ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2021, том 61, № 5, с. 800–812

ОБЩИЕ ЧИСЛЕННЫЕ МЕТОДЫ

УДК 519.65

МАЛОРАНГОВОЕ ПРЕДСТАВЛЕНИЕ НЕЙРОННЫХ СЕТЕЙ[1)]

© 2021 г. Ю. В. Гусак[1,*], Т. К. Даулбаев[1,], И. В. Оселедец[1,2], Е. С. Пономарев[1], А. С. Чихоцкий[1]**

[1] *121205 Москва, Большой бульвар, 30, стр. 1, Сколковский институт науки и технологий, Россия*

[2] *119333 Москва, ул. Губкина, 8, Институт вычислительной математики им. Г.И. Марчука Российской академии наук, Россия*

**e-mail: y.gusak@skoltech.ru*

***e-mail: t.daulbaev@skoltech.ru*

Поступила в редакцию 24.12.2020 г.
Переработанный вариант 24.12.2020 г.
Принята к публикации 14.01.2021 г.

Представлен новый метод ускорения глубоких нейронных сетей, который использует основные идеи сокращения размерности для решения уравнений в динамических системах. В основе предложенного метода лежит алгоритм поиска подматрицы максимального объема (MaxVol). Эффективность разработанного метода продемонстрирована на задаче ускорения предобученных нейронных сетей на задаче классификации изображений для трех разных наборов данных. Показано, что во многих практических задачах возможно эффективно заменить сверточные слои на полносвязные с малым числом параметров и меньшей вычислительной сложностью без существенной потери точности. Библ. 39. Фиг. 3. Табл. 4.

Ключевые слова: ускорение нейронных сетей, MaxVol, машинное обучение, анализ компонент.

DOI: 10.31857/S0044466921050100

1. ВВЕДЕНИЕ

Связь между глубокими нейронными сетями и системами обыкновенных дифференциальных уравнений (ОДУ) была показана в [1]–[4]. В упомянутых работах выход слоя нейронной сети при прямом проходе был представлен в виде состояния динамической системы в определенное время. Один из эффективных методов ускорения решения динамических систем – конструирование упрощенных моделей (см. [5]). Классический подход для построения таких моделей – метод дискретной эмпирической интерполяции (DEIM, см. [6]). Идея данного метода заключается в построении приближения вектора состояния вектором малой размерности в комбинации с эффективным пересчетом коэффициентов в полученном пространстве малой размерности с помощью нахождения подматрицы большого объема (т.е. с большим по значению определителем).

В настоящей работе мы используем связь нейросетевых алгоритмов и ОДУ для построения малорангового представления для предобученных сверточных и полносвязных нейронных сетей. Полученное малоранговое представление является полносвязной сетью с существенно меньшим числом нейронов в каждом слое, что значительно ускоряет работу модели.

Следуя подходу алгоритмов для ОДУ, предполагаем, что выходные тензоры для части слоев нейронной сети лежат в пространстве малой размерности. Мы называем это допущение *предположением о малоранговости*. Далее в работе показываем подкрепляющие его экспериментальные данные.

[1)]Работа выполнена при финансовой поддержке РФФИ (коды проектов 19-31-90172, 20-31-90127) и Минобрнауки РФ, проект 14.756.31.0001.

Итак, пусть $\mathbf{x}$ – объект из набора данных (например, картинка). Пусть $\mathbf{z}_k = \mathbf{z}_k(\mathbf{x})$ – векторизованный выходной тензор k-го слоя. Предположим, существует матрица $V_k \in \mathbb{R}^{D_k \times R_k}$ ($D_k \gg R_k$) такая, что

$$\mathbf{z}_k \cong V_k \mathbf{c}_k, \tag{1}$$

где $\mathbf{c}_k = \mathbf{c}_k(\mathbf{x})$ – вектора малой размерности, которые мы будем называть *векторами-вложениями*. Матрица V_k одинакова для всех $\mathbf{x}$.

Само линейное представление не снижает вычислительную сложность нейронной сети, потому что за каждой линейной операцией следует поэлементная нелинейная функция активации. Однако мы предлагаем способ приближенного пересчета векторов-вложений, при котором каждый следующий вектор малой размерности вычисляется на основании предыдущего малоразмерного вектора и фиксированных матриц. Данный метод мы назвали методом построения сетей меньшего порядка (Reduced-Order Network, или RON). В предположении о малоранговости наш метод может приблизить большинство сверточных нейронных сетей полносвязной сетью с существенно меньшим числом параметров и вычислительной сложностью. (Подразумеваем сети, состоящие из сверток, полносвязных слоев, неубывающих функций активации, нормализаций, максимум-пулингов, а также “остаточные” сверточные слои с непоследовательной связью слоев (блоки ResNet).)

Другими словами, вместо работы с большими по значению выходами слоев мы предлагаем проецировать вход всей нейронной сети в пространство малой размерности и работать с низкопараметрическим представлением во всех слоях. Выход последнего слоя нейронной сети переводится из малоразмерного пространства обратно в исходное с помощью линейного преобразования. В результате такого подхода вычислительная сложность всей глубокой нейронной сети существенно снижается.

На практике, даже если *предположение о малоранговости* выполняется не на всех слоях или только приближенно, получается восстановить точность модели с помощью нескольких итераций дообучения.

Эмпирически показываем, что наш алгоритм может быть эффективно использован как дополнение к методу ускорения нейронных сетей на основе прунинга каналов.

Основные результаты настоящей работы следующие.

• Предложен новый метод ускорения глубоких нейронных сетей, основанный на малоранговой аппроксимации и не требующий повторного обучения модели.

• Показано, как эффективно применять метод поиска прямоугольной матрицы максимального объема (rectangular maximum volume) для уменьшения размерности слоев, оценена ошибка такого приближения.

• Предложенный метод экспериментально проверен на серии вычислительных экспериментов по ускорению предобученных глубоких сверточных нейронных сетей (VGG and ResNet) на задаче классификации изображений для наборов данных CIFAR10, CIFAR100 и SVHN и полносвязных сетей LeNet на MNIST. В ряде случаев удалось существенно ускорить модель без потери качества.

• Продемонстрировано, что метод эффективен для ускорения сетей после процедуры прунинга, что позволило существенно ускорить уже ускоренную модель.

2. ОБЗОР ЛИТЕРАТУРЫ

В последние годы было предложено множество способов ускорить исполнение сверточных нейронных сетей (CNNs) (см. [25]). В этом разделе мы рассмотрим основные идеи различных семейств методов и выделим различия между ними и нашим подходом.

Многие различные методы имеют дело с предварительно обученной сетью, которую мы называем *сеть-Учитель*, и ускоренной сетью, называемую *сеть-Ученик*. Эта терминология взята из методов *дистилляции знаний* (knowledge distillation) (см. [26]–[29]), в которых выходы после softmax-слоя сети-Учителя используются как целевые метки для сети-Ученика.

В п. 5.4 мы сравнили наш подход с различными алгоритмами *прореживания каналов* (channel pruning). Эти методы нацелены на удаление избыточных каналов в весах различных слоев нейронной сети, тем самым ускоряя и сжимая ее. Каналы выбираются исходя из специального ин-

формационного критерия. Например, этим критерием может быть сумма абсолютных значений весов (см. [30]) или средняя доля нулей (см. [31]).

Есть два доминирующих подхода в прунинге.

Первый имеет дело с отдельно взятой сетью, которая тренируется с нуля с добавлением регуляризатора, усиливающего разреженность весов. Затем некоторые каналы признаются избыточными и удаляются (см. [18], [32]). Обычно это итерационный, вычислительно сложный процесс, особенно в случае глубоких сетей.

Второй подход задействует сеть-Учителя и сеть-Ученика. Ученик обучается минимизировать ошибку воспроизведения промежуточных выходов слоев сети-Учителя (см. [23], [31], [22]).

В [23] выбор каналов осуществляется с помощью LASSO регрессии, а реконструкция сети происходит в смысле среднеквадратичного отклонения. В работе ThiNet [22] стратегия выбора каналов зависит от статистики на следующих слоях.

В [18] было предложено умножать каждый канал на уникальное значение, получаемое в процессе обучения сети. Такой подход позволил произвести разреженную регуляризацию с помощью данных скалярных параметров. Прунинг может быть объединен с процессом поиска архитектуры, как в [13], где стратегия прунинга определяется с помощью LSTM сети, обучаемой алгоритмами обучения с подкреплением.

Наконец, в статье Discrimination-aware Channel Pruning (DCP) [12], с которой мы сравниваем свой алгоритм, к предобученной сети применена многоступенчатая схема прореживания. На каждом шаге алгоритма DCP сеть с предыдущего шага тренируется со специальным классификатором и дискриминационной функцией потерь (discriminative loss). Наименее информативные каналы либо сокращаются с фиксированной скоростью (DCP метод), либо выбираются с использованием жадного алгоритма (DCP-Adapt метод).

Другое семейство подходов к ускорению моделей – *малоранговые методы*, использующие матричное или тензорное разложение для выбора информативных параметров. В большом числе случаев существенное уменьшение вычислительной сложности достигается путем замены одного сверточного слоя на набор меньших сверточных слоев, что показано в [33]–[36], [20]. Отметим, что в большинстве малоранговых методов тензорные разложения применяются к весовым тензорам, а не к выходам слоев (см. [21]).

Наконец, стоит упомянуть методы снижения вычислительной стоимости модели путем дискретизации значений или *квантизации* (quantization) (см. [37], [38]). Такие методы могут значительно ускорить сети, но они обычно требуют специального оборудования для достижения теоретического ускорения на практике.

3. ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ И ОПРЕДЕЛЕНИЯ

Рассмотрим алгоритм выбора прямоугольной матрицы максимального объема (rectangular maximum volume) в п. 3.1 и метод нахождения подпространства вложений (embedding) малой размерности в п. 3.2. Обе эти операции играют ключевую роль в нашем методе.

3.1. Алгоритм поиска подматрицы максимального объема (MaxVol) и скетч-матрица

Прямоугольный алгоритм MaxVol – жадный алгоритм, который ищет в исходной матрице подматрицу из ее строк, имеющую максимальный объем. Объем для матрицы A определяется следующим образом:

$$\mathrm{vol}(A) = \det(A^{\top}A). \tag{2}$$

MaxVol имеет ряд практических применений (см. [7], [8]). В данной работе мы используем его для снижения размерности переопределенных систем.

Положим, $A \in \mathbb{R}^{D\times R}$ – матрица, где число строк намного превышает число столбцов (высокая матрица, $D \gg R$). Нам требуется решить следующую систему линейных уравнений:

$$A\mathbf{x} = \mathbf{b} \tag{3}$$

при фиксированной матрице A для любой правой части $\mathbf{b} \in \mathbb{R}^{D}$. Решением такой системы является

$$\mathbf{x} = A^{\dagger}\mathbf{b}, \tag{4}$$

где $A^\dagger = (A^\top A)^{-1}A^\top$ – псевдообращение Мура–Пенроуза матрицы A. Существует проблема, связанная с тем, что произведение матрицы на вектор с матрицей $A^\dagger$ размера $R \times D$ вычислительно сложно. Кроме того, для плохо обусловленных матриц поиск решения является нестабильным.

Вместо использования всех D уравнений мы выберем наиболее "репрезентативные". Для этого применим упомянутый алгоритм поиска подматрицы максимального объема (https://bitbucket.org/muxas/maxvolpy) для матрицы A. Результатом работы алгоритма является P индексов строк ($R \le P \ll D$), которые соответствуют избранным уравнениям в данной системе. Они используются для дальнейших вычислений. В нашей работе мы предполагаем, что значение параметра P лежит в интервале $[R, 2R]$.

Подматрица из P строк может быть представлена в виде матричного произведения SA, где $S \in \{0,1\}^{P\times D}$. Назовем *$S$ матрицей выбора строк*. Для удобства будем считать, что алгоритм поиска прямоугольной матрицы максимального объема возвращает матрицу выбора строк S. Тогда решением исходной системы уравнений (3) будет

$$\mathbf{x} = (SA)^\dagger(S\mathbf{b}). \tag{5}$$

Взятие строк по индексам в $\mathbf{b}$ – простая операция, поэтому итоговая стоимость подсчета $S\mathbf{b}$ равна $O(P)$. Если $(SA)^\dagger$ предварительно посчитана, то для любой новой правой части требуется посчитать лишь одно произведение матрицы на вектор с матрицей размера $R \times P$.

3.2. Вычисление вложений малой размерности

Пусть $Z \in \mathbb{R}^{N\times D}$ – выходная матрица для заданного слоя нейронной сети. Каждая строка этой матрицы соответствует одному элементу обучающей выборки, прошедшему через часть нейронной сети от начала и до рассматриваемого слоя включительно. Усеченное сингулярное разложение ранга R для $Z^\top \in \mathbb{R}^{D\times N}$ может быть посчитано следующим образом:

$$Z^\top \cong \underbrace{V}_{D\times R}\ \underbrace{\Sigma\mathbb{U}^\top}_{R\times N}. \tag{6}$$

Здесь матрица V соответствует линейному отображению вектора в подпространство вложений малой размерности.

4. ОСНОВНОЙ МЕТОД

Цель метода – построить аппроксимацию исходной нейронной сети-Учителя более быстрой нейронной сетью-Учеником.

Мы детально описываем метод на примере многослойной полносвязной сети в п. 4.1. Приложение к сверточным нейронным сетям (CNN) рассмотрено в п. 4.2, способ применения к "остаточным" сверточным нейронным сетям (семейство сетей ResNet) – в п. 4.3.

4.1. Многослойная полносвязная сеть

Рассмотрим работу алгоритма на примере многослойной полносвязной нейронной сети, называемой также многослойным персептроном (MLP).

Обозначим через ψ_k ($k = 1, 2, \ldots, K$) множество неубывающих поэлементных функций активаций. В это множество входят такие известные функции, как ReLU, ELU, Leaky ReLU и т.д. Для простоты изложения полагаем, что мы хотим ускорить всю сеть-Учителя, однако хотим подчеркнуть, что наш метод применим и для ускорения части исходной нейронной сети. Также без ограничения общности предположим, что свободные коэффициенты всех линейных слоев равны нулю.

Пусть $\mathbf{z}_0$ – входной тензор нейронной сети, который должен пройти через K слоев сети-Учителя. Происходят следующие операции:

$$\mathbf{z}_1 = \psi_1(W_1\mathbf{z}_0), \quad \mathbf{z}_2 = \psi_2(W_2\mathbf{z}_1), \quad \ldots, \quad \mathbf{z}_K = W_K\mathbf{z}_{K-1}, \tag{7}$$

где $W_k \in \mathbb{R}^{D_k\times D_{k-1}}$ – это матрица весов для k-го слоя.

Обозначим через $\mathbf{c}_1, \ldots, \mathbf{c}_K$ векторы-вложения, соответствующие промежуточным признакам (выходам скрытых слоев нейронной сети) $\mathbf{z}_1, \ldots, \mathbf{z}_K$. Обозначим линейное отображение векторов $\mathbf{z}_k$ в векторы $\mathbf{c}_k$ через $V_k \in \mathbb{R}^{D_k \times R_K}$. Это линейное отображение получается с помощью сингулярного разложения (SVD) и известно заранее. Важно отметить, что размерность k-го вложения R_K намного меньше размерности исходного тензора признаков D_k.

Предположение о малоранговости первого слоя приводит нас к следующему выражению:

$$\mathbf{z}_1 \cong \boxed{V_1\mathbf{c}_1 \cong \psi_1(W_1\mathbf{z}_0)}. \tag{8}$$

Выделенное выражение – это переопределенная линейная система с матрицами $V_1 \in \mathbb{R}^{D_1 \times R_1}$, вектором $\psi_1(W_1\mathbf{z}_0)$ и вектором неизвестных $\mathbf{c}_1$. Если $S_1 \in \mathbb{R}^{P_1 \times D_1}$ является матрицей выбора строк (см. п. 3.1) для матрицы V_1, то мы можем выписать выражение для вектора-вложения $\mathbf{c}_1$:

$$\mathbf{c}_1 \cong (S_1V_1)^\dagger S_1\psi_1(W_1z_0) = \underbrace{(S_1V_1)^\dagger}_{R_1 \times P_1}\psi_1(\underbrace{S_1W_1}_{P_1 \times D_1}\mathbf{z}_0). \tag{9}$$

Мы переставили матрицу выбора строк S_1 и поэлементную активационную функцию ψ, потому что их перестановка не меняет выражение.

Используя описанный выше подход, в том же предположении о малоранговости запишем выражение для подсчета $\mathbf{c}_2$ через вложение $\mathbf{c}_1$:

$$\mathbf{z}_2 \cong \psi_2(W_2\mathbf{z}_1) \cong \psi_2(W_2V_1\mathbf{c}_1) \cong V_2\mathbf{c}_2. \tag{10}$$

Получим линейную систему

$$\boxed{V_2\mathbf{c}_2 \cong \psi_2(W_2V_1\mathbf{c}_1)}. \tag{11}$$

Теперь применим прямоугольный алгоритм MaxVol. Если $S_2 \in \mathbb{R}^{P_2 \times D_2}$ – матрица выбора строк для матрицы V_2, то вектор $\mathbf{c}_2$ вычисляется следующим образом:

$$\mathbf{c}_2 \cong \underbrace{(S_2V_2)^\dagger}_{R_2 \times P_2}\psi_2(\underbrace{S_2W_2V_1}_{P_2 \times R_1}\mathbf{c}_1). \tag{12}$$

Продолжая процесс для всех последующих слоев, получим выражение для выхода нейронной сети-Ученика, а именно, $\mathbf{z}_k \cong V_K\mathbf{c}_K$:

$$\begin{gathered}
\mathbf{c}_1 \cong \underbrace{(S_1V_1)^\dagger}_{R_1 \times P_1}\psi_1(\underbrace{S_1W_1}_{P_1 \times D_1}\mathbf{z}_0), \\
\ldots \\
\mathbf{c}_k \cong \underbrace{(S_kV_k)^\dagger}_{R_K \times P_k}\psi_2(\underbrace{S_kW_kV_{k-1}}_{P_k \times R_{k-1}}c_{k-1}), \quad k = 1, 2, \ldots, K, \\
\mathbf{z}_K \cong V_K\mathbf{c}_K.
\end{gathered} \tag{13}$$

Предположим, что s_k – выход функции ψ_k. Тогда система уравнений (13) приобретет вид

$$\begin{gathered}
\mathbf{s}_1 \cong \psi_1(\underbrace{S_1W_1}_{P_1 \times D_1}\mathbf{z}_0), \\
\mathbf{s}_2 \cong \psi_2(\underbrace{S_2W_2V_1(S_1V_1)^\dagger}_{P_2 \times P_1}\mathbf{s}_1), \\
\ldots \\
\mathbf{s}_K \cong \psi_K(\underbrace{S_KW_KV_{K-1}(S_{K-1}V_{K-1})^\dagger}_{P_K \times P_{K-1}}\mathbf{s}_{K-1}), \\
\mathbf{z}_K \cong \underbrace{V_K(S_KV_K)^\dagger}_{D_k \times R_K}\mathbf{s}_K.
\end{gathered} \tag{14}$$

В результате вместо сети с K слоями размера $D_k \times D_{k+1}$ (см. (7)) мы получим намного более компактную сеть из $K+1$ слоя (см. (14)).

Наш подход можно представить в виде алгоритма (см. алгоритм 1):

Algorithm 1: Инициализация сети-Ученика

Input: Веса слоев сети-Учителя $\{W_1, \ldots, W_K\}$; список поэлементных функций активации $\{\psi_1, \ldots, \psi_K\}$; подвыборка из тренировочной выборки Z, размер которой есть (число элементов подвыборки) $\times$ (размер входного вектора); $\{R_1, \ldots, R_K\}$ – размеры вложений;

Output: Веса сети-Ученика $\{\widetilde{W}_0, \widetilde{W}_1, \ldots, \widetilde{W}_K\}$;

▷ Для упрощения реализации алгоритма мы храним все веса $\{V_k\}_{k=1}^K$, но на самом деле достаточно хранить лишь два.

for $k \leftarrow 1$ **to** K **do**

  $Z \leftarrow$ проход данных Z через k-й слой

  $\mathbb{U}, \Sigma, V_k \leftarrow \text{truncated\_svd}(Z^\top, R_K)$

  ▷ На практике мы не храним всю Z, и используем потоковый рандомизированный алгоритм сингулярного разложения.

  $S_k \leftarrow \text{rect\_max\_vol}(V_k)$

end

$\widetilde{W}_0 \leftarrow S_1 W_1$

for $k \leftarrow 1$ **to** $K-1$ **do**

  $\widetilde{W}_k \leftarrow S_k W_k V_{k-1} (S_{k-1} V_{k-1})^\dagger$

end

$\widetilde{W} \leftarrow V_K (S_K V_K)^\dagger$

return $\{\widetilde{W}_0, \widetilde{W}_1, \ldots, \widetilde{W}_K\}$

4.2. Сверточные нейронные сети

Свертка – линейное преобразование. Мы рассматриваем его как произведение матрицы на вектор и преобразуем сверточные слои в полносвязные. Обсудим два важных момента.

Сначала мы векторизуем все выходные данные. Теряем ли мы геометрическую структуру карты объектов? Только частично, потому что структурная информация также учтена в исходной весовой матрице.

Второй момент касается того, что число переметров в сверточной матрице больше, чем в соответствующем ей ядре. Однако эти числа сопоставимы после сжатия, если количество каналов в сверточных слоях не достаточно велико. В результате сеть-Ученик может быть не только быстрее, но и меньше, чем сеть-Учитель.

Пакетная нормализация (Batch normalization), как частно используемый слой нейронной сети, может быть объединен с полносвязным слоем. Таким образом, в сети-Студенте мы избавляемся от слоев пакетной нормализации, но сохраняем свойство нормализации.

Операция подвыборки (Maximum pooling), применяемая к выходам скрытых слоев, является локальной операцией, которая обычно отображает участок размера 2×2 в одно значение: максимальное значение в данной области. Наш метод позволяет сжимать сети с таким слоем: в процессе сэмплирования берется в 4 раза больше индексов и после применяется операция подвыборки (Maximum pooling).

4.3. "Остаточные" сверточные нейронные сети (Residual Networks)

Стандартные архитектуры с последовательными сверточными слоями (например, VGG) проигрывают в эффективности более современным архитектурам (см. [9]–[11]). Такие модели со-

держат несколько параллельных ветвей, выходы которых суммируются перед тем, как попасть на вход функции активации.

Аппроксимируем выход каждой ветви и весь результат следующим образом:

$$V\mathbf{c} \cong \psi(V_1\mathbf{c}_1 + \ldots + V_k\mathbf{c}_k). \tag{15}$$

Выражение (15) – переопределенная система линейных уравнений. Если S – матрица выбора строк для матрицы V, то вложение $\mathbf{c}$ вычисляется как

$$\mathbf{c} \cong (SV)^{\dagger}\psi(SV_1\mathbf{c}_1 + \ldots + SV_k\mathbf{c}_k). \tag{16}$$

Остальные шаги для "остаточной" сети вычисляются так же, как и для многослойного персептрона (см. п. 4.1).

4.4. Ошибка аппроксимации

Положим $\varepsilon_k = V_k\mathbf{c}_k - z_k$ – ошибка малоранговой аппроксимации, тогда

$$S_kV_k\mathbf{c}_k = (S_kV_k)^{\dagger}S_kz_k + (S_kV_k)^{\dagger}S_k\varepsilon_k \tag{17}$$

и ошибка нашего алгоритма равна $e_k := \left\|(S_kV_k)^{\dagger}S_k\varepsilon_k\right\|_2$. Учитывая информацию о нормированности $\left\|V_k^{\top}\right\|_2 = \|S_k\|_2 = 1$, получаем

$$\left\|(S_kV_k)^{\dagger}S_k\right\|_2 = \left\|V_k^{\top}V_k(S_kV_k)^{\dagger}S_k\right\|_2 \le \left\|V_k(S_kV_k)^{\dagger}\right\|_2. \tag{18}$$

Принимая во внимание лемму 4.3 и замечание 4.4 из статьи про прямоугольный алгоритм MaxVol [8], имеем

$$\left\|V_k(S_kV_k)^{\dagger}\right\|_2 \le \sqrt{1 + \frac{(D_k - P_K)r_k}{P_K + 1 - R_K}}. \tag{19}$$

(В этой статье исходная матрица обозначается через C.) Следовательно,

$$e_k \le \sqrt{1 + \frac{(D_k - P_K)R_K}{P_K + 1 - R_K}}\left\|\varepsilon_k\right\|_2. \tag{20}$$

Например, если $P_K = 1.5R_K$, то ошибка приближения e_k равна $O(\sqrt{D_k}\left\|\varepsilon_k\right\|_2)$ для $R_K = o(D_k)$.

5. ЭКСПЕРИМЕНТЫ

Представим результаты вычислительных экспериментов. Сначала приводится эмпирическое подтверждение предположения о малоранговости выходов скрытых слоев нейронной сети. Затем демонстрируется качество работы нашего алгоритма RON для ускорения полносвязных и сверточных нейронных сетей. В завершение проводится сравнение с существующими аналогами, результаты которых приведены в DCP [12] и [13].

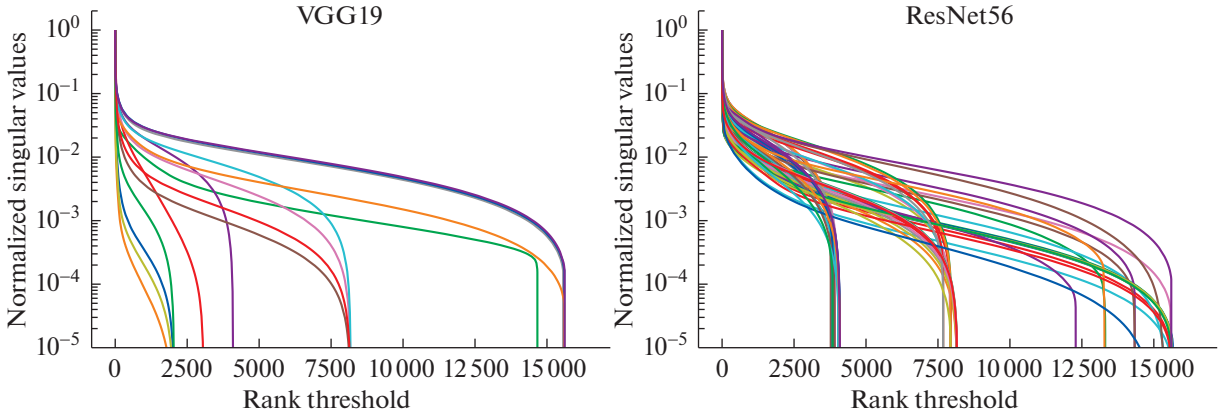
Наборы данных. Демонстрируем эффективность работы нашего алгоритма на четырех наборах данных: MNIST, CIFAR-10, CIFAR-100 и SVHN.

• MNIST – коллекция написанных от руки цифр, состоящая из 70000 картинок размера 28×28, включая 60 тысяч картинок обучающей выборки и 10 тысяч тестовой.

• CIFAR-10 – набор данных из 50000 обучающих и 10000 тестовых цветных картинок размера 32×32, на которых изображены объекты одного из 10 классов.

• CIFAR-100 содержит 100 классов, в каждом из которых по 500 обучающих и 100 тестовых изображений, параметры которых схожи с CIFAR-10.

• SVHN – датасет из фотографий с номерами уличных домов, содержащий 73257 обучающих и 26032 тестовых картинок размера 32×32.



Фиг. 1. Сингулярные числа для всех слоев для VGG19 и ResNet56, обученных на CIFAR-10. Значения нормированы на наибольшее сингулярное число для слоя. Видно, что большая их часть мала в относительном сравнении.

5.1. Сингулярные числа

Наш метод полагается на предположение о возможности эффективно отобразить выходы слоев в пространство малой размерности. Мы экспериментально проверяем это предположение для избранных архитектур на задаче классификации изображений. На фиг. 1 приведены графики для двух архитектур нейронной сети: VGG19 и ResNet56. На каждом из графиков изображены сингулярные значения промежуточных тензоров (выхода группы слоев или блока в случае ResNet). Заметим, что для части блоков сингулярные числа быстро убывают. Это означает, что их выходы могут быть хорошо приближены малоранговой аппроксимацией.

Для выбора ранга использовали два подхода: непараметрический алгоритм вариационной байесовской матричной факторизации (Variational Bayesian Matrix Factorization или VBMF из [14]) и выбор постоянного коэффициента снижения ранга.

Сингулярные числа вычисляются для матриц, содержащих данные из всей обучающей выборки. Мы использовали потоковой рандомизированный алгоритм вычисления сингулярного разложения (SVD) (см. [15], [16]), что позволило не хранить всю матрицу в памяти.

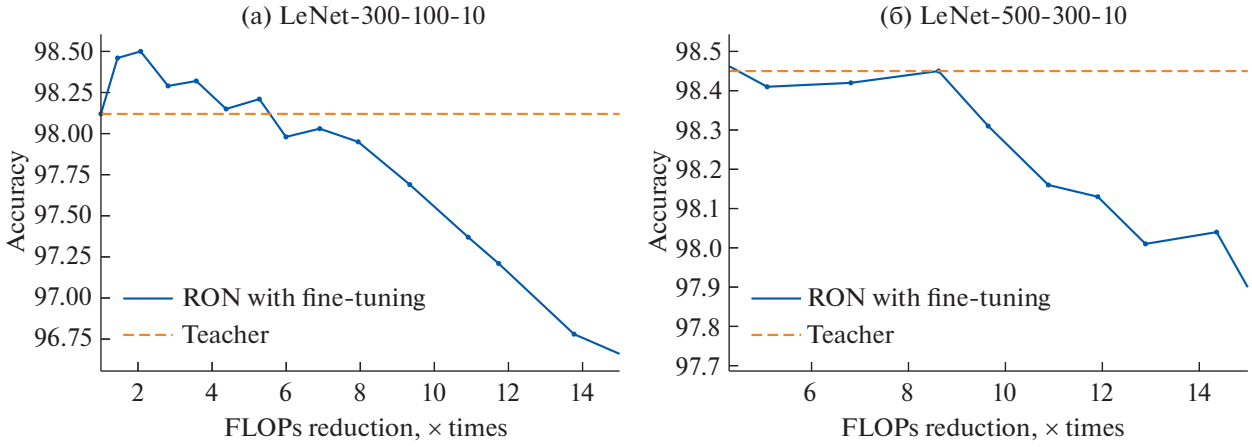
5.2. Ускорение полносвязных сетей

Для демонстрации работы метода на полносвязных сетях выбраны архитектуры LeNet-300-100 и LeNet-500-100 для классификации изображений из набора данных MNIST. LeNet-300-100 состоит из трех полносвязных слоев с матрицами размера 784×300, 300×100, 100×10 и функций активаций ReLU. В LeNet-500-100 на скрытых слоях содержится, соответственно, 500 и 100 нейронов. Осуществлено 15 итераций следующей процедуры. В начале мы обучили модель с шагом градиентного спуска `1e-3` в течение 25 эпох. Затем дообучили с меньшим шагом градиентного спуска (`5e-4`) также в течение 25 эпох. Затем применили наш метод (RON) с фиксированным уровнем снижения ранга: 0.7 и 0.75 соответственно. На фиг. 2 видно, что точность сетей убывает при более сильном ускорении.

В [13] модель LeNet-500-100 ускорена в ~7.85 раза без потери качества. Метод RON позволяет получить ускорение более чем в $\times 8$ раз (фиг. 2б).

5.3. Ускорение сверточных сетей

Мы применили наш метод к сверточным сетям архитектуры, схожей с VGG (см. [17]), для классификации на CIFAR-10, CIFAR-100 и SVHN. Во всех экспериментах лишь один раз использовали алгоритм RON, а затем дообучали получившуюся сеть, если это было необходимо. В процессе инициализации сети-Ученика (см. алгоритм 1) размеры вложений для CIFAR-10 выбирались с помощью VBMF, а для CIFAR-100 и SVHN – с помощью наперед



Фиг. 2. Метод RON для различных моделей LeNet.

заданного коэффициента сжатия. Мы использовали предобученные нейронные сети для максимальной чистоты эксперимента: для CIFAR-10 из репозитория Model-Zoo (https://github.com/SCUT-AILab/DCP/wiki/Model-Zoo). Данный репозиторий также содержит модель VGG-19 и ее прореженную методом DCP (см. [12]) версию. Для экспериментов на CIFAR-100 и SVHN использовались

- VGG-19 для CIFAR-100, (https://github.com/bearpaw/pytorch-classification)
- VGG-7 для SVHN. (https://github.com/aaron-xichen/pytorch-playground)

Затем мы применили RON к моделям вида VGG, ускорив несколько последних слоев. Например, в модели RON (8 to 16) были ускорены девять слоев с 8-го по 16 включительно.

RON без дообучения. Инициализируем сеть-Ученика способом, показанным в алгоритме 1, а затем измеряем достигнутое ускорение и качество полученной модели. Для VGG-19 на CIFAR-10 с помощью RON удалось построить модель, требующую в 1.53× меньшее число операций, чем исходная при улучшении качества на 0.09% без какого-либо дообучения (см. табл. 1). Мы сравнили применение нашего алгоритма к ускорению предобученной сети VGG-19 с алгоритмом из [13], чтобы наглядно показать, что без дообучения RON намного превосходит прунинг каналов до дообучения (см. [18], [13]) (см. модель с меткой “w/o fine-tuning”). Сверточные нейронные сети, которые были подвергнуты прунингу с параметром `pruning rate`, равным 0.1% (число FLOP в 1.23× меньше чем у исходной сети), демонстрируют падение качества более чем на 20% (см. фиг. 5 в [13]).

RON с дообучением. Если после ускорения модели алгоритмом RON провести ее дообучение, то зачастую можно добиться лучших показателей – качество восстановится. Так модель, ускоренная в 2.3×, показала качество на 0.28% лучше исходной сети для VGG-19 на CIFAR-10 (табл. 1). Процесс дообучения состоял из 250 эпох стохастического градиентного спуска с моментом 0.9 и размером батча 256. Шаг градиентного спуска изначально равнялся `1e-2` и сокращался в 2 раза каждые 10 эпох. При дообучении использовался дропаут (dropout).

Архитектуры VGG на CIFAR-100 (табл. 2) и SVHN (табл. 3) менее избыточны, их ускорение без потери качества меньше, чем для CIFAR-10.

Отметим, что шаги ускорения и дообучения могут быть применены пошагово, с постепенным увеличением числа сжатых слоев или степени их сжатия. Такой подход более вычислительно емкий, однако и для задачи прунинга каналов (см. [18], [12], [19], [13]) и для малоранговых методов (см. [20]) позволяет уменьшить падение качества при сжатии.

Применение RON после прунинга. Мотивация использования малорангового метода после процедуры прунинга заключается в следующем. Прунинг убирает наименее информативные каналы сверточного слоя, тем самым уменьшая и ускоряя сеть (например, в DCP из [12]). Однако сверточная сеть, составленная из самых информативных слоев (после прунинга), все еще может иметь малоранговую структуру и, значит, может быть ускорена методом RON. При применении

Таблица 1. Точность и теоретическое ускорение (уменьшение числа FLOP) для моделей, ускоренных методом RON на наборе данных CIFAR-10

Модель	Измененные слои	Acc@1 без дообучения	Acc@1 с дообучением	Сокращение числа FLOP
сеть-Учитель	–	–	93.70	1.00×
RON	10 to 16	**93.79**	**94.10**	**1.53**×
RON	9 to 16	93.46	**94.15**	**1.68**×
RON	8 to 16	90.58	**94.24**	**1.93**×
RON	7 to 16	85.79	**93.98**	**2.30**×
RON	6 to 16	72.53	93.12	3.01×
RON	5 to 16	58.12	91.88	3.66×
DCP [12]	–	–	93.96	2.00×
DCP + RON	10 to 16	**93.98**	**94.24**	**3.06**×
DCP + RON	9 to 16	**93.90**	**94.27**	**3.37**×
DCP + RON	8 to 16	91.82	**94.01**	**3.78**×
DCP + RON	7 to 16	88.88	**93.97**	**4.48**×
DCP + RON	6 to 16	81.30	93.26	5.56×
DCP + RON	5 to 16	64.12	91.5	7.21×

Примечание. DCP – метод прунинга из [12].

Таблица 2. VGG на CIFAR-100

Модель	Измененные слои	Acc@1 без дообуч.	Acc@5 без дообуч.	Acc@1 с дообуч.	Acc@5 с дообуч.	Ускорение на CPU	Сокращение числа FLOP
сеть-Учитель	–	–	–	71.95	89.41	1.00×	1.00×
RON 10×	8 to 16	70.81	88.51	**72.09**	**90.12**	**1.95**×	**1.66**×
RON 20×	8 to 16	63.94	85.12	71.89	89.95	2.15×	1.71×
RON 10×	10 to 16	60.68	82.36	70.87	90.46	1.72×	1.84×
RON 20×	10 to 16	44.07	68.29	69.69	89.78	2.19×	2.19×
RON 10×	12 to 16	42.77	67.34	66.84	88.16	2.22×	2.58×

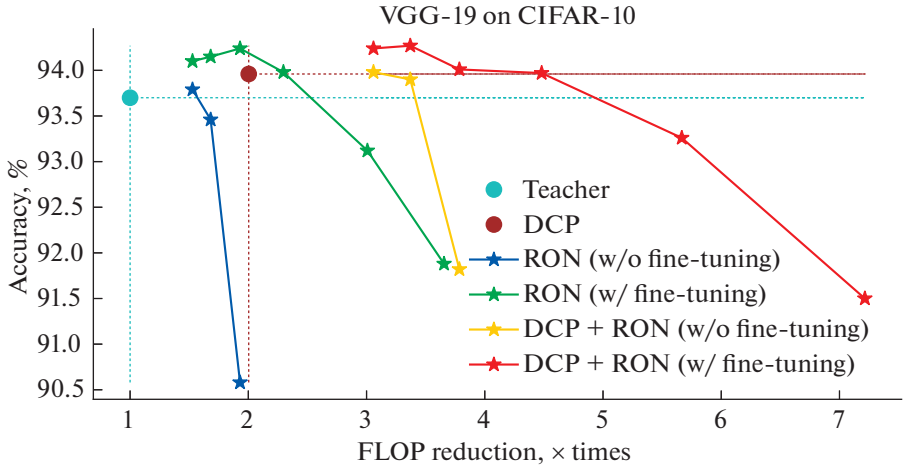
Примечание. RON $N\times$ означает ускоренную модель, у которой размерность последних слоев понижена в $N\times$ относительно исходной.

Таблица 3. VGG на SVHN

Модель	Измененные слои	Acc@1 без дообучения	Acc@1 с дообучением	Ускорение на CPU	Сокращение числа FLOP
сеть-Учитель	–	–	96.03	1.00×	1.00×
RON 10 ×	5 to 7	92.46	95.41	1.62×	1.30×
RON 20 ×	5 to 7	89.04	95.33	1.71×	1.53×
RON 20 ×	3 to 7	83.58	92.13	1.67×	1.65×

Примечание. RON $N\times$ означает ускоренную модель, у которой размерность последних слоев понижена в $N\times$ относительно исходной.

метода RON на модели VGG-19, уже прореженной методом DCP (см. [12]), при параметре `pruning rate` 0.3%, получилось добиться суммарного уменьшения числа операций в 4.48× раза при улучшении качества на 0.27% по сравнению с исходной сетью VGG-19 (фиг. 3).



Фиг. 3. Сравнение точности и теоретического ускорения (уменьшения числа FLOP) моделей, полученных методами DCP/RON/DCP + RON на наборе данных CIFAR-10. Если модель после применения RON дообучалась в течение нескольких эпох, то она имеет в скобках указание (w/ fine-tuning), в противном случае – (w/o fine-tuning).

5.4. Сравнение с другими методами

Достоинство нашего метода заключается в том, что его можно применять поверх других методов прореживания (channel pruning). Для демонстрации этого качества мы использовали уже прореженную нейронную сеть архитектуры VGG из [12] и ускорили ее. Полученное ускорение без потери качества составило 1.68× по сравнению с уже прореженной сетью или 3.36× по сравнению с исходной моделью.

Мы свели в одну табл. 4 результаты наших экспериментов и результаты из [12]. Также мы добавили к сравнению результаты из соответствующих работ для ThiNet [22], Channel pruning (CP) [23], Slimming [18] и метода width-multiplier [24].

6. ОБСУЖДЕНИЕ

Мы предложили метод, основанный на предположении о малоранговости матриц выходов слоев нейронной сети. Показали, что в ряде случаев полученная сеть-Ученик существенно быстрее исходной модели при сохранении той же точности даже без дообучения.

Минусом предложенного подхода является возможное увеличение числа параметров в результирующей полносвязной сети, когда исходной является сверточная сеть с широкими слоями. Однако наш подход эффективно работает поверх уже ускоренных методом прунинга моде-

Таблица 4. Сравнение на CIFAR-10

Модель	Сокращение числа FLOP	Падение качества, %
ThiNet [22]	2.00×	0.14
Network Sliming [18]	2.04×	0.19
Channel Pruning [23]	2.00×	0.32
Width-multiplier [24]	2.00×	0.38
Discrimination-aware Channel Pruning (DCP) [12]	2.00×	–0.17
DCP-Adapt [12]	2.86×	–0.58
RON (modified layers: 7 to 16) + fine-tuning	**2.30**×	**–0.18**
DCP + RON (modified layers: 9 to 16) + fine-tuning	**3.37**×	**–0.57**
DCP + RON (modified layers: 7 to 16) + fine-tuning	**4.48**×	**–0.27**

Примечание. VGG-19 (исходное качество 93.7%). Чем большее сокращение числа FLOP достигнуто при меньшем падении точности, тем лучше.

лей и в комбинации с ним может дать меньшее число параметров, чем было в исходной модели. В дальнейшем мы планируем использовать разреживание (sparsification) (см. [39]) и квантизацию (quantization) поверх нашего алгоритма для решения данной проблемы.

7. ЗАКЛЮЧЕНИЕ

Нами разработан метод ускорения нейронных сетей, основанный на проецировании выходов слоев в малопараметрическое подпространство. Основой метода послужили алгоритмы сингулярного разложения и поиска прямоугольной подматрицы максимального объема. Вычислительные эксперименты показали, что наш подход позволяет найти хорошее приближение исходной модели в полученном пространстве параметров новой, более быстрой, нейронной сети. А именно, в задаче классификации изображений для наборов данных CIFAR10 и CIFAR100, полученные нашим методом модели существенно ускоряют исходные и сопоставимы с ними по качеству даже без дополнительного дообучения. С дообучением исходную модель получилось ускорить в 4.48× раза без потери качества. Кроме обширных экспериментальных результатов в работе также приведена теоретическая оценка верхней границы ошибки приближения.

СПИСОК ЛИТЕРАТУРЫ

1. *Chen T.Q., Rubanova Y., Bettencourt J., Duvenaud D.K.* Neural ordinary differential equations // Adv. Neural Informat. Proc. Syst. 2018. P. 6572–6583.
2. *Grathwohl W., Chen R.T., Betterncourt J., Sutskever I., Duvenaud D.* Ffjord: Free-form continuous dynamics for scalable reversible generative models // Proc. Inter Conf. Learn. Represent. 2019.
3. *Gusak J., Markeeva L., Daulbaev T., Katrutsa A., Cichocki A., Oseledets I.* Towards Understanding Normalization in Neural ODEs// Inter. Conf. Learn. Represent. (ICLR) Workshop on Integration of Deep Neural Models and Differential Equations.
https://openreview.net/forum?id=mllQ3QNNr9d. 2020.
4. *Daulbaev T., Katrutsa A., Gusak J., Markeeva L., Cichocki A., Oseledets I.* Interpolation Technique to Speed Up Gradients Propagation in Neural ODEs. arXiv preprint arXiv:2003.05271. 2020.
5. *Quarteroni A., Rozza G., et al.* Reduced order methods for modeling and computational reduction // V. 9. Springer, 2014. № 5. С. 477–512.
6. *Chaturantabut S., Sorensen D.C.* Nonlinear model reduction via discrete empirical interpolation // SIAM J. Sci. Comput. 2010. V. 32. P. 2737–2764.
7. *Fonarev A., Mikhalev A., Serdyukov P., Gusev G., Oseledets I.* Efficient rectangular maximal-volume algorithm for rating elicitation in collaborative filtering // 2016 IEEE 16th Inter. Conf. on Data Mining (ICDM), IEEE. 2016. V. 1. P. 141–150.
8. *Mikhalev A., Oseledets I.V.* Rectangular maximum-volume submatrices and their applications // Linear Algebra and its Appl. 2018. V. 538. P. 187–211.
9. *He K., Zhang X., Ren S., Sun J.* Deep residual learning for image recognition // Proc. of the IEEE Conf. Comput. Vis. Pattern Recognit. 2016. P. 770–778.
10. *Zagoruyko S., Komodakis N.* Wide residual networks // Proc. British Machine Vis. Conf. (BMVC). 2016. P. 87.1–87.12.
11. *Huang G., Liu Z., Van Der Maaten L., Weinberger K.Q.* Densely connected convolutional networks // Proc. IEEE Conf. Comput. Vis. Pattern Recognit. 2017. P. 4700–4708.
12. *Zhuang Z., Tan M., Zhuang B., Liu J., Guo Y., Wu Q., Huang J., Zhu J.* Discrimination-aware channel pruning for deep neural networks // Adv. Neural Inform. Proc. Systems. 2018. V. 31. P. 881–892.
13. *Zhong J., Ding G., Guo Y., Han J., Wang B.* Where to prune: Using lstm to guide end-to-end pruning // Inter. Joint Conf. Artific. Intelligence. 2018. P. 3205–3211.
14. *Nakajima S., Sugiyama M., Babacan S.D., Tomioka R.* Global analytic solution of fully-observed variational bayesian matrix factorization // J. Machine Learn. Res. 2013. V. 14. P. 1–37.
15. *Woodruff D.P.* Sketching as a tool for numerical linear algebra // Foundat. and Trends in Theoret. Comput. Sci. 2014. V. 10. P. 1–157.
16. *Tsitsulin A., Munhkoeva M., Mottin D., Karras P., Oseledets I., Muller E.* Frede: Linear-space anytime graph embeddings // arXiv:2006.04746, 2020, url: https://arxiv.org/abs/2006.04746
17. *Simonyan K., Zisserman A.* Very deep convolutional networks for large-scale image recognition // arXiv:1409.1556, 2014, url: https://arxiv.org/abs/1409.1556
18. *Liu Z., Li J., Shen Z., Huang G., Yan S., Zhang C.* Learning efficient convolutional networks through network slimming // 2017 IEEE Inter. Conf. Comput. Vis. (ICCV). 2017.
19. *Gao X., Zhao Y., Dudzyak L., Mullins R., Xu C.zhong* Dynamic channel pruning: Feature boosting and suppression // Inter. Conf. Learn. Representat. 2019.

20. *Gusak J., Kholiavchenko M., Ponomarev E., Markeeva L., Blagoveschensky P., Cichocki A., Oseledets I.* Automated multi-stage compression of neural networks // IEEE/CVF Inter. Conf. Comput. Vis. Workshop (ICCVW). 2019.
21. *Cui C., Zhang K., Daulbaev T., Gusak J., Oseledets I., Zhang Z.* Active Subspace of Neural Networks: Structural Analysis and Universal Attacks // arXiv preprint arXiv:1910.13025. 2019.
22. *Luo J., Zhang H., Zhou H., Xie C., Wu J., Lin W.* Thinet: Pruning cnn filters for a thinner net // IEEE Transact. Pattern Anal. Mach. Intelligence. 2018. V. 41. Iss. 10. P. 2525–2538.
23. *He Y., Zhang X., Sun J.* Channel pruning for accelerating very deep neural networks // 2017 IEEE Inter. Conf. Comput. Vis. (ICCV). 2017.
24. *Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H.* MobileNets: Efficient Convolutional neural networks for mobile vision applications // arXiv:1704.04861, 2017, url: https://arxiv.org/abs/1704.04861
25. *Cheng Y., Wang D., Zhou P., Zhang T.* Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges // IEEE Signal Proc. Magazine. 2018. V. 35. Iss. 1. P. 126–136.
26. *Bucilua C., Caruana R., Niculescu-Mizil A.* Model Compression // Proc. 12th ACM SIGKDD internat. Conf. on Knowledge Discovery and Data Mining. 2006. P. 535–541.
27. *Hinton G., Vinyals O., Dean J.* Distilling the Knowledge in a Neural Network // NIPS Deep Learn. and Represent. Learn. Workshop. 2015.
28. *Romero A., Ballas N., Kahou S.E., Chassang A., Gatta C., Bengio Y.* FitNets: Hints for Thin Deep Nets // Proc. Internat. Conf. Learn. Represent. 2015.
29. *Zagoruyko S., Komodakis N.* Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer // Proc. Inter. Conf. Learn. Represent. 2017.
30. *Li H., Kadav A., Durdanovic I., Samet H., Graf H.P.* Pruning filters for efficient convnets // Proc. Inter. Conf. Learn. Represent. 2017.
31. *Hu H., Peng R., Tai Y.-W., Tang C.-K.* Network trimming: A data-driven neuron pruning approach towards efficient deep architectures // arXiv:1607.03250, 2016, url: https://arxiv.org/abs/1607.03250
32. *Wen W., Wu C., Wang Y., Chen Y., Li H.* Learning structured sparsity in deep neural networks // Adv. Neural Inform. Proc. Systems. 2016. P. 2074–2082.
33. *Denton E., Zaremba W., Bruna J., LeCun Y., Fergus R.* Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation // Adv. Neural Infor. Proc. Systems. 2014. V. 2. P. 1269–1277.
34. *Jaderberg M., Vedaldi A., Zisserman A.* Speeding up Convolutional Neural Networks with Low Rank Expansions // Proc. British Mach. Vis. Conf. (BMVC). 2014.
35. *Lebedev V., Ganin Y., Rakhuba M., Oseledets I., Lempitsky V.* Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition // Proc. 3rd Inter. Conf. Learn. Represent. 2015.
36. *Zhang X., Zou J., He K., Sun J.* Accelerating Very Deep Convolutional Networks for Classification and Detection // IEEE Transact. Pattern Anal. Mach. Intellig. 2016. V. 38. Iss. 10.
37. *Courbariaux M., Bengio Y., David J.-P.* Training deep neural networks with low precision multiplications // 3rd Inter. Conf. Learn. Represent. 2015.
38. *Gupta S., Agrawal A., Gopalakrishnan K., Narayanan P.* Deep Learning with Limited Numerical Precision // Proc. 32nd Inter. Conf. Inter. Conf. Mach. Learn. V. 37. 2015. P. 1737–1746.
39. *Molchanov D., Ashukha A., Vetrov D.* Variational dropout sparsifies deep neural networks // Proc. 34th Inter. Conf. Mach. Learn. 2017. V. 70. P. 2498–2507.