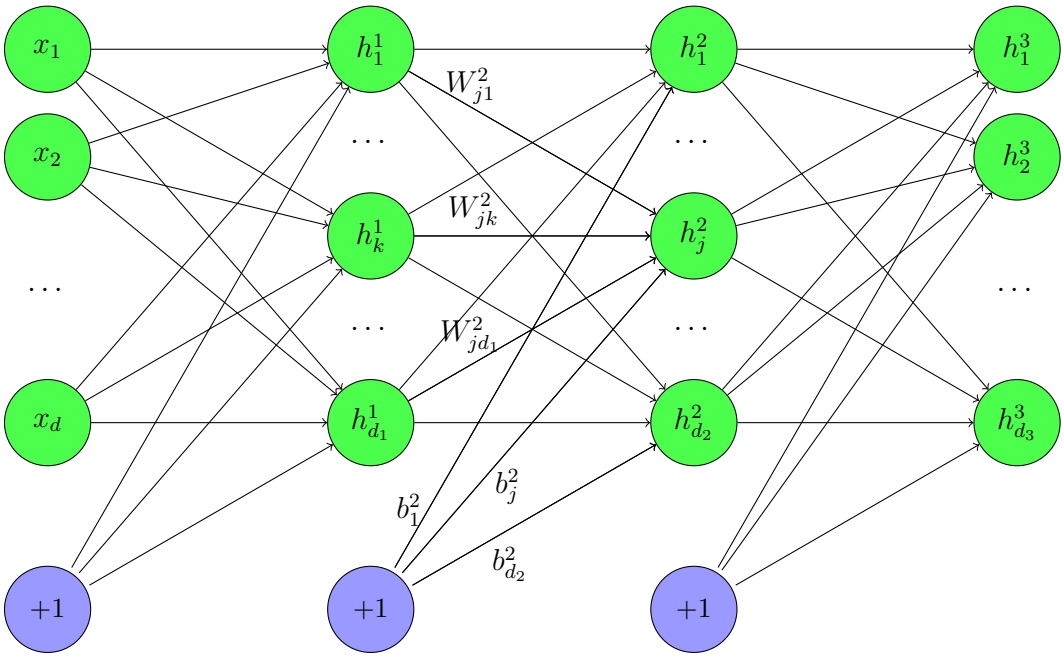

Рис. 1: Полносвязная нейронная сеть

Обучение нейронных сетей

Как было сказано выше, нейронная сеть задаёт параметрическое семейство функций $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{\theta})$ с вектором параметров $\boldsymbol{\theta}$. Для обучения нейронной сети требуется найти

такие значения параметров, при которых нейронная сеть наилучшим образом решает задачу классификации. Формально эта задача является задачей оптимизации, которая описана ниже.

Пусть дана обучающая выборка $X^\ell = \{(\boldsymbol{x}^i, y^i)\}_{i=1}^N$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{1, 2, \ldots, K\}$. Введем функцию потерь $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$, характеризующую величину неточности предсказания класса для i-го объекта выборки для алгоритма с параметрами $\boldsymbol{\theta}$. Для задачи классификации обычно используют кросс-энтропию:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) = -\sum_{k=1}^{K} [y_i = k] \log p_k(\boldsymbol{x}_i, \boldsymbol{\theta}).$$

Чем увереннее классификатор предсказывает правильный класс для i-ого объекта обучающей выборки, тем меньше значение указанной функции. Задача обучения заключается в поиске вектора параметров $\boldsymbol{\theta}$, минимизирующего среднее значение функции потерь на обучающей выборке:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) \to \min_{\boldsymbol{\theta}}.$$

Для решения указанной задачи оптимизации в случае больших выборок используют алгоритм стохастического градиентного спуска (SGD) или его модификации [8]. В простейшем варианте итерация алгоритма SGD выглядит следующим образом:

$$\begin{cases} i \sim \mathcal{U}\{1, 2, \ldots, N\} \\ \boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{old} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) \end{cases}$$

На каждой итерации выбирается случайный объект, вычисляется градиент по $\boldsymbol{\theta}$ функции потерь для выбранного объекта, затем выполняется сдвиг текущего значения параметров в направлении антиградиента функции потерь. Для эффективного вычисления градиента $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$ по параметрам нейронной сети разработан алгоритм “обратного распространения ошибки” [9]. Идея этого алгоритма заключается в том, что функция $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{\theta})$, заданная нейронной сетью, является композицией функций, и для дифференцирования такой функции можно применить правило дифференцирования сложной функции для каждого слоя в порядке от последнего к первому:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}^i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{i+1}} \times \frac{\partial \boldsymbol{h}^{i+1}}{\partial \boldsymbol{\theta}^i}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{i+1}} \times \frac{\partial \boldsymbol{h}^{i+1}}{\partial \boldsymbol{h}^i}.$$

Свёрточные нейронные сети

Для решения задачи классификации изображений были предложены свёрточные нейронные сети. На вход свёрточной нейронной сети поступает изображение $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{H \times W \times C}$ размера $H \times W$ пикселей, имеющее C цветовых каналов, заданное тензором интенсивностей всех цветовых каналов для каждого пикселя. Для учета специфики входных данных при обработке изображений используют свёрточные слои, в которых линейное преобразование является операцией свёртки. Это преобразование устроенно следующим образом. Выбирается набор из S фильтров с размерами $\ell \times \ell$ и C цветовыми каналами. Фильтры будем задавать тензором $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{\ell \times \ell \times C \times S}$. Для вычисления результата операции свёртки перебираются все фрагменты изображения $\boldsymbol{\mathcal{X}}$ размера $\ell \times \ell \times C$ и вычисляется взаимная корреляция между выбранным фрагментом и всеми фильтрами. Результаты этих вычислений объединяются в выходной тензор $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{H' \times W' \times S}$, где $H' = H - \ell + 1$, $W' = W - \ell + 1$. После чего к тензору $\boldsymbol{\mathcal{Y}}$ поэлементно применяется функция активации, далее этот тензор можно трактовать как изображение размера $H' \times W'$ с S цветовыми каналами и подать на вход следующему слою. Формально линейное преобразование, реализуемое свёрточным слоем записывается следующим образом:

$$\mathcal{Y}(x, y, s) = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \sum_{c=1}^{C} \mathcal{K}(i, j, c, s)\, \mathcal{X}(x + i - 1, y + j - 1, c).$$

Свёрточные слои имеют ряд преимуществ перед полносвязными:

- для вычисления результата свёрточного слоя требуется меньшие объёмы вычислительных ресурсов, чем для полносвязного слоя с такими же размерами входного и выходного массивов;
- свёрточный слой имеет меньше настраиваемых параметров, что уменьшает риск переобучения;

- использование свёрточных слоёв позволяет учесть специфику обрабатываемых данных: свёрточные слои на первых уровнях детектируют простые объекты (прямолинейные границы, перепады цвета) в разных участках изображения, на более глубоких уровнях происходит выделение более сложных объектов.

Кроме свёрточных слоёв в свёрточных нейронных сетях используются так называемые пулинг-слои (pooling). Эти слои предназначены для снижения пространственных размеров изображений, передаваемых по нейронной сети. В простейшем варианте работа пулинг-слоя заключается в выполнении прохода окном фиксированного размера по изображению и агрегации значений интенсивностей пикселей окна. Для агрегации интенсивностей используют заранее выбранную функцию, например максимум (max-pooling) или среднее (average-pooling). Выходом пулинг слоя является изображение меньшего размера, значения пикселей в котором соответствуют агрегированным значениям интенсивностей пикселей из соответствующего окна.

Более подробную информацию о свёрточных нейронных сетях можно найти в работах [1, 10, 9].

2.4 Уменьшение числа параметров решающего алгоритма

Как было сказано выше, при использовании ряда подходов к решению задачи обучения по прецедентам решающий алгоритм (функцию) a выбирают из некоторого фиксированного семейства $\mathcal{A}_\Theta$, параметризованного вектором из n параметров $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^n$. В этом случае правило, по которому объекту x сопоставляется предсказанное алгоритмом значение скрытой переменной $\hat{y} = a_{\boldsymbol{\theta}}(x)$ полностью определяется значением вектора параметров $\boldsymbol{\theta}$. Следовательно для хранения такого алгоритма в памяти компьютера достаточно сохранить параметры алгоритма. В связи с этим, возникает задача снижения объема памяти, необходимого для хранения параметров алгоритма. Однако, для решения этой задачи недостаточно просто закодировать параметры алгоритма без избыточности (решить задачу сжатия информации), так как необходимо обеспечить возможность эффективного применения обученного алгоритма на новых данных.

В данной работе рассматривается задача снижения числа параметров свёрточных нейронных сетей. Данная задача заключается в поиске альтернативной параметризации слоёв нейронных сетей. Формально, для уменьшения параметров выполняется переход к новому параметрическому семейству решающих алгоритмов $\mathcal{A}_\Phi \subset \mathcal{A}_\Theta$ с вектором из m параметров $\boldsymbol{\phi} \in \Phi \subset \mathbb{R}^m$, где $m < n$. Предполагается, что суженное семейство $\mathcal{A}_\Phi$ остаётся достаточно богатым в том смысле, что в рамках данного семейства удастся решить задачу обучения по прецедентам не значительно хуже, чем при использовании исходного семейства $\mathcal{A}_\Theta$. В качестве меры уменьшения числа параметров будем использовать величину $\frac{m}{n}$ — отношение числа параметров семейства $\mathcal{A}_\Phi$ к числу параметров исходного семейства $\mathcal{A}_\Theta$.

3 Обзор литературы

Во многих архитектурах нейронных сетей большинство параметров сосредоточенно в полносвязных слоях, как следствие большое количество работ, например [11, 12, 5, 13], посвящено данной проблеме. Однако после сжатия параметров полносвязных слоев, основная доля параметров оказывается сосредоточена в свёрточных слоях [5]. Кроме того, многие современные нейросетевые архитектуры не имеют крупных полносвязных слоев [14, 15]. В результате, недавно было предложено несколько техник сокращения числа параметров свёрточных слоев [16, 17, 18, 19, 20, 21].

Один из наиболее популярных подходов сжатия свёрточных слоев основан на удалении наименее важных весов из тензора параметров. Другой подход, получивший распространение, основан на идее квантизации весов, которая заключается в том, что веса могут принимать значения из заранее заданного набора чисел. Более подробно данные методы, а также их комбинирование описано в [16, 17, 19]. Предложенный в данной работе подход допускает возможность комбинирования с данными методами. Для увеличения коэффициента сжатия описанные техники могут быть объединены с кодированием Хаффмана, как это описано в [16], которое также может быть использовано совместно с разработанным в рамках данной работы методом.

Так же в последнее время появляются работы, использующие матричные и тензорные разложения для представления матрицы весов свёрточного слоя. Например, в работе [20] используется CP-разложение для матрицы весов, а в [21] — представление матрицы весов в виде произведения Кронекера для ускорения нейронных сетей. Кроме того использование данных техник позволило авторам добиться сжатия свёрточных слоев.

4 TT-разложение

Разложение тензорного поезда (Tensor Train, TT) — это способ компактного представления многомерных массивов, предложенный И. В. Оселедцом в 2012 году [6]. Представление тензора в TT-формате позволяет не только компактно хранить тензор, но и эффективно применять операции линейной алгебры к TT-тензорам. Ниже будет приведено формальное описание TT-разложения многомерных массивов и приведены некоторые важные свойства данного разложения.

TT-разложением для d-мерного тензора $\boldsymbol{\mathcal{A}}$ размера $n_1 \times n_2 \times \ldots \times n_d$ называется такой набор матриц $\boldsymbol{G}_k[i_k] \in \mathbb{R}^{r_{k-1} \times r_k}$, где $i_k = 1, \ldots, n_k$, $k = 1, \ldots, d$, и $r_0 = r_d = 1$, что каждый элемент тензора $\boldsymbol{\mathcal{A}}$ может быть вычислен следующим образом

$$\mathcal{A}(i_1, i_2, \ldots, i_d) = \boldsymbol{G}_1[i_1]\boldsymbol{G}_2[i_2] \ldots \boldsymbol{G}_d[i_d]. \tag{1}$$

Набор чисел $\{r_i\}_{i=0}^d$ называется рангами TT-разложения или *TT-рангами*, набор чисел $\{n_i\}_{i=0}^d$ называется *модами* тензора $\boldsymbol{\mathcal{A}}$, матрицы $\boldsymbol{G}_k[\cdot]$ называются *TT-ядрами*.

Будем говорить, что d-мерный тензор $\boldsymbol{\mathcal{A}}$ представлен в TT-формате, если задано некоторое его TT-разложение. Отметим, что матрицы $\boldsymbol{G}_k[i_k]$ соответствующие одной и той же размерности k должны иметь одинаковый размер $r_{k-1} \times r_k$. Кроме того ранги $r_0 = r_d = 1$. Это требование необходимо для того, чтобы результатом матричного произведения (1) являлась скалярная величина.

Введем обозначения $n = \max\limits_{k=1,2,\ldots,d} n_k$, $r = \max\limits_{k=0,1,\ldots,d} r_k$. Из представления (1) можно заметить, что для описания тензора в TT-формате требуется $\sum\limits_{k=1}^{d} n_k r_{k-1} r_k = \mathcal{O}(dnr^2)$ параметров в то время как при перечислении всех элементов тензора требуется

$\prod_{k=1}^{d} n_k = \mathcal{O}(n^d)$ параметров. Тем самым представление тензора в TT-формате с малыми значениями TT-рангов позволяет снизить объём памяти, необходимый для хранения тензора.

Любой тензор можно представить в TT-формате с наперёд заданной точностью, используя алгоритм TT-SVD (см. теорему 2.2 Оселедца [6]), но, возможно, такое представление будет иметь большие TT-ранги. Кроме того, используя модификацию алгоритма TT-SVD, любой тензор можно квазиоптимально представить в TT-формате с TT-рангами не превышающими наперед заданные ограничения. Таким образом, возникает задача поиска баланса между желаемой точностью TT-представления тензора и объёмом памяти, необходимым для хранения такого представления.

В данной работе будет рассмотрен следующий способ применения TT-разложения. Пусть необходимо вычислить вектор $\boldsymbol{y} \in \mathbb{R}^N$, являющийся произведением матрицы $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ и вектора $\boldsymbol{x} \in \mathbb{R}^M$, то есть:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}.$$

Перепишем это соотношение поэлементно:

$$y(i) = \sum_{j=1}^{M} W(i,j)x(j), \qquad i = 1, 2, \ldots, N.$$

В случае работы с большой матрицей $\boldsymbol{W}$ может оказаться выгодно представить матрицу как многомерный тензор и перейти к выполнению операций над TT-представлением данного тензора. С этой целью вводится специальное TT-представление для матриц.

Формально, рассмотрим два набора чисел $\{n_k\}_{k=1}^d$, $\{m_k\}_{k=1}^d$ и два взаимно однозначных отображения $(i_1, i_2, \ldots, i_d) \leftrightarrow i$, $(j_1, j_2, \ldots j_d) \leftrightarrow j$ такие, что:

$$\begin{gathered} \prod_{k=1}^{d} n_k = N \qquad \prod_{k=1}^{d} m_i = M \\ 1 \leqslant i_k \leqslant n_k, \qquad 1 \leqslant j_k \leqslant m_k, \quad k = 1, 2, \ldots d. \end{gathered} \tag{2}$$

Теперь рассмотрим d-мерный тензор $\boldsymbol{\mathcal{W}}$, имеющий размерности $(m_1 n_1) \times (m_2 n_2) \times \ldots \times (m_d n_d)$. Каждому элементу матрицы $\boldsymbol{W}$ сопоставим элемент тензора $\boldsymbol{\mathcal{W}}$ по

следующему правилу:

$$W(i,j) = W((i_1, i_2, \ldots, i_d), (j_1, j_2, \ldots, j_d)) = \mathcal{W}((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)),$$

то есть выполним переход от скалярных индексов i и j к составным индексам $(i_1, i_2, \ldots, i_d)$ и $(j_1, j_2, \ldots, j_d)$ и для индексации размерностей тензора $\boldsymbol{\mathcal{W}}$ будем использовать парные индексы $(i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)$. Аналогичные действия выполним для векторов $\boldsymbol{x}$ и $\boldsymbol{y}$ и представим их в виде d-мерных тензоров $\boldsymbol{\mathcal{X}}$ и $\boldsymbol{\mathcal{Y}}$:

$$x(j) = \mathcal{X}(j_1, j_2, \ldots, j_d)$$
$$y(i) = \mathcal{Y}(i_1, i_2, \ldots, i_d)$$

.

Наконец, представим тензор $\boldsymbol{\mathcal{W}}$ в TT-формате:

$$\mathcal{W}((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)) = \boldsymbol{G}_1[i_1, j_1]\boldsymbol{G}_2[i_2, j_2] \ldots \boldsymbol{G}_d[i_d, j_d].$$

В результате получим окончательное представление для элементов матрицы $\boldsymbol{W}$:

$$W(i;j) = W((i_1, i_2, \ldots, i_d); (j_1, j_2, \ldots, j_d)) = \boldsymbol{G}_1[i_1, j_1]\boldsymbol{G}_2[i_2, j_2] \ldots \boldsymbol{G}_d[i_d, j_d]. \quad (3)$$

Представление матрицы $\boldsymbol{W}$ в виде (3) будем называть TT-форматом или TT-представлением матрицы $\boldsymbol{W}$.

Подставляя TT-представление матрицы $\boldsymbol{W}$ в выражение $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ и представив вектора $\boldsymbol{x}$ и $\boldsymbol{y}$ с помощью тензоров $\boldsymbol{\mathcal{X}}$ и $\boldsymbol{\mathcal{Y}}$, получим:

$$\mathcal{Y}(i_1, i_2, \ldots, i_d) = \sum_{j_1=1}^{m_1} \sum_{j_2=1}^{m_2} \ldots \sum_{j_d=1}^{m_d} \left[\left(\boldsymbol{G}_1[i_1, j_1]\boldsymbol{G}_2[i_2, j_2] \ldots \boldsymbol{G}_d[i_d, j_d] \right) \mathcal{X}(j_1, j_2, \ldots, j_d) \right]. \quad (4)$$

Учтём, что в выражении (4) в круглых скобках заключено произведение d матриц (TT-ядер матрицы $\boldsymbol{W}$). Чтобы отразить этот факт, запишем выражение для матричного произведения явно, используя для индексации столбцов и строк матриц $\boldsymbol{G}_k[i_k, j_k]$ индексы α_{k-1}, α_k соответственно. Перепишем выражение (4), добавив суммирование по индексам $\alpha_1, \alpha_2, \ldots, \alpha_{d-1}$ (суммирование по $\alpha_0 = \alpha_d = 1$ выполнять не нужно, так как указанные индексы имеют единственное возможное значение):

$$\mathcal{Y}(i_1, i_2, \ldots, i_d) = \sum_{\substack{j_1, j_2, \ldots, j_d \\ \alpha_1, \alpha_2, \ldots \alpha_{d-1}}} \Big[(G_1[i_1, j_1](1, \alpha_1) G_2[i_2, j_2](\alpha_1, \alpha_2) \ldots G_d[i_d, j_d](\alpha_{d-1}, 1)) \cdot$$

$$\cdot \mathcal{X}(j_1, j_2, \ldots, j_d) \Big].$$

Отметим, что в полученном выражении можно выполнить изменение порядка суммирования и группировку слагаемых таким образом, что для вычисления произведения матрицы на вектор с использованием TT-формата потребуется $\mathcal{O}(dr^2 n \max\{M, N\})$ операций, где $r = \max\limits_{k=0,1,\ldots,d} r_k$, $n = \max\limits_{k=1,2,\ldots,d} n_k$. Для явного вычисления произведения требуется $\mathcal{O}(NM) = \mathcal{O}(n^d m^d)$ операций, где $m = \max\limits_{k=1,2,\ldots,d} m_k$. Таким образом, представление матрицы $\boldsymbol{W}$ в TT-формате позволяет снизить объём вычислительных ресурсов (объём памяти и число операций), необходимый для вычисления произведения $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$.

5 Метод сжатия полносвязных слоёв

В работе [5] предложен метод для сжатия полносвязных слоёв нейронных сетей. Полносвязный слой осуществляет линейное преобразование над входным вектором $\boldsymbol{x} \in \mathbb{R}^M$, параметрами слоя являются матрица весов $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ и вектор сдвига $\boldsymbol{b} \in \mathbb{R}^N$. Результирующий вектор $\boldsymbol{y} \in \mathbb{R}^N$ вычисляется по формуле:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}.$$

Отметим, что большая часть параметров полносвязного слоя сосредоточена в матрице $\boldsymbol{W}$. Поэтому интерес представляет уменьшение числа параметров необходимых для хранения именно матрицы $\boldsymbol{W}$.

Идея описываемого метода заключается в представлении матрицы $\boldsymbol{W}$ в виде многомерного тензора $\boldsymbol{\mathcal{W}}$ и применении TT-разложения к полученному тензору. Эта процедура подробно описана в разделе 4, с учётом выражений (2), (3), (4), преобразование, осуществляемое полносвязным слоем, можно записать в виде:

$$\mathcal{Y}(i_1, i_2, \ldots, i_d) = \sum_{j_1, j_2, \ldots, j_d} \left[\left(\boldsymbol{G}_1[i_1, j_1] \boldsymbol{G}_2[i_2, j_2] \ldots \boldsymbol{G}_d[i_d, j_d] \right) \mathcal{X}(j_1, j_2, \ldots, j_d) \right] + \\ + \mathcal{B}(i_1, i_2, \ldots, i_d). \tag{5}$$

Здесь вектор $\boldsymbol{b}$ также представлен в виде многомерного тензора $\boldsymbol{\mathcal{B}}$. Выражение (5) позволяет параметризовать полносвязный слой с помощью TT-ядер $\boldsymbol{G}_1[\cdot, \cdot]$, $\boldsymbol{G}_2[\cdot, \cdot]$, …, $\boldsymbol{G}_d[\cdot, \cdot]$ и вектора $\boldsymbol{b}$. Параметризованный таким образом полносвязных слой будем называть TT-полносвязным слоем. TT-полносвязный слой имеет меньшее число настраиваемых параметров по сравнению с исходным полносвязным слоем. Тем не менее, TT-полносвязный слой оказывается достаточно гибким для того, чтобы нейросеть со сжатыми слоями была способна решить задачу классификации без значительной потери в качестве по сравнению с исходной сетью. Для обучения сети с описанной модификацией слоёв применим алгоритм SGD, всё что необходимо изменить это производить оптимизацию функции потерь по новым параметрам полносвязных слоёв. Формулы для вычисления градиента функции потерь приведены в работе [5].

В задаче классификации изображений ImageNet ILSVRC-2012 [22] для нейросетей архитектуры VGG [4] авторам данного подхода удалось добиться сжатия отдельных полносвязных слоёв более чем в 190 000 раз без значительной потери в качестве классификации.

Рассматриваемый подход имеет ряд ограничений. Во-первых, авторам работы не удалось произвести совместное обучение нескольких TT-полносвязных слоёв. Данная проблема была решена в рамках курсовой работы автора данной работы путем применения техники Batch Normalization [23] к нейронным сетям с TT-полносвязными слоями. Во-вторых, при сжатии полносвязных слоёв свёрточных нейронных сетей в сотни тысяч раз, сжатие всей сети оказывается не таким значительным, так как при уменьшении числа параметров полносвязных слоёв большая часть параметров концентрируется в свёрточных слоях. В данной работе разработан подход для уменьшения числа параметров свёрточных слоёв, расширяющий описанные выше идеи.

6 Предложенный метод

6.1 Представление свёртки в виде матричного произведения

Свёрточный слой осуществляет преобразование исходного трехмерного тензора $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{H \times W \times C}$ в выходной трехмерный тензор $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{H-\ell+1 \times W-\ell+1 \times S}$, выполняя свёртку тензора $\boldsymbol{\mathcal{X}}$ с тензором весов $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{\ell \times \ell \times C \times S}$ по следующей формуле:

$$\mathcal{Y}(x, y, s) = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \sum_{c=1}^{C} \mathcal{K}(i, j, c, s)\, \mathcal{X}(x + i - 1, y + j - 1, c). \tag{6}$$

Для ускорения вычисления операции (6), осуществляемой свёрточным слоем, большинство современных программ для обучения глубинных нейронных сетей формулируют данную операцию в терминах умножения матриц. В частности, такая формулировка допускает эффективную имплементацию вычислений на графических процессорах. Мы будем использовать данную формулировку в качестве мотивации для предложенного в данной работе метода. Дальнейшая часть раздела будет посвящена описанию данного метода вычисления свёртки.

Для удобства изложения будем использовать обозначения $H' = H - \ell + 1$ и $W' = W - \ell + 1$. Преобразуем выходной тензор $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{H' \times W' \times S}$ в матрицу $\boldsymbol{Y}$ размера $H'W' \times S$ по следующей формуле

$$\mathcal{Y}(x, y, s) = Y(x + H'(y - 1), s),$$

где $x = 1, \ldots, H', y = 1, \ldots, W', s = 1, \ldots, S$.

Сформируем по входному тензору $\boldsymbol{\mathcal{X}}$ матрицу $\boldsymbol{X}$ размера $H'W' \times \ell^2 C$, k-ая строка которой соответствует подтензору входного тензора $\boldsymbol{\mathcal{X}}$ размера $\ell \times \ell \times C$, использующемуся для вычисления k-ой строки матрицы $\boldsymbol{Y}$.

$$\mathcal{X}(x + i - 1, y + j - 1, c) = X(x + H'(y - 1), i + \ell(j - 1) + \ell^2(c - 1)),$$

где $x = 1, \ldots, H',\ y = 1, \ldots, W',\ i, j = 1, \ldots, \ell$. Наконец, преобразуем тензор весов $\boldsymbol{\mathcal{K}}$ в матрицу $\boldsymbol{K}$ размера $\ell^2 C \times S$

$$\mathcal{K}(i, j, c, s) = K(i + \ell(j - 1) + \ell^2(c - 1), s).$$

Используя введенные матрицы $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{K}$ мы можем выразить преобразование (6), как умножение матриц, а именно $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{K}$.

6.2 Метод сжатия свёрточных слоёв

В этом разделе описаны два метода представления тензора параметров $\boldsymbol{\mathcal{K}}$ свёрточного слоя в TT-формате. Первый подход заключается в применении TT-разложения напрямую к 4-х мерному тензору $\boldsymbol{\mathcal{K}}$:

$$\mathcal{K}(x, y, c, s) = \boldsymbol{G}_1[x]\boldsymbol{G}_2[y]\boldsymbol{G}_3[c]\boldsymbol{G}_4[s]. \tag{7}$$

Однако данный подход не оказывается эффективным на практике. Рассмотрим свёрточный слой с фильтрами размера 1×1. В этом случае матрица $\boldsymbol{K}$, описанная в предыдущем разделе, будет иметь размер $C \times S$ и использование TT-формата вида (7) эквивалентно использованию низкорангового формата для матриц. Однако, в работе [5] было экспериментально показано, что TT-формат для матриц оказывается более эффективным чем низкоранговое разложение. Поэтому перейдем к рассмотрению представления тензора $\boldsymbol{\mathcal{K}}$, которое совпадает с TT-форматом матрицы $\boldsymbol{K}$ для свёрток с фильтрами размера 1×1.

Рассмотрим матрицу $\boldsymbol{K} \in \mathbb{R}^{\ell^2 C \times S}$, связанную с тензором $\boldsymbol{\mathcal{K}}$ по правилу:

$$\mathcal{K}(x, y, c, s) = K(x + \ell(y-1) + \ell^2(c-1), s).$$

Применим TT-разложение к матрице $\boldsymbol{K}$ следующим образом. Рассмотрим числа C и S в виде произведения натуральных чисел $\{C_i\}_{i=1}^d, \{S_i\}_{i=1}^d$ таких, что $C = \prod\limits_{i=1}^{d} C_i$, $S = \prod_{i=1}^{d} S_i$. Представим матрицу $\boldsymbol{K}$ как $(d+1)$-мерный тензор $\tilde{\boldsymbol{\mathcal{K}}}$, k-ая размерность которого есть $C_k S_k$ для $k = 1, 2, \ldots, d$ и ℓ^2 для $k = 0$:

$$\tilde{\mathcal{K}}((x + \ell(y-1), 1), (c_1, s_1), \ldots, (c_d, s_d)) = K(x + \ell(y-1) + \ell^2(c'-1), s'),$$

где

$$c' = c_1 + \sum_{i=2}^{d}(c_i - 1)\prod_{j=1}^{i-1} c_j, \quad 1 \leqslant c_i \leqslant C_i, \quad s' = s_1 + \sum_{i=2}^{d}(s_i - 1)\prod_{j=1}^{i-1} s_j, \quad 1 \leqslant s_i \leqslant S_i.$$

Теперь рассмотрим TT-формат для тензора $\tilde{\boldsymbol{\mathcal{K}}}$:

$$\tilde{\mathcal{K}}((x+\ell(y-1),1),(c_1,s_1),\ldots,(c_d,s_d)) = \tilde{\boldsymbol{G}}_0[x+\ell(y-1),1]\boldsymbol{G}_1[c_1,s_1]\ldots\boldsymbol{G}_d[c_d,s_d].$$

Для упрощения нотации будем индексировать 0-е ядро с помощью индексов x и y: $\boldsymbol{G}_0[x,y] = \tilde{\boldsymbol{G}}_0[x+\ell(y-1),1]$. Подставляя ядро $\boldsymbol{G}_0$ в выписанные выше формулы окончательно получим следующее представление тензора $\boldsymbol{\mathcal{K}}$:

$$\mathcal{K}(x,y,c',s') = \mathcal{K}(x,y,(c_1,\ldots,c_d),(s_1,\ldots,s_d)) = \boldsymbol{G}_0[x,y]\boldsymbol{G}_1[c_1,s_1]\ldots\boldsymbol{G}_d[c_d,s_d]. \quad (8)$$

Для уменьшения числа параметров свёрточного слоя будем использовать новые параметры $\boldsymbol{G}_0[\cdot,\cdot]$, $\boldsymbol{G}_1[\cdot,\cdot]$, …, $\boldsymbol{G}_d[\cdot,\cdot]$. Выпишем формулы для вычисления результата преобразования свёртки для предложенной модификации свёрточного слоя. Представим входной 4-х мерный тензор $\boldsymbol{\mathcal{X}}$ в виде $(d+2)$-мерного тензора $\tilde{\boldsymbol{\mathcal{X}}}$ размера $H \times W \times C_1 \times C_2 \times \ldots \times C_d$. TT-свёрточный слой преобразует тензор $\tilde{\boldsymbol{\mathcal{X}}}$ в тензор $\tilde{\boldsymbol{\mathcal{Y}}}$ размера $H' \times W' \times S_1 \times S_2 \times \ldots \times S_d$ по формуле:

$$\begin{aligned}\tilde{\mathcal{Y}}(x,y,s_1,\ldots,s_d) = \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\sum_{c_1,\ldots,c_d}\Big[&\tilde{\mathcal{X}}(x+i-1,y+j-1,c_1,\ldots c_d)\cdot \\ &\cdot \boldsymbol{G}_0[x,y]\boldsymbol{G}_1[c_1,s_1]\ldots\boldsymbol{G}_d[c_d,s_d]\Big]\end{aligned} \quad (9)$$

Идея предложенной модификации свёрточного слоя проиллюстрирована на рисунке 2.

Параметризованный таким образом свёрточный слой будем называть TT-свёрточным слоем. Легко видеть, что TT-свёрточный слой использует $\mathcal{O}(\ell^2 r + d\bar{C}\bar{S}r^2)$ параметров, где r — максимальный ранг TT-разложения (8), $\bar{C} = \max\limits_{i=1,\ldots,d} C_i$, $\bar{S} = \max\limits_{i=1,\ldots,d} S_i$. В то же время исходный свёрточный слой содержит $\mathcal{O}(\ell^2\bar{C}^d\bar{S}^d)$ параметров. Варьируя параметр r, можно контролировать степень сжатия сверточных слоев.

Для обучения нейронной сети с TT-свёрточными слоями необходимо выполнять оптимизацию функцию потерь по введенным выше параметрам TT-свёртки при помощи метода стохастического градиентого спуска. В программной реализации описанного метода необходимые градиенты функции потерь по TT-ядрам вычислялись автоматически средствами библиотеки TensorFlow [24].

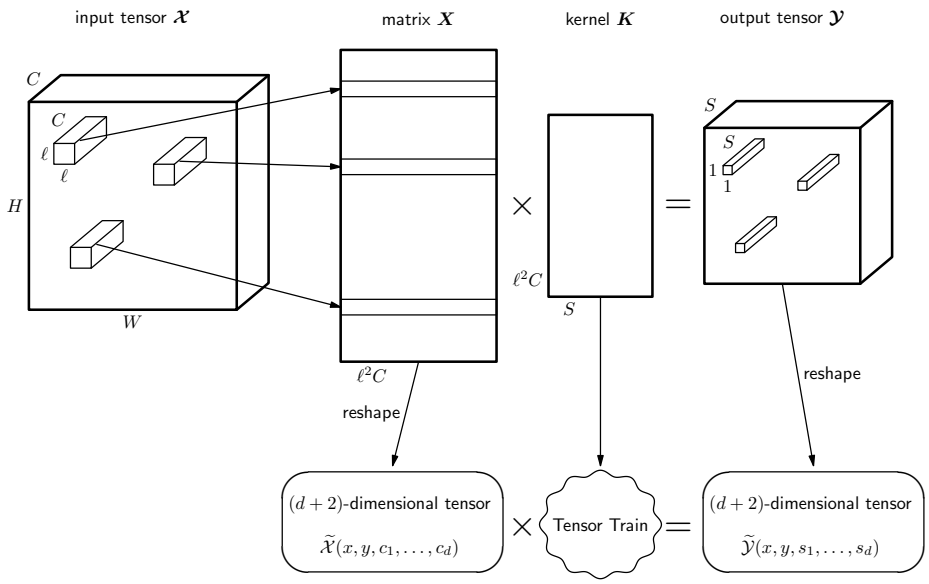

Рис. 2: Иллюстрация идеи предложенного подхода

7 Вычислительные эксперименты

В данной секции приводится описание и результаты экспериментального исследования предложенного метода. Предложенный метод был программно реализован на языке Python с использованием библиотеки TensorFlow [24], предназначенной для работы с графами вычислений. Программная реализация метода и программы для воспроизведения описанных экспериментов доступны в репозитории https://github.com/timgaripov/TensorNet-TF.

Экспериментальная оценка предложенного метода сжатия была проведена на наборе изображений CIFAR-10 [25]. Данный набор данных состоит из двух частей обучающей и валидационной, которые включают 50 000 и 10 000 объектов соответственно. Объекты представляют собой изображения в формате RGB размера 32×32. Каждое изоображение отнесено к одному из 10 классов по 6000 изображений для каждого класса. Во всех экспериментах обучение нейронных сетей проводилось методом стохастического градиентного спуска с моментами, где коэффициент момента брался равным 0.9.

Останов алгоритма осуществлялся через 100 эпох, где под одной эпохой понимается проход алгоритма по всей обучающей выборке. В качестве расписания для

множителя перед градиентом (learning rate) использовалась следующая схема: исходный learning rate брался равным 0.1 и уменьшался в 10 раз каждые 30 эпох.

В качестве базовых архитектур были выбраны две свёрточные сети (будем называть их conv и conv-fc). Архитектура первой сети (conv):

1. свёрточный слой (64 выходных каналов)
2. нормализующий слой (batch normalization layer [23])
3. ReLU нелинейность
4. max-пулинг (с окном 3×3 и шагом 2)
5. свёрточный слой (128 выходных каналов)
6. нормализующий слой (batch normalization layer)
7. ReLU нелинейность
8. max-пулинг (с окном 3×3 и шагом 2)
9. свёрточный слой (128 выходных каналов)
10. нормализующий слой (batch normalization layer)
11. ReLU нелинейность
12. свёрточный слой (128 выходных каналов)
13. average-пулинг (с окном 4×4 и шагом 4)
14. полносвязный слой с 128 входными и 10 выходными нейронами.

Фильтры во всех свёрточных слоях имеют размер 3×3.

В качестве первого эксперимента проведено сравнение предложенного TT-свёрточного слоя (9) с «наивным подходом» (7), то есть методом, в котором тензор весов свёрточного слоя рассматривается, как четырехмерный тензор и представляется в TT-формате. Результаты представлены в таблице 1.

Вторая базовая сеть (conv-fc) была получена из первой путем замены average-пулинг слоя на два полносвязных слоя размера 8192×1536 и 1536×512 соответственно. Результаты экспериментов для данной архитектуры приведены в таблице 2. Для того, чтобы сравнить полученный метод с работой, посвященной сжатию только полносвязных слоёв, были проведены эксперименты по сжатию только полносвязных слоёв второй архитектуры. Представляя данные два полносвязных слоя в TT-формате, удалось получить сжатие сети в 21 раз с потерей 0.7%.

Модель	точность (%)	сжатие
conv (базовая сеть)	90.7	1
TT-conv	89.9	2.02
TT-conv	89.2	2.53
TT-conv	89.3	3.23
TT-conv	88.7	4.02
TT-conv (naive)	88.3	2.02
TT-conv (naive)	87.6	2.90

Таблица 1: Результаты сжатия первой базовой сети ('conv'); 'TT-conv': сжимаются свёрточные слои нейронной сети; 'TT-conv (naive)': сжатие свёрточных слоев наивным подходом (7), в котором TT-разложение к тензору весов применено напрямую. Разные строки с одной и той же моделью соответствуют разному выбору рангов и мод TT-разложения. Приведены значения точности классификации на валидационной выборке.

Модель	точность (%)	сжатие
conv-fc (базовая сеть)	90.5	1
conv-TT-fc	90.3	10.72
conv-TT-fc	89.8	19.38
conv-TT-fc	89.8	21.01
TT-conv-TT-fc	90.1	9.69
TT-conv-TT-fc	89.7	41.65
TT-conv-TT-fc	89.4	82.87

Таблица 2: Результаты сжатия второй базовой сети ('conv-fc'), где большая часть параметров приходится на полносвязные слои; 'conv-TT-fc': сжимаются только полносвязные слои нейронной сети; 'TT-conv-TT-fc': сжимаются полносвязные и свёрточные слои нейронной сети. Разные строки с одной и той же моделью соответствуют разному выбору рангов и мод TT-разложения. Приведены значения точности классификации на валидационной выборке.

Изначально, большая часть параметров нейронной сети приходилась на полносвязные слои, однако после сжатия их при помощи представления матрицы весов в TT-формате, количество параметров свёрточных слоев начинает значительно превосходить количество оставшихся параметров TT-полносвязных слоев. Таким образом, свёрточные слои не позволяют сильно сократить количество параметров архитектуры только сжатием полносвязных слоев. Замена свёрточных слоев на TT-свёрточные слои позволяет сжать сеть в 80 раз с потерей 1.1% качества.

По результатам проведенных экспериментов можно сделать следующие выводы.

- Предложенная параметризация TT-свёрточного слоя (8) действительно оказывается эффективнее наивного подхода (7).
- Использование TT-свёрточных слоёв позволяет производить сжатие многослойных свёрточных нейронных сетей без значительных потерь в качестве классификации. Метод позволяет выбрать желаемое соотношение между числом параметров сети и качеством классификации за счёт выбора мод и рангов TT-разложения.
- Предложенный метод допускает возможность его совместного применения с предложенным ранее методом для сжатия полносвязных слоёв.

8 Заключение

В рамках данной работы был разработан метод уменьшения числа параметров свёрточных слоёв нейронных сетей. Предложенный метод основан на специальном представлении тензора параметров свёрточного слоя в виде многомерного массива и хранении этого массива в TT-формате. Данный подход представляет собой альтернативный способ параметризации преобразования свёртки, который позволяет снизить число параметров преобразования и эффективно выполнять данное преобразование.

В работе экспериментально показана возможность обучения многослойных свёрточных нейронных сетей, свёрточные слои в которых модифицированы предложенным методом. Кроме того, показана возможность успешного комбинированного сжатия свёрточных и полносвязных слоёв.

По результатам работы сделаны доклады на международных конференциях: NIPS workshop, 2016 (Барселона) [26]; «Ломоносов», 2017 (Москва) [27].

Список литературы

[1] *Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E.* Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. — 2012. — С. 1097–1105.

[2] Rethinking the inception architecture for computer vision / Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe и др. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — С. 2818–2826.

[3] *Kalchbrenner Nal, Grefenstette Edward, Blunsom Phil.* A convolutional neural network for modelling sentences // *arXiv preprint arXiv:1404.2188.* — 2014.

[4] *Simonyan Karen, Zisserman Andrew.* Very Deep Convolutional Networks for Large-Scale Image Recognition // *arXiv preprint arXiv:1409.1556.* — 2014.

[5] Tensorizing neural networks / Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, Dmitry P Vetrov // Advances in Neural Information Processing Systems. — 2015. — С. 442–450.

[6] *Oseledets I. V.* Tensor-Train Decomposition // *SIAM J. Scientific Computing.* — 2011. — Т. 33, № 5. — С. 2295–2317.

[7] *Воронцов К. В.* Математические методы обучения по прецедентам (теория обучения машин). — Москва, 2011.

[8] *Bottou Léon.* Large-scale machine learning with stochastic gradient descent // Proceedings of COMPSTAT'2010. — Springer, 2010. — С. 177–186.

[9] *Goodfellow Ian, Bengio Yoshua, Courville Aaron.* Deep learning. — MIT Press, 2016.

[10] *LeCun Yann, Bengio Yoshua, Hinton Geoffrey.* Deep learning // *Nature.* — 2015. — Т. 521, № 7553. — С. 436–444.

[11] Compressing neural networks with the hashing trick / Wenlin Chen, James Wilson, Stephen Tyree и др. // International Conference on Machine Learning. — 2015. — С. 2285–2294.

[12] *Xue Jian, Li Jinyu, Gong Yifan.* Restructuring of deep neural network acoustic models with singular value decomposition. // Interspeech. — 2013. — С. 2365–2369.

[13] Low-rank matrix factorization for deep neural network training with high-dimensional output targets / Tara N Sainath, Brian Kingsbury, Vikas Sindhwani и др. // Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on / IEEE. — 2013. — С. 6655–6659.

[14] Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia и др. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2015. — С. 1–9.

[15] Deep residual learning for image recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — С. 770–778.

[16] *Han Song, Mao Huizi, Dally William J.* Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding // *arXiv preprint arXiv:1510.00149.* — 2015.

[17] Learning both weights and connections for efficient neural network / Song Han, Jeff Pool, John Tran, William Dally // Advances in Neural Information Processing Systems. — 2015. — С. 1135–1143.

[18] Predicting parameters in deep learning / Misha Denil, Babak Shakibi, Laurent Dinh и др. // Advances in Neural Information Processing Systems. — 2013. — С. 2148–2156.

[19] PerforatedCNNs: Acceleration through elimination of redundant convolutions / Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, Pushmeet Kohli // Advances in Neural Information Processing Systems. — 2016. — С. 947–955.

[20] Speeding-up convolutional neural networks using fine-tuned cp-decomposition / Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba и др. // *arXiv preprint arXiv:1412.6553.* — 2014.

[21] *Wang Min, Liu Baoyuan, Foroosh Hassan.* Factorized Convolutional Neural Networks // *arXiv preprint arXiv:1608.04337.* — 2016.

[22] Imagenet large scale visual recognition challenge / Olga Russakovsky, Jia Deng, Hao Su и др. // *International Journal of Computer Vision.* — 2015. — Т. 115, № 3. — С. 211–252.

[23] *Ioffe Sergey, Szegedy Christian.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // Proceedings of the 32nd International Conference on Machine Learning (ICML-15). — 2015. — С. 448–456.

[24] Tensorflow: Large-scale machine learning on heterogeneous distributed systems / Martín Abadi, Ashish Agarwal, Paul Barham и др. // *arXiv preprint arXiv:1603.04467.* — 2016.

[25] *Krizhevsky Alex.* Learning Multiple Layers of Features from Tiny Images. — 2009.

[26] Ultimate tensorization: compressing convolutional and FC layers alike / Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, Dmitry Vetrov // *arXiv preprint arXiv:1611.03214.* — 2016.

[27] *Гарипов Т. И.* Применение тензорного разложения для сжатия свёрточных нейронных сетей // Сборник тезисов XXIV Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов-2017» секция «Вычислительная математика и кибернетика». — Издательский отдел факультета вычислительной математики и кибернетики МГУ имени М. В. Ломоносова, 2017. — С. 10–11.