

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК



ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО
УНИВЕРСИТЕТА

BULLETIN

OF THE SOUTH URAL
STATE UNIVERSITY

СЕРИЯ

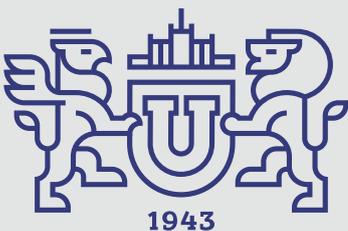
**ВЫЧИСЛИТЕЛЬНАЯ
МАТЕМАТИКА
И ИНФОРМАТИКА**

2023, том 12, № 4

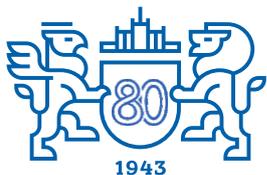
SERIES

**COMPUTATIONAL
MATHEMATICS
AND SOFTWARE ENGINEERING**

2023, volume 12, no. 4



ВЕСТНИК



ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО
УНИВЕРСИТЕТА

2023
Т. 12, № 4

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий,
в которых должны быть опубликованы результаты диссертаций
на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

Редакционная коллегия

Л.Б. Соколинский, д.ф.-м.н., проф., *гл. редактор*
М.Л. Цымблер, д.ф.-м.н., доц., *зам. гл. редактора*
Я.А. Краева, *отв. секретарь*
А.И. Гоглачев, *техн. редактор*

Редакционный совет

С.М. Абдуллаев, д.г.н., профессор
А. Андреев, PhD, профессор (Германия)
В.И. Бердышев, д.ф.-м.н., акад. РАН, *председатель*
В.В. Воеводин, д.ф.-м.н., чл.-кор. РАН
Дж. Донгарра, PhD, профессор (США)

С.В. Зыкин, д.т.н., профессор
И.М. Куликов, д.ф.-м.н.
Д. Маллманн, PhD, профессор (Германия)
А.В. Панюков, д.ф.-м.н., профессор
Р. Продан, PhD, профессор (Австрия)
Г.И. Радченко, к.ф.-м.н., доцент (Австрия)
В.П. Танана, д.ф.-м.н., профессор
В.И. Ухоботов, д.ф.-м.н., профессор
В.Н. Ушаков, д.ф.-м.н., чл.-кор. РАН
М.Ю. Хачай, д.ф.-м.н., чл.-кор. РАН
А. Черных, PhD, профессор (Мексика)
П. Шумяцкий, PhD, профессор (Бразилия)



BULLETIN

OF THE SOUTH URAL
STATE UNIVERSITY

2023

Vol. 12, no. 4

SERIES

“COMPUTATIONAL
MATHEMATICS AND SOFTWARE
ENGINEERING”

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya “Vychislitel'naya Matematika i Informatika”

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Editorial Board

L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russia)

M.L. Zymbler, South Ural State University (Chelyabinsk, Russia)

Ya.A. Kraeva, South Ural State University (Chelyabinsk, Russia)

A.I. Goglavchev, South Ural State University (Chelyabinsk, Russia)

Editorial Council

S.M. Abdullaev, South Ural State University (Chelyabinsk, Russia)

A. Andrzejak, Heidelberg University (Germany)

V.I. Berdyshev, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)

J. Dongarra, University of Tennessee (USA)

M.Yu. Khachay, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)

I.M. Kulikov, Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of RAS (Novosibirsk, Russia)

D. Mallmann, Julich Supercomputing Centre (Germany)

A.V. Panyukov, South Ural State University (Chelyabinsk, Russia)

R. Prodan, Alpen-Adria-Universität Klagenfurt (Austria)

G.I. Radchenko, Silicon Austria Labs (Graz, Austria)

P. Shumyatsky, University of Brasilia (Brazil)

V.P. Tanana, South Ural State University (Chelyabinsk, Russia)

A. Tchernykh, CICESE Research Center (Mexico)

V.I. Ukhobotov, Chelyabinsk State University (Chelyabinsk, Russia)

V.N. Ushakov, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)

V.V. Voevodin, Lomonosov Moscow State University (Moscow, Russia)

S.V. Zykin, Sobolev Institute of Mathematics, Siberian Branch of the RAS (Omsk, Russia)

Содержание

ОБЗОР ПРИМЕНЕНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ И ПАРАЛЛЕЛЬНЫХ АРХИТЕКТУР В ЗАДАЧАХ ФРАГМЕНТАЦИИ ГОРНЫХ ПОРОД М.В. Ронкин, Е.Н. Акимова, В.Е. Мислов, К.И. Решетников	5
ИССЛЕДОВАНИЕ НЕЙРОСЕТЕВОГО МЕТОДА РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ Н.А. Ольховский	55
МЕТОДЫ УПРАВЛЕНИЯ WORK-STEALING ДЕКАМИ В ДИНАМИЧЕСКИХ ПЛАНИРОВЩИКАХ МНОГОПРОЦЕССОРНЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ Е.А. Аксёнова, А.В. Соколов	76
ВОЗМОЖНОСТИ ПАРАЛЛЕЛИЗМА ПРИ ИДЕНТИФИКАЦИИ КВАЗИЛИНЕЙНОГО РЕКУРРЕНТНОГО УРАВНЕНИЯ М.С.А. Аботалёб, Т.А. Макаровских, А.В. Панюков	94

Contents

REVIEW ON APPLICATION OF DEEP NEURAL NETWORKS AND PARALLEL ARCHITECTURES FOR ROCK FRAGMENTATION PROBLEMS M.V. Ronkin, E.N. Akimova, V.E. Misilov, K.I. Reshetnikov	5
STUDY OF NEURAL NETWORK MODELS FOR LINEAR PROGRAMMING N.A. Olkhovsky	55
CONTROL METHODS OF WORK-STEALING DEQUES IN DYNAMIC SCHEDULERS OF MULTIPROCESSOR PARALLEL COMPUTATIONS E.A. Aksenova, A.V. Sokolov	76
POSSIBILITIES OF PARALLELISM UNDER IDENTIFYING A QUASI-LINEAR RECURRENT EQUATION M.S.A. Abotaleb, T.A. Makarovskikh, A.V. Panyukov	94



This issue is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

ОБЗОР ПРИМЕНЕНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ И ПАРАЛЛЕЛЬНЫХ АРХИТЕКТУР В ЗАДАЧАХ ФРАГМЕНТАЦИИ ГОРНЫХ ПОРОД

© 2023 М.В. Ронкин¹, Е.Н. Акимова^{1,2}, В.Е. Мисилов^{1,2}, К.И. Решетников¹

¹Уральский федеральный университет

(620002 Екатеринбург, ул. Мира, д. 19),

²Институт математики и механики им. Н.Н. Красовского УрО РАН

(620108 Екатеринбург, ул. С. Ковалевской, д. 16)

E-mail: m.v.ronkin@urfu.ru, aen15@yandex.ru, v.e.misilov@urfu.ru, ki.reshetnikov@urfu.ru

Поступила в редакцию: 14.07.2023

Оценка производительности добычи полезных ресурсов, в том числе определение геометрических размеров объектов горной породы в открытом карьере, является одной из наиболее важных задач в горнодобывающей промышленности. Задача фрагментации горных пород решается с помощью методов компьютерного зрения, таких как экзemplярная сегментация или семантическая сегментация. В настоящее время для решения таких задач для цифровых изображений используются нейронные сети глубокого обучения. Нейронные сети требуют больших вычислительных мощностей для обработки цифровых изображений высокого разрешения и больших наборов данных. Для решения этой проблемы в литературе предлагается использование облегченных архитектур нейронных сетей, а также методов оптимизации производительности, таких как параллельные вычисления с помощью центральных, графических и специализированных процессоров. В обзоре рассматриваются последние достижения в области нейронных сетей глубокого обучения для решения задач компьютерного зрения применительно к фрагментации горных пород и вопросы повышения производительности реализаций нейронных сетей на различных параллельных архитектурах.

Ключевые слова: компьютерное зрение, сверточные нейронные сети, глубокое обучение, экзemplярная сегментация, семантическая сегментация, обнаружение объектов, параллельные вычисления, задачи горнодобывающей промышленности, фрагментация горных пород.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Ронкин М.В., Акимова Е.Н., Мисилов В.Е., Решетников К.И. Обзор применения глубоких нейронных сетей и параллельных архитектур в задачах фрагментации горных пород // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 5–54. DOI: 10.14529/cmse230401.

Введение

Цифровизация горнодобывающей промышленности относится к области так называемой Индустрии 4.0. В настоящее время многие горнодобывающие процессы автоматизируются с использованием подходов на основе нейронных сетей (НС) глубокого обучения (Deep learning, DL). Примерами таких процессов являются: бурение, взрывные работы, транспортировка полезных ископаемых и их переработка. Несмотря на широкое использование нейронных сетей в горнодобывающей промышленности данный подход остается не достаточно исследованным по сравнению с другими видами промышленности. Более того, текущее состояние научной области не позволяет полностью автоматизировать процессы и на большинстве этапов обработки горных работ по-прежнему требуется работа оператора.

Предварительный анализ использования DL в горнодобывающей промышленности показывает, что в большинство случаев задачи сводятся к использованию сверточных нейронных сетей (СНС) в системах компьютерного зрения [1]. Примерами такого подхода яв-

ляются следующие задачи: классификация типов руды [2] классификация обогащенных пород [3]; классификация типов минералов [4]; оценка распределения размеров частиц [5]; анализ результатов бурения [6, 7]; анализ спутниковых снимков карьера [8, 9]; исследование карьера [10]; распознавание типов местности [11]; автономное управление транспортными средствами [12]; оценка качества взрывных работ на карьерах [13], и многие другие.

Результаты исследований авторов в области использования компьютерного зрения для оценки фрагментации горных пород демонстрируют перспективность данного подхода. Предложенные методы напрямую связаны с задачей определения асбестовых прожилок на кусках породы при обработке в условиях конвейерной ленты [14] и в условиях открытого карьера [15]. Проблему оценки фрагментации горных пород можно рассматривать как самостоятельную задачу или как задачу, связанную с оценкой производительности карьера [16].

В настоящее время оценка производительности карьера обычно проводится либо визуально, либо в лаборатории. Специалисты геологической службы могут проводить оперативный визуальный контроль практически в режиме реального времени, но с относительно низкой точностью. Стационарный лабораторный контроль обеспечивает высокую точность, но является трудоемким [17]. В результате лабораторный контроль может дать только среднюю оценку по всем рабочим площадкам.

С другой стороны, оценки, сделанные геологической службой (путем визуального анализа), могут быть очень субъективными. Как правило, специалисты геологической службы не могут формально описать свои алгоритмы или критерии, которые они используют для оценки производительности карьера. Более того, оценки, сделанные разными специалистами, могут существенно отличаться между собой [15]. В научной литературе обсуждаются методы автоматизации для решения таких специфических задач с использованием различных систем компьютерного зрения.

Специфику использования компьютерного зрения можно объяснить следующим образом. Типичные проблемы компьютерного зрения требуют большого количества извлекаемых признаков, которые не могут быть формально описаны. Глубокое обучение позволяет автоматически извлекать полезное признаковое описание даже из необработанных наборов данных. С другой стороны, эта задача требует довольно больших баз данных и вычислительных ресурсов, что приводит к соответствующим проблемам. Наиболее критичными проблемами являются требования к производительности вычислительных систем и требования к пространству для хранения моделей и данных [18].

Анализ литературы в области использования нейронных сетей в задачах фрагментации горных пород показывает, что в большинстве случаев используются только базовые сверточные архитектуры нейросетей без какой-либо оптимизации. Важным аспектом использования сверточных нейронных сетей является уменьшение объема памяти и оптимизация вычислений для низкопроизводительных устройств.

Целью данной статьи является обзор современных тенденций в области оптимизации глубокого обучения нейронных сетей для решения задач компьютерного зрения в задачах фрагментации горных пород и других подобных задач. Данная работа является продолжением и расширением обзорной статьи [19]. В настоящей работе рассматриваются последние достижения в области нейронных сетей глубокого обучения для решения задач компьютерного зрения и вопросы повышения производительности реализаций нейронных сетей на различных параллельных архитектурах.

В рамках достижения указанной цели мы выделяем следующие подзадачи:

- Использование современных архитектур кодировщиков признаков, специально разработанных для работы в реальном времени.
- Использование современных архитектур головных подсетей моделей, соответствующих задачам компьютерного зрения, в частности, экземплярной сегментации, семантической сегментации и других.
- Техники оптимизации сети для использования на конечных вычислительных устройствах.

Новизна исследований в статье заключается в обзоре методов оптимизации и ускорения решения задач фрагментации в горнодобывающей промышленности. Мы ограничиваем обзор тенденциями в подходах глубокого обучения в компьютерном зрении, современными архитектурами кодировщиков признаков, а также оптимизацией их обучения и работы. Улучшения должны быть направлены на повышение вычислительной производительности при сохранении или повышении точности. Этого можно достичь путем сочетания параллельных вычислений при оптимизации нейронных сетей и использованием современных кодировщиков признаков.

Настоящая статья имеет следующую структуру. В разделе 1 кратко описана методология нашего исследования. В разделе 2 мы обсуждаем проблемы компьютерного зрения, возникающие при оценке фрагментации горных пород и аналогичные задачи в горнодобывающей промышленности. В разделе 3 мы рассматриваем современное состояние и тенденции в глубоком обучении компьютерного зрения применительно к фрагментации горных пород и аналогичных задач в горнодобывающей промышленности. В разделе 4 мы проводим обзор реализации нейронных сетей на различных типах параллельных вычислительных устройств. Основное внимание уделяется графическим процессорам, особенности использования многопроцессорных вычислений. В заключении мы формулируем основные тенденции, описанные в литературе, и наши выводы относительно перспективных методов и подходов к рассмотренной теме.

1. Методология

В нашем исследовании мы сформулировали и ответили на следующие вопросы.

- Вопрос 1. Какие методы решения задач компьютерного зрения применяются при оценке фрагментации горных пород и в смежных областях горнодобывающей промышленности?
- Вопрос 2. Какие подходы к решению соответствующих задач компьютерного зрения являются наиболее современными?
- Вопрос 3. Какие параллельные архитектуры и методы оптимизации используются при реализации сверточных нейронных сетей?

Для сбора релевантных статей по каждому вопросу мы сформулировали запросы для поисковых систем Google Scholar и Scopus, объединив соответствующие ключевые слова.

- Запрос 1: rock fragmentation, blast quality estimation, open-pit mining, mining productivity estimation, mining industry problem, computer vision.
- Запрос 2: computer vision, deep learning, computer vision neural network architectures, feature extractors for computer vision, semantic segmentation, instance segmentation, real-time instance segmentation.

- Запрос 3: deep learning, convolutional neural networks, parallel computing, high performance, real-time performance.

При отборе работ мы использовали следующие критерии:

- Статьи должны описывать специфику использования компьютерного зрения в задачах, связанных с фрагментацией горных пород. Мы рассматриваем подходы на основе глубокого обучения нейронных сетей и классические системы компьютерного зрения.
- Статьи должны описывать современное состояние, тенденции и трудности применения глубокого обучения в компьютерном зрении относительно рассматриваемых задач. Сюда входит современное состояние т.н. кодировщика признаков (feature extractor, backbone) сверточных нейронных сетей. Также сюда входит современное состояние головных подсетей для решения задач, связанных с сегментацией в реальном времени.
- В работах должно быть описано использование параллельных архитектур для реализации нейронных сетей или методов оптимизации, связанных с нейронными сетями. Это включает специфику различных архитектур, а именно многоядерных CPU, GPGPU и специализированных ускорителей, стратегии распараллеливания и оптимизации математических операций.
- Объем работы не должен быть менее шести страниц.
- Мы рассматриваем востребованные работы, опубликованные после 2017 года или имеющие не менее 10 цитирований.

2. Современное состояние компьютерного зрения в задаче фрагментации горных пород

Анализ и оценка эффективности и производительности добычи горных пород, особенно в открытом карьере, является одной из приоритетных проблем горной промышленности. Эта задача напрямую связана с оценкой фрагментации горных пород [13, 14, 17, 20–22].

Добыча в открытом карьере предполагает сбор и переработку фрагментов горных пород, полученных в результате взрывных работ на ряде относительно небольших рабочих площадках открытого карьера. На рис. 1 показан пример рабочей площадки. Снимок сделан на Баженовском асбестовом месторождении, Россия. Полученные куски породы доставляются на обогатительную фабрику для дальнейшей переработки. В результате необходимый выход продукции для всего карьера получается из комбинации всех оценок производительности рабочих площадок внутри карьера. Качественная оценка и контроль текущей производительности на каждой рабочей площадке карьера необходимы для успешного управления карьером и фабрикой.

Пример на рис. 1 соответствует исследованию, проведенному авторами [15, 16]. Работа [16] посвящена проблеме оценки фрагментации горных пород. Работа [15] посвящена оценке производительности карьера с использованием систем компьютерного зрения путем сегментации кусков породы и асбестовых прожилок внутри них. Оба исследования демонстрируют потребность в быстрых и точных методах глубокого обучения нейронных сетей в горной промышленности. Особенно это важно при использовании мобильных устройств, а также при измерениях в реальном времени. Приведенные примеры являются типичными проблемами горнодобывающей промышленности, рассматриваемыми в данном обзоре. Схема алгоритма оценки выхода продукции асбестового волокна, который был предложен в работе [15], показана на рис. 2.

Задача, показанная на рис. 2, заключается в оценке содержания асбеста в кусках породы в открытом карьере [15]. Для каждого выбранного куска породы было сделано изображение высокого разрешения, на котором были сегментированы асбестовые прожилки. Среднее отношение площади асбестовых прожилок к площади соответствующего куска породы на изображении принимается за меру производительности карьера.

Наборы данных с размеченными изображениями для задачи на рис. 2 приводятся в работе [23] для разрабатываемых участков карьеров и в работе [24] для кусков породы с асбестовыми прожилками.



Рис. 1. Изображение рабочей площадки полученной после проведения взрывных работ

Методы оценки фрагментации горных пород при помощи систем компьютерного зрения относятся к «непрямым» методам [25]. Анализ работ в области таких систем показал, что этот подход известен в литературе как один из наиболее точных. Большинство работ можно разделить на два класса. Первый основан на применении классических методов компьютерного зрения, таких как Watershed (алгоритм водораздела). Второй подход включает использование современных методов глубокого обучения нейронных сетей. В рамках этого подхода задача фрагментации горных пород может быть решена как задача сегментации контура объекта с использованием подхода семантической сегментации (см. [20, 21, 26]) либо как задача семантической сегментации [13]. Также в ряде случаев могут быть рассмотрены подходы на основе обнаружения объектов [16]. Различия этих подходов проиллюстрированы на рис. 3.

В литературе авторы отмечают, что точность для классического подхода зависит от времени суток, погодных, сезонных и других условий [27–29]. Показано, что классический подход хорошо работает только на изображениях высокого разрешения и не дает достаточно точных результатов для больших масштабов изображений. Тем не менее, такие алгоритмы могут быть менее ресурсоемкими, чем подходы на основе глубокого обучения нейронных

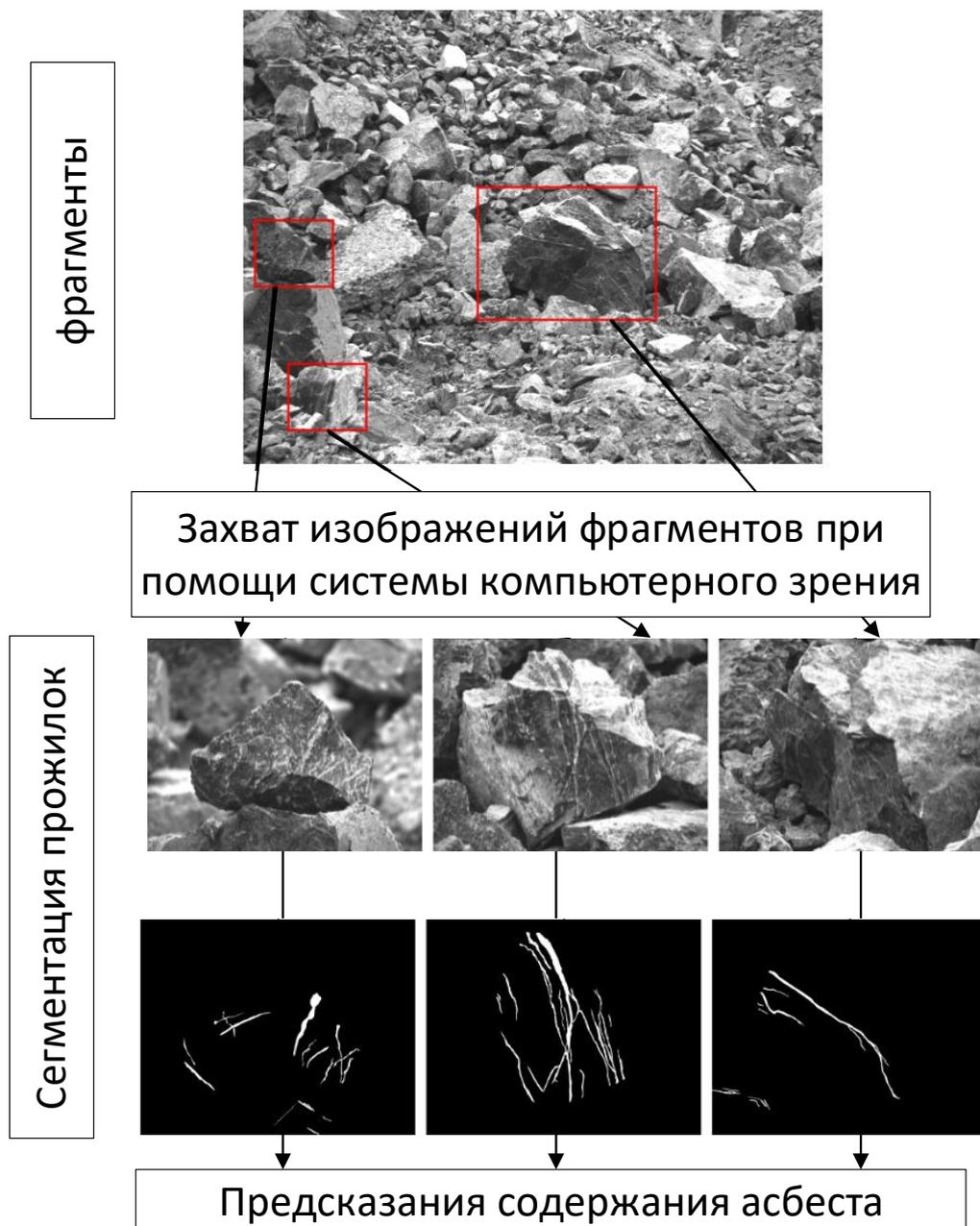


Рис. 2. Схема оценки продуктивности асбестовой породы внутри открытого карьера

сетей, применяемые в настоящее время в этой области [13, 25]. Отметим, что большинство классических решений на основе компьютерного зрения уже существует в коммерческом программном обеспечении, например, в пакетах Gold Size, Wipfrag и других. При этом данное программное обеспечение не специализировано для обработки изображений рабочих участков открытых карьеров [13, 25, 30–32]. В классическом подходе распределение размера фрагментов оценивается по значению площади круга сопоставимого размера [32]. Это значение «эквивалентного размера» может быть некорректной мерой и не отражает напрямую максимальный размер фрагмента породы, который наиболее важен с практической точки зрения в рассматриваемых задачах [13, 25]. При этом современные архитектуры нейронных сетей для задач экземплярной сегментации (instance segmentation) и семантической сегмен-

тации (semantic segmentation) способны достигать сравнимой производительности, а также более высокой точности и стабильности результатов при различных внешних условиях. Подходы на основе нейронных сетей позволяют преодолеть основные недостатки классических подходов компьютерного зрения [27–29]. По этой причине далее в работе будут рассмотрены подходы на основе нейронных сетей глубокого обучения.

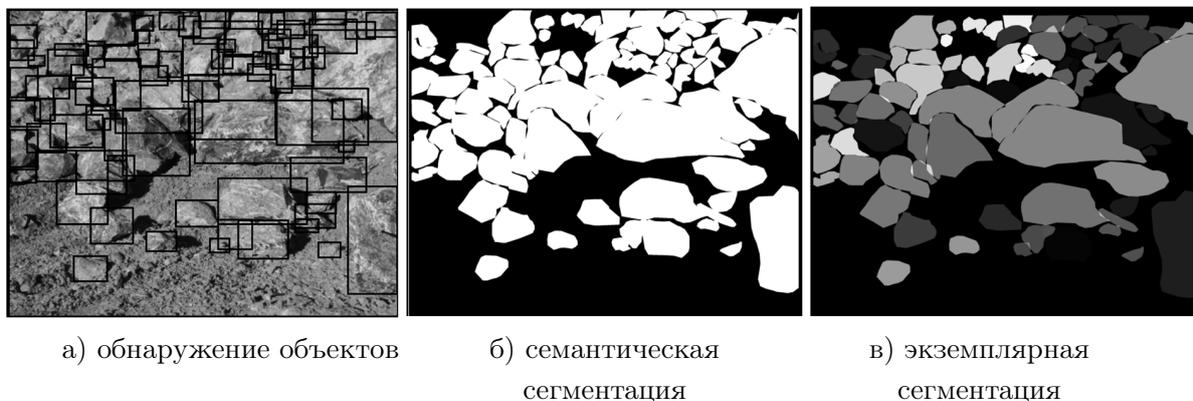


Рис. 3. Иллюстрация примера разметки изображения в наборе данных для задач

Задача семантической сегментации предполагает попиксельную классификацию. Выход нейронной сети должен иметь те же пространственные размеры, что и вход, а количество каналов должно быть равно количеству классов. Каждый пиксель для всех каналов взвешивается (с помощью Softmax) для определения класса, к которому он принадлежит. Среди всех сетей семантической сегментации наиболее популярным выбором является архитектура U-Net [33]. Будучи предложенной в 2015 году, эта архитектура остается сегодня одной из наиболее популярных. Рисунок 4 иллюстрирует архитектуру U-Net и принцип попиксельной классификации с применением функции Softmax. Архитектура U-Net реализует архитектуру вида кодировщик-декодировщик. Ключевой особенностью архитектуры является использование карт признаков из кодировщика в декодировщике. С 2015 года было предложено множество изменений в архитектуре U-Net. Большинство из них рассматривались для медицинских приложений [34, 35], а некоторые — для анализа микроскопических изображений [36]. Среди работ, в которых используются сети семантической сегментации, можно выделить следующие примеры. В работе [21] предложена классификация угля и пустых пород на конвейерной ленте с использованием пользовательской сверточной нейронной сети. Авторы собрали набор данных из 300 изображений, содержащих только уголь, смесь камней и угольную породу. Этого было достаточно для достижения точности распознавания 90%. Авторы [17] предлагают использовать архитектуру U-Net на смешанных изображениях для сегментации пустых пород в целях последующей сортировки.

В работе [20] рассматривается проблема определения размера руды как проблема сегментации контуров (задача семантической сегментации). Авторы предлагают использование модифицированной архитектуры Res-U-Net для определения размера руды как в условиях конвейерной ленты, так и в условиях открытого карьера. В обоих случаях получена точность около 90%. Кроме того, авторы протестировали классический алгоритм водораздела (watershed), который показывает сопоставимую точность для крупномасштабных изображений фрагментов скальной породы на конвейерной ленте, но имеет гораздо меньшую точность для изображений открытого карьера.

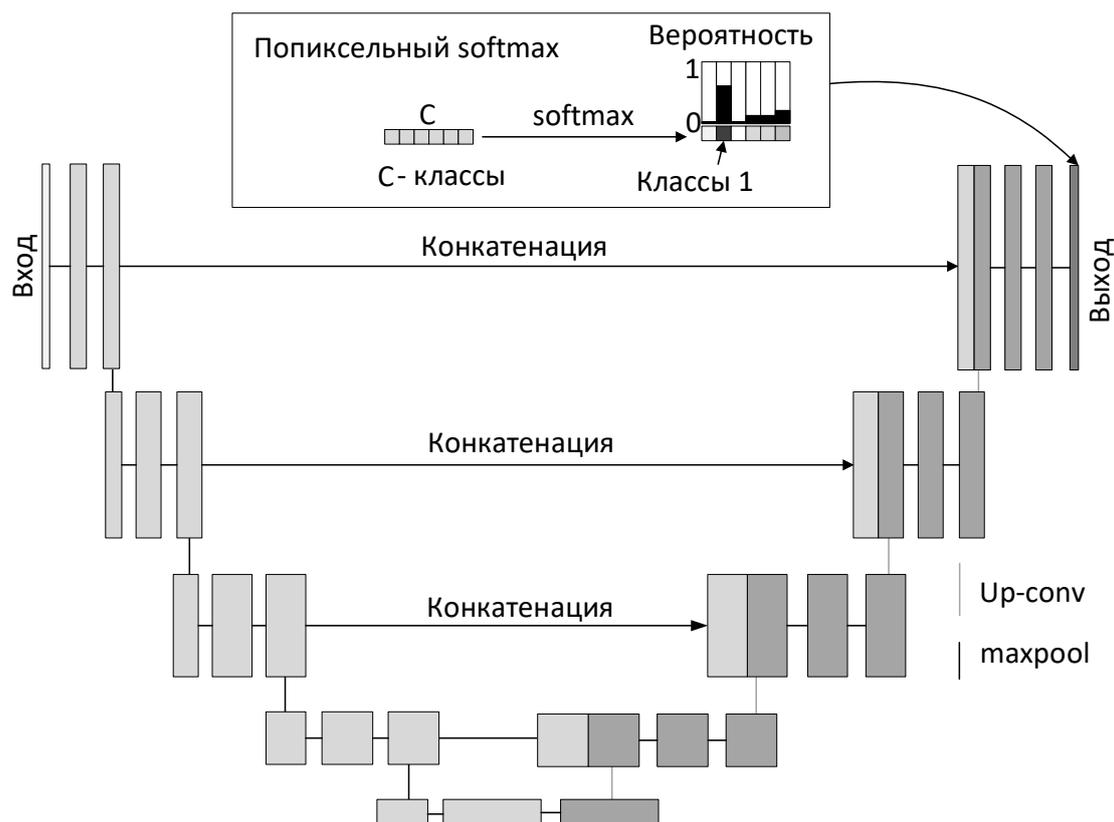


Рис. 4. Иллюстрация архитектуры U-Net и попиксельной классификации softmax

В работе [26] предложено объединить задачу сегментации при помощи U-Net с алгоритмом водораздела [37] для группировки результатов попиксельной сегментации в отдельные объекты. В статье [38] предложено использовать модифицированный алгоритм водораздела для сегментации в условиях карьера.

В статье [14] используется архитектура U-Net с модифицированными блоками для вычисления размера кусков породы и асбестовых прожилок внутри них на заводской конвейерной ленте.

Задача экземплярной сегментации в применении к глубоким нейронным сетям часто рассматривается как задача обнаружения объектов (заклочение объектов в ограничивающие рамки) с последующей семантической сегментацией экземпляров в каждой ограничивающей рамке [39]. Существуют другие подходы, например, основанные на идее разделения результатов семантической сегментации на отдельные объекты [40]. Наиболее распространенной архитектурой для экземплярной сегментации является Mask R-CNN [41]. Эта архитектура является многоэтапной. Mask R-CNN состоит из общего кодировщика признаков (например, ResNet [42]); подсети предварительного предложения регионов-кандидатов (Region Proposal Network, RPN) и головной подсети. Сеть RPN предназначена для предварительного обнаружения областей изображения, содержащих объекты потенциального интереса. Для каждой области определяются габаритные размеры, координаты центра и объектность (вероятность того, что область не является фоном). Среди таких областей в головную подсеть подаются только области с площадью пересечения меньше заданного порога и объектностью выше заданного порога. Алгоритм отбора называется немаксималь-

ным подавлением (Non-Maximum suppression, NMS). Результаты предварительного отбора «вырезаются» из карт признаков на выходе кодировщика и приводятся к единому размеру при помощи билинейной интерполяции. Головная подсеть состоит из подсети классификации объектов и подсети регрессии параметров ограничивающих рамок, а также из подсети семантической сегментации для каждого региона-кандидата. После работы сети алгоритм NMS применяется к окончательным регионам-кандидатам для отбора результатов [41]. Иллюстрация архитектуры Mask R-CNN показана на рис. 5. Впервые подход Mask R-CNN был предложен в 2017 году, но до сих пор остается одним из самых популярных решений для экземплярной сегментации [43].

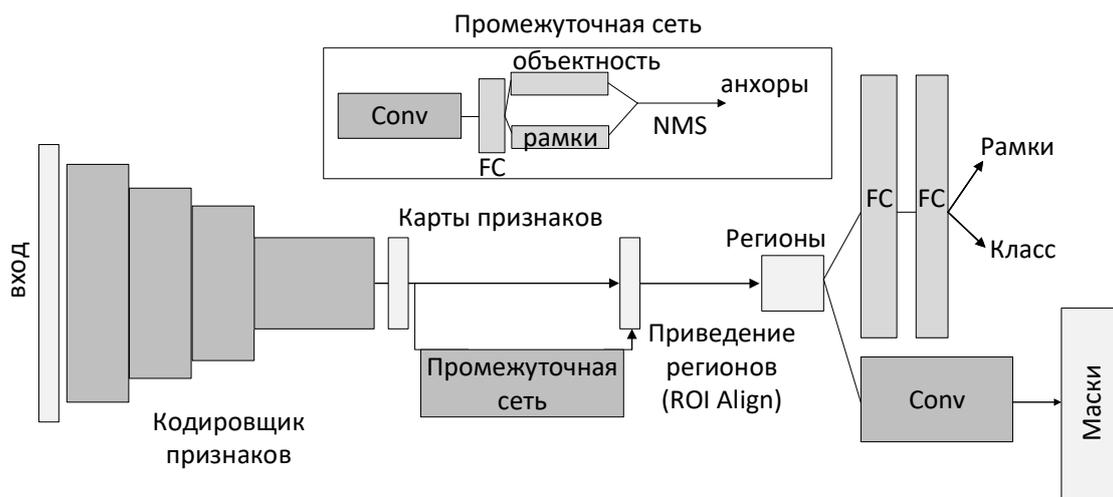


Рис. 5. Иллюстрация архитектуры Mask R-CNN

Способ использования Mask R-CNN в задачах фрагментации был предложен в работе [13], где авторы достигли точности около 90% для рабочих площадок внутри карьерах. Кроме того, было показано, что алгоритмы глубокого обучения работают лучше, чем обычные подходы компьютерного зрения для изображений высокого разрешения. Также в работе отмечена зависимость точности классических алгоритмов от времени суток, освещенности, сухой погоды и других внешних условий.

Аналогичные результаты получены в работе [44]. В статье сравниваются традиционный подход и подход на основе глубокого обучения для крупномасштабных изображений рабочих площадок внутри карьера. Авторы заявили о преимуществах Mask R-CNN для крупномасштабных изображений. Многие современные авторы утверждают, что традиционные подходы теряют оточность от таких факторов, как большой разброс размеров фрагментов, перекрывающихся экземпляров и качества фотографий, см. [13, 28, 44, 45].

В работе [15] предложен алгоритм обнаружения фрагментов породы на разрабатываемых участках карьера при помощи Mask R-CNN, а также предложено использование модифицированной архитектуры U-Net для сегментации прожилок асбестового волокна для изображения каждого из выделенных фрагментов породы.

Алгоритм состоит из четырех этапов.

1. Выявление кусков породы на полном изображении карьера.
2. Получение изображений каждого выбранного куска породы.
3. Сегментация асбестовых прожилок для каждого изображения фрагмента.

4. Получение средней оценки производительности (содержания) асбеста в карьере.

В статье [16] предлагается использовать архитектуру YOLOv5 [46] для обнаружения кусков породы и оценки их размера. Показано, что можно рассматривать задачу фрагментации как проблему обнаружения объектов. Отмечается, что использование архитектуры YOLOv5 позволяет в 10 раз ускорить решение проблемы фрагментации изображений карьеров без существенной потери точности.

Таблица 1. Современное состояние методов компьютерного зрения в задачах фрагментации горных пород

Ссылка	Комментарий
[33]	Архитектура U-Net, оригинальная публикация
[34–36]	Обзор архитектуры U-Net
[47–52]	Обзор семантической сегментации
[14, 17, 20, 21, 38, 40]	Семантическая сегментация для задачи фрагментации
[41]	Архитектура Mask R-CNN, оригинальная публикация
[13, 15, 16, 44]	Mask R-CNN в задаче фрагментации
[39, 40, 43, 49, 53–55]	Обзор сегментации экземпляров

Результаты анализа литературы по применению компьютерного зрения для решения задач фрагментации горных пород и аналогичных задач в горнодобывающей промышленности показывают, что наиболее часто применяемые методы глубокого обучения нейронных сетей основаны на хорошо зарекомендовавших себя архитектурах U-Net и Mask R-CNN. Авторы не уделяют большого внимания вычислительной оптимизации предлагаемых ими алгоритмов (моделей). Хорошо известно, что эти подходы являются ресурсоемкими и неэффективно работают на низкопроизводительных конечных устройствах и в задачах, требующих реального масштаба времени вычислений [56].

Проведенный анализ рассмотренных задач сегментации показывает, что такие архитектуры состоят из подсети кодировщика признаков (энкодер, feature extractor, backbone), промежуточной подсети (neck part) и головной подсети (решающая подсеть, head part, decision part). При этом часто именно кодировщик признаков оказывает наибольшее влияние на точность и вычислительную сложность модели. Заметим, что в упомянутых работах не уделяется внимание проблеме выбора кодировщика признаков и оптимизации его работы.

Основное решение соответствующих проблем состоит в следующем: [47–50, 57, 58]:

- использование современных, специально разработанных для реального масштаба времени архитектур кодировщиков признаков;
- обоснованный с точки зрения компромисса «точность — скорость работы» выбор общей архитектуры решения задачи;
- техника оптимизации модели нейронной сети для конечных вычислительных устройств.

Обзор наиболее важных работ по этому разделу представлен в табл. 1.

В следующем разделе описывается современное состояние в области архитектур нейронных сетей глубокого обучения для решения вышеописанных задач компьютерного зрения.

Все рассмотренные в следующем разделе архитектурные решения могут быть использованы в оптимизации задачи фрагментации горных пород.

3. Методы глубокого обучения

3.1. Архитектура сверточных нейронных сетей

В разделе представлен обзор тенденций в области глубокого обучения нейронных сетей в задачах компьютерного зрения, релевантных исследуемой тематике. В частности, рассмотрены современные архитектуры кодировщиков признаков с точки зрения их исторического развития. Отметим, что основным методом исследования архитектур кодировщиков признаков является экспериментальное тестирование для решения типичных задач классификации. Поэтому данный тип архитектур рассмотрен через призму именно этой задачи. После решения задачи классификации такой кодировщик признаков может быть перенесен на другие задачи компьютерного зрения.

Идея использования сверточной архитектуры (сверточная нейронная сеть, СНС, Convolutional Neural Network, CNN) в компьютерном зрении была предложена в 1989 году для распознавания почтовых индексов [59]. Современный подход к построению архитектур сверточных нейронных сетей был предложен в 1998 году в модели LeNet [60]. Иллюстрация LeNet показана на рис. 6. С этой архитектуры начинается исследование подхода глубоких нейронных сетей, в котором производится автоматическое извлечение признаков из «сырых» данных.

Автоматизированное извлечение признаков является основным преимуществом использования глубокого обучения в компьютерном зрении. Хорошо известно, что большинство задач компьютерного зрения трудно описать формально. Такое формальное описание можно заменить экстраполяцией результатов для набора примеров (обучающей выборки). Для этого на практике необходим хорошо подобранный и достаточно большой набор данных. Для решения задачи в такой постановке глубокая нейронная сеть включает два основных компонента: кодировщик признаков и подсеть, принимающая решения. Кодировщик признаков СНС состоит из последовательных комбинаций сверточных и вспомогательных слоев. Отметим, что применительно к компьютерному зрению операция свертки имеет несколько интуитивных преимуществ: инвариантность по масштабу, расположению и вращению признаков, а также меньшее количество параметров по сравнению с другими подходами [61, 62]. В настоящее время эти преимущества делают сверточные нейронные сети основным способом решения задач компьютерного зрения.

Современное развитие глубокого обучения СНС началось со сверточной архитектуры AlexNet (2012) [63, 64]. AlexNet состоит из 7 слоев. Сеть включает в себя такие особенности, как функция активации Rectified Linear Unit (ReLU); слой регуляризации Dropout; набор встроенных техник расширения (augmentation) изображений; слой max pooling для уменьшения размерности и другие приемы оптимизации модели [63]. Это была первая попытка интегрировать все современные на тот момент подходы как к построению архитектуры сети, так и к процедуре ее обучения.

После AlexNet следующие несколько лет прогресса глубокого обучения сверточных нейронных сетей были посвящены тому, чтобы сделать архитектуру более глубокой. Идея заключалась в увеличении так называемого рецептивного поля [61].

Предложены следующие модификации, которые сейчас являются распространенной практикой:

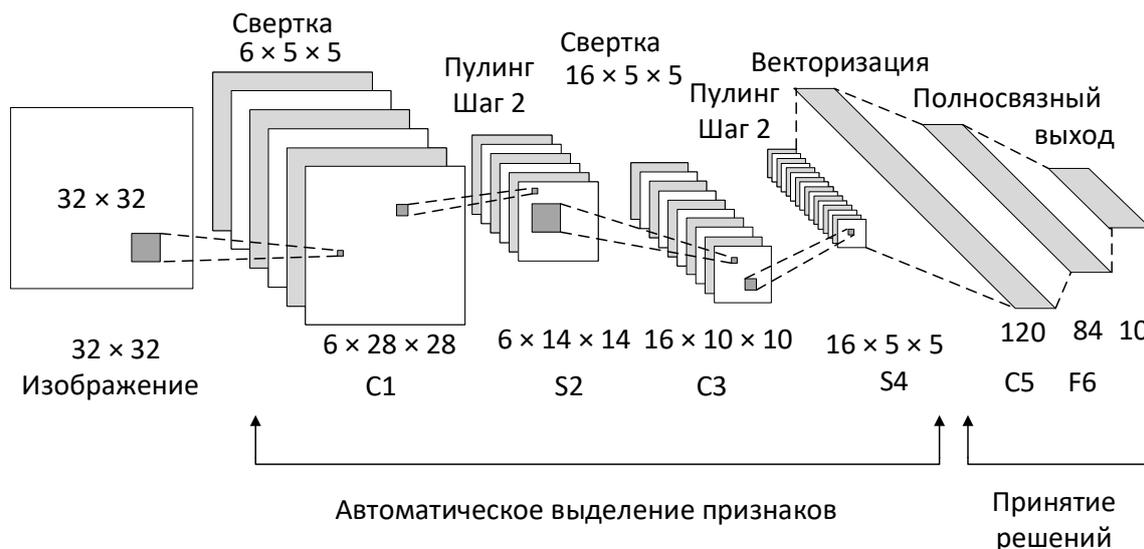


Рис. 6. Иллюстрация архитектуры сверточной нейронной сети LeNet

- оптимизация сверточного слоя с использованием различных подходов: каскадная свертка [65]; пространственно разделяемая свертка [66]; точечная свертка [67]; свертка по глубине [68];
- блок слоев с различным рецептивным полем [69];
- блоки слоев с параллельным соединением [42];
- пакетная нормализация как основной способ регуляризации [70];
- метод глобального пулинга [67];
- адаптивная оптимизация статистических моментов низшего порядка на основе градиента (ADAM) [71].

«Гонка за глубиной» закончилась в 2016 году, когда была создана модифицированная версия ResNet, включающая более 1000 слоев [72]. Было отмечено, что количество слоев больше 150 не имеет существенного эффекта. Идея остаточных слоев (residual connection, skip connection) стали доминирующими в проектировании архитектур сверточной нейронной сети. Показано, что этот прием устраняет практически все негативные эффекты, которые могут возникнуть в процессе обучения сверточной нейронной сети. Иллюстрация схемы блоков ResNet показана на рис. 7. Архитектура ResNet остается самой популярной архитектурой СНС в литературе [73]. В последние годы было предложено множество модификаций ResNet [74]. Блок ResNeXt [75] предполагает разделение основной ветви блока ResNet на несколько для повышения различия извлекаемых признаков. Блок Xception [68] расширяет результаты ResNeXt, используя отдельную свертку для каждой карты признаков. Блок DenseNet [76] предлагает объединить несколько слоев с каскадом остаточных связей для увеличения обмена информацией внутри каждого такого блока. Блок ResNet-D [77] улучшает архитектуру блока ResNet и предлагает множество практических приемов для ее обучения. Подход ViT [78] предлагает приемы для эффективного «переноса обучения» весовых параметров ResNet-подобных архитектур, а также правила «тонкой» настройки и способы избежать недостатков пакетной нормализации. Подход RegNet [79], предлагает несколько эффективных модификаций блоков ResNet, полученных методами автоматического поиска конфигураций блоков.

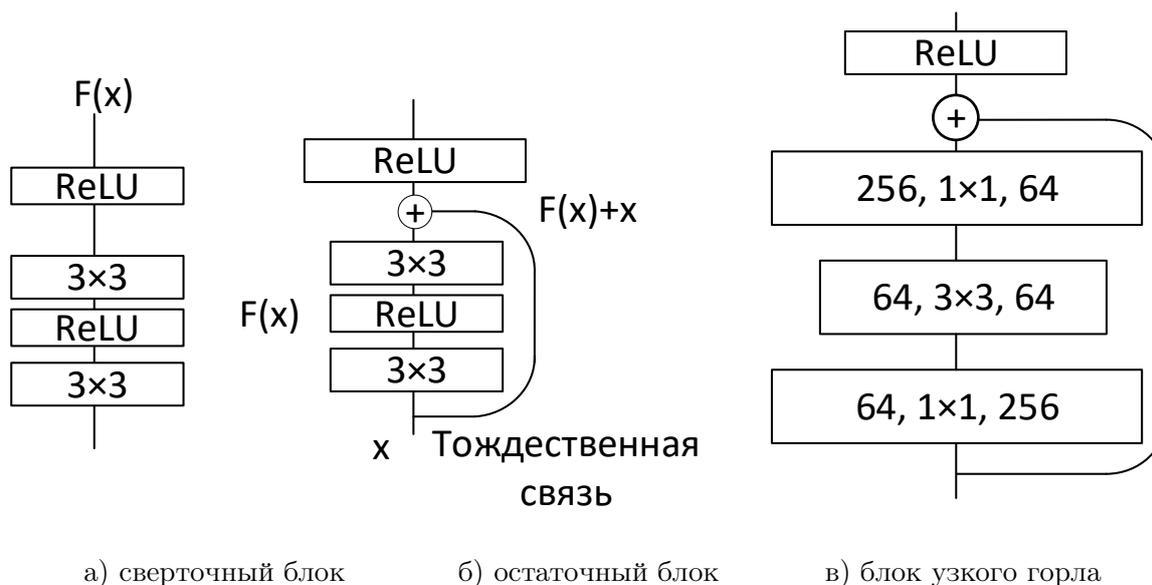


Рис. 7. Иллюстрация различных блоков остаточного соединения

Другой важный этап развития глубокого обучения СНС заключался в оптимизации архитектуры для малопроизводительных конечных устройств. Ключевое достижение 2018 года было реализовано в архитектуре MobileNetv2 [80]. Эта архитектура состоит из 17 слоев, объединенных в блоки с оптимизированными структурами. В 2018–2020 годах прогресс в оптимизации архитектур осуществлялся с помощью методов автоматизированного поиска архитектуры, известных как Neural Architecture Search (NAS) [81, 82]. Кроме того, оптимизация архитектур выполнялась с использованием механизма внимания. Среди реализаций механизма внимания основным стал подход Squeeze and Excitation (SE). Основная идея слоя SE-слоя заключается в перевзвешивании каналов выхода сверточного блока со множителями, вычисленными на основе дополнительных полносвязных слоев [83]. Наиболее успешным результатом использования NAS и SE-слоев стала архитектура EfficientNet [84]. Базовый блок EfficientNet стал базовым для многих современных исследований в области СНС в различных задачах компьютерного зрения. Схема блока EfficientNet показана на рис. 8.

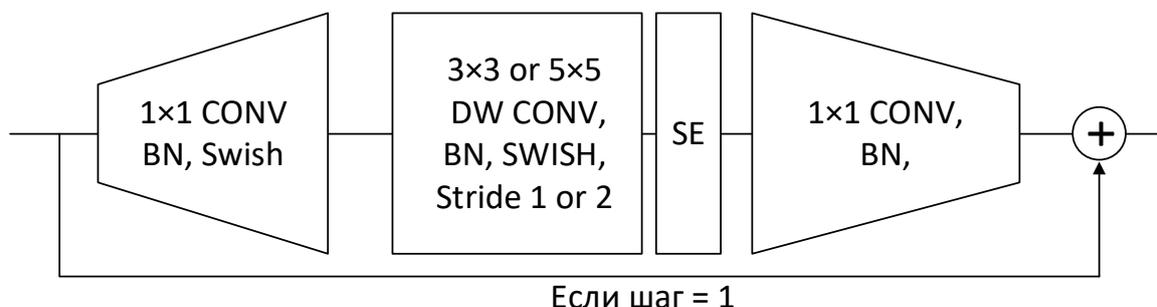


Рис. 8. Схема блочной архитектуры EfficientNet V1

3.2. Архитектуры нейронных сетей: трансформеры

Современные тенденции исследования систем компьютерного зрения включают не только сверточные блоки, но и использование блоков трансформеров [85, 86]. Архитектура блоков трансформеров предполагает только полносвязные слои. Входное изображение для трансформера делится на сетку небольших участков изображения (patches). Основной блок трансформера состоит из многоголового блока внимания (multi-head self-attention) [87] со слоем нормализации [88] и полносвязного слоя, также с нормализацией. Первым успешным примером реализации сети, построенной только из трансформеров, стала архитектура ViT в 2020 году [85]. Отметим, что ViT имеет весьма много параметров для рассматриваемых приложений и требует большого набора данных для обучения. Замечено, что блоки трансформера легко обучаются «глобальным» признакам изображений, но с трудом обучаются «локальным» и относительно небольшим признакам. Следующие шаги по модификации сети были посвящены поиску решения этой проблемы.

Современные решения включают:

- вычисление информации о внимании только для некоторых окон вместо всех участков изображения; окна могут различаться между слоями (SWIN) [58];
- использование последовательной комбинации сверточных слоев и слоев трансформеров (CoAtNet) [89];
- комбинация в каждом блоке операций свертки и операций блока трансформера (MaxVit) [90];
- поиск наиболее эффективных с вычислительной точки зрения модификаций блока трансформера (MobileVit) [91];
- поиск путей замены механизма внимания на другую нелинейную операцию для глобального извлечения признаков в блоке подобном трансформеру (MLP-Mixer) [92];
- использование мета-обучения, в т. ч. «дистилляции знаний» для конкретных задач; DeiT [93]; улучшение качества обучения целевых моделей путем подбора меток с помощью дополнительных моделей [94]; использование кросс-дата аугментации (аугментация меток и данных) [95].

Отметим, что лучшие на сегодня архитектуры-трансформеры имеют более одного миллиарда параметров и проходят предварительное обучение на сверхбольших наборах данных, имеющих порядка сотен миллионов изображений [96]. Иллюстрация архитектуры ViT показана на рис. 9.

3.3. Решающая подсеть

Как показано в обзоре, в настоящее время основными подходами к использованию компьютерного зрения в горнодобывающей промышленности являются архитектура семантической сегментации U-Net [33] и архитектура экземплярной сегментации Mask R-CNN [41]. Эти архитектуры являются наиболее популярными в соответствующих областях и могут рассматриваться как базовые модели. Обе сети содержат кодировщики признаков, основанные в основном на базовом блоке ResNet и специфичные для задач головные подсети. Важно отметить, что в исходном виде эти архитектуры не обеспечивают высокой производительности и точности по сравнению с более современными подходами (см. например [97–99], а также сравнение современного состояния на веб-сайте <https://paperswithcode.com/sota>).

В работе [19] отмечается, что среди современных достижений в области быстрых методов решения задач экземплярной сегментации (в реальном времени) выделяются такие ар-

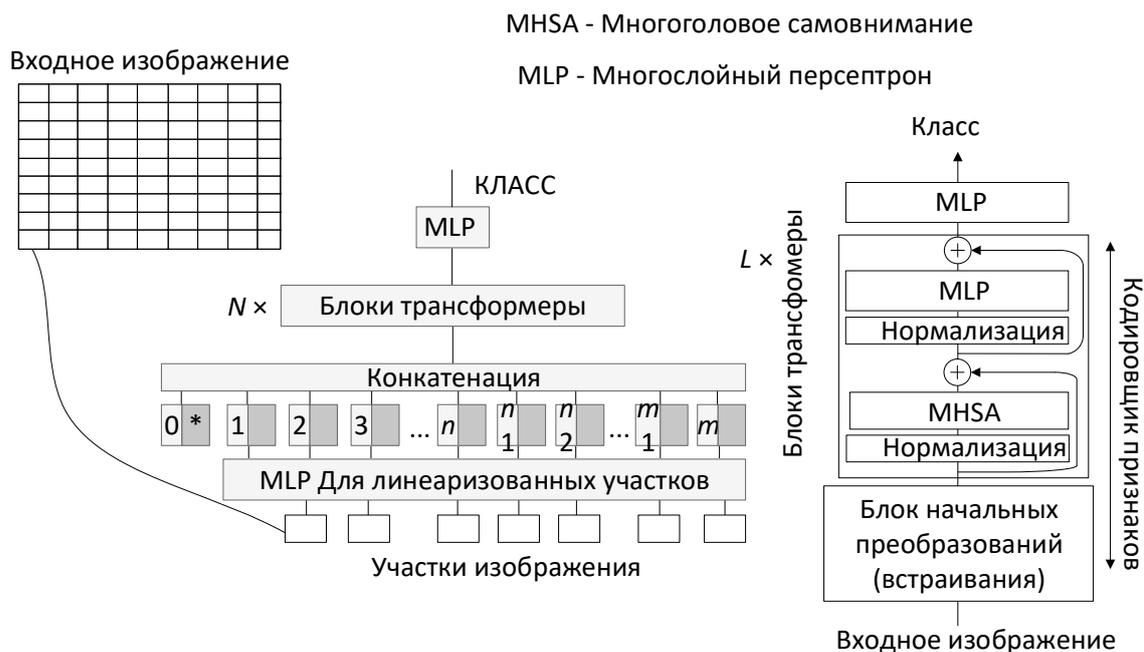


Рис. 9. Иллюстрация архитектуры ViT

хитектуры, как YOLACT [97] и ее модификации, такие как YOLACT++ [100], SOLOv2 [101] и SparseInst [98]. Отметим, что данные методы используют подход на основе поиска центральных позиций объектов и их сегментации, в отличие от сегментации на основе поиска регионов (как в Mask R-CNN). Модель обучается различать пиксели, принадлежащие либо одному и тому же, либо разным объектам. Например, подход YOLACT генерирует маски-прототипы по всему изображению, предсказывая наборы параметров для каждого экземпляра. Окончательные маски строятся после процедуры NMS. Архитектура YOLACT++ улучшает результаты базовых моделей за счет модификаций кодировщика признаков и добавления т. н. деформируемых сверток к головной подсети. Подход SOLO [102] основан на предположении, что экземпляры можно разделить по их центральному положению и размерам. Положения центров вычисляются путем деления изображения на ячейки и вычисления положения центра внутри каждой из них. Размеры экземпляров определяются на основе признаков кодировщика с т. н. пирамидальной структурой. Архитектура SOLO v2 улучшает предыдущие результаты, генерируя динамически весовые параметры для каждого потенциального экземпляра. Эта концепция называется динамическим ядром. Для каждого экземпляра генерируется своя маска (размером с изображение). Подход SparseInst аналогичен SOLOv2, но имеет две головных подсети, работающие с такими масками. Первая используется для создания масок с учетом классов, а вторая — для оценок положения и размеров объектов в каждой маске. Маски умножаются на предсказанные маски для создания масок сегментации.

Среди достойных внимания методов быстрого обнаружения объектов следует отметить архитектуры YOLOv5–YOLOv8. Эти архитектуры можно использовать как в режиме обнаружения объектов, так и в режиме экземплярной сегментации. Также в некоторых случаях в данных архитектурах выделяют отдельный подход — ориентированный поиск объектов, когда выделяют не только ограничивающие рамки для объектов, но угол наклона рамки (такой, чтобы рамка имела наименьшую площадь) [46, 99, 103–106]. Указанные архитекту-

ры наследуют единый подход YOLO, в котором предполагается, что в входное изображение делится на одинаковые ячейки. Каждый объект определяется как принадлежащий одной из этих ячеек. Каждая ячейка соответствует одному вектору на выходе модели. Вектор может включать в себя один или несколько наборов с предсказаниями размеров, класса и координат объекта. Результат объектов определяется с помощью алгоритма NMS [107]. Семейство архитектур YOLO разрабатывается с использованием большого количества экспериментов с архитектурой и методами обучения, которые называются «Bag of Specials» и «Bag of Fribies», соответственно [105, 106, 108]. В основе этих экспериментов лежит подход архитектуры YOLOv4, иллюстрация которой показана на рис. 10.

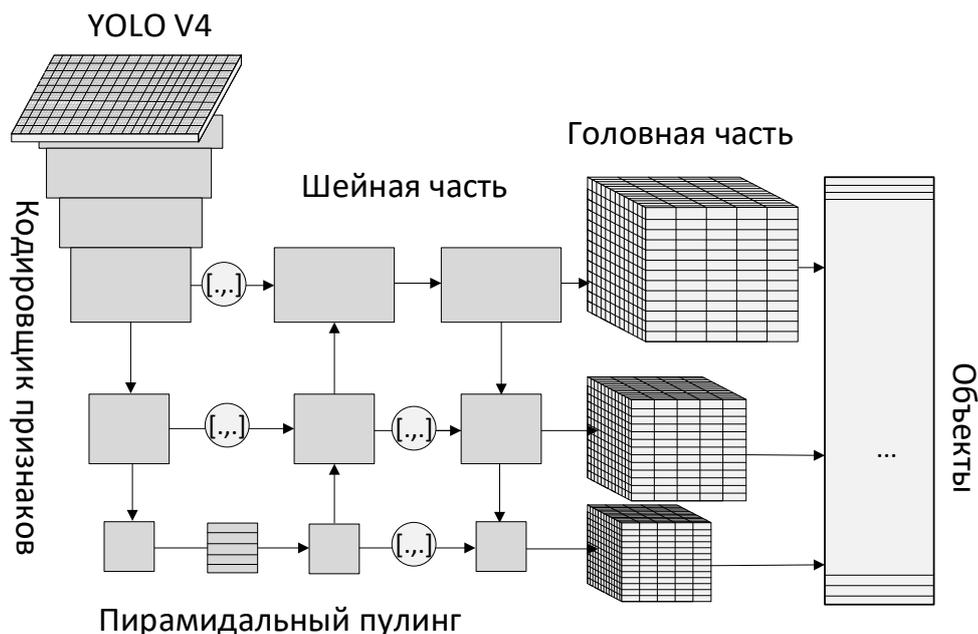


Рис. 10. Иллюстрация архитектуры YOLOv4

3.4. Тренды компьютерного зрения

Последние тенденции в компьютерном зрении можно обобщить следующим образом.

Оптимизация операции свертки. Например, для EfficientNet V2 [109] было замечено, что т. н. глубокая разделенная свертка (DepthWise Separable Convolution) становится вычислительно эффективной только для больших слоев, несмотря на меньшее количество параметров. Таким образом, в начальных блоках было предложено заменить ее на обычную свертку.

Оптимизация операции нормализации. Пакетная нормализация (BatchNorm) может снизить точность в случае предварительного обучения на большом наборе данных, при изменении размера пакета и в других случаях. Одним из способов решения этой проблемы является использование альтернативы — послойной нормализации (LayerNorm) [88]. Другой способ предполагает замену пакетной нормализации набором отдельных операций. Этот прием был реализован в NFNet [110]. Как правило замена пакетной нормализации сопровождается дополнительными операциями.

Автоматический поиск наиболее подходящей архитектуры. Подход поиска архитектуры нейросети (NAS) имеет множество преимуществ и недостатков. Основным недостатком

является зависимость пространства поиска от опыта исследователя. Также отбор архитектур из пространства поиска ограничен доступными вычислительными ресурсами и опытом (знаниями) исследователей, которые занимаются этой работой [81, 82].

Компромисс между блоками трансформерами, полностью и сверточными слоями. Как описано выше, этот вопрос остается открытым. Основными проблемами являются поиск наименьшей архитектуры (с точки зрения параметров), избежание необходимости работы с большими наборами данных для предварительного обучения и ускорение процедуры обучения.

Необходимость оптимизации головных подсетей для решения конкретных задач. Как показано выше, выбор архитектуры головной подсети позволяет достичь определенного компромисса между точностью и скоростью работы. Наиболее производительными подходами на текущий момент являются архитектуры семейства YOLO в задачах обнаружения объекта и архитектуры SOLO в задачах экземплярной сегментации [55, 111]. Наиболее точными архитектурами для решения соответствующих классов задач являются архитектуры типа Cascade-Mask-R-CNN и Hybrid task cascade for instance segmentation (HTC) [112].

Сравнение современных кодировщиков признаков на некоторых репрезентативных тестовых наборах можно найти в работе [96].

Основное обобщение ключевых публикаций представлено в табл. 2 и 3.

3.5. Архитектуры компьютерного зрения для фрагментации горных пород

В результате рассмотренных тенденций можно сделать следующий вывод относительно компромисса между ростом точности и низкой вычислительной производительностью. В компьютерном зрении современные архитектуры СНС используют как блоки свертки, так и трансформеры в виде ResNet-подобных блоков. Сверточный слой может быть реализован как в традиционном виде (для начальных слоев), так и в виде DepthWise-сверток для остальных блоков. Метод свертки DepthWise имеет преимущества в количестве обучаемых параметров и, вероятно, обладает большей потенциальной репрезентативной способностью. Блоки трансформеры в целом являются ресурсоемкими, но некоторые модификации, такие как MobileVit или SWIN, оптимизированы для достижения наилучшего компромисса между точностью и скоростью работы.

Наиболее эффективная комбинация блоков кодировщика признаков для конкретной задачи может быть определена только экспериментально. Тем не менее, в задачах фрагментации горных пород архитектуры Mask R-CNN и U-Net могут быть использованы в качестве базовых решений для соответствующих задач. Подходы, основанные на архитектурах SOLO и YOLO, можно рассматривать как основу для перспективных подходов для планируемых исследований.

При выборе архитектур нейронных сетей важно учитывать специфику подходов к переносу обучения. В частности, требуются модификации операций пакетной нормализации (BatchNorm) и работы с большими наборами данных на стадии предварительного обучения. Также итоговая модель может быть обучена с использованием таких техник, как дистилляция знаний.

Таблица 2. Тренды компьютерного зрения: кодировщики признаков

Ссылка	Архитектура	Комментарии
[60]	LeNet	Первая сверточная нейронная сеть; SGD с импульсом; инициализация весов;
[63]	AlexNet	Первая из эпохи современного глубокого обучения сверточная нейронная сеть; аугментация данных; дропаут
[65]	VGGNet	Каскадная сверточная сеть
[71]		Адаптивная оптимизация статистических моментов низшего порядка на основе градиента (ADAM)
[67]	NIN	Концепция блока; точечная свертка; глобальное среднее объединение
[69]	Inception	Блок с изменением рецептивного поля
[70]		Пакетная нормализация
[66]	Inception V3	Пространственно разделяемая свертка
[42]	ResNet	Остаточная связь; блок «узкого горлышка»
[75]	ResNeXt	Многоканальный блок «узкого горлышка»
[68]	Xception	DepthWise глубокая разделяемая свертка
[76]	DenseNet	Блок с несколькими тождественными связями
[80]	MobileNet V2	MobileNet архитектура; Обратные остаточные блоки с пространственно разделяемыми свертками
[83]	SE Net	Слой сжатия и возбуждения (Squeeze and Excitation layer)
[84, 109]	EfficientNet	Оптимизированный блок, полученный с помощью NAS
[77]	ResNet-D	Улучшенный ResNet
[78]	BiT	Эффективный перенос обучения
[79]	RegNet	Оптимизация блока ResNet
[110]	NFNet	Альтернатива пакетной нормализации
[85]	ViT	Первая эффективная архитектура визуального трансформера
[93]	DeiT	Дистилляция знаний, алгоритм мета-обучения для архитектур-трансформеров
[58]	SWIN	Вычисление информации о внимании только для некоторых окон, а не для всех
[89]	CoAtNet	Комбинация сверточного блока и блока-трансформера
[91]	MobileVit	Эффективный мобильный трансформер
[92]	MLP-Mixer	Блок типа трансформер без внимания

Таблица 3. Тренды компьютерного зрения: решающие подсети

Ссылка	Архитектура	Комментарии
[41]	Mask R-CNN	Базовая архитектура для экземплярной сегментации
[97, 100]	YOLOACT, ++	Архитектура экземплярной сегментации для низкопроизводительных устройств
[101, 102]	SOLO V1,V2	Архитектура экземплярной сегментации и динамическая свертка
[98]	SparseInst	Архитектура быстрой сегментации в режиме реального времени с улучшением результатов SOLO
[33]	U-Net	Базовая архитектура семантической сегментации
[113]	DeepLab v3+	Современная архитектура семантической сегментации
[108]	YOLOv4	Архитектура YOLO 4
[46, 99, 103, 104]	YOLOv5–v8	Современные архитектуры YOLO
[81, 82]		Обзор метода поиска архитектуры нейронной сети (NAS)
[64]		Исторический обзор архитектур
[85, 86]		Обзор архитектур трансформеров
[94]		Мета-обучение
[95]		Аугментация данных и кросс-дата аугментация
[105, 106]		Обзор архитектур решения задач обнаружения объектов в реальном времени
[96]		Результаты для бенчмарка ImageNet
[55]		Результаты тестов для экземплярной сегментации в реальном времени
[111]		Результаты тестов для задачи обнаружения объектов в реальном времени

Актуальны и другие вопросы: ускорение времени обучения и работы СНС, а также снижение требований к объему памяти для модели. Особенно это важно для низкопроизводительных систем. В следующем разделе обсуждаются параллельные архитектуры и методы оптимизации для эффективной реализации нейронных сетей.

4. Реализация на параллельных архитектурах

4.1. Специфика центральных процессоров

Центральные процессоры (CPU) можно рассматривать как многоядерные и многопоточные архитектуры, оптимизированные для векторных и матричных операций. Например, настольные процессоры могут иметь до 64 ядер и 128 потоков (с точки зрения аппаратной многопоточности) и до 72 ядер и 288 потоков в серверных процессорах. Такие процессоры

можно использовать как основу для обучения и работы нейронных сетей. Преимущества данного подхода заключаются в следующем:

- отсутствие необходимости в дополнительном оборудовании;
- простота работы с любыми фреймворками (большинство современных фреймворков поддерживает архитектуру x86-64);
- наличие дополнительных инструментов оптимизации производительности (OpenVINO [114], NNPack [115], oneDNN [116], CcT [117]), а также инструментов для оптимизации и распараллеливания нейронных сетей, например, Intel BigDL [118];
- оптимизированные наборы SIMD-инструкций (AMX, AVX-512VNNI и другие), поддерживающие форматы с ограниченной точностью BF16 и INT8 [119];
- возможность использования не только для машинного обучения, но и для вспомогательных задач.

Одним из передовых направлений в развитии процессоров являются гетерогенные архитектуры, включающие как CPU, так и GPU, FPGA и другие типы ускорителей, использующие унифицированную кэш-память и SoC. Примером такого подхода является ускорители нейронных сетей Intel на базе процессора Miryard. Ожидается, что такой подход позволит достичь максимальной эффективности при низкой стоимости конечного устройства и ограниченном энергопотреблении.

4.2. Специфика графических процессоров

Графические процессоры (GPU) — это тип вычислительных устройств, построенных по принципу матричных архитектур с синхронным параллелизмом на уровне инструкций (так называемые SIMT-архитектуры, «одиночный поток команд, множественный поток данных»). Этот тип архитектуры ориентирован на обработку больших массивов регулярных данных (включая изображения). Первоначально задачами, для которых разрабатывались такие процессоры, были специальные операции обработки изображений, которые выполнялись на аппаратном уровне. Большинство таких операций включает умножение матриц и реализуются с помощью операций сложения и умножения с плавающей точкой (Fused Multiply-Add, FMA). Поэтому ряд современных графических процессоров специализируется именно на этом типе операций. Такие ускорители называются General Purpose Graphics Processor Unit (GPGPU) [120].

В настоящее время GPGPU представляют собой многопроцессорные системы, состоящие из набора потоковых мультипроцессоров (SMT). Мультипроцессоры также могут быть объединены в группы, кластеры (или срезы). Каждый мультипроцессор SIMT выполняет блок (или поток) инструкций SIMD с помощью набора функциональных модулей.

Графические ускорители стали популярным инструментом для обучения и выполнения алгоритмов машинного обучения. В первую очередь это связано с тем, что алгоритмы машинного обучения (включая глубокие нейронные сети) хорошо поддаются распараллеливанию. Более того, алгоритмы обучения и выполнения нейронных сетей включают набор последовательных операций, которые выполняются одна за другой над одним набором данных. Большинство вычислительных операций нейронной сети сводится к FMA. Среди преимуществ графических ускорителей в таких задачах можно выделить способность к распараллеливанию и обеспечение высокой вычислительной мощности (десятки TFLOPs в формате FP32 и сотни TFLOPs в формате FP16 с использованием тензорных ядер), а

также продвинутые технологии для организации GPU-серверов. Для процесса обучения нейронных сетей графические ускорители, как правило, обеспечивают наилучшую производительность по отношению к стоимости. Кроме того, GPU часто являются единственным оборудованием, которое способно масштабироваться с вычислительными требованиями определенных алгоритмов и моделей глубокого обучения.

В настоящее время наиболее популярными в задачах машинного обучения являются графические ускорители компании NVIDIA [121]. Такая популярность обусловлена как поддержкой со стороны NVIDIA, так и использованием технологии CUDA большинством фреймворков для обучения нейронных сетей. Использование в ускорителях технологии разреженных матриц и форматов INT8/INT4 позволяет ускорить выполнение до 10 раз. Реальная производительность графических ускорителей, как правило, ниже пиковой и зависит от оптимизации кода. Такая оптимизация в ручном режиме часто игнорируется, так как является очень трудоемкой. Возможно использование автоматической оптимизации с помощью утилит, таких как NVIDIA TensorRT [122].

4.2.1. Использование пониженной точности на GPU

Одной из ключевых проблем при реализации нейронных сетей является большой объем оперативной памяти, необходимый для хранения модели. В настоящее время наиболее популярными методами сжатия моделей являются квантование с использованием разреженности весов и округление функций активации. Квантование может быть линейным или нелинейным [123]. Одной из основных тенденций в нейронных сетях и других алгоритмах машинного обучения является использование арифметических операций с пониженной и смешанной точностью. Операции со смешанной точностью предполагают, что хотя бы один из операндов имеет пониженную точность. Примерами таких форматов являются TensorFloat-32 (TF32) [124], Brain Float (BF32 и BF16) [125] и другие.

Одним из частных случаев является так называемая бинаризованная нейронная сеть (BNN) [126]. Ее преимущество заключается в максимальном использовании пропускной способности памяти и минимальном размере модели [127, 128]. Также интерес представляют архитектуры, основанные на формате INT2 $\{-1, 0, 1\}$ — они относятся к классу Ternary Neural Networks (TNN) [129]. Главная особенность TNN заключается в автоматическом округлении малых значений весов до нуля во время обучения. При этом потеря точности может быть снижена до 1% по сравнению с традиционным форматом FP32 [130].

Ряд современных работ показывают, что наиболее перспективным способом является обучение сети в форматах с плавающей запятой с последующей тонкой настройкой или переобучением в целочисленных форматах. Цель такого подхода — уменьшить размер обученной модели и времени выполнения при минимальной потере точности [131]. Следует отметить, что в этом случае нет необходимости оптимизировать все слои нейронной сети. Можно использовать различные степени квантования для параметров нейронной сети, чтобы уменьшить потери в точности. Верхние слои глубоких нейронных сетей, а также входной слой, требуют большего диапазона весов и значений функций активации, чем нижние слои. Также отмечается, что при квантовании требуется процесс переобучения (или дообучения) нейронной сети [132]. Например, в сверточных нейронных сетях наибольшая вычислительная нагрузка приходится на многомерные свертки в верхних слоях кодировщика признаков. Таким образом, они должны быть приоритетными для оптимизации [133].

Современные GPGPU позволяют выполнять операции FMA с несколькими вариантами глубины квантования [120]. В частности, поддерживаются операции со смешанной точностью, например, операции FMA вида $D = A \cdot B + C$, где операнды A и B имеют половинную точности [134]. Использование операндов с пониженной точностью уменьшает объем ОЗУ, занимаемый параметрами нейронной сети.

Для ускорения обработки вычислений в глубоких нейронных сетях в ряде архитектур графических ускорителей используются так называемые модули тензорных вычислений. В зависимости от архитектуры GPU, эти модули могут отличаться как по количеству, так и по поддерживаемой ими функционалу. Например, в процессорах архитектуры NVIDIA Ampere [135] в каждом блоке используется тензорное ядро, поддерживающее операции смешанной точности FP16/FP32, FP16, BF16, TF32, FP64, INT8, INT4, бинарные операции, а также операции над разреженными матрицами в соотношении 2 нуля на 4 параметра. Более того, каждый тензорный модуль представляет собой массив, например, $4 \times 4 \times 4$, предназначенный для выполнения операций FMA $D = A \cdot B + C$ в цикле, где A , B , C и D — матрицы размера 4×4 [134].

Современные фреймворки для нейронных сетей предоставляют инструменты для автоматического и адаптивного применения методов смешанной точности [136–138]. Они облегчают задачу квантования модели и балансировки ее производительности, эффективности и точности.

4.2.2. Использование разреженных матриц на GPU

На практике обученная нейронная сеть имеет большинство весовых коэффициентов, равных или близких к нулю. Такие веса могут быть удалены или обрезаны [139, 140]. Отмечается, что в сверточных сетях разреженность может достигать 90% без существенных потерь в точности. Также отмечается, что в TNN разреженность может достигать до 50% без значительных потерь в точности. В экспериментах [130] авторы получили 1% потери точности при разреженности 65% для архитектуры GoogLeNet. Также отметим, что в современных графических ускорителях используются технологии оптимизации вычислений операций над разреженными матрицами [141]. Эта техника сокращает время работы нейронных сетей.

Отметим, что разреженность нейронных сетей может достигаться автоматически в ходе обучения модели. Это связано с частым использованием полулинейного модуля (ReLU) в качестве функции активации в СНС. Это может ускорить процедуру обучения нейронной сети. Например, в архитектуре NVIDIA Ampere [135], используется предположение, что операнды инструкций для разреженных матриц — это матрицы, где есть два ненулевых значения в каждом векторе с четырьмя входными значениями (разреженность строк 2:4). Благодаря такой структуре матрицы ее можно сжимать, уменьшая требуемый объем памяти и пропускную способность почти вдвое (разреженное тензорное ядро).

4.2.3. Оптимизация доступа к памяти

Скорость доступа к памяти может ограничивать производительность нейронных сетей. Обычно для оценки производительности системы в отношении вычислительной производительности и пропускной способности памяти используется модель Roofline [142]. Эта модель включает предельные технические характеристики системы и может быть использована

для оценки производительности реальной программы. Анализ модели Roofline показывает, что для достижения более высокой производительности следует применять одновременно несколько методов оптимизации. Повышение производительности может быть достигнуто, например, за счет оптимизации вычислительных модулей, использования адаптированных форматов данных и оптимизации вычислительных алгоритмов. Также значительное повышение производительности может быть получено за счет оптимизации циклов. Для этого могут применяться различные методы, такие как переупорядочивание циклов, конвейеризация циклов, разворачивание циклов и другие. Переупорядочивание циклов уменьшает количество обращений к памяти между итерациями цикла. При разворачивании цикла несколько итераций выполняются одновременно параллельно. Метод конвейеризации циклов выполняет итерации цикла с перекрытием таким образом, что следующая итерация начинается до завершения предыдущей. Оптимальное количество не перекрывающихся итераций может варьироваться в зависимости от алгоритма вычислений. Этот параметр также влияет на количество обращений к памяти и должен зависеть от архитектуры аппаратного обеспечения. Оптимизация цикла может быть выполнена путем использования двойных буферов для хранения результатов [143]. В некоторых работах [144] предлагается разворачивание ядра для СНС. Такая оптимизация подразумевает замену одного сверточного ядра, например, размера 5×5 , несколькими ядрами размером 3×3 . Оптимизация цикла также может быть достигнута за счет использования регулярной структуры нейронных сетей [133].

В дополнение к стратегиям, описанным выше, для оптимизации циклов могут использоваться методы более высокого уровня. К таким методам относятся блокирование (тайлинг) циклов и чередование циклов. Блокирование циклов — это разбиение циклов на более мелкие компоненты, блоки (тайлы). Все входные данные для одного блока хранятся в специальном буфере или кэш-памяти. Цикл выполняется с каждым блоком по очереди. Во время чередования циклов компоненты цикла переставляются таким образом, чтобы соседние компоненты использовали одни и те же данные. Таким образом, не нужно перезагружать данные из памяти для отдельных компонентов. Эти методы позволяют контролировать размер и порядок блоков. Таким образом, можно оптимизировать циклы для различных слоев нейронной сети, например, СНС [133].

Для оптимизации операции свертки [145, 146], можно использовать несколько типов алгоритмов: пространственная свертка (CONV), векторизованная свертка (например, im2col), быстрая свертка Винограда (Winograd) и частотная свертка. Алгоритм Winograd является самым быстрым, но при этом он требует отдельной реализации для различных размеров сверточных ядер, что не всегда выполнимо. Частотная свертка имеет умеренную скорость, но может быть реализована только для размеров ядра, равных степеням двойки. Пространственная свертка является самой медленной, но она имеет самые низкие требования к объему кэш-памяти и пропускной способности шины доступа к памяти.

4.3. Использование нескольких графических процессоров

Достичь высокого параллелизма нейронных сетей можно путем повышения эффективности вычислительных операций и распределение вычисления всей сети по нескольким измерениям. Например, разбив данные на мини-пакеты послойно или тензорно, можно разделить прямой проход и обратное распространение между параллельными процессорами. Существует несколько основных стратегий распределения рабочей нагрузки [147]:

- распределение входных данных — параллелизм данных;
- распределение структуры нейросети — параллелизм модели (тензорная нарезка);
- конвейерная обработка — параллелизм модели (послойный);
- комбинированная стратегия — гибридный параллелизм.

4.3.1. Параллелизм данных

Среди методов обучения нейронной сети популярным является стохастический градиентный спуск (Stochastic Gradient Descent, SGD). При его применении количество обновлений весов уменьшается за счет вычисления потерь выборки в мини-пакетах, а затем усреднения градиента по мини-пакетам [148]. Данные обрабатываются за N выборок. Мини-пакетные методы используются как компромисс между традиционным SGD, который использует всю выборку за один раз, и пакетными методами, использующими весь набор данных на каждой итерации. В [149] доказано, что SGD с мини-пакетами обеспечивает сходимость, аналогичную традиционному SGD. Самый простой подход к распараллеливанию — разделить обработку мини-пакетов на несколько вычислительных устройств, поскольку большинство операций не зависят от количества элементов мини-пакета. Этот подход известен как шаблонный параллелизм (Pattern Parallelism) [150].

Параллелизм данных имеет основной недостаток, связанный с чрезмерными коммуникационными затратами между вычислительными узлами, поскольку веса должны быть синхронизированы между каждым вычислительным узлом. Такие коммуникационные затраты увеличиваются с ростом размера модели, что значительно затрудняет масштабируемость параллелизма данных [150].

Большинство фреймворков глубокого обучения сегодня поддерживают параллелизм данных для одного GPU, нескольких GPU или кластера из нескольких GPU. Масштаб параллелизма данных естественным образом определяется размером мини-пакета. За исключением пакетной нормализации, которая обычно применяется между слоями и функциями активации, все операции обрабатывают одну выборку за один раз. Это означает, что процедуры прямого прохождения через СНС и обратного распространения ошибки почти полностью независимы. На этапе обновления весов распределенные результаты усредняются, чтобы получить градиент для всего пакета данных. Поскольку все параметры глубокой нейронной сети должны быть доступны всем устройствам, они дублируются.

Существует несколько факторов, которые влияют на производительность синхронного SGD с параллелизмом данных. Чтобы сохранить точность, размеры мини-пакетов обычно ограничены. Если мы увеличиваем размер выше определенного порога, то мы теряем точность [151]. В то же время при масштабировании системы до нескольких рабочих узлов в кластере размер обучающей партии увеличивается линейно. Таким образом, в обучающей системе с большим количеством узлов размер набора данных для каждого узла системы должен быть небольшим. Малый размер выборки данных на вычислительном узле способствует сокращению времени вычислений и высоким затратам на синхронизацию, что делает алгоритмы передачи данных важным фактором производительности в больших обучающих системах [152].

Применяя различные модификации [153, 154] в процессе обучения, можно увеличить размер мини-пакета без значительной потери точности. Хотя проблема обобщающей способности сети все еще существует, она не настолько серьезна, как это было заявлено в предыдущих работах. Узким местом (ограничивающим масштабируемость) является опера-

ция пакетной нормализации, для выполнения которой требуется синхронизация. Поскольку пакетная нормализация применяется многократно в некоторых архитектурах глубоких нейронных сетей, она становится слишком затратной. Для смягчения этой проблемы популярными подходами к пакетной нормализации используют небольшие подмножества из мини-пакета (например, 32 изображения) для независимой нормализации. Если для каждого процессора запланировано хотя бы 32 изображения, то синхронизация становится локальной, что, в свою очередь, увеличивает потенциал масштабирования.

В работе [155] предлагается использовать весовую нормализацию для отделения нормы параметра от его направления путем перепараметризации. Это уменьшает вычислительную сложность с $O(\log N)$ до $O(1)$ за счет устранения внутренних зависимостей внутри мини-пакета. По мнению авторов этого метода, весовая нормализация снижает необходимость пакетной нормализации и обеспечивает сопоставимую точность при использовании упрощенной версии пакетной нормализации без коррекции дисперсии.

В литературе были предложены и другие подходы к параллелизму данных. Например, в ParallelSGD [156] алгоритм SGD выполняется с помощью мини-пакетов, несколько раз параллельно, распределяя набор данных между процессорами. После того, как все экземпляры SGD сходятся, полученные веса объединяются и усредняются, что обеспечивает параллелизм данных.

ParallelSGD был разработан с использованием парадигмы программирования MapReduce [157, 158]. Используя MapReduce, можно планировать параллельные задачи для выполнения на мультипроцессорах, распределенных средах и нескольких GPU. Ранее потенциал масштабирования MapReduce был протестирован в различных задачах машинного обучения, включая нейронные сети [159–161], что способствовало необходимости перехода от однопроцессорного обучения к системам с распределенной памятью. Отметим, что его высокая универсальность затрудняет создание реализаций, специально предназначенных для нейронных сетей глубокого обучения.

Современные реализации используют высокопроизводительные коммуникационные интерфейсы (такие как MPI) [162] для достижения тонкого параллелизма. Это уменьшает задержку за счет асинхронного выполнения кода, конвейеризации, непостоянных коммуникаций и использования параллелизма внутри вычислительных устройств [163–166]. В последнем случае мини-пакеты разбиваются на микро-пакеты, которые распределяются для параллельной или последовательной обработки. Это уменьшает объем памяти, позволяя использовать более быстрые методы.

Фреймворк PyTorch реализует автоматический параллелизм данных [167] путем разделения входных данных по указанным устройствам с помощью разбиения пакета на части. Модуль реплицируется на каждой машине и каждом устройстве, и каждая такая реплика обрабатывает часть входных данных. Во время обратного прохода градиенты от каждого узла усредняются.

4.3.2. Параллелизм модели

Второй стратегией, используемой для распараллеливания глубоких нейронных сетей, является параллелизм модели, также известный как параллелизм сети [168, 169].

Эта стратегия представляет собой распределение частей глубокой нейронной сети между компьютерами. Каждое вычислительное устройство выполняет обновление весов внутри назначенного ему подмножества модели. В результате интенсивность обмена данными зна-

чительно снижается по сравнению с параллелизмом данных. Более того, такой подход позволяет обучать более крупные модели и обходить ограничения на размер, накладываемые памятью, доступной одному процессору [149].

Здесь вычисления распределяются по нейронам в каждом слое или по измерениям в четырехмерном тензоре. В этом случае мини-пакет копируется на все процессоры, и разные части глубокой нейронной сети вычисляются на разных процессорах. Это снижает потребление памяти (поскольку вся сеть больше не хранится в одном месте), но добавляет дополнительные коммуникации между уровнями.

Поскольку при этом размер мини-пакетов не меняется, то компромисс между использованием данных и коммуникациями отсутствует. Архитектура глубоких нейронных сетей создает внутренние зависимости между слоями, которые, в свою очередь, генерируют коммуникации, определяющие общую производительность. Например, полносвязные слои подразумевают коммуникации «все со всеми» (в отличие от стратегии параллелизма данных, которая позволяет избежать этих коммуникаций), поскольку нейроны одного слоя связаны со всеми нейронами следующего слоя.

Метод сокращения числа коммуникаций между полносвязными слоями заключается в использовании алгоритма матричного умножения Кэннона, модифицированного для использования в нейронных сетях [170]. Сообщается, что такой алгоритм обеспечивает более высокую эффективность и скорость по сравнению с простым разделением на многослойные сети меньшего масштаба.

Для СНС применение модельного параллелизма имеет ограниченную эффективность [171]. Если выборки разделены между процессорами по частям (или каналам), то для вычисления результата каждой свертки придется получать результаты от всех остальных процессоров, поскольку эта операция суммирует результаты. Для частичного решения этой проблемы были предложены локально связанные сети (Locally Connected Network, LCN) [172–175]. Они также выполняют свертки, но для каждой области они применяют несколько локальных фильтров, обеспечивающих разделение по измерениям, устраняя необходимость в коммуникациях «все со всеми».

Использование локально связанных сетей дает возможность запустить СНС на 5000 узлах CPU с трехузловым кластере с несколькими GPU [176]. Отсутствие обмена весами приводит к тому, что обучение не зависит от коммуникаций, что дает значительный потенциал для масштабирования. Успешное применение тех же методов в СНС требует тонкого контроля параллелизма. Обмен весами является неотъемлемой частью работы СНС. Он помогает уменьшить объем памяти и улучшить коммуникации. Таким образом, стандартные операции свертки используются более широко, чем локально связанные сети.

4.3.3. Конвейеризация

Под конвейеризацией глубокого обучения можно понимать либо перекрытие вычислений между различными слоями по мере поступления данных, либо разделение глубокой нейронной сети (Deep Neural Network, DNN) по глубине с закреплением уровней за конкретными процессорами. Конвейер можно рассматривать как форму параллелизма данных (элементы или выборки обрабатываются параллельно), а также как параллелизм модели (длина конвейера определяется структурой DNN).

Первая форма конвейеризации может быть использована для перекрытия этапов прямого прохождения, обратного распространения и обновления весов [177, 178]. Эта схема

широко используется на практике и повышает эффективность использования ресурсов за счет сокращения времени простоя процессора. На более высоком уровне детализации архитектуры нейронных сетей могут быть разработаны по принципу вычислений перекрывающихся слоев, как это происходит в сетях с глубоким суммированием (Deep Stacking Network, DSN) [179]. В DSN каждый полносвязный слой вычисляется на отдельном шаге. В конечном итоге результаты всех предыдущих шагов передаются для входа следующего слоя. Благодаря свободной зависимости данных, этот метод позволяет частично вычислять каждый слой параллельно.

Применительно к послойному разделению [180, 181], многопроцессорный конвейер имеет ряд преимуществ как перед параллелизмом данных, так и перед параллелизмом моделей. Во-первых, нет необходимости хранить все параметры на всех процессорах во время прямого прохода и обратного распространения ошибки (как в случае с модельным параллелизмом). Во-вторых, существует фиксированное число конечных точек связи между процессорами на границах уровней, поскольку исходный и целевой процессоры всегда известны. Более того, поскольку процессоры всегда вычисляют одни и те же слои, веса можно хранить в кэше, чтобы уменьшить накладные расходы на доступ к памяти. Недостатки конвейеризации заключаются в том, что данные (выборки) должны поступать с определенной скоростью для полного использования системы, а задержка пропорциональна количеству процессоров.

При простой реализации модельного параллелизма одновременно активен только один GPU. PipeDream [182] предлагает решение проблемы с помощью конвейеризации, но конвейерный модельный параллелизм вводит проблемы стабильности и согласованности обновления весов. Поскольку в конвейере одновременно обрабатывается несколько мини-пакетов, последующий мини-пакет может начать обучение до того, как предыдущая обновит веса. Проблема застоя/согласованности приводит к нестабильному обучению и снижает точность модели. Grіpe [183] предлагает метод конвейера, отличный от PipeDream, чтобы полностью избежать этих проблем. Он разбивает каждый мини-пакет на несколько микро-пакетов, а затем конвейеризует выполнение каждого набора микро-пакетов по ячейкам. Веса обновляются синхронно для всех накопленных градиентов после завершения обработки мини-пакета. Несмотря на то, что этот подход может увеличить использование GPU по сравнению с неконвейерным параллелизмом моделей, он снижает пропускную способность GPU. Для ее повышения был предложен механизм SpecTrain, который предсказывает будущие веса на ранних стадиях конвейера [171].

4.3.4. Комбинированный параллелизм

Комбинация нескольких стратегий параллелизма может преодолеть недостатки каждой стратегии. Существует несколько успешных подходов к реализации гибридных методов.

Например, AlexNet [64, 184] — сверточная нейронная сеть, где большинство вычислений выполняется в сверточных слоях, но большая часть параметров принадлежит полносвязным слоям. В работе [185] авторы использовали несколько стратегий параллелизма и коммуникационных оптимизаций. Для обучения сети они добились ускорения в 400 раз, используя 512 графических процессоров.

AMPNet [186] — асинхронная реализация обучения DNN на процессорах, которая использует промежуточное представление для реализации параллелизма модели. В частности, параллельные подзадачи внутри и между слоями формируются и планируются асин-

хронно. Кроме того, асинхронное выполнение динамического потока управления позволяет конвейерно выполнять прямой проход, обратное распространение и обновление весов. Основным преимуществом AMPNet является древовидная рекуррентная нейронная сеть, которая использует переменную длину выборки и динамический поток управления (в отличие от однородных СНС).

Наконец, распределенная система глубокого обучения DistBelief [187] сочетает в себе все три стратегии параллелизма. В этой реализации обучение проводится одновременно на нескольких экземплярах модели, причем каждый экземпляр обучается на разных выборках (параллелизм данных). Внутри каждого экземпляра DNN происходит распределение как по нейронам в одном слое (параллелизм модели), так и по разным слоям (конвейеризация). Таким образом, в DistBelief конвейеризация не ограничивается различными ядрами CPU на одном узле.

4.3.5. Особенности параллельных сверточных нейронных сетей

В этом разделе мы рассмотрим различные подходы к распараллеливанию СНС. Обучение глубоких СНС на больших наборах данных затратно и требует много времени.

Архитектура сверточных нейронных сетей позволяет сократить время обучения сети, поскольку первые четыре слоя не являются полностью связанными между собой. Эта часть сети может быть распараллелена, что позволяет сократить время обучения. Существует множество методик распараллеливания алгоритмов нейронных сетей, например, распараллеливание операций на нижних слоях, распределение их по разным прямым и обратным потокам и т.д. Для выбора оптимального подхода необходимо учитывать характеристики сети и особенности архитектуры компьютера. Разработка параллельных алгоритмов для нейронных сетей осуществляется при наличии как минимум двух физических процессоров.

Существуют две проблемы при разработке масштабируемых параллельных сверточных нейронных сетей в вычислительных средах с распределенной памятью. Одна из них — высокая степень зависимости данных, которая проявляется при обновлении параметров модели между двумя соседними мини-пакетами, а другая — большой объем данных, которые необходимо передавать по каналам связи. Для их решения используются различные стратегии, включая перекрытие межпроцессных взаимодействий [188].

В работе [188] предложен подход для совмещения передачи данных и вычислений. В алгоритме обратного распространения ошибки градиент функции стоимости вычисляются над всеми параметрами сети. Поскольку нет зависимости данных между градиентами на разных слоях модели, то обмен данными для обмена градиентами между всеми вычислительными узлами может осуществляться одновременно с расчетами для других слоев. Необходимо дублировать вычисления и передачу данных для полного использования аппаратных ресурсов и достижения высокого ускорения. В научной литературе есть работы, посвященные эффективному распределению рабочей нагрузки для снижения затрат на коммуникации между узлами. Работа [188] демонстрирует влияние дублирования на масштабируемость обучения СНС, когда архитектура СНС распараллеливается с помощью параллелизма данных. Модели обучаются с помощью синхронного SGD, поэтому результаты параллельного и последовательного обучения одинаковы. Более оптимальная стратегия распараллеливания основана на двух ключевых приемах, направленных на максимизацию перекрытия между вычислениями и коммуникациями узлов. На начальном этапе все градиенты параметров объединяются в два больших блока с последующим их уменьшением

по всем узлам с использованием асинхронной связи. Далее, исходя из анализа зависимости данных, необходимо максимизировать перекрытие путем выбора оптимального размера каждого фрагмента градиента. Кроме того, вычисления градиента повторяются в нескольких полносвязных слоях. В зависимости от архитектуры модели пересчет градиента может значительно снизить коммуникационные затраты. Это также позволяет перекрыть время связи с временем прямого распространения ошибки для следующего мини-пакета. Данный метод позволил добиться ускорения обучения модели VGG-A до 77.97 раз при использовании 128 узлов.

4.4. Специфика FPGA и ASIC

Многие исследования и коммерческие проекты предлагают разработку специализированных аппаратных ускорителей для обучения и исполнения нейронных сетей [144, 189–191]. Наиболее важной причиной для разработки специализированных архитектур является максимальное использование параллелизма, присущего моделям машинного обучения и нейронным сетям, особенно глубоким нейронным сетям. Как правило, CPU, GPU и другие ускорители общего назначения не достигают оптимальной производительности в конкретных задачах. В то же время высокая универсальность таких процессоров означает большее энергопотребление и высокую цену. Даже если процессоры имеют специальные модули для обработки нейронных сетей, они лишь дополняют основную архитектуру [192].

В настоящее время как в литературе, так и в коммерческих реализациях предложено большое количество специализированных архитектур вычислительных ускорителей. Часто специализированные ускорители основаны на технологии Field-Programmable Gate Array (FPGA) или Application Specific Integrated Circuit (ASIC). Основным преимуществом FPGA является возможность организации достаточно гибкой параллельной вычислительной системы с заданной точностью. Современные программируемые логические интегральные схемы (ПЛИС) могут включать несколько различных вариантов вычислений, таких как таблицы соответствия (LUT), цифровая обработка сигналов (DSP) и двоичная логика (flip-flop). Вычисления могут выполняться на любом из этих компонентов. Во многих случаях ПЛИС реализуются как гетерогенные системы с контроллером или процессором (технология System-on-Chip). Это позволяет разработчикам обеспечить наиболее гибкий подход к аппаратной и программной реконфигурации блока исполнения.

Технология FPGA — популярная технология для реализации аппаратных ускорителей нейронных сетей. Также ускорители могут быть изготовлены на основе технологии ASIC [193]. Следует отметить, что обычно специализированные архитектуры FPGA основаны на принципах SIMD CPU или SIMT GPGPU. Помимо анализа аппаратных ускорителей, в некоторых работах рассматриваются программно-аппаратные архитектуры с возможностью реконфигурации [194]. Существуют следующие преимущества специализированных ускорителей нейронных сетей, помимо энергоэффективности, цены и размера:

- оптимизированный набор инструкций (Instructure Set Architecture, ISA);
- более высокая степень параллелизации, степень повторного использования данных и буферизации (с использованием встроенных буферов First In, First Out, FIFO);
- возможность обработки специализированных форматов данных (задаваемая пользователем битовая глубина), а также разреженных вычислений.

В некоторых реализациях вместо ISA вычисления нейронной сети выполняются с помощью машины конечных состояний [195].

Таблица 4. Параллельные архитектуры, используемые для нейронных сетей

Тип оборудования	Ссылки	Преимущества	Недостатки
CPU	[114–119]	Широкая доступность, широкая поддержка	Ограниченная производительность, высокое энергопотребление
GPGPU	[120–122]	Лучшая производительность по цене, хорошая масштабируемость	Высокое энергопотребление, высокая цена
FPGA/ASIC	[119, 130, 144, 189–193, 196, 197]	Низкое энергопотребление, низкая цена, хорошая масштабируемость	Ограниченная производительность

Отмечается, например, [130], что ускорители FPGA лучше подходят для СНС, чем GPU и CPU, поскольку они могут создавать произвольные конфигурации вычислительных модулей. Эти модули обычно формируются в виде элемента обработки (Processing Element, PE). Блок PE представляет собой элемент скалярных, векторных или матричных вычислений. В литературе рассматриваются архитектуры элемента обработки с SIMD (обычно для процессоров CPU) или SIMT (для GPGPU). Эти конфигурации представлены в классе темпоральных архитектур [119]. Специализированные конфигурации PE для ускорителей DNN включают архитектуры Very Long Instruction Word (VLIW) и Decoupled Access/Execute, а также систолические массивы [196]

Таблица 5. Методы ускорения реализации нейронных сетей

Техника	Ссылки	Описание
Сокращенная и смешанная точность	[123–125, 130, 136–138, 199–202]	Уменьшает размер моделей, ускоряет обучение и выполнение
Разреженные матрицы	[130, 139, 140]	Сокращает время выполнения
Оптимизация доступа к памяти	[133, 143–146]	Оптимизация циклов, оптимизация свертки матрицы
Параллелизм данных	[148–150, 152, 156, 167, 171]	Распределение входных данных
Параллелизм модели	[168, 169, 172–175, 203]	Распределение структуры сети
Конвейеризация	[171, 177–183]	Распределение слоев сети или этапов обучения
Комбинированный параллелизм	[185, 186]	Комбинация нескольких стратегий параллелизма

Архитектуры типа систолических массивов реализуют принцип обработки потока данных и относятся к классу пространственных ускорителей [119]. Систолические массивы могут быть как статическими, так и реконфигурируемыми. Большинство ускорителей DNN используют статические массивы, построенные по технологии ASIC [197]. В настоящее время наиболее распространенным типом систолического массива является Tensor Processor Unit (TPU) [193].

Систолические массивы обладают такими преимуществами, как высокая степень повторного использования данных, низкие требования к памяти и низкое энергопотребление. Целью использования этих типов архитектуры является снижение влияния задержки памяти вне кристалла за счет оптимизации структуры процессора под архитектуру нейронных сетей [198]. Систолические тензорные массивы также могут быть оптимизированы для разряженных матричных умножений [197]. Многочисленные источники (см. обзор [119]) говорят о том, что помимо TPU перспективными могут быть и некоторые другие типы пространственных архитектур ускорителей нейронных сетей. Эти системы предназначены для ускорения операций обобщенного матричного умножения и свертки с использованием множества элементов с регулярной организацией.

В табл. 4 и 5 представлен обобщенный обзор литературы для этого раздела.

Заключение

Проведен обзор применения методов глубокого обучения в компьютерном зрении для решения задач фрагментации горных пород и задач в горнодобывающей промышленности. Результаты анализа показали, что большинство современных работ сосредоточено на проведении оценок параметров фрагментов горных пород с использованием систем компьютерного зрения на базе сверточных нейронных сетей глубокого обучения. При этом рассматриваются подходы семантической сегментации в сочетании с дополнительными операциями или подходы на основе экземплярной сегментации. Для реализации обучения и работы нейронных сетей используются параллельные вычислительные архитектуры.

Перечислим основные тенденции, представленные в литературе. Тенденции разделены в соответствии с методологией исследования и поставленными выше вопросами.

Вопрос 1. Какие методы решения задач компьютерного зрения применяются при оценке фрагментации горных пород и в смежных областях горнодобывающей промышленности?

- Сверточные нейронные сети являются одним из самых популярных подходов к решению проблемы оценки фрагментации горных пород.
- Оригинальные архитектуры U-Net и Mask R-CNN могут применяться в качестве базовых решений.
- Задачи требуют быстрых вычислений при сохранении высокой точности. Эти требования могут быть выполнены с помощью современных облегченных архитектур нейронных сетей, с одной стороны, и методов оптимизации вычислений для глубокого обучения, с другой стороны.

Вопрос 2. Какие подходы к решению соответствующих задач компьютерного зрения являются наиболее современными?

- Использование современных архитектур кодировщиков признаков, достигающих наилучшей точности при условии ориентации на работу в реальном времени.
- Поиск наиболее эффективных комбинаций сверток и других операций, в том числе блоков трансформеров и операций внимания.
- Стратегия обучения и регуляризации модели может включать различные приемы: кросс-аугментация данных, пакетная нормализация и мета-обучение для снижения вычислительных затрат при сохранении высокой точности.
- Быстрые модели решения задач обнаружения объектов и экземплярной сегментации (модели реального времени) могут быть модифицированы с помощью различных кодировщиков, обеспечивающих снижение вычислительной сложности при минимальных потерях точности.

Вопрос 3. Какие параллельные архитектуры и методы оптимизации используются при реализации сверточных нейронных сетей?

- Наиболее популярным средством для реализации искусственных нейронных сетей в научных и промышленных приложениях являются графические процессоры.
- Для реализации архитектур и процессов их обучения большинство производителей вычислительных устройств и разработчиков программного обеспечения вводят возможности вычислений со смешанной и пониженной точностью, а также работы с разреженными матрицами. Данная тенденция связана с попытками сократить время выполнения вычислительных операций, а также уменьшить объем данных в памяти. Это позволяет увеличить размер пакетов при обучении нейронных сетей, а также обеспечить работу сетей на низкопроизводительных устройствах.

Сформулируем основные результаты нашего обзора.

- Обзор посвящен проблемам использования систем компьютерного зрения для решения задач оценки фрагментации горных пород и других аналогичных задач горнодобывающей промышленности. Как показывают рассмотренные публикации, в настоящее время исследователи в этой области продолжают использовать базовые устаревшие подходы компьютерного зрения как с нейронными сетями глубокого обучения, так и без них. Большинство этих подходов относится к работам, опубликованным до 2017 года.
- Мы хотим обратить внимание читателей и исследователей на последние достижения в области компьютерного зрения (с 2017 года по настоящее время) и на использование этих результатов в задачах фрагментации горных пород.
- Результаты обзора показывают, что в исследуемой области наиболее целесообразно использовать подход глубокого обучения нейронных сетей для задач экземплярной сегментации и обнаружения объектов в реальном времени с различными вариантами кодировщиков признаков и решающих подсетей. Выбор конкретной архитектуры, плюсы и минусы различных подходов, а также рекомендации по их использованию являются предметом отдельного исследования.

Подводя итоги, можно сделать вывод, что в области оценки фрагментации горных пород и связанных с ней задач в горнодобывающей промышленности возможен значитель-

ный прорыв по отношению к текущему состоянию области исследований. Перспективными являются вопросы достижения компромисса между современным состоянием архитектур компьютерного зрения с глубоким обучением и оптимизацией их работы для вычислительных устройств с помощью рассмотренных методов.

Исследование выполнено за счет совместного гранта Российского научного фонда и Правительства Свердловской области №22-21-20051, <https://rscf.ru/project/22-21-20051/>.

Литература

1. Fu Y., Aldrich C. Deep learning in mining and mineral processing operations: a review // IFAC-PapersOnLine. 2020. Vol. 53, no. 2. P. 11920–11925. DOI: 10.1016/j.ifacol.2020.12.712.
2. Zhou W., Wang H., Wan Z. Ore Image Classification Based on Improved CNN // Computers and Electrical Engineering. 2022. Vol. 99. P. 107819. DOI: 10.1016/j.compeleceng.2022.107819.
3. Liu X., Wang H., Jing H., *et al.* Research on intelligent identification of rock types based on faster R-CNN method // IEEE Access. 2020. Vol. 8. P. 21804–21812. DOI: 10.1109/ACCESS.2020.2968515.
4. Amiripallia S.S., Rao G.N., Beharaa J., Sanjay K. Mineral Rock Classification Using Convolutional Neural Network // First International Conference on Recent Trends in Computing (ICRTC 2021), Virtual, Kopargaon, India, May 21–22, 2021. Advances in Parallel Computing. Vol. 39 / ed. by M. Rajesh, K. Vengatesan, M. Gnanasekar, *et al.* Amsterdam, The Netherlands: IOS Press, 2021. P. 499–505. DOI: 10.3233/APC210235.
5. Karimpouli S., Tahmasebi P. Segmentation of digital rock images using deep convolutional autoencoder networks // Computers & Geosciences. 2019. Vol. 126. P. 142–150. DOI: 10.1016/j.cageo.2019.02.003.
6. He M., Zhang Z., Ren J., *et al.* Deep convolutional neural network for fast determination of the rock strength parameters using drilling data // International Journal of Rock Mechanics and Mining Sciences. 2019. Vol. 123. P. 104084. DOI: 10.1016/j.ijrmms.2019.104084.
7. Alzubaidi F., Mostaghimi P., Swietojanski P., *et al.* Automated lithology classification from drill core images using convolutional neural networks // Journal of Petroleum Science and Engineering. 2021. Vol. 197. P. 107933. DOI: 10.1016/j.petro.2020.107933.
8. Chen T., Hu N., Niu R., *et al.* Object-Oriented Open-Pit Mine Mapping Using Gaofen-2 Satellite Image and Convolutional Neural Network, for the Yuzhou City, China // Remote Sensing. 2020. Vol. 12, no. 23. P. 3895. DOI: 10.3390/rs12233895.
9. Baek J., Choi Y. Deep neural network for predicting ore production by truck-haulage systems in open-pit mines // Applied Sciences. 2020. Vol. 10, no. 5. P. 1657. DOI: 10.3390/app10051657.
10. Williams J., Singh J., Kumral M., Ramirez Ruiseco J. Exploring deep learning for digit-limit optimization in open-pit mines // Natural Resources Research. 2021. Vol. 30, no. 3. P. 2085–2101. DOI: 10.1007/s11053-021-09864-y.
11. Somua-Gyimah G., Frimpong S., Nyaaba W., Gbadam E. A computer vision system for terrain recognition and object detection tasks in mining and construction environments //

- 2019 SME Annual Conference and Expo and CMA 121st National Western Mining Conference, Denver, CO, USA, February 24–27, 2019. Society for Mining, Metallurgy and Exploration (SME), 2019.
12. Zeng F., Jacobson A., Smith D., *et al.* Lookup: Vision-only real-time precise underground localisation for autonomous mining vehicles // 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, May 20–24, 2019. IEEE, 2019. P. 1444–1450. DOI: 10.1109/ICRA.2019.8794453.
 13. Vu T., Bao T., Hoang Q.V., *et al.* Measuring blast fragmentation at Nui Phao open-pit mine, Vietnam using the Mask R-CNN deep learning model // Mining Technology. 2021. Vol. 130, no. 4. P. 232–243. DOI: 10.1080/25726668.2021.1944458.
 14. Zyuzin V., Ronkin M., Porshnev S., Kalmykov A. Computer vision system for the automatic asbestos content control in stones // Big Data and AI Conference 2020, Moscow, Russian Federation, September 17–18, 2020. IOP Publishing: Journal of Physics: Conference Series. Vol. 1727, 2021. P. 012014. DOI: 10.1088/1742-6596/1727/1/012014.
 15. Zyuzin V., Ronkin M., Porshnev S., Kalmykov A. Automatic Asbestos Control Using Deep Learning Based Computer Vision System // Applied Sciences. 2021. Vol. 11, no. 22. P. 10532. DOI: 10.3390/app112210532.
 16. Ronkin M., Kalmykov A., Reshetnikov K., Zyuzin V. Investigation of Object Detection Based Method for Open-Pit Blast Quality Estimation // 2022 Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT), Yekaterinburg, Russian Federation, September 19–21, 2022. IEEE, 2022. P. 248–251. DOI: 10.1109/USBREIT56278.2022.9923353.
 17. Gao R., Sun Z., Li W., *et al.* Automatic coal and gangue segmentation using U-Net based fully convolutional networks // Energies. 2020. Vol. 13, no. 4. P. 829. DOI: 10.3390/en13040829.
 18. Sangaiah A.K. Deep learning and parallel computing environment for bioengineering systems. St. Louis, MO, USA: Academic Press, 2019. 280 p.
 19. Ronkin M. V., Akimova E. N., Misilov V. E. Review of deep learning approaches in solving rock fragmentation problems // AIMS Mathematics. 2023. Vol. 8, no. 10. P. 23900–23940. DOI: 10.3934/math.20231219.
 20. Liu X., Zhang Y., Jing H., *et al.* Ore image segmentation method using U-Net and Res_Unet convolutional networks // RSC advances. 2020. Vol. 10, no. 16. P. 9396–9406. DOI: 10.1039/C9RA05877J.
 21. Si L., Xiong X., Wang Z., Tan C. A deep convolutional neural network model for intelligent discrimination between coal and rocks in coal mining face // Mathematical Problems in Engineering. 2020. Vol. 2020. P. 2616510. DOI: 10.1155/2020/2616510.
 22. Su C., Xu S.-j., Zhu K.-y., Zhang X.-c. Rock classification in petrographic thin section images based on concatenated convolutional neural networks // Earth Science Informatics. 2020. Vol. 13, no. 4. P. 1477–1484. DOI: 10.1007/s12145-020-00505-1.
 23. Ronkin M., Reshetnikov K., Zyuzin V. Open-Pits asbestos. 2022. DOI: 10.17632/pfdbfpfygh. (accessed: 16.12.2022).

24. Ronkin M., Reshetnikov K., Zyuzin V., *et al.* Asbest veins in the open pit conditions. 2022. DOI: 10.17632/y2jfk63tpd. (accessed: 16.12.2022).
25. Babaeian M., Ataei M., Sereshki F., *et al.* A new framework for evaluation of rock fragmentation in open pit mines // Journal of Rock Mechanics and Geotechnical Engineering. 2019. Vol. 11, no. 2. P. 325–336. DOI: 10.1016/j.jrmge.2018.11.006.
26. Li H., Pan C., Chen Z. and Wulamu A., Yang A. Ore image segmentation method based on U-Net and watershed // Comput. Mater. Contin. 2020. Vol. 65. P. 563–578. DOI: 10.32604/cmc.2020.09806.
27. Mkwelo S., Nicolls V., De Jager G. Watershed-based segmentation of rock scenes and proximity-based classification of watershed regions under uncontrolled lighting // SAIEE Africa Research Journal. 2005. Vol. 96, no. 1. P. 28–34. DOI: 10.23919/SAIEE.2005.9488146.
28. Bamford T., Esmaceli K., Schoellig A.P. A deep learning approach for rock fragmentation analysis // International Journal of Rock Mechanics and Mining Sciences. 2021. Vol. 145. P. 104839. DOI: 10.1016/j.ijrmms.2021.104839.
29. Jung D., Choi Y. Systematic review of machine learning applications in mining: Exploration, exploitation, and reclamation // Minerals. 2021. Vol. 11, no. 2. P. 148. DOI: 10.3390/min11020148.
30. Franklin J.A., Katsabanis T. Measurement of blast fragmentation. Rotterdam, the Netherlands: A. A. Balkema, 1996. 324 p.
31. Tosun A. A modified Wipfrag program for determining muckpile fragmentation // Journal of the Southern African Institute of Mining and Metallurgy. 2018. Vol. 118, no. 10. P. 1113–1199. DOI: 10.17159/2411-9717/2018/v118n10a13.
32. Latham J.-P., Kemeny J., Maerz N., *et al.* A blind comparison between results of four image analysis systems using a photo-library of piles of sieved fragments // Fragblast. 2003. Vol. 7, no. 2. P. 105–132. DOI: 10.1076/frag.7.2.105.15899.
33. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional networks for biomedical image segmentation // International Conference on Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Munich, Germany, October 5–9, 2015. Proceedings, Part III. Vol. 9351 / ed. by N. Navab, J. Hornegger, W. Wells, A. Frangi. Springer, 2015. P. 234–241. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24574-4_28.
34. Siddique N., Paheding S., Elkin C.P., Devabhaktuni V. U-net and its variants for medical image segmentation: A review of theory and applications // IEEE Access. 2021. Vol. 9. P. 82031–82057. DOI: 10.1109/ACCESS.2021.3086020.
35. Yin X.-X., Sun L., Fu Y., *et al.* U-Net-Based Medical Image Segmentation // Journal of Healthcare Engineering. 2022. Vol. 2022. P. 4189781 DOI: 10.1155/2022/4189781.
36. Wu J., Liu W., Li C., *et al.* A State-of-the-art Survey of U-Net in Microscopic Image Analysis: from Simple Usage to Structure Mortification // CoRR. 2022. Vol. abs/2202.06465. arXiv: 2202.06465. URL: <https://arxiv.org/abs/2202.06465>.
37. Beucher S. Use of watersheds in contour detection // International Workshop on Image Processing: Real-time Edge and Motion detection/estimation, Rennes, France, September 17–21, 1979. CCETT, 1979.

38. Guo Q., Wang Y., Yang S., Xiang Z. A method of blasted rock image segmentation based on improved watershed algorithm // *Scientific Reports*. 2022. Vol. 12, no. 1. P. 1–21. DOI: 10.1038/s41598-022-11351-0.
39. Gu W., Bai S., Kong L. A review on 2D instance segmentation based on deep neural networks // *Image and Vision Computing*. 2022. Vol. 120. P. 104401. DOI: 10.1016/j.imavis.2022.104401.
40. Hafiz A.M., Bhat G.M. A survey on instance segmentation: state of the art // *International journal of multimedia information retrieval*. 2020. Vol. 9, no. 3. P. 171–189. DOI: 10.1007/s13735-020-00195-x.
41. He K., Gkioxari G., Dollár P., Girshick R. Mask R-CNN // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Vol. 42, no. 2. P. 386–397. DOI: 10.1109/TPAMI.2018.2844175.
42. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 27–30, 2016. IEEE, 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.
43. Ramesh C.S., *et al.* A Review on Instance Segmentation Using Mask R-CNN // *Proceedings of the International Conference on Systems, Energy & Environment (ICSEE) 2021, Kerala, India, January 22–23, 2021*. SSRN, 2021. P. 183–186. DOI: 10.2139/ssrn.3794272.
44. Schenk F., Tscharf A., Mayer G., Fraundorfer F. Automatic muck pile characterization from UAV images // *ISPRS Geospatial Week 2019, Enschede, The Netherlands, June 10–14, 2019*. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2019. Vol. IV-2/W5. P. 163–170. DOI: 10.5194/isprs-annals-IV-2-W5-163-2019.
45. Maitre J., Bouchard K., Bédard L.P. Mineral grains recognition using computer vision and machine learning // *Computers & Geosciences*. 2019. Vol. 130. P. 84–93. DOI: 10.1016/j.cageo.2019.05.009.
46. Jocher G., Chaurasia A., Stoken A., *et al.* ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. DOI: 10.5281/zenodo.7347926.
47. Zaidi S.S.A., Ansari M.S., Aslam A., *et al.* A survey of modern deep learning based object detection models // *Digital Signal Processing*. 2022. P. 103514. DOI: 10.1016/j.dsp.2022.103514.
48. Mo Y., Wu Y., Yang X., *et al.* Review the state-of-the-art technologies of semantic segmentation based on deep learning // *Neurocomputing*. 2022. Vol. 493. P. 626–646. DOI: 10.1016/j.neucom.2022.01.005.
49. Minaee S., Boykov Y.Y., Porikli F., *et al.* Image segmentation using deep learning: A survey // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021. Vol. 44, no. 7. P. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.
50. Yuan X., Shi J., Gu L. A review of deep learning methods for semantic segmentation of remote sensing imagery // *Expert Systems with Applications*. 2021. Vol. 169. P. 114417. DOI: 10.1016/j.eswa.2020.114417.
51. PapersWithCode.com. Semantic segmentation benchmarks. URL: <https://paperswithcode.com/task/semantic-segmentation> (accessed: 25.11.2022).

52. PapersWithCode.com. Real-time semantic segmentation benchmarks. URL: <https://paperswithcode.com/task/real-time-semantic-segmentation/latest> (accessed: 25.11.2022).
53. Carvalho O.L.F.d., Carvalho Junior O.A. de, Albuquerque A.O.d., *et al.* Instance segmentation for large, multi-channel remote sensing imagery using Mask-RCNN and a mosaicking approach // Remote Sensing. 2020. Vol. 13, no. 1. P. 39. DOI: 10.3390/rs13010039.
54. PapersWithCode.com. Instance segmentation benchmarks. URL: <https://paperswithcode.com/task/instance-segmentation> (accessed: 25.11.2022).
55. PapersWithCode.com. Real-time Instance Segmentation on MSCOCO. URL: <https://paperswithcode.com/sota/real-time-instance-segmentation-on-mscoco> (accessed: 25.11.2022).
56. Hossain S., Lee D.-j. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices // Sensors. 2019. Vol. 19, no. 15. P. 3371. DOI: 10.3390/s19153371.
57. Strudel R., Garcia R., Laptev I., Schmid C. Segmenter: Transformer for semantic segmentation // 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, October 10–17, 2021. IEEE, 2022. P. 7262–7272. DOI: 10.1109/ICCV48922.2021.00717.
58. Liu Z., Lin Y., Cao Y., *et al.* Swin transformer: Hierarchical vision transformer using shifted windows // 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, October 10–17, 2021. IEEE, 2022. P. 10012–10022. DOI: 10.1109/ICCV48922.2021.00986.
59. LeCun Y., Boser B., Denker J.S., *et al.* Backpropagation applied to handwritten zip code recognition // Neural computation. 1989. Vol. 1, no. 4. P. 541–551. DOI: 10.1162/neco.1989.1.4.541.
60. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE. 1998. Vol. 86, no. 11. P. 2278–2324. DOI: 10.1109/5.726791.
61. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge, MA, USA: MIT Press, 2016. 800 p. URL: <http://www.deeplearningbook.org>.
62. Zhang A., Lipton Z.C., Li M., Smola A.J. Dive into deep learning // Cambridge, UK: Cambridge University Press, 2023. URL: <https://D2L.ai>
63. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet classification with deep convolutional neural networks // Communications of the ACM. 2017. Vol. 60, no. 6. P. 84–90. DOI: 10.1145/3065386.
64. Alom M.Z., Taha T.M., Yakopcic C., *et al.* The history began from AlexNet: A comprehensive survey on deep learning approaches // CoRR. 2018. Vol. abs/1803.01164. arXiv: 1803.01164. URL: <https://arxiv.org/abs/1803.01164>.
65. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition // 3rd International Conference on Learning Representations, ICLR 2015, San

- Diego, CA, USA, May 7-9, 2015. Conference Track Proceedings // ed. by Y. Bengio, Y. LeCun. URL: <https://arxiv.org/abs/1409.1556>.
66. Szegedy C., Vanhoucke V., Ioffe S., *et al.* Rethinking the inception architecture for computer vision // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 27–30, 2016. IEEE, 2016. P. 2818–2826. DOI: 10.1109/CVPR.2016.308.
67. Lin M., Chen Q., Yan S. Network in network // 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014. Conference Track Proceedings / ed. by Y. Bengio, Y. LeCun. URL: <https://arxiv.org/abs/1312.4400>.
68. Chollet F. Xception: Deep learning with depthwise separable convolutions // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 21–26, 2017. IEEE, 2017. P. 1251–1258. DOI: 10.1109/CVPR.2017.195.
69. Szegedy C., Liu W., Jia Y., *et al.* Going deeper with convolutions // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, June 7–12, 2015. IEEE, 2015. P. 1–9. DOI: 10.1109/CVPR.2015.7298594.
70. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift // Proceedings of the 32nd International Conference on Machine Learning, ICML'15, Lille, France, July 7–9, 2015. Proceedings of Machine Learning Research. Vol. 37 / ed. by F. Bach, D. Blei. PMLR, 2015. P. 448–456. URL: <https://proceedings.mlr.press/v37/ioffe15.html>.
71. Kingma D.P., Ba J. Adam: A method for stochastic optimization // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015. Conference Track Proceedings / ed. by Y. Bengio, Y. LeCun. URL: <https://arxiv.org/abs/1412.6980>.
72. He K., Zhang X., Ren S., Sun J. Identity mappings in deep residual networks // European conference on computer vision – ECCV 2016, Amsterdam, The Netherlands, October 11–14, 2016. Proceedings, Part IV. Vol. 9908 / ed. by B. Leibe, J. Matas, N. Sebe, M. Welling. Springer, 2016. P. 630–645. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-46493-0_38.
73. PapersWithCode.com. Convolutional neural networks. URL: <https://paperswithcode.com/methods/category/convolutional-neural-networks> (accessed: 25.11.2022).
74. PapersWithCode.com. Most popular image models. URL: <https://paperswithcode.com/methods/category/image-models> (accessed: 25.11.2022).
75. Xie S., Girshick R., Dollár P., *et al.* Aggregated residual transformations for deep neural networks // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 21–26, 2017. IEEE, 2017. P. 5987–5995. DOI: 10.1109/CVPR.2017.634.
76. Huang G., Liu Z., Van Der Maaten L., Weinberger K.Q. Densely connected convolutional networks // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 21–26, 2017. IEEE, 2017. P. 2261–2269. DOI: 10.1109/CVPR.2017.243.
77. He T., Zhang Z., Zhang H., *et al.* Bag of tricks for image classification with convolutional neural networks // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition

- (CVPR), Long Beach, CA, USA, June 15–20, 2019. IEEE, 2020. P. 558–567. DOI: 10.1109/CVPR.2019.00065.
78. Kolesnikov A., Beyer L., Zhai X., *et al.* Big transfer (bit): General visual representation learning // European conference on computer vision – ECCV 2020, Glasgow, UK, August 23–28, 2020. Proceedings, Part V. Vol. 12350 / ed. by A. Vedaldi, H. Bischof, T. Brox, JM. Frahm. Springer, 2020. P. 491–507. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-58558-7_29.
79. Radosavovic I., Kosaraju R.P., Girshick R., *et al.* Designing network design spaces // 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, June 13–19, 2020. IEEE, 2020. P. 10425–10433. DOI: 10.1109/CVPR42600.2020.01044.
80. Sandler M., Howard A., Zhu M., *et al.* MobileNetV2: Inverted Residuals and Linear Bottlenecks // 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, June 18–23, 2018. IEEE, 2018. P. 4510–4520. DOI: 10.1109/CVPR.2018.00474.
81. Kyriakides G., Margaritis K. An introduction to neural architecture search for convolutional networks // CoRR. 2020. Vol. abs/2005.11074. arXiv: 2005.11074. URL: <https://arxiv.org/abs/2005.11074>.
82. He X., Zhao K., Chu X. AutoML: A survey of the state-of-the-art // Knowledge-Based Systems. 2021. Vol. 212. P. 106622. DOI: 10.1016/j.knosys.2020.106622.
83. Hu J., Shen L., Sun G. Squeeze-and-excitation networks // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 42, no. 8. P. 2011–2023. DOI: 10.1109/TPAMI.2019.2913372.
84. Tan M., Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks // Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, June 9–15, 2019. Proceedings of Machine Learning Research. Vol. 97 / ed. by K. Chaudhuri, R. Salakhutdinov. PMLR, 2019. P. 6105–6114. URL: <https://proceedings.mlr.press/v97/tan19a.html>.
85. Dosovitskiy A., Beyer L., Kolesnikov A., *et al.* An image is worth 16x16 words: Transformers for image recognition at scale // 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021. URL: <https://arxiv.org/abs/2010.11929>.
86. Khan S., Naseer M., Hayat M., *et al.* Transformers in vision: A survey // ACM Computing Surveys. 2021. Vol. 54, no. 10s. P. 1–41. DOI: 10.1145/3505244.
87. Vaswani A., Shazeer N., Parmar N., *et al.* Attention is all you need // 31st Annual Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, December 4–9, 2017. Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon, U. von Luxburg, S. Bengio, *et al.* Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
88. Ba J.L., Kiros J.R., Hinton G.E. Layer normalization // CoRR. 2016. Vol. abs/1607.06450. arXiv: 1607.06450. URL: <https://arxiv.org/abs/1607.06450>.
89. Dai Z., Liu H., Le Q.V., Tan M. Coatnet: Marrying convolution and attention for all data sizes // 35th Conference on Neural Information Processing Systems (NeurIPS

- 2021), Virtual, Online, December 6–14, 2021. Advances in Neural Information Processing Systems. Vol. 34 / ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, *et al.* Curran Associates, Inc., 2021. P. 3965–3977. URL: <https://proceedings.neurips.cc/paper/2021/hash/20568692db622456cc42a2e853ca21f8-Abstract.html>.
90. Tu Z., Talebi H., Zhang H., *et al.* MaxViT: Multi-axis Vision Transformer // 17th European conference on computer vision – ECCV 2022, Tel Aviv, Israel, October 23–27, 2022. Proceedings, Part XXIV. Vol. 13684 / ed. by S. Avidan, G. Brostow, M. Cissé, *et al.* Springer, 2022. P. 459–479. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-20053-3_27.
91. Mehta S., Rastegari M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer // The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022. URL: <https://arxiv.org/abs/2110.02178>.
92. Tolstikhin I.O., Houlsby N., Kolesnikov A., *et al.* MLP-Mixer: An all-MLP Architecture for Vision // 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Virtual, Online, December 6–14, 2021. Advances in Neural Information Processing Systems. Vol. 34 / ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, *et al.* Curran Associates, Inc., 2021. P. 24261–24272. URL: <https://proceedings.neurips.cc/paper/2021/hash/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Abstract.html>.
93. Touvron H., Cord M., Douze M., *et al.* Training data-efficient image transformers & distillation through attention // Proceedings of the 38th International Conference on Machine Learning, Virtual, July 18–24, 2021. Proceedings of Machine Learning Research. Vol. 139 / ed. by M. Meila, T. Zhang. PMLR, 2021. P. 10347–10357. URL: <https://proceedings.mlr.press/v139/touvron21a>.
94. Hospedales T.M., Antoniou A., Micaelli P., Storkey A.J. Meta-learning in neural networks: A survey // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021. Vol. 44, no. 9. P. 5149–5169. DOI: 10.1109/TPAMI.2021.3079209.
95. Naveed H. Survey: Image mixing and deleting for data augmentation // CoRR. 2023. Vol. abs/2106.07085. arXiv: 2106.07085. URL: <https://arxiv.org/abs/2106.07085>.
96. PapersWithCode.com. Image Classification on ImageNet. URL: <https://paperswithcode.com/sota/image-classification-on-imagenet> (accessed: 25.11.2022).
97. Bolya D., Zhou C., Xiao F., Lee Y.J. Yolact: Real-time instance segmentation // 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), October 27 – November 2, 2019. IEEE, 2020. P. 9157–9166. DOI: 10.1109/ICCV.2019.00925.
98. Cheng T., Wang X., Chen S., *et al.* Sparse Instance Activation for Real-Time Instance Segmentation // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, June 18–24, 2022. IEEE, 2022. P. 4433–4442. DOI: 10.1109/CVPR52688.2022.00439.
99. Wang C.-Y., Bochkovskiy A., Liao H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors // 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada, June 20–22, 2023. IEEE, 2023. P. 7464–7475.

100. Bolya D., Zhou C., Xiao F., Lee Y.J. Yolact++: Better real-time instance segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. Vol. 44, no. 2. P. 1108–1121. DOI: 10.1109/TPAMI.2020.3014297.
101. Wang X., Zhang R., Kong T., *et al.* Solov2: Dynamic and fast instance segmentation // Annual Conference on Neural Information Processing Systems (NeurIPS 2020), Virtual, December 6–12, 2020. Advances in Neural Information Processing Systems. Vol. 33 / ed. by H. Larochelle, and M. Ranzato, R. Hadsell, *et al.* Curran Associates, Inc., 2020. P. 17721–17732. URL: <https://proceedings.neurips.cc/paper/2020/hash/cd3afef9b8b89558cd56638c3631868a-Abstract.html>.
102. Wang X., Kong T., Shen C., *et al.* Solo: Segmenting objects by locations // European conference on computer vision – ECCV 2020, Glasgow, UK, August 23–28, 2020. Proceedings, Part V. Vol. 12350 / ed. by A. Vedaldi, H. Bischof, T. Brox, J.M. Frahm. Springer, 2020. P. 649–665. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-58523-5_38.
103. Li C., Li L., Jiang H., *et al.* YOLOv6: A single-stage object detection framework for industrial applications // CoRR. 2022. Vol. abs/2209.02976. arXiv: 2209.02976. URL: <https://arxiv.org/abs/2209.02976>.
104. Jocher G. Ultralytics YOLOv8. URL: <https://github.com/ultralytics/ultralytics> (accessed: 20.04.2023).
105. Diwan T., Anirudh G., Tembhurne J.V. Object detection using YOLO: Challenges, architectural successors, datasets and applications // Multimedia Tools and Applications. 2023. Vol. 82, P. 9243–9275. DOI: 10.1007/s11042-022-13644-y.
106. Jiang P., Ergu D., Liu F., *et al.* A Review of Yolo algorithm developments // Procedia Computer Science. 2022. Vol. 199. P. 1066–1073. DOI: 10.1016/j.procs.2022.01.135.
107. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, June 27–30, 2016. IEEE, 2016. P. 779–788. DOI: 10.1109/CVPR.2016.91.
108. Bochkovskiy A., Wang C.-Y., Liao H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection // CoRR. 2020. Vol. abs/2004.10934. arXiv: 2004.10934. URL: <https://arxiv.org/abs/2004.10934>.
109. Tan M., Le Q. EfficientNetv2: Smaller models and faster training // Proceedings of the 38th International Conference on Machine Learning, Virtual, July 18–24, 2021. Proceedings of Machine Learning Research. Vol. 139 / ed. by M. Meila, T. Zhang. PMLR, 2021. P. 10096–10106. URL: <http://proceedings.mlr.press/v139/tan21a.html>.
110. Brock A., De S., Smith S.L., Simonyan K. High-performance large-scale image recognition without normalization // Proceedings of the 38th International Conference on Machine Learning, Virtual, July 18–24, 2021. Proceedings of Machine Learning Research. Vol. 139 / ed. by M. Meila, T. Zhang. PMLR, 2021. P. 1059–1071. URL: <https://proceedings.mlr.press/v139/brock21a.html>.
111. PapersWithCode.com. Real-Time Object Detection. URL: <https://paperswithcode.com/task/real-time-object-detection> (accessed: 25.11.2022).

112. Chen K., Pang J., Wang J., *et al.* Hybrid Task Cascade for Instance Segmentation // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, June 15–20, 2019. IEEE, 2020. P. 4974–4983. DOI: 10.1109/CVPR.2019.00511.
113. Chen L.-C., Zhu Y., Papandreou G., *et al.* Encoder-decoder with atrous separable convolution for semantic image segmentation // European conference on computer vision – ECCV 2018, Munich, Germany, September 8–14, 2018. Proceedings, Part VII. Vol. 11211 / ed. by V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss. Springer, 2018. P. 801–818. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-01234-2_49.
114. Intel. Open VINO. 2023. URL: <https://docs.openvino.ai/2023.0/home.html> (accessed: 23.08.2023).
115. Georgia Tech and Facebook Artificial Intelligence Research. NNpack acceleration package for neural networks on multi-core CPUs. 2022. URL: <https://github.com/Maratyszczka/NNPACK> (accessed: 01.10.2022).
116. Intel. oneDNN Intel math kernel library for deep neural networks (Intel MKL-DNN) and deep neural network library (DNNL). 2022. URL: <https://github.com/oneapi-src/oneDNN> (accessed: 01.10.2022).
117. Hadjis S., Abuzaid F., Zhang C., Ré C. Caffe con Troll: Shallow Ideas to Speed Up Deep Learning // DanaC'15: Proceedings of the Fourth Workshop on Data analytics in the Cloud, Melbourne, Australia, May 31–June 4, 2015 / ed. by A. Katsifodimos. New York: ACM, 2015. P. 1–4. DOI: 10.1145/2799562.2799641.
118. Dai, J. J., Ding, D., Shi, D., *et al.* BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, June 18–24, 2022. IEEE, 2022. P. 21407–21414. DOI: 10.1109/CVPR52688.2022.02076.
119. Capra M., Bussolino B., Marchisio A., *et al.* An Updated Survey of Efficient Hardware Architectures for Accelerating Deep Convolutional Neural Networks // Future Internet. 2020. Vol. 12, no. 7. P. 113. DOI: 10.3390/fi12070113.
120. Dumas II J.D. Computer architecture: Fundamentals and principles of computer design. CRC Press, 2017. 447 p.
121. NVIDIA Corporation. Artificial Neural Network. 2022. URL: <https://developer.nvidia.com/discover/artificial-neural-network> (accessed: 01.10.2022).
122. NVIDIA Corporation. TensorRT SDK. 2023. URL: <https://developer.nvidia.com/tensorrt> (accessed: 08.23.2023).
123. Gholami A., Kim S., Dong Z., *et al.* A Survey of Quantization Methods for Efficient Neural Network Inference. Low-Power Computer Vision. New York: Chapman and Hall/CRC, 2022, P. 291–326. DOI: 10.1201/9781003162810-13.
124. Kikuchi Y., Fujita K., Ichimura T., *et al.* Calculation of Cross-correlation Function Accelerated by Tensor Cores with TensorFloat-32 Precision on Ampere GPU // 22nd International Conference – ICCS 2022, London, UK, June 11–23, 2022. Proceedings, Part II. Vol. 13351 / ed. by D. Groen, C. de Mulatier, M. Paszynski, *et al.* Springer, 2022. P. 277–290. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-08754-7_37.

125. Burel S., Evans A., Anghel L. Zero-Overhead Protection for CNN Weights // 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Athens, Greece, October 6–8, 2021. IEEE, 2021. P. 1–6. DOI: 10.1109/DFT52944.2021.9568363.
126. Simons T., Lee D.-J. A Review of Binarized Neural Networks // Electronics. 2019. Vol. 8, no. 6. P. 661. DOI: 10.3390/electronics8060661.
127. Wang Y., Feng B., Ding Y. QGTC: Accelerating Quantized Graph Neural Networks via GPU Tensor Core // PPOPP'22: Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Seoul, Republic of Korea, April 2–6, 2022. New York: ACM, 2022. P. 107–119. DOI: 10.1145/3503221.3508408.
128. Feng B., Wang Y., Geng T., *et al.* APNN-TC: Accelerating Arbitrary Precision Neural Networks on Ampere GPU Tensor Cores // SC '21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MO, USA, November 14–19, 2021. New York: ACM, 2021. P. 1–12. DOI: 10.1145/3458817.3476157.
129. Alemdar H., Leroy V., Prost-Boucle A., Pétrot F. Ternary neural networks for resource-efficient AI applications // 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, May 14–19, 2017. IEEE, 2017. P. 2547–2554. DOI: 10.1109/IJCNN.2017.7966166.
130. Nurvitadhi E., Venkatesh G., Sim J., *et al.* Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? // FPGA'17: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, February 22–24, 2017. New York: ACM, 2017. P. 5–14. DOI: 10.1145/3020078.3021740.
131. Li Z., Wallace E., Shen S., *et al.* Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers // Proceedings of the International Conference on Machine Learning, Virtual, July 13–18, 2020. Proceedings of Machine Learning Research. Vol. 119 / ed. by H. Daumé III, A. Singh. PMLR, 2020. P. 5958–5968. URL: <https://proceedings.mlr.press/v119/li20m.html>.
132. Qiu J., Wang J., Yao S., *et al.* Going Deeper with Embedded FPGA Platform for Convolutional Neural Network // FPGA'16: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, February 21–23, 2016. New York: ACM, 2016. P. 26–35. DOI: 10.1145/2847263.2847265.
133. Li C., Yang Y., Feng M., *et al.* Optimizing Memory Efficiency for Deep Convolutional Neural Networks on GPUs // SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, November 13–18, 2016. IEEE, 2016. P. 633–644. DOI: 10.1109/SC.2016.53.
134. NVIDIA Corporation. Mixed-Precision Programming with CUDA 8. 2016. URL: <https://developer.nvidia.com/blog/mixed-precision-programming-cuda-8/> (accessed: 01.10.2022).
135. Anzt H., Tsai Y.M., Abdelfattah A., *et al.* Evaluating the Performance of NVIDIA's A100 Ampere GPU for Sparse and Batched Computations // 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), Virtual, November 12, 2020. IEEE, 2021. P. 26–38. DOI: 10.1109/PMBS51919.2020.00009.

136. Tian R., Zhao Z., Liu W., *et al.* SAMP: A Toolkit for Model Inference with Self-Adaptive Mixed-Precision // CoRR. 2022. Vol. abs/2209.09130. arXiv: 2209.09130. URL: <https://arxiv.org/abs/2209.09130>.
137. Linux Foundation. Automatic Mixed Precision package - torch.amp. 2022. URL: <https://pytorch.org/docs/stable/amp.html> (accessed: 01.10.2022).
138. Honka T. Automatic Mixed Precision Quantization of Neural Networks using Iterative Correlation Coefficient Adaptation: PhD thesis / Honka Tapio. Tampere University, Finland, 2021. URL: <https://trepo.tuni.fi/handle/10024/135952>.
139. Liang T., Glossner J., Wang L., *et al.* Pruning and quantization for deep neural network acceleration: A survey // Neurocomputing. 2021. Vol. 461. P. 370–403. DOI: 10.1016/j.neucom.2021.07.045.
140. Wimmer P., Mehnert J., Condurache A.P. Dimensionality Reduced Training by Pruning and Freezing Parts of a Deep Neural Network, a Survey // Artificial Intelligence Review. 2023. DOI: 10.1007/s10462-023-10489-1.
141. Sun W., Li A., Geng T., *et al.* Dissecting Tensor Cores via Microbenchmarks: Latency, Throughput and Numerical Behaviors // IEEE Transactions on Parallel and Distributed Systems. 2023. Vol. 34, no. 1. P. 246–261. DOI: 10.1109/TPDS.2022.3217824.
142. Wang Y., Yang C., Farrell S., *et al.* Time-Based Roofline for Deep Learning Performance Analysis // 2020 IEEE/ACM Fourth Workshop on Deep Learning on Supercomputers (DLS), Atlanta, GA, USA, November 11, 2020. IEEE, 2020. P. 10–19. DOI: 10.1109/DLS51937.2020.00007.
143. Li Y., Liu Z., Xu K., *et al.* A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks // ACM Journal on Emerging Technologies in Computing Systems. 2018. Vol. 14, no. 2. P. 1–16. DOI: 10.1145/3154839.
144. Wu R., Guo X., Du J., Li J. Accelerating Neural Network Inference on FPGA-Based Platforms – A Survey // Electronics. 2021. Vol. 10, no. 9. P. 1025. DOI: 10.3390/electronics10091025.
145. Habib G., Qureshi S. Optimization and acceleration of convolutional neural networks: A survey // Journal of King Saud University — Computer and Information Sciences. 2022. Vol. 34, no. 7. P. 4244–4268. DOI: 10.1016/j.jksuci.2020.10.004.
146. Mittal S. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform // Journal of Systems Architecture. 2019. Vol. 97. P. 428–442. DOI: 10.1016/j.sysarc.2019.01.011.
147. Xu W., Zhang Y., Tang X. Parallelizing DNN Training on GPUs: Challenges and Opportunities // WWW'21: Companion Proceedings of the Web Conference 2021, Ljubljana, Slovenia, April 19–23, 2021. New York: ACM, 2021. P. 174–178. DOI: 10.1145/3442442.3452055.
148. Le Q.V., Ngiam J., Coates A., *et al.* On optimization methods for deep learning // ICML'11: Proceedings of the 28th International Conference on International Conference on Machine Learning, Bellevue, WA, USA, June 28 – July 2, 2011. Madison, WI, USA: Omnipress, 2011. P. 265–272. DOI: 10.5555/3104482.3104516.

149. Shamir O. Without-Replacement Sampling for Stochastic Gradient Methods // 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 5–10, 2016. Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee, M. Sugiyama, U. Luxburg, *et al.* Curran Associates, Inc., 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/c74d97b01eae257e44aa9d5bade97baf-Abstract.html>.
150. Wei J., Zhang X., Ji Z., *et al.* Deploying and scaling distributed parallel deep neural networks on the Tianhe-3 prototype system // Scientific Reports. 2021. Vol. 11. P. 20244. DOI: 10.1038/s41598-021-98794-z.
151. Smith S.L., Le Q.V. A Bayesian Perspective on Generalization and Stochastic Gradient Descent // 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018. Conference Track Proceedings. URL: <https://arxiv.org/abs/1710.06451>.
152. Que C., Zhang X. Efficient Scheduling in Training Deep Convolutional Networks at Large Scale // IEEE Access. 2018. Vol. 6. P. 61452–61456. DOI: 10.1109/ACCESS.2018.2875407.
153. Xiang S., Li H. On the Effects of Batch and Weight Normalization in Generative Adversarial Networks // CoRR. 2017. Vol. abs/2005.11074. arXiv: 1704.03971. URL: <https://arxiv.org/abs/1704.03971>.
154. Gitman I., Ginsburg B. Comparison of Batch Normalization and Weight Normalization Algorithms for the Large-scale Image Classification // CoRR. 2017. Vol. abs/1709.08145. arXiv: 1709.08145. URL: <https://arxiv.org/abs/1709.08145>.
155. Dukler Y., Gu Q., Montufar G. Optimization Theory for ReLU Neural Networks Trained with Normalization Layers // Proceedings of the International Conference on Machine Learning, Virtual, July 13–18, 2020. Proceedings of Machine Learning Research. Vol. 119 / ed. by H. Daumé III, A. Singh. PMLR, 2020. P. 2751–2760. URL: <https://proceedings.mlr.press/v119/dukler20a.html>.
156. Yu H., Yang S., Zhu S. Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning // Proceedings of the AAAI Conference on Artificial Intelligence AAAI-19, Honolulu, HI, USA, January 27 – February 1, 2019. Vol. 33, no. 1. Palo Alto, CA, USA: AAAI Press, 2019. P. 5693–5700. DOI: 10.1609/aaai.v33i01.33015693.
157. Xu J., Wang J., Qi Q., *et al.* Effective Scheduler for Distributed DNN Training Based on MapReduce and GPU Cluster // Journal of Grid Computing. 2021. Vol. 19. P. 8. DOI: 10.1007/s10723-021-09550-6.
158. Si T.N., Van Hung T., Ngoc D.V., Le Q.N. Using Stochastic Gradient Descent On Parallel Recommender System with Stream Data // 2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD), Danang, Vietnam, August 4–6, 2022. IEEE, 2022. P. 88–93. DOI: 10.1109/BCD54882.2022.9900664.
159. Sukanya J., Gandhi K.R., Palanisamy V. An assessment of machine learning algorithms for healthcare analysis based on improved MapReduce // Advances in Engineering Software. 2022. Vol. 173. P. 103285. DOI: 10.1016/j.advengsoft.2022.103285.

160. Asadianfam S., Shamsi M., Kenari A.R. TVD-MRDL: traffic violation detection system using MapReduce-based deep learning for large-scale data // *Multimedia Tools and Applications*. 2021. Vol. 80, no. 2. P. 2489–2516. DOI: 10.1007/s11042-020-09714-8.
161. Kul S., Sayar A. Sentiment Analysis Using Machine Learning and Deep Learning on Covid 19 Vaccine Twitter Data with Hadoop MapReduce // *Proceedings of the 6th International Conference on Smart City Applications (SCA2021)*, Virtual Safranbolu, Turkey, October 27–29, 2021. *Innovations in Smart Cities Applications Volume 5*. Vol. 393 / ed. by M. Ben Ahmed, A.A. Boudhir, I.R. Karas, *et al.* Springer, 2022. P. 859–868. *Lecture Notes in Computer Science*. DOI: 10.1007/978-3-030-94191-8_69.
162. Snir M., Otto S. W., Huss-Lederman S., *et al.* MPI: The complete reference: The MPI core. Cambridge, MA, USA: MIT Press, 1998. 427 p.
163. Thao Nguyen T., Wahib M., Takano R. Hierarchical Distributed-Memory Multi-Leader MPI-Allreduce for Deep Learning Workloads // *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, Takayama, Japan, November 27–30, 2018. IEEE, 2018. P. 216–222. DOI: 10.1109/CANDARW.2018.00048.
164. Awan A.A., Bédorf J., Chu C.-H., *et al.* Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation // *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Larnaca, Cyprus, May 14–17, 2019. IEEE, 2019. P. 498–507. DOI: 10.1109/CCGRID.2019.00064.
165. Bhagirath, Mittal N., Kumar S. Machine Learning Computation on Multiple GPU's using CUDA and Message Passing Interface // *2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC)*, Greater Noida, India, October 18–19, 2019. IEEE, 2020. P. 18–22. DOI: 10.1109/PEEIC47157.2019.8976714.
166. Ghazimirsaeed S.M., Anthony Q., Shafi A., *et al.* Accelerating GPU-based Machine Learning in Python using MPI Library: A Case Study with MVAPICH2-GDR // *2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*, Virtual, November 9–19, 2020. IEEE, 2021. P. 1–12. DOI: 10.1109/MLHPCAI4S51975.2020.00010.
167. Linux Foundation. Distributed Data Parallel - torch.nn. 2022. URL: <https://pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html> (accessed: 01.10.2022).
168. Jia Z., Zaharia M., Aiken A. Beyond Data and Model Parallelism for Deep Neural Networks // *Machine Learning and Systems (MLSys 2019)*, Stanford, CA, USA, March 31 – April 2, 2019. *Proceedings of Machine Learning and Systems*, Vol. 1 / ed. by A. Talwalkar, V. Smith, M. Zaharia. MLSYS, 2019. P. 1–13. URL: https://proceedings.mlsys.org/paper_files/paper/2019/hash/b422680f3db0986ddd7f8f126baaf0fa-Abstract.html.
169. Xu A., Huo Z., Huang H. On the Acceleration of Deep Learning Model Parallelism With Staleness // *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, June 13–19, 2020. IEEE, 2020. P. 2085–2094. DOI: 10.1109/CVPR42600.2020.00216.

170. Ericson L., Mbuva R. On the Performance of Network Parallel Training in Artificial Neural Networks // CoRR. 2017. Vol. abs/1701.05130. arXiv: 1701.05130. URL: <https://arxiv.org/abs/1701.05130>.
171. Chen C.-C., Yang C.-L., Cheng H.-Y. Efficient and Robust Parallel DNN Training through Model Parallelism on Multi-GPU Platform // CoRR. 2018. Vol. abs/1809.02839. arXiv: 1809.02839. URL: <https://arxiv.org/abs/1809.02839>.
172. Bruna J., Zaremba W., Szlam A., LeCun Y. Spectral Networks and Locally Connected Networks on Graphs // 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014. Conference Track Proceedings / ed. by Y. Bengio, Y. LeCun. URL: <http://arxiv.org/abs/1312.6203>.
173. Chen Y.-h., Moreno I.L., Sainath T., *et al.* Locally-connected and convolutional neural networks for small footprint speaker recognition // Proceedings of 16th Annual Conference of the International Speech Communication Association (INTERSPEECH-2015), Dresden, Germany, September 6–10, 2015. ISCA, 2015. P. 1136–1140. DOI: 10.21437/Interspeech.2015-297.
174. Wadekar S.N. Locally connected neural networks for image recognition: PhD thesis / Wadekar Shakti Nagnath. Purdue University Graduate School, West Lafayette, IN, USA, 2019. URL: https://hammer.purdue.edu/articles/thesis/LOCALLY_CONNECTED_NEURAL_NETWORKS_FOR_IMAGE_RECOGNITION/11328404/1.
175. Ankile L.L., Hegglund M.F., Krangle K. Deep Convolutional Neural Networks: A survey of the foundations, selected improvements, and some current applications // CoRR. 2020. Vol. abs/2011.12960. arXiv: 2011.12960. URL: <https://arxiv.org/abs/2011.12960>.
176. Coates A., Huval B., Wang T., *et al.* Deep learning with COTS HPC systems // Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, June 17–19, 2013. Proceedings of Machine Learning Research. Vol. 28, no. 3 / ed. by S. Dasgupta, D. McAllester. PMLR, 2013. P. 1337–1345. URL: <https://proceedings.mlr.press/v28/coates13.html>.
177. Gironés R.G., Salcedo A.M. Forward-backward parallelism in on-line backpropagation // International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Alicante, Spain, June 2–4, 1999. Proceedings, Volume II. Vol. 1607 / ed. by J. Mira, J.V. Sánchez-Andrés. Springer, 1999. P. 157–165. Lecture Notes in Computer Science. DOI: 10.1007/BFb0100482.
178. Xiao D., Yang C., Wu W. Efficient DNN training based on backpropagation parallelization // Computing. 2022. Vol. 104. P. 2431–2451. DOI: 10.1007/s00607-022-01094-1.
179. Zhang H., Dai Y., Li H., Koniusz P. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, June 15–20, 2019. IEEE, 2020. P. 5971–5979. DOI: 10.1109/CVPR.2019.00613.
180. Xiang Y., Kim H. Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference // 2019 IEEE Real-Time Systems Symposium (RTSS), Hong Kong, China, December 3–6, 2019. IEEE, 2020. P. 392–405. DOI: 10.1109/RTSS46320.2019.00042.

181. Shi S., Tang Z., Wang Q., *et al.* Layer-wise Adaptive Gradient Sparsification for Distributed Deep Learning with Convergence Guarantees // 24th European Conference on Artificial Intelligence (ECAI 2020), Santiago de Compostela, Spain, August 29 – September 8, 2020. Frontiers in Artificial Intelligence and Applications. Vol. 325 / ed. by G. De Giacomo, A. Catala, B. Dilkina, *et al.* Amsterdam, The Netherlands: IOS Press, 2020. P. 1467–1474. DOI: 10.3233/FAIA200253.
182. Harlap A., Narayanan D., Phanishayee A., *et al.* PipeDream: Fast and Efficient Pipeline Parallel DNN Training // CoRR. 2018. Vol. abs/1806.03377. arXiv: 1806.03377. URL: <https://arxiv.org/abs/1806.03377>.
183. Huang Y., Cheng Y., Bapna A., *et al.* GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism // Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, December 8–14, 2019. Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach, H. Larochelle, A. Beygelzimer, *et al.* Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/093f65e080a295f8076b1c5722a46aa2-Abstract.html>.
184. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks // Communications of the ACM. 2017. Vol. 60, no. 6. P. 84–90. DOI: 10.1145/3065386.
185. Sun P., Feng W., Han R., *et al.* Optimizing Network Performance for Distributed DNN Training on GPU Clusters: ImageNet/AlexNet Training in 1.5 Minutes // CoRR. 2019. Vol. abs/1902.06855. arXiv: 1902.06855. URL: <https://arxiv.org/abs/1902.06855>.
186. Gaunt A.L., Johnson M.A., Riechert M., *et al.* AMPNet: Asynchronous Model-Parallel Training for Dynamic Neural Networks // CoRR. 2017. Vol. abs/1705.09786. arXiv: 1705.09786. URL: <https://arxiv.org/abs/1705.09786>.
187. Dean J., Corrado G., Monga R., *et al.* Large Scale Distributed Deep Networks // Conference on Neural Information Processing Systems (NIPS 2012), Lake Tahoe, NV, USA, December 3–8, 2012. Advances in Neural Information Processing Systems. Vol. 25 / ed. by F. Pereira, C. Burges, L. Bottou, K. Weinberger. Curran Associates, Inc., 2012. URL: <https://proceedings.neurips.cc/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html>.
188. Lee S., Jha D., Agrawal A., *et al.* Parallel Deep Convolutional Neural Network Training by Exploiting the Overlapping of Computation and Communication // 2017 IEEE 24th International Conference on High Performance Computing (HiPC), Jaipur, India, December 18–21, 2017. IEEE, 2018. P. 183–192. DOI: 10.1109/HiPC.2017.00030.
189. Hu Y., Liu Y., Liu Z. A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC // 2022 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China, January 7–9, 2022. IEEE, 2022. P. 100–107. DOI: 10.1109/ICCRD54409.2022.9730377.
190. Kim J.-Y. Chapter Five - FPGA based neural network accelerators // Advances in Computers (Vol. 122): Hardware Accelerator Systems for Artificial Intelligence and Machine Learning. Elsevier, 2021. P. 135–165. DOI: 10.1016/bs.adcom.2020.11.002.

191. Mittal S., Vibhu. A survey of accelerator architectures for 3D convolution neural networks // Journal of Systems Architecture. 2021. Vol. 115. P. 102041. DOI: 10.1016/j.sysarc.2021.102041.
192. Omondi A.R., Rajapakse J.C. FPGA implementations of neural networks. Dordrecht, The Netherlands: Springer, 2006. 365 p.
193. Reagen B., Adolf R., Whatmough P., *et al.* Deep learning for computer architects. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2017. 123 p. DOI: 10.2200/S00783ED1V01Y201706CAC041.
194. Genc H., Kim S., Amid A., *et al.* Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration // 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, December 5–9, 2021. IEEE, 2021. P. 769–774. DOI: 10.1109/DAC18074.2021.9586216.
195. Ding W., Huang Z., Huang Z., *et al.* Designing efficient accelerator of depthwise separable convolutional neural network on FPGA // Journal of Systems Architecture. 2019. Vol. 97. P. 278–286. DOI: 10.1016/j.sysarc.2018.12.008.
196. Hu Y.H., Kung S.-Y. Systolic Arrays // Handbook of Signal Processing Systems / ed. by S.S. Bhattacharyya, E.F. Deprettere, R. Leupers, J. Takala. Cham: Springer International Publishing, 2019. P. 939–977. DOI: 10.1007/978-3-319-91734-4_26.
197. Liu Z.-G., Whatmough P.N., Mattina M. Systolic Tensor Array: An Efficient Structured-Sparse GEMM Accelerator for Mobile CNN Inference // IEEE Computer Architecture Letters. 2020. Vol. 19, no. 1. P. 34–37. DOI: 10.1109/LCA.2020.2979965.
198. Chen K.-C., Ebrahimi M., Wang T.-Y., Yang Y.-C. NoC-Based DNN Accelerator: A Future Design Paradigm // NOCS '19: Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip, New York, NY, USA, October 17–18, 2019. New York: ACM, 2019. P. 1–8. DOI: 10.1145/3313231.3352376.
199. Sun X., Choi J., Chen C.-Y., *et al.* Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks // Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada, December 8–14, 2019. Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach, H. Larochelle, A. Beygelzimer, *et al.* Curran Associates, Inc., 2019. P. 4900–4909. URL: <https://proceedings.neurips.cc/paper/2019/hash/65fc9fb4897a89789352e211ca2d398f-Abstract.html>.
200. Jia X., Song S., He W., *et al.* Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes // CoRR. 2018. Vol. abs/1807.11205. arXiv: 1807.11205. URL: <https://arxiv.org/abs/1807.11205>.
201. Yang J.A., Huang J., Park J., *et al.* Mixed-Precision Embedding Using a Cache // CoRR. 2020. Vol. abs/2010.11305. arXiv: 2010.11305. URL: <https://arxiv.org/abs/2010.11305>.
202. Courbariaux M., Hubara I., Soudry D., *et al.* Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1 // CoRR. 2016. Vol. abs/1602.02830. arXiv: 1602.02830. URL: <https://arxiv.org/abs/1602.02830>.
203. Nour B., Cherkaoui S. How Far Can We Go in Compute-less Networking: Computation Correctness and Accuracy // IEEE Network. 2022. Vol. 36, no. 4. P. 197–202. DOI: 10.1109/MNET.012.2100157.

Ронкин Михаил Владимирович, к.т.н., доцент, кафедра информационных технологий и систем управления, Уральский федеральный университет (Екатеринбург, Российская Федерация)

Акимова Елена Николаевна, д.ф.-м.н., профессор, кафедра информационных технологий и систем управления, Уральский федеральный университет; в.н.с., отдел некорректных задач анализа и приложений, Институт математики и механики им. Н.Н. Красовского Уральского отделения Российской академии наук (Екатеринбург, Российская Федерация)

Мисилов Владимир Евгеньевич, к.ф.-м.н., доцент, кафедра информационных технологий и систем управления, Уральский федеральный университет; н.с., отдел некорректных задач анализа и приложений, Институт математики и механики им. Н.Н. Красовского Уральского отделения Российской академии наук (Екатеринбург, Российская Федерация)

Решетников Кирилл Игоревич, аспирант, кафедра информационных технологий и систем управления, Уральский федеральный университет (Екатеринбург, Российская Федерация)

DOI: 10.14529/cmse230401

REVIEW ON APPLICATION OF DEEP NEURAL NETWORKS AND PARALLEL ARCHITECTURES FOR ROCK FRAGMENTATION PROBLEMS

© 2023 M.V. Ronkin¹, E.N. Akimova^{1,2}, V.E. Misilov^{1,2}, K.I. Reshetnikov¹

¹Ural Federal University (Mira Street 19, Ekaterinburg, 620002 Russia),

²Krasovskii Institute of Mathematics and Mechanics, Ural Branch of RAS
(S. Kovalevskaya Street 16, Ekaterinburg, 620108 Russia)

E-mail: m.v.ronkin@urfu.ru, aen15@yandex.ru, v.e.misilov@urfu.ru, ki.reshetnikov@urfu.ru

Received: 14.07.2023

Evaluation of mining productivity, including the determination of the geometric dimensions of rock objects in an open pit, is one of the most critical tasks in the mining industry. The problem of rock fragmentation is usually solved using computer vision methods such as instance segmentation or semantic segmentation. Today, deep learning neural networks are used to solve such problems for digital images. Neural networks require a lot of computing power to process high-resolution digital images and large datasets. To address this issue, in literature, lightweight architectural neural networks are proposed, as well as parallel computing using CPU, GPU, and specialized accelerators. The review discusses the latest advances in the field of deep learning neural networks for solving computer vision problems in relation to rock fragmentation and aspects of improving the performance of neural network implementations on various parallel architectures.

Keywords: computer vision, convolutional neural networks, deep learning, instance segmentation, semantic segmentation, object detection, parallel computing, mining industry problems, rock fragmentation.

FOR CITATION

Ronkin M.V., Akimova E.N., Misilov V.E., Reshetnikov K.I. Review on Application of Deep Neural Networks and Parallel Architectures for Rock Fragmentation Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 5–54. (in Russian) DOI: 10.14529/cmse230401.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

ИССЛЕДОВАНИЕ НЕЙРОСЕТЕВОГО МЕТОДА РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

© 2023 Н.А. Ольховский

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: olkhovskii@susu.ru

Поступила в редакцию: 20.10.2023

В статье исследован метод определения вектора движения по гиперплоскостям, ограничивающим допустимый многогранник многомерной задачи линейного программирования на основе визуальных образов, подаваемых на вход нейронной сети прямого распространения. Алгоритм визуализации строит в окрестности точки, расположенной на ограничивающей гиперплоскости, рецептивное поле. Для каждой точки рецептивного поля вычисляется скалярное смещение до поверхности гиперплоскости. На основании вычисленного смещения каждой точке рецептивного поля присваивается скалярная величина. Полученный визуальный образ подается на вход нейронной сети прямого распространения, которая вычисляет на ограничивающей гиперплоскости направление максимального увеличения целевой функции. В статье предложена усовершенствованная форма крестообразного рецептивного поля. Описано построение обучающего множества на основе случайно сгенерированных ограничивающих гиперплоскостей и целевых функций в многомерных пространствах. Разработана масштабируемая архитектура нейронной сети с изменяемым числом скрытых слоев. Произведен подбор гиперпараметров нейронной сети. В вычислительных экспериментах подтверждена высокая (более 98%) точность работы крестообразного рецептивного поля. Исследована зависимость точности результатов нейронной сети от числа скрытых слоев и продолжительности обучения.

Ключевые слова: линейное программирование, метод поверхностного движения, искусственная нейронная сеть, глубокое обучение.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Ольховский Н.А. Исследование нейросетевого метода решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 55–75. DOI: 10.14529/cmse230402.

Введение

Одной из фундаментальных задач современной прикладной математики является задача линейного программирования (ЛП) с большим числом параметров [1]. Оптимизационные модели, основанные на многопараметрической (многомерной) задаче ЛП встречаются в системах поддержки принятия решений в экономике [2, 3], в системах управления беспилотными летательными аппаратами [4], в управлении технологическими процессами [5–7], при построении логистических цепочек [8–10], в оперативном управлении и планировании [11, 12].

До сих пор одним из наиболее распространенных способов решения задач ЛП был класс алгоритмов, разработанных на основе симплекс-метода [13]. Было установлено, что симплекс-метод эффективен для решения большого класса задач ЛП. В частности, симплекс-метод эффективно использует преимущества любой гиперразреженности в задачах ЛП [14]. Однако симплекс-метод обладает некоторыми фундаментальными особенностями, которые ограничивают его использование для решения больших задач ЛП. Во-первых, в определенных случаях симплекс-метод должен выполнять итерации по всем вершинам симплекса, что соответствует экспоненциальной временной сложности [15–17]. Во-

вторых, в большинстве случаев симплекс-метод успешно решает задачи ЛП, содержащие до 50 000 переменных, однако при решении задач больших размерностей часто наблюдается потеря точности [18], которая не может быть компенсирована даже применением таких мощных вычислительных процедур, как «аффинное масштабирование» или «итеративное уточнение» [19]. В-третьих, в общем случае последовательный характер симплекс-метода затрудняет распараллеливание в многопроцессорных системах с распределенной памятью [20]. Были предприняты многочисленные попытки создать масштабируемую параллельную реализацию симплекс-метода, но все они оказались безуспешными [21]. Во всех случаях граница масштабируемости составляла от 16 до 32 процессорных узлов (см., например, [22]).

Хачиян доказал [23], используя вариант метода эллипсоидов (предложенный в 1970-х годах Шором [24], Юдиным и Немировским [25]), что задачи ЛП могут быть решены за полиномиальное время. Однако попытки применить этот подход на практике оказались безуспешными, поскольку в подавляющем большинстве случаев метод эллипсоида демонстрировал гораздо худшую эффективность по сравнению с симплекс-методом. Позже Кармаркар [26] предложил алгоритм внутренней точки с полиномиальным временем, который можно было использовать на практике. Этот алгоритм породил целую область современных методов внутренней точки [27], которые способны решать большие задачи ЛП с миллионами переменных и миллионами уравнений [28–32]. Более того, эти методы являются самокорректирующимися, а следовательно, обеспечивают высокую точность вычислений. Общим недостатком методов внутренней точки является необходимость найти некоторую допустимую точку, удовлетворяющую всем ограничениям задачи ЛП, перед началом вычислений. Нахождение такой внутренней точки может быть сведено к решению дополнительной задачи ЛП [33].

В работе [34] предложен новый подход к решению задачи ЛП, названный «метод поверхностного движения». В основе метода лежит идея о том, что точка максимального значения целевой функции (2) принадлежит границе допустимой области $\Gamma(M)$, и определить ее можно, двигаясь по поверхности многогранника постоянно в сторону максимального увеличения целевой функции. В общих чертах алгоритм поверхностного движения состоит из следующих шагов. В начале берется произвольная точка на границе допустимой области задачи ЛП. Сделать это можно, в частности, при помощи фейеровских отображений способом, описанным в статье [35]. Затем вычисляется вектор движения \mathbf{d} по поверхности допустимой области, соответствующий направлению максимального возрастания целевой функции. После чего происходит движение в заданном направлении до ограничивающего ребра. В этой точке вычисляется новый вектор \mathbf{d} , и указанные шаги повторяются до тех пор, пока вектор \mathbf{d} не окажется нулевым. Статья [34] содержит доказательство, что сделать это можно всегда за конечное число шагов. Проблема метода поверхностного движения состоит в том, что не представлен конструктивный способ нахождения вектора \mathbf{d} .

Решением проблемы с определением вектора \mathbf{d} может стать сочетание искусственных нейронных сетей (ИНС) с оригинальным методом визуализации многомерных задач ЛП, предложенным в [36]. Известно, что нейронные сети прямого распространения демонстрируют впечатляющие результаты в задачах, связанных с распознаванием сложных визуальных образов. С другой стороны, в [36] приведен хорошо масштабируемый параллельный алгоритм, позволяющий за конечное, заранее прогнозируемое время, строить визуальные образы, доступные для обработки нейронными сетями. В настоящей статье исследован подход, объединяющий технологию глубоких нейронных сетей с возможностями параллельных

вычислений для нахождения вектора \mathbf{d} , указывающего направление максимального увеличения целевой функции.

Статья имеет следующую структуру. В разделе 1 кратко изложены основные положения метода визуализации задач ЛП и метода поверхностного движения, формирующие теоретическую основу настоящего исследования. В разделе 2 описана новая структура рецептивного поля, получившая название «крестообразная». Раздел 3 описывает разработку архитектуры фундаментальной нейросетевой модели, вычисляющей вектор движения \mathbf{d} по гиперплоскостям, ограничивающим допустимую область. В разделе 4 приведены результаты вычислительных экспериментов. Заключение суммирует полученные результаты и рассматривает возможные направления дальнейших исследований.

1. Теоретические основы

Рассмотрим основные положения метода визуализации многомерных задач ЛП [36] и метода поверхностного движения [34]. В общем виде задачу линейного программирования можно представить следующим образом:

$$\bar{\mathbf{x}} = \arg \max \{ \langle \mathbf{c}, \mathbf{x} \rangle \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n \}, \quad (1)$$

где $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ и $\mathbf{c} \neq \mathbf{0}$. Причем ограничение $\mathbf{x} \geq \mathbf{0}$ входят в систему $A\mathbf{x} \leq \mathbf{b}$ в форме $-x_i \leq 0$ для всех $i \in \{1, 2, \dots, n\}$. Здесь вектором \mathbf{c} обозначен градиент целевой функции, максимум которой необходимо найти:

$$f(\mathbf{x}) = c_1x_1 + \dots + c_nx_n. \quad (2)$$

Обозначим через \mathcal{P} множество индексов, нумерующих строки матрицы A :

$$\mathcal{P} = \{1, \dots, m\}. \quad (3)$$

Пусть $\mathbf{a}_i \in \mathbb{R}^n$ обозначает вектор, представляющий i -тую строку матрицы A . Мы предполагаем, что $\mathbf{a}_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через \hat{H}_i замкнутое полупространство, определяемое неравенством $\langle \mathbf{a}_i, \mathbf{x} \rangle \leq b_i$, а через H_i — ограничивающую его гиперплоскость:

$$\hat{H}_i = \{ \mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{a}_i, \mathbf{x} \rangle \leq b_i \}; \quad (4)$$

$$H_i = \{ \mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{a}_i, \mathbf{x} \rangle = b_i \}. \quad (5)$$

Целевая проекция точки $\mathbf{z} \in \mathbb{R}^n$ на гиперплоскость H_i вычисляется по формуле

$$\gamma_i(\mathbf{z}) = \mathbf{z} + \beta_i(\mathbf{z})\mathbf{c}, \quad (6)$$

где

$$\beta_i(\mathbf{z}) = - \frac{\langle \mathbf{a}_i, \mathbf{z} \rangle - b_i}{\langle \mathbf{a}_i, \mathbf{c} \rangle} \|\mathbf{c}\|. \quad (7)$$

При этом гиперплоскость H_i не должна быть параллельной к вектору \mathbf{c} . Скалярная величина $\beta_i(\mathbf{z})$ называется целевым смещением точки \mathbf{z} относительно гиперплоскости H_i .

Определим допустимый многогранник

$$M = \bigcap_{i \in \mathcal{P}} \hat{H}_i, \quad (8)$$

представляющий множество допустимых точек задачи ЛП (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что $M \neq \emptyset$, то есть задача ЛП (1) имеет решение. Обозначим через $\Gamma(M)$ множество граничных точек многогранника M . Под граничной точкой множества $M \subset \mathbb{R}^n$ понимается точка в \mathbb{R}^n , для которой любая открытая ее окрестность в \mathbb{R}^n имеет непустое пересечение как с множеством M , так и с его дополнением.

Целевой гиперплоскостью $H_c(\mathbf{z})$, проходящей через точку \mathbf{z} , называется гиперплоскость, задаваемая формулой

$$H_c(\mathbf{z}) = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{c}, \mathbf{x} \rangle = \langle \mathbf{c}, \mathbf{z} \rangle\}. \quad (9)$$

Полупространство \hat{H}_i называется рецессивным [37], если

$$\langle \mathbf{a}_i, \mathbf{c} \rangle > 0. \quad (10)$$

Определим

$$\mathcal{I} = \{i \in \mathcal{P} \mid \langle \mathbf{a}_i, \mathbf{c} \rangle > 0\}, \quad (11)$$

то есть \mathcal{I} представляет множество индексов, для которых полупространство \hat{H}_i является рецессивным. Поскольку допустимый многогранник M представляет собой ограниченное множество, имеем

$$\mathcal{I} \neq \emptyset. \quad (12)$$

Определим рецессивный многогранник

$$\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i. \quad (13)$$

Очевидно, что \hat{M} является выпуклым, замкнутым, неограниченным множеством. Из (8) и (11) следует

$$M \subset \hat{M}. \quad (14)$$

Обозначим через $\Gamma(\hat{M})$ множество граничных точек рецессивного многогранника \hat{M} . Согласно утверждению 3 в [37] имеем

$$\bar{\mathbf{x}} \in \Gamma(\hat{M}), \quad (15)$$

то есть решение задачи ЛП (1) лежит на границе рецессивного многогранника \hat{M} .

Целевая проекция $\hat{\gamma}(\mathbf{z})$ точки $\mathbf{z} \in \mathbb{R}^n$ на границу $\Gamma(\hat{M})$ рецессивного многогранника \hat{M} вычисляется по формуле

$$\hat{\gamma}(\mathbf{z}) = \mathbf{z} + \hat{\beta}(\mathbf{z})\mathbf{c}, \quad (16)$$

где $\hat{\beta}(\mathbf{z}) = \min \{\beta_i(\mathbf{z}) \mid i \in \mathcal{I}\}$. Скалярная величина $\hat{\beta}(\mathbf{z})$ называется целевым смещением точки \mathbf{z} относительно границы рецессивного многогранника. На основе целевых смещений строится цифровой образ задачи ЛП.

Метод поверхностного движения, описанный в [34], строит на поверхности допустимого многогранника путь из произвольной граничной точки $\mathbf{u}^{(0)} \in \hat{M} \cap \Gamma(M)$ до точки $\bar{\mathbf{x}}$, являющейся решением задачи ЛП (1). Перемещение по поверхности рецессивного многогранника происходит в направлении наибольшего увеличения целевой функции. Реализация метода поверхностного движения приведена в виде алгоритма 1 в [34].

Ключевым пунктом алгоритма является построение вектора движения \mathbf{d} для текущего приближения $\mathbf{u}^{(k)} \in \hat{M} \cap \Gamma(M)$. Рассмотрим его подробнее. Сначала строится n -мерный

диск D , являющийся пересечением целевой гиперплоскости $H_c(\mathbf{u}^{(k)})$, проходящей через точку $\mathbf{u}^{(k)}$, и n -мерного шара $V_r(\mathbf{u}^{(k)})$ малого радиуса r с центром в точке $\mathbf{u}^{(k)}$.

$$D = H_c(\mathbf{u}^{(k)}) \cap V_r(\mathbf{u}^{(k)}). \quad (17)$$

На гипердиске вычисляется точка \mathbf{v} с максимальным смещением относительно границы рецессивного многогранника \hat{M} .

$$\mathbf{v} = \arg \max\{\hat{\beta}(\mathbf{x}) \mid \mathbf{x} \in D\}. \quad (18)$$

Далее вычисляется точка \mathbf{w} , являющаяся целевой проекцией \mathbf{v} на границу рецессивного многогранника.

$$\mathbf{w} = \hat{\gamma}(\mathbf{v}). \quad (19)$$

Отметим, что, согласно утверждению 4 из [34], мы всегда можем подобрать такой радиус r для $V_r(\mathbf{u}^{(k)})$, чтобы исходная точка $\mathbf{u}^{(k)}$ и целевая проекция \mathbf{w} принадлежали одной и той же гиперплоскости. Вектор \mathbf{d} движения по поверхности многогранника M , соответствующий направлению наибольшего увеличения целевой функции, определяется как разность целевой проекции \mathbf{w} и точки $\mathbf{u}^{(k)}$.

$$\mathbf{d} = \mathbf{w} - \mathbf{u}^{(k)}. \quad (20)$$

Для практической реализации метода поверхностного движения необходим эффективный способ вычисления вектора \mathbf{d} . Идея эффективного решения этой проблемы, предложенная в работе [36], состоит в следующем: построить на суперкомпьютере визуальный образ окрестности точки $\mathbf{u}^{(k)}$ и передать его на вход глубокой нейронной сети, обученной выдавать координаты вектора \mathbf{d} .

Для построения визуального образа задачи ЛП в [36] строится гиперкубическое рецептивное поля. Гиперкубическое рецептивное поле $\mathfrak{G}_{\text{cube}}(\mathbf{z}, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $\mathbf{z} \in H_c$ и рангом $\eta \in \mathbb{N}$ представляет собой множество точек, являющихся узлами гиперкубической решетки размерности $n - 1$ с фиксированным расстоянием δ между узлами, которая имеет 2η ячеек по каждому измерению. Общее количество точек в гиперкубическом рецептивном поле вычисляется по формуле

$$K_{\text{cube}} = (2\eta + 1)^{n-1}. \quad (21)$$

Пример гиперкубического рецептивного поля в пространстве \mathbb{R}^3 приведен на рис. 1а.

Образом $\mathcal{I}(\mathbf{z}, \eta, \delta)$, порожденным рецептивным полем $\mathfrak{G}(\mathbf{z}, \eta, \delta)$, является упорядоченное множество вещественных чисел

$$\mathcal{I}(\mathbf{z}, \eta, \delta) = \left\{ \hat{\beta}(\mathbf{g}) \mid \mathbf{g} \in \mathfrak{G}(\mathbf{z}, \eta, \delta) \right\}. \quad (22)$$

Полученный образ подается на вход глубокой нейронной сети (DNN). Правильно DNN вычисляет координаты вектора \mathbf{d} .

$$\mathbf{d} = \text{DNN} \left(\mathcal{I}(\mathbf{u}^{(k)}, \eta, \delta) \right). \quad (23)$$

Основная проблема гиперкубического рецептивного поля состоит в том, что с увеличением размерности пространства, число точек рецептивного поля в соответствии с формулой (21)

Утверждение 4 из [36] показывает, что линейное подпространство размерности $(n - 1)$, порожденное ортогональными векторами c_1, \dots, c_{n-1} , является гиперплоскостью, параллельной гиперплоскости H_c . Обозначим через E_c следующий ортонормированный базис:

$$E_c = \left\{ e^{(i)} = \frac{c^{(i)}}{\|c\|} \mid i \in \{1, \dots, n - 1\} \right\}. \quad (26)$$

Определение 1. Крестообразным рецептивным полем $\mathfrak{G}_{\text{cross}}(z, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $z \in H_c$ и рангом $\eta \in \mathbb{N}$ будем называть конечное упорядоченное множество точек, расположенных на осях ортонормированного базиса E_c с центром координат в точке z , определяющего линейное многообразие H_c , с фиксированным расстоянием δ между соседними точками. На каждой полуоси базиса располагается η точек. Центр координат также включается в крестообразное рецептивное поле.

Крестообразное рецептивное поле может быть построено с помощью алгоритма 1. Прокомментируем его шаги. На шаге 1 создается пустое множество точек \mathfrak{G} . На шагах 2–19 цикл **for** на каждой итерации заполняет точками одну ось базиса E_c . На шагах 3–18 цикл **for** создает одну точку. На шаге 4 инициализируется точка s , все координаты которой заполнены нулями. Затем цикл **for** на шагах 5–16 вычисляет по очереди каждую координату s_j в базисе E_c и сохраняет ее в s . Если номер координаты не совпадает с номером обрабатываемой оси, то такая координата равна 0 (шаги 6–7). Иначе s_j присваивается координата по оси $e^{(j)}$ (шаги 8–14). Если точка расположена на отрицательной полуоси, то координате по очереди присваиваются значения $\{-\eta, \dots, -1\}$ (шаги 9–10); на положительной полуоси координате присваиваются значения $\{1, \dots, \eta\}$ (шаги 11–13). На шаге 15 вычисленная координата добавляется к s . По окончании записи всех координат в s , на шаге 17 новая точка со сдвигом z добавляется в множество точек рецептивного поля \mathfrak{G} . После того, как множество точек сформировано, на шаге 20 к нему добавляется центральная точка поля. Шаг 21 завершает работу алгоритма.

Пример крестообразного рецептивного поля в пространстве \mathbb{R}^3 приведен на рис. 16. Общее количество точек крестообразного поля вычисляется по формуле

$$K_{\text{cross}} = 2\eta(n - 1) + 1. \quad (27)$$

Действительно, алгоритм 1 в цикле **for** на шагах 2–19 строит точки на каждой из $n - 1$ осей, задаваемых векторами из E_c . На каждой оси во вложенном цикле **for** (шаги 3–18) от нулевой точки строится η точек в положительном и η точек в отрицательном направлениях. Всего получается $2\eta(n - 1)$ точек. В завершение к получившемуся множеству точек на шаге 20 добавляется центральная точка. В сумме получаем $2\eta(n - 1) + 1$.

Поскольку хранить постоянно весь массив точек нецелесообразно, на практике координаты точки можно вычислять динамически по ее номеру. Именно этот подход используется при построении образа рецептивного поля в алгоритме 2.

Дадим краткие комментарии по шагам этого алгоритма. Шаг 1 инициализирует пустое множество \mathfrak{J} , которое будет содержать целевые смещения точек рецептивного поля. Затем цикл **for** перебирает (шаги 2–19) все точки рецептивного поля, кроме центральной. На шаге 3 вычисляется номер оси, на которой расположена точка. На шаге 4 вычисляется номер точки на оси. На шагах 5–10 вычисляются координаты точки. На шаге 5 в качестве исходных принимаются координаты центральной точки рецептивного поля. В зависимости от того, расположена вычисляемая точка на отрицательной или положительной полуоси (шаг 6), к

Алгоритм 1 Построение крестообразного рецептивного поля $\mathfrak{G}_{\text{cross}}(z, \eta, \delta)$

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```

1:  $\mathfrak{G} := \emptyset$ 
2: for  $t = 1 \dots n - 1$  do
3:   for  $i = 1 \dots 2\eta$  do
4:      $s := \mathbf{0}$ 
5:     for  $j = 1 \dots n - 1$  do
6:       if  $j \neq t$  then
7:          $s_j = 0$ 
8:       else
9:         if  $i \leq \eta$  then
10:           $s_j := (i - \eta - 1)\delta$ 
11:        else
12:           $s_j := (i - \eta)\delta$ 
13:        end if
14:      end if
15:       $s := s + s_j e^{(j)}$ 
16:    end for
17:     $\mathfrak{G} := \mathfrak{G} \cup \{s + z\}$ 
18:  end for
19: end for
20:  $\mathfrak{G} := \mathfrak{G} \cup \{z\}$ 
21: stop

```

исходным координатам добавляется перемещение в отрицательную сторону (шаг 7), либо в положительную (шаг 9). Центральная точка на этом этапе не обрабатывается, так как она для всех осей общая. Далее на шагах 11–18 с помощью формулы (16) вычисляется целевое смещение полученной точки рецептивного поля относительно границы рецессивного многогранника. После вычисления всех точек рецептивного поля, кроме центральной, на шаге 20 к образу добавляется еще одно смещение, равное нулю, так как центральная точка рецептивного поля всегда располагается на поверхности допустимого многогранника. Следующее утверждение дает оценку временной сложности описанного алгоритма.

Утверждение 1. Алгоритм 2 имеет временную сложностью $O(n^2m)$.

Доказательство. Рассмотрим алгоритм 2. Шаги 5, 7, 9, 11 и 13 имеет временную сложность $O(n)$. Остальные шаги, за исключением операторов цикла **for**, имеют вычислительную сложность $O(1)$. Количество итераций внутреннего цикла **for** (шаг 12) можно оценить как $O(m)$. Следовательно, шаги 12–17 имеют суммарную временную сложность $O(nm)$. Количество итераций внешнего цикла **for** (шаг 2) может быть оценено как $O(n)$. Таким образом, временная сложность алгоритма 2 в целом может быть оценена как $O(n^2m)$. \square

Алгоритм 2 Построение крестообразного образа $\mathcal{J}_{\text{cross}}(\mathbf{z}, \eta, \delta)$

Require: $\mathbf{z} \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```

1:  $\mathcal{J} := \emptyset$ 
2: for  $k = 1 \dots 2\eta(n-1)$  do
3:    $l := \lfloor (k-1)/2\eta \rfloor + 1$ 
4:    $p := (k-1) \bmod 2\eta + 1$ 
5:    $\mathbf{g} := \mathbf{z}$ 
6:   if  $p \leq \eta$  then
7:      $\mathbf{g} := \mathbf{g} + (p - \eta - 1)\delta \mathbf{e}^{(l)}$ 
8:   else
9:      $\mathbf{g} := \mathbf{g} + (p - \eta)\delta \mathbf{e}^{(l)}$ 
10:  end if
11:   $\hat{\beta} := -\frac{\langle \mathbf{a}_1, \mathbf{g} \rangle - b_1}{\langle \mathbf{a}_1, \mathbf{c} \rangle} \|\mathbf{c}\|$ 
12:  for  $i = 2 \dots m$  do
13:     $\beta_i := -\frac{\langle \mathbf{a}_i, \mathbf{g} \rangle - b_i}{\langle \mathbf{a}_i, \mathbf{c} \rangle} \|\mathbf{c}\|$ 
14:    if  $\beta_i \leq \hat{\beta}$  then
15:       $\hat{\beta} := \beta_i$ 
16:    end if
17:  end for
18:   $\mathcal{J} := \mathcal{J} \cup \{\hat{\beta}\}$ 
19: end for
20:  $\mathcal{J} := \mathcal{J} \cup \{0\}$ 
21: stop

```

3. Построение нейронной сети для движения по гиперплоскости

Пусть в \mathbb{R}^n заданы случайный вектор \mathbf{a} и гиперплоскость $H_{\mathbf{a}}$, ортогональная вектору \mathbf{a} . Без ограничения общности мы можем полагать, что гиперплоскость $H_{\mathbf{a}}$ проходит через нулевой вектор:

$$H_{\mathbf{a}} = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{a}, \mathbf{x} \rangle = 0\}. \quad (28)$$

Зададим произвольный вектор \mathbf{c} такой, что

$$\langle \mathbf{a}, \mathbf{c} \rangle > 0. \quad (29)$$

Гиперплоскость $H_{\mathbf{a}}$ представляет собой случайную грань некоторого допустимого многогранника M , а вектор \mathbf{c} — градиент случайной целевой функции. При этом полупространство $\hat{H}_{\mathbf{a}}$, задаваемое формулой

$$\hat{H}_{\mathbf{a}} = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{a}, \mathbf{x} \rangle \leq 0\}, \quad (30)$$

является рецессивным по отношению к вектору \mathbf{c} . Очевидно, что в контексте задачи ЛП $M \subset \hat{H}_{\mathbf{a}}$ и $H_{\mathbf{a}} \cap \Gamma(M) \neq \emptyset$.

Для построения цифрового образа гиперплоскости $H_{\mathbf{a}}$, используем целевую гиперплоскость в точке $\mathbf{0}$:

$$H_{\mathbf{c}}(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{c}, \mathbf{x} \rangle = \mathbf{0}\}. \quad (31)$$

Воспользовавшись (24) и (26), сформируем базис $E_{\mathbf{c}} \subset H_{\mathbf{c}}(\mathbf{0})$. С помощью алгоритма 4 из [36] построим гиперкубический образ $\mathcal{J}_{\text{cube}}(\mathbf{0}, \eta, \delta)$, и с помощью алгоритма 2, описанного выше, построим крестообразный образ $\mathcal{J}_{\text{cross}}(\mathbf{0}, \eta, \delta)$. Далее мы будем обозначать эти образы как $\mathcal{J}_{\text{cube}}$ и $\mathcal{J}_{\text{cross}}$. С помощью функции

$$\varphi_{\mathcal{J}}(\rho) = 511(\rho - \min(\mathcal{J})) / (\max(\mathcal{J}) - \min(\mathcal{J})) - 256 \quad (32)$$

выполним нормализацию обоих образов:

$$\bar{\mathcal{J}}_{\text{cube}} = \{\varphi_{\mathcal{J}_{\text{cube}}}(\rho) \mid \rho \in \mathcal{J}_{\text{cube}}\}, \quad (33)$$

$$\bar{\mathcal{J}}_{\text{cross}} = \{\varphi_{\mathcal{J}_{\text{cross}}}(\rho) \mid \rho \in \mathcal{J}_{\text{cross}}\}. \quad (34)$$

Для построения обучающего множества необходимо сопоставить нормализованному образу правильный вектор движения $\mathbf{d}_{\mathbf{a}}$ по гиперплоскости $H_{\mathbf{a}}$. Обозначим через $\pi_{\mathbf{a}}(\mathbf{c})$ ортогональную проекцию вектора \mathbf{c} на гиперплоскость $H_{\mathbf{a}}$:

$$\pi_{\mathbf{a}}(\mathbf{c}) = \mathbf{c} - \frac{\langle \mathbf{a}, \mathbf{c} \rangle}{\|\mathbf{a}\|^2} \mathbf{a}. \quad (35)$$

Очевидно, что ортогональная проекция $\pi_{\mathbf{a}}(\mathbf{c})$ указывает направление максимального увеличения значения целевой функции на гиперплоскости $H_{\mathbf{a}}$. Таким образом

$$\mathbf{d}_{\mathbf{a}} = \pi_{\mathbf{a}}(\mathbf{c}). \quad (36)$$

В качестве правильного ответа вместо n координат вектора $\mathbf{d}_{\mathbf{a}}$ в \mathbb{R}^n будем указывать $n - 1$ координату в базисе $E_{\mathbf{c}}$. Обозначим через $\mathbf{g}_{\mathbf{a}} \in H_{\mathbf{c}}$ ортогональную проекцию вектора $\mathbf{d}_{\mathbf{a}}$ на гиперплоскость $H_{\mathbf{c}}$:

$$\mathbf{g}_{\mathbf{a}} = \mathbf{d}_{\mathbf{a}} - \frac{\langle \mathbf{c}, \mathbf{d}_{\mathbf{a}} \rangle}{\|\mathbf{c}\|^2} \mathbf{c}. \quad (37)$$

Используя (36) и (35), данную формулу можно преобразовать к виду

$$\mathbf{g}_{\mathbf{a}} = \frac{\langle \mathbf{c}, \mathbf{a} \rangle}{\|\mathbf{c}\|^2} \mathbf{c} - \mathbf{a}. \quad (38)$$

Коэффициентом угла наклона вектора $\mathbf{g}_{\mathbf{a}}$ к базисному вектору $\mathbf{e}^{(i)}$ назовем косинус угла между векторами $\mathbf{g}_{\mathbf{a}}$ и $\mathbf{e}^{(i)}$:

$$\cos \alpha_i = \frac{\langle \mathbf{e}^{(i)}, \mathbf{g}_{\mathbf{a}} \rangle}{\|\mathbf{g}_{\mathbf{a}}\|}. \quad (39)$$

Взятые вместе, коэффициенты углов наклона образуют вектор правильного ответа

$$\mathbf{y}_{\mathbf{a}} = \left(\frac{\langle \mathbf{e}^{(1)}, \mathbf{g}_{\mathbf{a}} \rangle}{\|\mathbf{g}_{\mathbf{a}}\|}, \dots, \frac{\langle \mathbf{e}^{(n-1)}, \mathbf{g}_{\mathbf{a}} \rangle}{\|\mathbf{g}_{\mathbf{a}}\|} \right). \quad (40)$$

Для настройки оптимальных гиперпараметров нейронной сети была разработана гипермодель, представленная на рис. 2. Эта гипермодель включает входной слой, блок скрытых

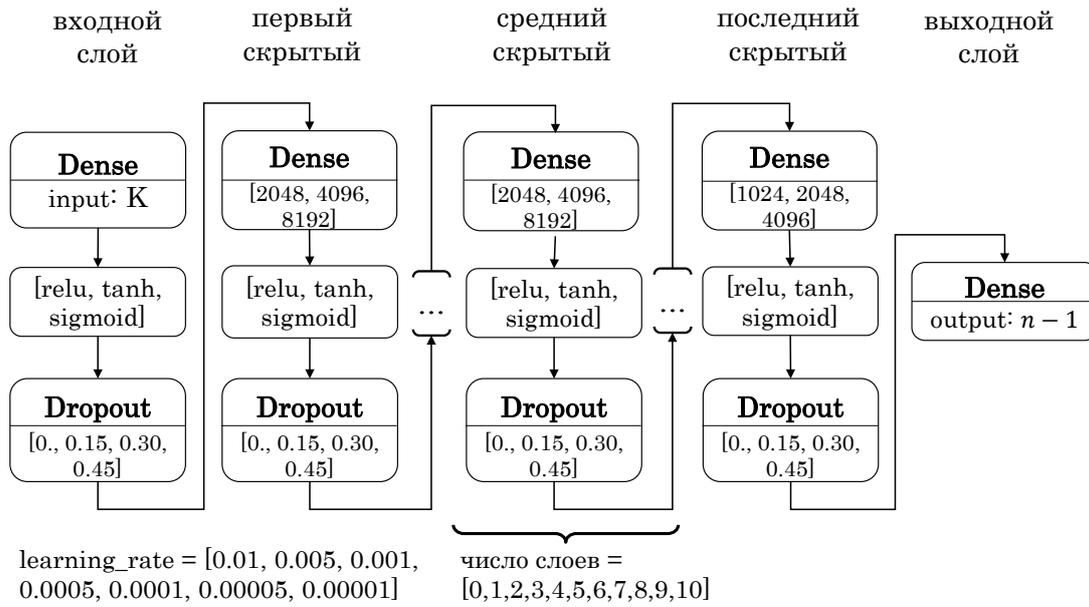


Рис. 2. Архитектура гипермодели

Таблица 1. Параметры рецептивного поля

Параметр	Семантика	Значение
n	размерность пространства	10
η	ранг рецептивного поля	5
δ	расстояние между соседними точками	1
K_{cube}	число точек рецептивного поля	91

слоев и выходной слой. Блок скрытых слоев состоит из первого скрытого слоя, переменного числа средних скрытых слоев и последнего скрытого слоя. Все слои являются полностью связанными (dense connection). Выбор функции активации для всех слоев осуществлялся из набора $\{ReLU, sigmoid, tanh\}$. Входной слой имеет K нейронов, соответствующих точкам используемого рецептивного поля. Выходной слой имеет $n - 1$ нейронов, соответствующих числу коэффициентов углов наклона вектора \mathbf{g}_a .

Подбор гиперпараметров выполнялся на обучающем множестве, сгенерированном в пространстве \mathbb{R}^{10} с конфигурацией рецептивного поля, представленной в табл. 1. Для генерации случайных координат векторов \mathbf{a} и \mathbf{c} использовалось стандартное нормальное распределение. Число нейронов для скрытых слоев подбиралось из набора $\{1024, 2048, 4096, 8192\}$. Количество средних скрытых слоев подбиралось из набора $\{0, 1, \dots, 8\}$. Заметим, что максимально возможное число скрытых слоев при таком выборе параметров равнялось 10, что соответствует заданной размерности пространства $n = 10$.

Основываясь на представленной гипермодели, был выполнен байесовский поиск оптимального набора гиперпараметров с использованием платформы W&B [38]. При обучении был использован оптимизатор *RMSProp* [39]. Размер блока обучающих прецедентов (batch size) равнялся 128. В качестве функции потерь (loss) была использована косинусная мера (cosine similarity)

$$CS = - \frac{\sum_{k=1}^{n-1} (\mathbf{a}_k \cdot \mathbf{y}_k)}{\sqrt{\sum_{k=1}^{n-1} \mathbf{a}_k^2} \cdot \sqrt{\sum_{k=1}^{n-1} \mathbf{y}_k^2}}. \quad (41)$$

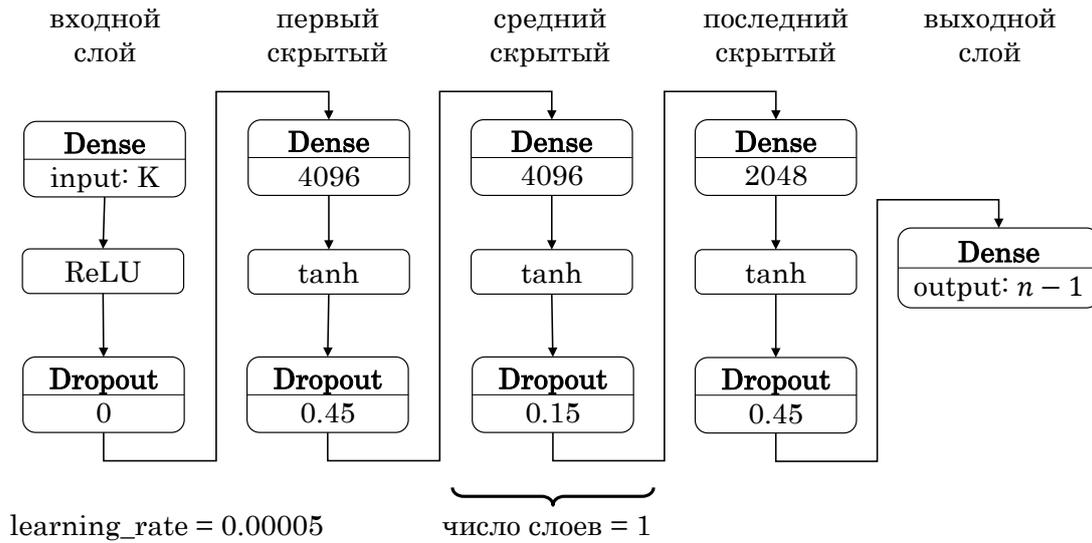


Рис. 3. Архитектура нейронной сети

Здесь α_k — это значения коэффициентов наклона, предсказанные нейронной сетью, y_k — коэффициенты наклона, рассчитанные на основе формулы (40). Набор обучающих данных из 100 000 прецедентов был разделен следующим образом:

- обучающая выборка: 80000 элементов;
- тестовая выборка: 15000 элементов;
- валидационная выборка: 5000 элементов.

Для обучения и тестирования нейронных сетей использовался комплекс «Нейрокомпьютер» Южно-Уральского государственного университета [40], оборудованный графическими процессорами nVidia Tesla V100. Обучение производилось при помощи библиотек `keras` и `TensorFlow`. В результате была получена глубокая ИНС, архитектура которой представлена на рис. 3. Для оценки качества работы нейронных сетей была использована оригинальная модификация средней абсолютной ошибки, получившая название «средняя абсолютная нормализованная ошибка» MANE (mean absolute normalized error):

$$\text{MANE} = \frac{1}{n} \cdot \sum_{i=1}^n \left| \frac{y_i}{\|\mathbf{y}\|} - \frac{\alpha_i}{\|\boldsymbol{\alpha}\|} \right|. \quad (42)$$

Создание обучающего множества и обучение искусственной нейронной сети производилось по схеме, представленной на рис. 4. Сначала программа генерации случайных гиперплоскостей, приняв в качестве входного параметра число Q , генерирует Q пар $\{\mathbf{a}, \mathbf{c}\}$, где каждая пара состоит из случайного вектора \mathbf{a} , определяющего гиперплоскость (28) и случайного вектора целевой функции \mathbf{c} . Массив сгенерированных данных помещается в файл, передаваемый программе визуализации. Визуализатор принимает входные параметры, определяющие положение точек рецептивного поля:

- ранг рецептивного поля η ;
- плотность рецептивного поля δ ;
- форма рецептивного поля `cube` (гиперкубическое) или `cross` (крестообразное).

В результате работы визуализатора формируется файл прецедентов, содержащий Q строк. В каждой строке записаны K нормализованных по формуле (32) коэффициентов, форми-

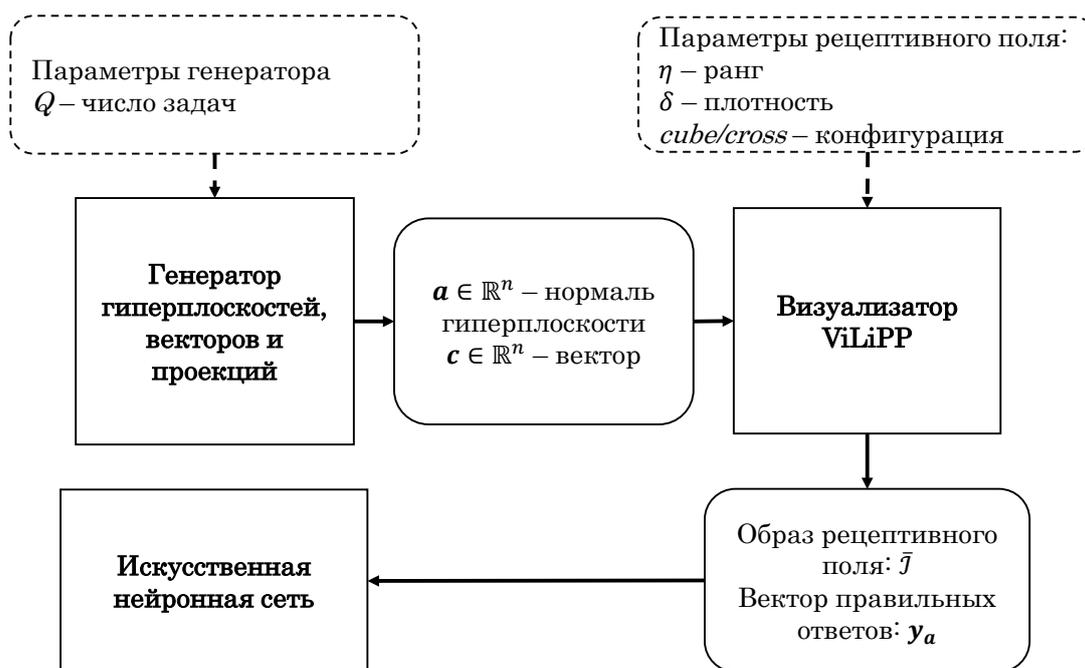


Рис. 4. Архитектура программного комплекса для построения обучающего множества

рующих образ \tilde{J} , и $n - 1$ коэффициентов угла наклона, формирующих вектор правильного ответа y_a .

4. Вычислительные эксперименты

В первой серии экспериментов исследовалось влияние конфигурации и числа точек рецептивного поля на точность работы ИНС. На основе архитектуры, представленной на рис. 3, были построены 10 нейронных сетей, отличающихся числом входных нейронов. Число входных нейронов соответствовало гиперкубической и крестообразной конфигурациям рецептивного поля. Для каждой конфигурации перебиралось значение ранга η от 1 до 5. Результаты представлены на рис. 5. На графиках видно, что крестообразное рецептивное поле незначительно уступает в точности гиперкубическому. При этом для размерности $n = 4$ наилучшие результаты крестообразное поле демонстрирует, начиная с ранга 2 (рис. 5б). Гиперкубическое рецептивное поле для этой размерности демонстрирует наилучший результат также при ранге 2, однако дальнейшее увеличение ранга приводит к ухудшению точности (рис. 5а). Основываясь на полученных результатах, для дальнейших экспериментов с задачами больших размерностей было выбрано крестообразное рецептивное поле ранга 5.

Во второй серии вычислительных экспериментов была исследована зависимость точности ИНС от числа скрытых слоев. Для пространств \mathbb{R}^{10} и \mathbb{R}^{30} был построен набор нейронных сетей на основе архитектуры, представленной на рис. 3, для которых число скрытых слоев варьировалось от 2 до n . Результаты представлены на рис. 6. Для размерности 10 число скрытых слоев практически не влияет на точность работы ИНС (рис. 6а). Однако для размерности 30 лучший результат показывает ИНС с пятью скрытыми слоями (рис. 6б).

В третьей серии экспериментов была предпринята попытка улучшить точность работы ИНС за счет увеличения времени обучения. Были протестированы ИНС для пространств \mathbb{R}^{10} и \mathbb{R}^{30} . Обучение нейронной сети в \mathbb{R}^{10} происходило на протяжении 900 эпох, в про-

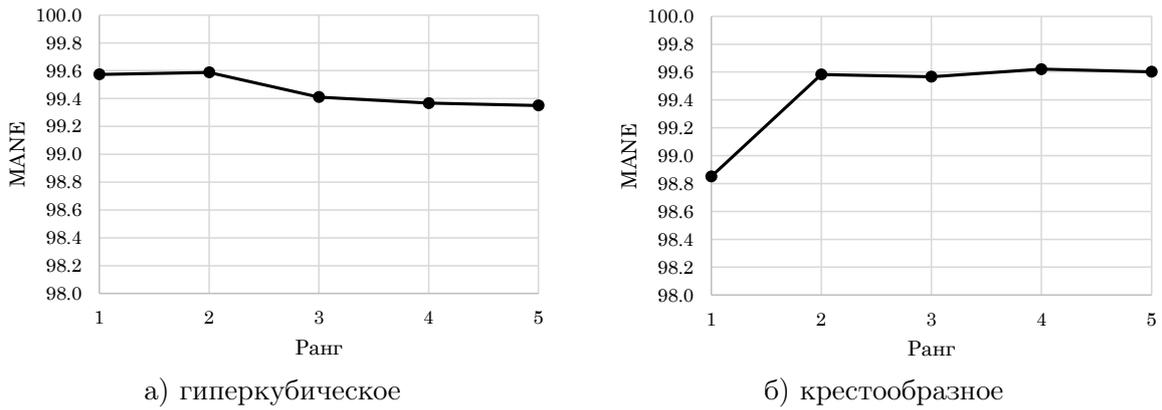


Рис. 5. Зависимость точности от конфигурации и ранга рецептивного поля в \mathbb{R}^4

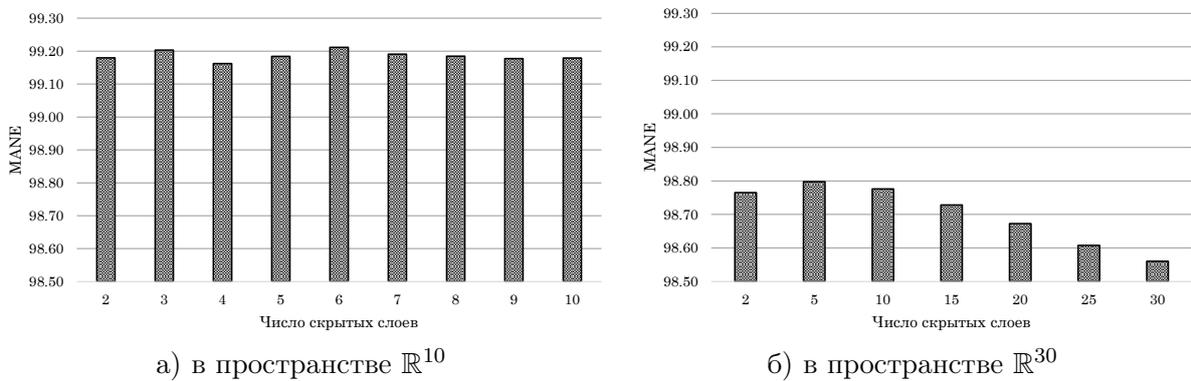


Рис. 6. Зависимость точность ИНС от числа скрытых слоев

странстве \mathbb{R}^{30} — 800 эпох. И в том и в другом случае время обучения составило 60 мин. Результаты представлены на рис. 7. Обращает на себя внимание тот факт, что в \mathbb{R}^{10} максимальная точность достигается на 400 эпохе и далее существенно уже не меняется. В \mathbb{R}^{30} точность увеличивалась практически линейно вплоть до 800 эпохи. Таким образом, можно сделать предположение, что с ростом размерности пространства время обучения ИНС существенно увеличивается.

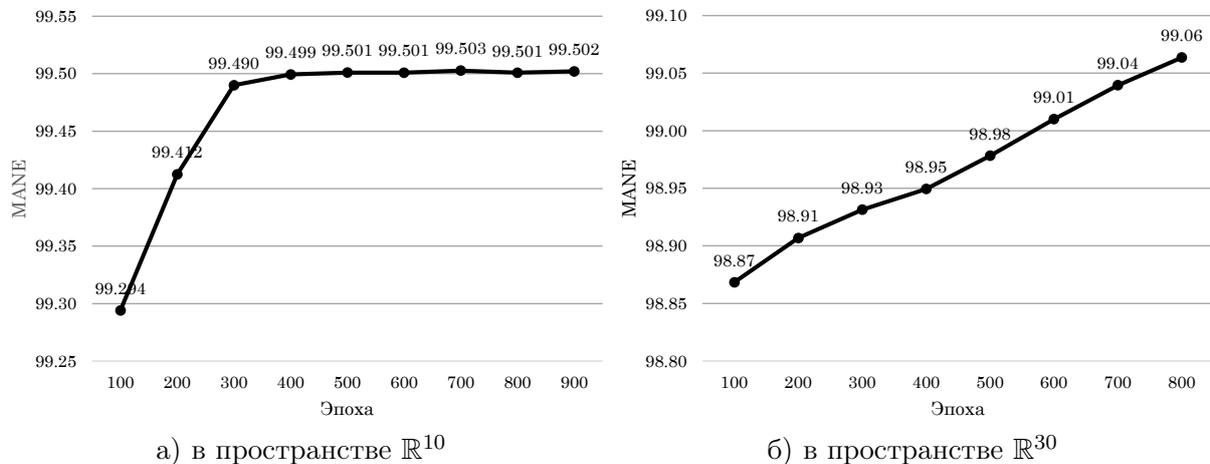


Рис. 7. Зависимость точность ИНС от количество эпох обучения

Заключение

В настоящей работе исследован метод, позволяющий при помощи глубокой нейронной сети прямого распространения вычислять направление движения по гиперплоскостям, ограничивающим допустимый многогранник задачи ЛП. Вычисляемое направление движения соответствует максимальному увеличению значений целевой функции. Улучшен ранее разработанный метод визуализации поверхности многогранника. Представлена новая крестообразная конфигурация рецептивного поля, позволяющая избежать экспоненциального роста числа точек визуализации при увеличении размерности задачи ЛП. Для крестообразной конфигурации рецептивного поля разработан метод генерации обучающих прецедентов, по которому созданы обучающие наборы данных, визуализирующие положение случайных гиперплоскостей и целевых векторов в пространствах размерности до 30 включительно. С использованием обучающих наборов разработана масштабируемая архитектура глубокой ИНС с изменяемым числом скрытых слоев и проведены несколько серий вычислительных экспериментов. Результаты экспериментов свидетельствуют о том, что предложенная крестообразная конфигурация рецептивного поля с точки зрения точности результатов, выдаваемых ИНС, практически не отличается от гиперкубической конфигурации. Испытания крестообразного рецептивного поля показали, что точность основанной на нем ИНС не зависит от числа скрытых слоев. Точность более 98% по метрике MANE достигается в \mathbb{R}^{10} и \mathbb{R}^{30} ИНС с двумя скрытыми слоями на 50 эпохе. Установлено, что для крестообразной конфигурации наилучшая точность достигается при ранге 3 и с дальнейшим увеличением ранга существенно не изменяется. Скорость обучения зависит от размерности пространства. В \mathbb{R}^{10} наилучший результат достигается на 400 эпохе обучения и в дальнейшем не изменяется. В \mathbb{R}^{30} до 800 эпохи точность продолжает расти линейно и достигает значения 99.06%.

В качестве направления дальнейших исследований отметим следующие шаги, реализация которых позволит построить принципиально новый способ решения многомерных задач линейного программирования.

- Разработка нейросетевой модели для вычисления вектора \mathbf{d} в точке, лежащей на ребре — многообразии размерности $n - k$, образованном пересечением k гиперплоскостей.
- Разработка ансамбля нейросетевых моделей для вычисления вектора \mathbf{d} в любой точке допустимого многогранника размерности n .
- Исследование возможности применения сверточных нейронных сетей при распознавании сложных структур на поверхности допустимого многогранника.

Исследование выполнено при финансовой поддержке РФФ (проект № 23-21-00356).

Литература

1. Соколинская И.М., Соколинский Л.Б. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии – XI международная конференция (ПаВТ'2017): Короткие статьи и описания плакатов, Казань, 3–7 апреля, 2017. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471–484. URL: <http://omega.sp.susu.ru/pavt2017/short/014.pdf>.
2. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.

3. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion // IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.
4. Seregin G. Lecture Notes on Regularity Theory for the Navier-Stokes Equations. WORLD SCIENTIFIC, 2014. DOI: 10.1142/9314.
5. Demin D. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, no. 4 (87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.
6. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers // 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). IEEE, 2017. P. 1–4. DOI: 10.1109/ICIEAM.2017.8076177.
7. Zagorskina E., Barbasova T., Shnaider D. Intelligent Control System of Blast-furnace Melting Efficiency // 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). IEEE, 2019. P. 0710–0713. DOI: 10.1109/SIBIRCON48586.2019.8958221.
8. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements // IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/itr2.12047.
9. Scholl M., Minnerup K., Reiter C., *et al.* optimization of a Thermal Management System for Battery Electric Vehicles // 2019 Fourteenth International Conference on Ecological Vehicles and Renewable Energies (EVER). IEEE, 2019. P. 1–10. DOI: 10.1109/EVER.2019.8813657.
10. Meisel S. Dynamic Vehicle Routing // Anticipatory Optimization for Dynamic Decision Making. Vol. 51. New York, NY: Springer, 2011. P. 77–96. Operations Research/Computer Science Interfaces Series. DOI: 10.1007/978-1-4614-0505-4_6.
11. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis // Real-Time Systems. Wiley, 2002. P. 41–85. DOI: 10.1002/0471224626.ch3.
12. Kopetz H. Real-Time Scheduling // Real-Time Systems. Boston, MA: Springer, 2011. P. 239–258. Real-Time Systems Series. DOI: 10.1007/978-1-4419-8237-7_10.
13. Dantzig G.B. Linear programming and extensions. Princeton university press, 1998. 656 p.
14. Hall J.A.J., McKinnon K.I.M. Hyper-Sparsity in the Revised Simplex Method and How to Exploit it // Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.
15. Klee V., Minty G.J. How good is the simplex algorithm? // Inequalities — III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. Academic Press, 1972. P. 159–175.
16. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement // Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.
17. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.
18. Bartels R.H., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions // Handbook for Automatic Computation. Berlin, Heidelberg: Springer Berlin Heidelberg, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.

19. Tolla P. A Survey of Some Linear Programming Methods // Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: Wiley, 2014. P. 157–188. DOI: 10.1002/9781119005216.ch7.
20. Hall J.A.J. Towards a practical parallelisation of the simplex method // Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.
21. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // Algorithms, Probability, Networks, and Games. Vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Springer, 2015. P. 281–307. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24024-4_17.
22. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems // Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.
23. Khachiyan L. Polynomial algorithms in linear programming // USSR Computational Mathematics and Mathematical Physics. 1980. Vol. 20, no. 1. P. 53–72. DOI: 10.1016/0041-5553(80)90061-0.
24. Shor N.Z. Cut-off method with space extension in convex programming problems // Cybernetics. 1977. Vol. 13, no. 1. P. 94–96. DOI: 10.1007/BF01071394.
25. Юдин Д.Б., Немировский А.С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач // Экономика и математические методы. 1976. № 2. С. 357–369.
26. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.
27. Gondzio J. Interior point methods 25 years later // European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.
28. Fathi-Hafshejani S., Mansouri H., Peyghami M.R., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term // Optimization. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.
29. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming // Numerical Algebra, Control & Optimization. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.
30. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs // International Conference on Graphic and Image Processing (ICGIP 2011). Vol. 8285. International Society for Optics, Photonics, 2011. P. 828502. DOI: 10.1117/12.913019.
31. Kheirfam B., Haghghi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function // Optimization. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.
32. Xu Y., Zhang L., Zhang J. A full-modified-Newton step infeasible interior-point algorithm for linear optimization // Journal of Industrial and Management Optimization. 2015. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.
33. Roos C., Terlaky T., Vial J.-P. Interior Point Methods for Linear Optimization. Springer-Verlag, 2005. 500 p. DOI: 10.1007/b100325.
34. Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования // Вычислительные методы и программирование. 2023. Т. 24, № 4. С. 408–429. DOI: 10.26089/NumMet.v24r428.

35. Соколинский Л.Б., Соколинская И.М. Об одном итерационном методе решения задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2020. Т. 21, № 3. С. 329–340. DOI: 10.26089/NUMMET.V21R328.
36. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103.
37. Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46. DOI: 10.14529/cmse230201.
38. Biewald L. Experiment Tracking with Weights and Biases // Software available from wandb.com. 2020. January.
39. Goodfellow I., Bengio Y., Courville A. Deep Learning (Adaptive Computation and Machine Learning). Vol. 8. 2016.
40. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

Ольховский Николай Александрович, младший научный сотрудник, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230402

STUDY OF NEURAL NETWORK MODELS FOR LINEAR PROGRAMMING

© 2023 N.A. Olkhovsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: olkhovskina@susu.ru

Received: 20.10.2023

The article explores a method for determining the motion vector from hyperplanes that bound the feasible polytop of a multidimensional linear programming problem. The method is based on visual images fed to the input of a feedforward neural network. The visualization algorithm constructs a receptive field in the vicinity of a point located on the bounding hyperplane. For each point of the receptive field, the scalar bias to the hyperplane surface is calculated. Based on the calculated bias, each receptive field point is assigned with a scalar value. The resulting visual image is fed to the input of a feed-forward neural network, which calculates the direction of maximum increase in the objective function on the bounding hyperplane. The article proposes an improved form of the cross-shaped receptive field. The construction of a training set based on randomly generated bounding hyperplanes and objective functions in multidimensional spaces is described. A scalable neural network architecture with a variable number of hidden layers has been developed. The hyperparameters of the neural network were selected. Computational experiments confirmed the high (more than 98%) accuracy of the cross-shaped receptive field. The dependence of the accuracy of the neural network results on the number of hidden layers and the duration of training was studied.

Keywords: linear programming, surface movement method, artificial neural network, deep learning.

FOR CITATION

Olkhovsky N.A. Study of Neural Network Models for Linear Programming. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 55–75. (in Russian) DOI: 10.14529/cmse230402.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Sokolinskaya I.M., Sokolinsky L.B. On the Solution of Linear Programming Problems in the Age of Big Data. Parallel Computational Technologies. Vol. 753 / ed. by L.B. Sokolinsky, M.L. Zymbler. Cham: Springer, 2017. P. 86–100. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-67035-5_7.
2. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery. Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.
3. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion. IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.
4. Seregin G. Lecture Notes on Regularity Theory for the Navier-Stokes Equations. WORLD SCIENTIFIC, 2014. DOI: 10.1142/9314.
5. Demin D. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state. Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, no. 4 (87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.
6. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers. 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). IEEE, 2017. P. 1–4. DOI: 10.1109/ICIEAM.2017.8076177.
7. Zagorskina E., Barbasova T., Shnaider D. Intelligent Control System of Blast-furnace Melting Efficiency. 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). IEEE, 2019. P. 0710–0713. DOI: 10.1109/SIBIRCON48586.2019.8958221.
8. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements. IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/itr2.12047.
9. Scholl M., Minnerup K., Reiter C., *et al.* optimization of a Thermal Management System for Battery Electric Vehicles. 2019 Fourteenth International Conference on Ecological Vehicles and Renewable Energies (EVER). IEEE, 2019. P. 1–10. DOI: 10.1109/EVER.2019.8813657.
10. Meisel S. Dynamic Vehicle Routing. Anticipatory Optimization for Dynamic Decision Making. Vol. 51. New York, NY: Springer, 2011. P. 77–96. Operations Research/Computer Science Interfaces Series. DOI: 10.1007/978-1-4614-0505-4_6.
11. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis. Real-Time Systems. Wiley, 2002. P. 41–85. DOI: 10.1002/0471224626.ch3.

12. Kopetz H. Real-Time Scheduling. Real-Time Systems. Boston, MA: Springer, 2011. P. 239–258. Real-Time Systems Series. DOI: 10.1007/978-1-4419-8237-7_10.
13. Dantzig G.B. Linear programming and extensions. Princeton university press, 1998. 656 p.
14. Hall J.A.J., McKinnon K.I.M. Hyper-Sparsity in the Revised Simplex Method and How to Exploit it. Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.
15. Klee V., Minty G.J. How good is the simplex algorithm?. Inequalities — III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. Academic Press, 1972. P. 159–175.
16. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement. Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.
17. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.
18. Bartels R.H., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions. Handbook for Automatic Computation. Berlin, Heidelberg: Springer Berlin Heidelberg, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.
19. Tolla P. A Survey of Some Linear Programming Methods. Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: Wiley, 2014. P. 157–188. DOI: 10.1002/9781119005216.ch7.
20. Hall J.A.J. Towards a practical parallelisation of the simplex method. Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.
21. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method. Algorithms, Probability, Networks, and Games. Vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Springer, 2015. P. 281–307. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24024-4_17.
22. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems. Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.
23. Khachiyan L. Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics. 1980. Vol. 20, no. 1. P. 53–72. DOI: 10.1016/0041-5553(80)90061-0.
24. Shor N.Z. Cut-off method with space extension in convex programming problems. Cybernetics. 1977. Vol. 13, no. 1. P. 94–96. DOI: 10.1007/BF01071394.
25. Yudin D., Nemirovsky A. Information complexity and efficient methods for solving convex extremal problems. Economics and mathematical methods (Ekonomika i matematicheskie metody). 1976. No. 2. P. 357–369. (in Russian).
26. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.
27. Gondzio J. Interior point methods 25 years later. European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.

28. Fathi-Hafshejani S., Mansouri H., Peyghami M.R., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term. *Optimization*. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.
29. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming. *Numerical Algebra, Control & Optimization*. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.
30. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs. *International Conference on Graphic and Image Processing (ICGIP 2011)*. Vol. 8285. International Society for Optics, Photonics, 2011. P. 828502. DOI: 10.1117/12.913019.
31. Kheirfam B., Haghghi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function. *Optimization*. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.
32. Xu Y., Zhang L., Zhang J. A full-modified-Newton step infeasible interior-point algorithm for linear optimization. *Journal of Industrial and Management Optimization*. 2015. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.
33. Roos C., Terlaky T., Vial J.-P. *Interior Point Methods for Linear Optimization*. Springer-Verlag, 2005. 500 p. DOI: 10.1007/b100325.
34. Olkhovsky N.A., Sokolinsky L.B. A new method of linear programming. *Numerical Methods and Programming (Vychislitel'nye Metody i Programirovanie)*. 2023. Vol. 24, no. 4. P. 408–429. (in Russian) DOI: 10.26089/NUMMET.V24R428.
35. Sokolinskaya I.M., Sokolinsky L.B. On an iterative method for solving linear programming problems on cluster computing systems. *Numerical Methods and Programming (Vychislitel'nye Metody i Programirovanie)*. 2020. Vol. 21, no. 3. P. 329–340. DOI: 10.26089/NumMet.v21r328.
36. Olkhovsky N.A., Sokolinsky L.B. Visualizing Multidimensional Linear Programming Problems. *Parallel Computational Technologies*. Vol. 1618 / ed. by L.B. Sokolinsky, M.L. Zymbler. Cham: Springer, 2022. P. 172–196. *Communications in Computer and Information Science*. DOI: 10.1007/978-3-031-11623-0_13.
37. Sokolinsky L.B., Sokolinskaya I.M. Apex Method: A New Scalable Iterative Method for Linear Programming. *Mathematics*. 2023. Vol. 11, no. 7. P. 1654. DOI: 10.3390/math11071654.
38. Biewald L. Experiment Tracking with Weights & Biases. Software available from wandb.com. 2020. January.
39. Goodfellow I., Bengio Y., Courville A. *Deep Learning (Adaptive Computation and Machine Learning)*. Vol. 8. 2016.
40. Bilenko R.V., Dolganina N.Y., Ivanova E.V., Rekachinsky A.I. High-performance Computing Resources of South Ural State University. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*. 2022. Vol. 11, no. 1. P. 15–30. DOI: 10.14529/cmse220102.

МЕТОДЫ УПРАВЛЕНИЯ WORK-STEALING ДЕКАМИ В ДИНАМИЧЕСКИХ ПЛАНИРОВЩИКАХ МНОГОПРОЦЕССОРНЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

© 2023 Е.А. Аксёнова, А.В. Соколов

*Институт прикладных математических исследований,
обособленное подразделение исследовательского центра
«Карельский научный центр Российской академии наук»
(185910 Петрозаводск, ул. Пушкинская, д. 11)
E-mail: aksenova@krc.karelia.ru, sokavs@gmail.com*

Поступила в редакцию: 21.07.2023

В параллельных планировщиках задач, работающих по стратегии work-stealing, каждый процессор имеет свой дек задач. Один конец дека используется для добавления и извлечения задач только владельцем, а другой — для перехвата задач другими процессорами. В статье предлагается обзор методов управления work-stealing деками, которые используются при реализации work-stealing планировщиков параллельных задач, а также представлено описание поставленных и решенных нашим коллективом задач оптимального управления деками для стратегии work-stealing. Принцип алгоритмов оптимального управления деками в двухуровневой памяти заключается в том, что при переполнении выделенного участка быстрой памяти происходит перераспределение элементов (задач) дека между уровнями памяти. В быстрой памяти остаются элементы из концов дека, так как с ними будет происходить работа в ближайшее время, а элементы средней части дека хранятся в медленной памяти. В таком случае необходимо определить оптимальное количество элементов, которое нужно оставить в быстрой памяти, в зависимости от критерия оптимальности и параметров системы.

Ключевые слова: имитационные и марковские модели оптимального управления структурами данных, оптимальное кэширование деков, оптимальное управление work-stealing деками, оптимизация work-stealing планировщиков, управляемые случайные блуждания.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аксёнова Е.А., Соколов А.В. Методы управления work-stealing деками в динамических планировщиках многопроцессорных параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 76–93. DOI: 10.14529/cmse230403.

Введение

Дек (от англ. deque — double ended queue) — это структура данных, в которой добавление новых элементов и удаление существующих производится с обоих концов (рис. 1). Дек поддерживает FIFO- и LIFO-операции, поэтому с помощью дека можно реализовать как стек, так и очередь.

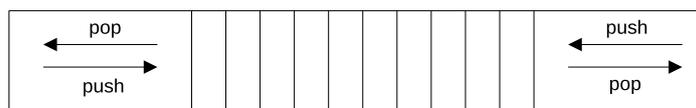


Рис. 1. Дек

В параллельных планировщиках задач, работающих по стратегии work-stealing, каждый процессор имеет свой дек (deque) задач. Один конец дека используется для добавления и извлечения задач только владельцем, а другой — для перехвата задач другими процессорами. Когда процессор создает новую задачу, он добавляет задачу в свой дек; когда процессору нужна задача, он берет задачу из дека. Если процессор узнает, что его дек пуст, он перехватывает задачи у другого процессора. Добавление и удаление элементов (задач или указателей на задачи) выполняется на одном конце дека, который работает как LIFO-стек, а перехват (кража) элементов происходит на другом конце дека — как в FIFO-очереди (рис. 2).

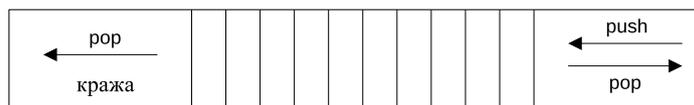


Рис. 2. Work-stealing дек

В данной статье предлагается обзор методов управления work-stealing деками, которые используются при реализации work-stealing планировщиков параллельных задач, а также представлено описание поставленных и решенных нашим коллективом задач оптимального управления work-stealing деками. Принцип алгоритмов оптимального управления деками в двухуровневой памяти заключается в том, что при переполнении отведенного деку участка быстрой памяти происходит перераспределение элементов между уровнями памяти: в быстрой памяти остаются элементы из концов дека, так как с ними будет происходить работа в ближайшее время, а элементы средней части дека хранятся в памяти второго уровня (рис. 3).

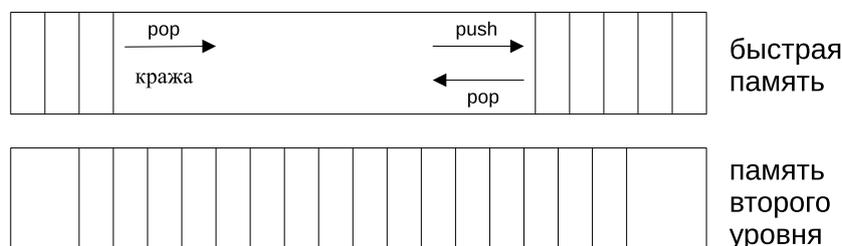


Рис. 3. Work-stealing дек в двухуровневой памяти

Основной целью наших работ по этой тематике являлось построение и анализ математических и имитационных моделей с целью оптимизации работы с деками, расположенными в общей многоуровневой памяти. Параметрами для этих моделей являются число деков, вероятности операций с деками на каждом шаге дискретного времени (возможно как последовательное, так и параллельное выполнение операций), размеры памяти всех уровней, временные характеристики операций работы с деками в разных уровнях памяти. В качестве критериев оптимальности рассматривались максимизация суммы средних времен работы каждого дека до перераспределения памяти, максимизация наименьшего среднего времени работы каждого дека до перераспределения памяти, минимизация суммы средних затрат на перераспределения памяти, возникающие в случае переполнения или опустошения быстрой памяти каждым деком.

Традиционно дека разрабатывается с предположением, что указатели задач хранятся в этих структурах данных, а объекты задач находятся в куче памяти. Путем изменения организации деков задач так, чтобы они могли содержать объекты задач вместо указателей, удалось повысить производительность более чем в 2.5 раза для приложений, привязанных к ЦП, и уменьшить количество промахов кэша последнего уровня на 30% по сравнению с work-stealing планировщиками Intel Threading Building Blocks и Intel/MIT Cilk.

Статья организована следующим образом. В разделе 1 дан обзор статей, в которых описаны различные методы балансировки задач в многопроцессорных системах. В разделе 2 представлены задачи оптимального управления деками для стратегии work-stealing, решенные нашим коллективом, а так же описание реализации экспериментального динамического work-stealing планировщика. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

1. Методы балансировки задач в многопроцессорных системах

Стратегии балансировки параллельных вычислений разделяют на статические и динамические [1]. Статическая балансировка используется, когда известны все свойства и особенности выполняемых задач. В этом случае можно заранее определить оптимальное расписание задач (например, минимизировать среднее время решения). Такие задачи считаются NP-полными [2] и встречаются достаточно редко. При динамической балансировке планировщик во время работы использует некую относительно простую стратегию балансировки, которая дает результат близкий к оптимальному (например, наименьшее время выполнения задачи) [3]. В свою очередь, в динамической балансировке выделяют централизованную, когда есть специальный поток для распределения работы между другими потоками, и нецентрализованную [4, 5]. При нецентрализованной балансировке у каждого потока имеется своя очередь задач (подпрограмм) для выполнения и потоки сами осуществляют распределение. Например, поток, у которого слишком много задач, может их перераспределить между другими потоками (метод «work-dealing» [6, 7]). Если у потока заканчиваются задачи для выполнения, он их запрашивает (метод «work-requesting» [8]) или перехватывает (метод «work-stealing» [9]) у другого потока.

Теоретически доказано [9], что стратегия work-stealing дает распределение задач близкое к оптимальному (минимизация времени выполнения задачи), и на практике она себя также зарекомендовала [10]. Ее реализацию можно встретить во многих планировщиках задач, например, Cilk [11], dotNET TPL [12], X10 [13] и других. В этом методе балансировки нагрузки каждый процессор решает ряд задач, указатели на которые хранятся в деке этого процессора. Когда процессор создает новую задачу, он добавляет указатель на задачу в свой дек; когда процессору нужна задача, он берет указатель на задачу из вершины дека. Если процессор узнает, что его дек пуст, он перехватывает указатели на задачи у другого процессора. Первые две операции выполняются как в LIFO-стеке, а перехват происходит из основания дека — как в FIFO-очереди. В терминологии Д. Кнута такая структура данных называется деком с ограниченным входом [14]. Количество элементов, извлекаемых за один перехват, может различаться. Так, в [15] предлагалось перехватывать один элемент, в [6] — половину элементов.

Механизм балансировки загрузки применяется в многопроцессорных системах и вычислительных сетях. Для повышения эффективности «work-stealing» метода балансировки

загрузки исследуют способы реализации и управления структурами данных. На основе этого метода разрабатываются различные планировщики задач. Например, в [15] исследуется проблема эффективного планирования структурированных многопоточных вычислений на параллельных компьютерах, предлагается планировщик с перехватом работы для многопоточных вычислений с зависимостями.

В работе [16] авторы разделяют методы балансировки нагрузки параллельных задач на статическую и динамическую, а также предлагают стратегию динамической балансировки нагрузки для однородной многопроцессорной системы и применяют ее к сети, называемой сетью *Folded Crossed Cube*. Результаты экспериментов показывают, что вместе со временем выполнения достигается меньший коэффициент дисбаланса нагрузки. В статье [17] рассмотрено применение теоретических результатов к задаче балансировки нагрузки в стохастических динамических сетях с неполной информацией о текущих состояниях агентов и изменяющимся набором каналов связи. Установлены условия достижения оптимального уровня балансировки нагрузки. Производительность системы оценивается как аналитически, так и путем моделирования.

В статье [18] исследуются интерактивные веб-сервисы, которые все чаще управляют критически важными бизнес-задачами, такими как поиск, реклама, игры, покупки и финансы. Показано, как обобщить кражу работы, которая используется для минимизации времени выполнения одного параллельного задания, для оптимизации целевой задержки в интерактивных сервисах с несколькими параллельными запросами. Авторы разрабатывают новый адаптивный метод кражи задач — метод контроля хвостов, который уменьшает количество запросов, не достигающих целевой задержки. Этот подход реализуется в библиотеке *Intel Threading Building Blocks* и оценивается на различных рабочих нагрузках. Метод контроля хвостов существенно снижает количество запросов, превышающих желаемую целевую задержку, и обеспечивает относительное улучшение до 58% по сравнению с различными базовыми методами.

В работе [19] анализируется *Intel Threading Building Blocks* — библиотека C++ для параллельного программирования. Шаблоны библиотеки для общих параллельных циклов построены на вложенном параллелизме и планировщике, перехватывающем задачи. В статье обсуждаются способы оптимизации, в которых алгоритм для оптимизации работы учитывает данные о кражах задач.

В статье [20] авторы утверждают, что кража задач — это эффективный метод реализации балансировки нагрузки при детальном параллелизме задач. Обычно для этой цели используются параллельные деки. Недостатком многих параллельных деков является то, что они требуют дорогостоящих ограничений памяти для локальных операций с деками. В этой статье предлагается новый неблокирующий дек для кражи задач, основанный на разделенной очереди задач. В деке используется динамическая точка разделения между общей и частной частями дека. Авторы представляют «Lace» — реализацию кражи задач с интерфейсом, похожим на *work-stealing* библиотеку кражи задач *Wool* [21].

В статье [22] основное внимание уделяется многоядерным алгоритмам ветвей и границ для решения крупномасштабных задач оптимизации на основе перестановок. Исследуется пять стратегий кражи задач с новой структурой данных, которая называется целочисленно-векторной матрицей. В этих стратегиях каждый поток имеет частную матрицу, позволяющую локально управлять набором подзадач. Стратегии различаются способом выбора потока-жертвы и степенью детализации украденных задач. Результаты оценки эффектив-

ности подхода на основе целочисленно-векторной матрицы показали, что предложенный алгоритм кражи задач превосходит алгоритм кражи на основе связанных списков по процессорному времени, использованию памяти и количеству выполняемых операций кражи.

В работе [23] представлены три структуры данных без блокировок для распределения приоритетных задач: приоритетная структура с кражей задач, централизованная с ослабленной семантикой и гибридная, сочетающая обе концепции. На примере задачи поиска в графе кратчайшего пути от одной вершины (SSSP, Single Source Shortest Path) показано, как различные подходы влияют на расстановку приоритетов, и предоставлены верхние границы количества проверенных узлов. Авторы утверждают, что распределение приоритетных задач обеспечивает интуитивно понятный и простой способ распараллеливания проблемы SSSP, которая является сложной задачей. Экспериментальные данные подтверждают хорошую масштабируемость полученного алгоритма.

В работе [24] предлагается планировщик (LAWS, Locality-Aware Work-Stealing), который использует общую кэш-память и систему памяти NUMA. В LAWS распределитель задач с балансировкой нагрузки используется для равномерного разделения и хранения набора данных программы на всех узлах памяти и выделения задачи интерфейсу, в котором узел локальной памяти хранит свои данные. Для балансировки нагрузки применяется трехуровневый планировщик с кражей задач. Результаты экспериментов показывают, что LAWS может повысить производительность программ, связанных с памятью, до 54.2% по сравнению с традиционными work-stealing планировщиками.

Асимметричные многоядерные архитектуры (AMC, Asymmetric Multi-Core), в которых ядра разных процессоров имеют разную производительность и энергопотребление, широко используются от крупных центров обработки данных до мобильных смартфонов из-за их высокой производительности и энергоэффективности. Однако существующие способы распределения задач часто приводят к низкой производительности параллельных программ на новых архитектурах AMC из-за несбалансированной рабочей нагрузки, кэш-промахов и удаленного доступа к памяти. Чтобы решить эту проблему, авторы [25] предлагают систему выполнения (SAWS, Selective Asymmetry-aware Work-Stealing), которая может сократить удаленный доступ к памяти, одновременно распределяя рабочую нагрузку между асимметричными ядрами. SAWS состоит из планировщика задач с учетом асимметричности и планировщика выбора кражи задач. Планировщик задач с учетом асимметричности правильно распределяет задачи по асимметричным процессорам, чтобы большинство задач могли получить доступ к данным из узла локальной памяти, а рабочая нагрузка балансировалась в соответствии с вычислительными возможностями различных процессоров. После этого используется планировщик выбора кражи задач для дальнейшей балансировки рабочей нагрузки во время выполнения. Результаты экспериментов на реальной системе показывают, что SAWS повышает производительность программ, связанных с памятью, до 59.3% по сравнению с традиционными work-stealing планировщиками в архитектурах AMC.

В статье [26] представлен масштабируемый планировщик с адаптивным алгоритмом кражи задач (SLAW, Scalable Locality-aware Adaptive Work-stealing scheduler), который выбирает способ распределения каждой задачи во время выполнения. Планировщик SLAW также устанавливает ограничения на использование стека и кучи, необходимые для хранения задач.

В работе [27] разработаны модели и проведен анализ нескольких рандомизированных алгоритмов кражи работы в динамических условиях. С помощью дифференциальных

уравнений в этих моделях показано предельное поведение систем при возрастании числа процессоров до бесконечности. Этот подход дает возможность моделировать разные системы и обеспечивать точные численные аппроксимации поведения системы даже тогда, когда количество процессоров относительно невелико.

В [28] предлагается новый механизм оптимизации планировщиков путем сбора информации во время выполнения программы. С этой целью, используется матрица взаимодействия потоков (TIM, Thread-Interaction Matrix), в которой хранится статистика взаимодействия потоков.

2. Модели оптимального управления work-stealing деками

В [29] описана реализация экспериментального динамического work-stealing планировщика. Параллельные планировщики задач, работающие по стратегии work-stealing, дают распределение задач близкое к оптимальному, и при этом обладают низкими накладными расходами по времени, используемой памяти и межпроцессорным синхронизациям. Основная идея этой стратегии: когда у процессора заканчиваются задачи для выполнения, он их забирает (крадет) у другого процессора. Один из недостатков этой стратегии — большое количество краж, возникающих при выполнении относительно маленьких задач. В этой статье описана реализация work-stealing планировщика на языке C++, в которой возможно извлекать определенное количество задач за одну кражу, и предложен способ оценки вероятностей операций, выполняющихся в планировщике. Такая модификация при правильном выборе количества украденных задач позволяет значительно уменьшить общее количество краж.

Для представления динамических деков в памяти используются разные способы. Можно воспользоваться методом «связное представление» [30]. Модель такого метода будет схожа с уже построенной моделью связного представления стеков и очередей [31]. Можно использовать способ двухсвязного страничного представления [32]. Математическая модель для такого представления деков будет близка к разработанной модели для других структур данных [27], только для стеков и очередей достаточно иметь односвязный список страниц. Также, возможно представление параллельных структур в общей памяти (которая заранее не делится), когда они двигаются друг за другом по кругу, начиная с некоторого начального места в памяти. Этот метод работы был предложен в [33] и анализировался в [34, 35] для FIFO-очередей. Использование этого метода для work-stealing деков было запатентовано [36].

Отметим, что среди многоядерных архитектур есть архитектуры без кэш-памяти. Например, в архитектуре AsAP-II каждое ядро имеет два FIFO-буфера, а в архитектуре SEAForth — два стека (для хранения данных и адресов возврата). В этих архитектурах очереди и стеки реализованы циклически и отделены друг от друга с возможностью потери элементов из-за переполнения. Но в наших работах исследуются ситуации, когда для хранения нескольких структур данных используется разделяемая память. В некоторых случаях это может свести к минимуму количество потерянных элементов. На основе этих архитектур аппаратно могут быть реализованы work-stealing деками. В этом случае важно исследовать оптимальную организацию двух деков, а затем, в случае произвольного количества ядер, можно получить желаемые чипы, составив их из «двухдековых». В [37] рассмотрена задача и предложена математическая модель оптимального разбиения памяти одного уровня для двух work-stealing деков.

В [38] представлен анализ математических моделей процесса работы с двумя циклическими деками, расположенными в общей памяти. Параметрами этих моделей являются вероятности операций на каждом шаге дискретного времени (возможно как последовательное, так и параллельное выполнение операций). Модели строятся в виде случайных блужданий по целочисленной решетке на плоскости. На основе вышеупомянутых моделей решены задачи оптимального разделения памяти при некоторых стратегиях перехвата элементов. В качестве критерия оптимальности рассматривается максимальное среднее время до переполнения памяти. Проведены статистические исследования по оценке вероятностей операций работы с деками для нескольких типов задач, выполняемых в реализованном планировщике — вычисление чисел Фибоначчи с помощью рекурсии, задача о рюкзаке, решаемая методом ветвей и границ, умножение матриц, сортировка слиянием, обход графа задач. Для полученных вероятностей операций работы с деками проведены численные эксперименты по анализу разработанных моделей. Разработаны алгоритмы и программы на языке C для построения матриц переходных вероятностей P при произвольных значениях m (размер быстрой памяти), s (размер первого дека), o (количество перехваченных элементов), вероятностей операций, а также нахождения оптимального разбиения памяти между деками, в зависимости от вероятностных характеристик деков, и оптимального количества элементов для перехвата. Для решения поставленных задач использовался аппарат управляемых случайных блужданий, поглощающих цепей Маркова, система LAPACK.

В [39] предложены математические модели работы с n последовательными циклическими деками, расположенными в общей памяти. Математические модели строятся в виде случайных блужданий по целочисленной решетке в n -мерном пространстве. Операции с заданными вероятностями происходят на каждом шаге дискретного времени. Решалась задача нахождения оптимального разбиения памяти между n деками и задача определения оптимального количества элементов для кражи. Критерием оптимальности являлось максимальное среднее время до переполнения памяти.

В [40] анализировались два метода представления деков: один из распространенных методов — раздельное последовательное циклическое представление деков, и новый метод, где общая память для деков заранее не делится и они двигаются друг за другом по кругу. Ранее эти методы анализировались нами для представления FIFO-очередей в сетевых приложениях, где для некоторых значений параметров системы метод «друг за другом» давал лучший результат. В работе представлен анализ модели процесса работы с двумя последовательными деками, когда они двигаются друг за другом по кругу в общей памяти. Предложены математическая и имитационная модели данного процесса и проведены численные эксперименты. Математическая модель была построена, как случайное блуждание по целым точкам в пирамиде. Имитационная модель строится с помощью метода Монте-Карло. Используемая стратегия work-stealing — перехват одного элемента. В качестве критерия оптимальности было рассмотрено максимальное среднее время работы до переполнения памяти.

В работе [41] представлены разработка, анализ и сравнение моделей и методов управления деками в ограниченной разделяемой памяти. Для решения поставленных задач использовалась модель в виде управляемого случайного блуждания и имитационное моделирование. Для случая трех деков рассмотрены следующие способы управления:

- каждый из трех деков располагается в своей отдельной области памяти;
- три дека двигаются друг за другом по кругу;

– комбинированный способ — два дека располагаются друг за другом, один отдельно.

В [42] исследовался способ повышения производительности work-stealing планировщиков за счет усовершенствования внутренних механизмов обработки данных. Традиционно дека разрабатывается с предположением, что указатели задач хранятся в этих структурах данных, а объекты задач находятся в куче памяти. Путем изменения организации деков задач таким образом, чтобы они могли содержать объекты задач вместо указателей, нам удалось повысить производительность более чем в 2.5 раза для приложений, привязанных к ЦП, и уменьшить количество промахов кэша последнего уровня на 30% по сравнению с work-stealing планировщиками Intel Threading Building Blocks и Intel/MIT Cilk. Сравнения производились на Intel Pentium N3530 2.16GHz, 4 CPU, 64 L1, с Linux 4.4.0 и на ARMv7l rev5, 4 CPU, 32 L1, с Linux 4.4.8. Использовались следующие тесты: умножение матриц, сортировка слиянием, рекурсивное вычисление чисел Фибоначчи, решение задачи о рюкзаке, обход графа в глубину. Реализации планировщика для общей и распределенной памяти размещены в свободных репозиториях [43] и [44] соответственно.

В работе [45] предлагается использовать разработанный нашим коллективом work-stealing планировщик [29, 42] для обучения сверточных нейронных сетей. В работе [46] авторы предлагают новые решения для оптимизации сельского хозяйства, основанные на использовании встроенных компьютерных систем и сенсорных сетей. Использование таких систем обработки информации поможет вытеснить мир, основанный на воздушных (беспилотниках) и наземных роботах. Эти инструменты будут выполнять точные задачи с помощью систем определения местоположения, которые предоставляют точную информацию. Чтобы всего этого добиться, необходимо провести тщательное изучение информационных систем сельскохозяйственных полей с использованием точных датчиков. Также нужны новые инструменты для управления данными, чтобы обрабатывать их в режиме реального времени. Правильное распределение этих данных с помощью различных инструментов (и в нашей работе [42] по мнению авторов статьи приводится один такой инструмент), требует их точной предварительной обработки, что позволит правильно обрабатывать информацию для повышения урожайности и качества продукции сельскохозяйственных полей. Также оптимизация работы с данными в реальном времени важна во многих других приложениях, например, в военных.

В [47] рассмотрена задача оптимального управления work-stealing деком в двухуровневой памяти. Предполагается, что известны вероятности параллельных операций с деком. Задача состоит в нахождении числа элементов FIFO- и LIFO-частей дека, которые при перераспределении дека будут оставлены в быстрой памяти, чтобы максимизировать среднее время работы до перераспределения памяти.

В [48] рассмотрена задача оптимального управления work-stealing деком в двухуровневой памяти. Предполагается, что известны вероятности параллельных операций с деком и временные характеристики уровней памяти. Задача состоит в нахождении оптимального количества элементов FIFO- и LIFO-частей дека, которые при перераспределении дека должны быть оставлены в быстрой памяти. В качестве критерия оптимальности рассмотрены минимальные средние затраты на перераспределение памяти, которые возникают в случае переполнения или опустошения быстрой памяти. Такой критерий позволяет учитывать конкретные скорости доступа к уровням памяти и применять разработанные методы к разным сочетаниям быстрой и медленной памяти.

В [49] анализируется задача оптимального управления двумя деками в двухуровневой памяти (например, регистры — оперативная память, когда известны вероятности параллельных операций с деками. Рассмотрен классический последовательный циклический метод представления дека в памяти. В случае переполнения дека в быстрой памяти или опустошения FIFO-части или LIFO-части происходят необходимые обмены между быстрой и медленной памятью и сдвиги элементов в быстрой памяти для перемещения в оптимальное состояние, которое необходимо найти. Задача состоит в том, чтобы найти оптимальное разбиение общей быстрой памяти для деков и определить оптимальное состояние каждого дека в каждом разделе после перераспределения памяти, т.е. найти оптимальное количество элементов, взятых с обеих сторон дека, которое надо оставлять в быстрой памяти, если дек заполнен или опустошен. Критерием оптимальности для совместного использования памяти является максимизация суммы средних времен работы каждого дека до перераспределения памяти и максимизация наименьшего среднего времени работы каждого дека до перераспределения памяти.

В [50] рассматривается задача оптимального управления двумя деками в двухуровневой памяти. Вероятности параллельных операций с деками известны. Задача состоит в том, чтобы найти оптимальное разделение быстрой памяти для деков и определить оптимальное количество элементов для обоих концов каждого дека, которое сохраняется в быстрой памяти после перераспределения памяти. В качестве критерия оптимальности рассматривается минимальная сумма средних затрат на перераспределения памяти, возникающие в случае переполнения или опустошения быстрой памяти для каждого дека. Этот критерий позволяет учитывать конкретные скорости доступа к уровням памяти и применять разработанные методы к различным сочетаниям быстрой и медленной памяти.

Заключение

В статье дан обзор моделей и методов оптимального параллельного управления work-stealing деками. Эта тематика является важной для оптимизации динамической балансировки параллельных вычислений. Например, возможен и такой вариант задач, когда некоторые дека могут быть очень большими и переполняют быструю память. Тогда декам нужно выделить отдельный раздел быстрой памяти и работать с ними так, как предложено для случая двух деков. Но в тоже время, некоторые дека имеют небольшой размер, поэтому их можно располагать друг за другом, пока они не переполнят быструю память.

В задаче параллельного управления двумя деками в двухуровневой памяти можно рассмотреть способ работы, когда в одном разделе быстрой памяти остаются LIFO-части деков — два стека, в которых происходят включения и исключения элементов, а в другом разделе быстрой памяти остаются две FIFO-части, из которых элементы только исключаются (кражи). Средние части деков находятся в медленной памяти, обращение к ним происходит при переполнении или опустошении LIFO-частей или FIFO-частей деков, расположенных в быстрой памяти. Возможны разные способы представления LIFO- и FIFO-частей в памяти: две FIFO-части расположены отдельно или объединены в одну очередь, две LIFO-части растут навстречу друг другу. Нужно определить, как разделить быструю память между двумя деками, какой способ организации LIFO- и FIFO-частей деков и какое начальное распределение памяти выбрать. Этот выбор зависит от характеристик уровней памяти, вероятностей операций со структурами данных и рассматриваемого критерия оптимальности. Можно рассмотреть три критерия оптимальности: максимизация среднего времени до

перераспределения быстрой памяти для двух деков; минимизация суммы средних затрат на перераспределение быстрой памяти для двух деков; минимизация наибольших средних затрат на перераспределение быстрой памяти для двух деков. Также можно попытаться обобщить эту задачу на произвольное число деков. В этом случае надо будет рассматривать разные варианты совместного расположения в быстрой памяти FIFO- и LIFO-частей деков.

Исследования, выполненные нашим коллективом, поддержаны грантами РФФИ № 15-01-03404 «Математические модели и оптимальные алгоритмы управления для некоторых методов работы с памятью в параллельных и сетевых устройствах» (2015–2017 гг.) и № 18-01-00125 «Математические модели и алгоритмы оптимального параллельного управления динамическими структурами данных и их реализация в планировщике многопроцессорных параллельных вычислений» (2018–2020 гг.).

Литература

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. Elsevier, 2008. 508 p.
2. Yu-Kwong K., Ishfaq A. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors // ACM Computing Surveys. 1999. Vol. 31, no. 4. P. 406–471. DOI: 10.1145/344588.344618.
3. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems // International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.
4. Beaumont O., Carter L., Ferrante J., *et al.* Centralized versus Distributed Schedulers for Bag-of-Tasks Applications // IEEE Transactions on Parallel and Distributed Systems. 2008. Vol. 19, no. 5. P. 698–709. DOI: 10.1109/TPDS.2007.70747.
5. Xia Y., Prasanna V. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping // Job Scheduling Strategies for Parallel Processing. Springer, 2010. Lecture Notes in Computer Science. Vol. 6253, P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.
6. Hendler D., Shavit N. Non-Blocking Steal-Half Work Queues // Proceedings of the 21st Annual Symposium on Principles of Distributed Computing, PODC '02, Monterey, California, July 21–24, 2002. New York, ACM, 2002. P. 280–289. DOI: 10.1145/571825.571876.
7. Hendler D., Shavit N. Work Dealing // Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '02, Winnipeg, Canada, August 10–13, 2002. New York, ACM, 2002. P. 164–172. DOI: 10.1145/564870.564900.
8. Acar U.A., Chargueraud A., Rainey M. Scheduling Parallel Programs by Work Stealing with Private Deques // ACM SIGPLAN Notices. 2013. Vol. 48, no. 8. P. 219–228. DOI: 10.1145/2517327.2442538.
9. Arora N.S., Blumofe R.D., Plaxton C.G. Thread Scheduling for Multiprogrammed Multiprocessors // Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 1998. P. 119–129. DOI: 10.1145/277651.277678.

10. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey // International Journal of Parallel Programming. 2018. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
11. Blumof R.D., Joerg C.F., Kuszmaul B.C., *et al.* Cilk: An Efficient Multithreaded Runtime System // ACM SIGPLAN Notices. 1995. Vol. 30, no. 8. P. 207–216. DOI: 10.1145/209937.209958.
12. Leijen D., Schulte W., Burckhardt S. The Design of a Task Parallel Library // ACM SIGPLAN Notices. 2009. Vol. 44, no. 10. P. 227–242. DOI: 10.1145/1639949.1640106.
13. Tardieu O., Wang H., Lin H. A Work-Stealing Scheduler for X10's Task Parallelism with Suspension // Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP '12, New Orleans, Louisiana, USA, February 25–29, 2012. New York, ACM, 2012. P. 267–276. DOI: 10.1145/2145816.2145850.
14. Knuth D. The Art of Computer Programming. Volume 1. Addison-Wesley Professional, 1997. 672 p.
15. Blumof R.D., Leiserson C.E. Scheduling Multithreaded Computations by Work Stealing // Journal of the ACM. 1999. Vol. 46, no. 5. P. 720–748. DOI: 10.1145/324133.324234.
16. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System // International Journal of Applied Evolutionary Computation. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.
17. Amelina N., Fradkov A., Jiang Y., Vergados D.J. Approximate Consensus in Stochastic Networks With Application to Load Balancing // IEEE Transactions on Information Theory. 2015. Vol. 61, no. 4. P. 1739–1752. DOI: 10.1109/TIT.2015.2406323.
18. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency // ACM SIGPLAN Notices. 2016. Vol. 51, no. 8. P. 1–13. DOI: 10.1145/3016078.2851151.
19. Robison A., Voss M., Kukanov A. Optimization via Reflection on Work Stealing in TBB // Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '08, Miami, FL, USA, April 14–18, 2008. IEEE, 2008. P. 1–8. DOI: 10.1109/IPDPS.2008.4536188.
20. Dijk T., Pol J.C. Lace: Non-blocking Split Deque for Work-Stealing // Parallel Processing Workshops. Springer, 2014. Lecture Notes in Computer Science. Vol. 8806, P. 206–217. DOI: 10.1007/978-3-319-14313-2_18.
21. Faxén K.-F. Wool-A work stealing library // ACM SIGARCH Computer Architecture News. 2008. Vol. 36, no. 5. P. 93–100. DOI: 10.1145/1556444.1556457.
22. Gmys J., Leroy R., Mezmaš M., *et al.* Work Stealing with Private Integer-Vector-Matrix Data Structure for Multi-core Branch-and-Bound Algorithms // Concurrency and Computation Practice and Experience. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: 10.1002/cpe.3771.
23. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-based Priority Scheduling // ACM SIGPLAN Notices. 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.
24. Chen Q., Guo M., Guan H. LAWS: Locality-Aware Work-Stealing for Multi-socket Multi-core Architectures // Proceedings of the 28th ACM International Conference on

- Supercomputing, ICS'14, Munich, Germany, June 10–13, 2014. New York, ACM, 2014. P. 3–12. DOI: 10.1145/2597652.2597665.
25. Guo H., Chen Q., Guo M., Xu L. SAWS: Selective Asymmetry-Aware Work-Stealing for Asymmetric Multi-core Architectures // Proceedings of the IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12–14, 2016. IEEE, 2016. P. 116–123. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0027.
26. Guo Y., Zhao J., Cave V., Sarkar V. SLAW: A Scalable Locality-aware Adaptive Work-stealing Scheduler for Multi-core Systems // ACM SIGPLAN Notices. 2010. Vol. 45, no. 5. P. 341–342. DOI: 10.1145/1837853.1693504.
27. Mitzenmacher M. Analyses of Load Stealing Models Based on Differential Equations // Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA/PODC98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 1998. P. 212–221. DOI: 10.1145/277651.277687.
28. Wangkai J., Xiangjun P. SLITS: Sparsity-Lightened Intelligent Thread Scheduling // Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '23, Orlando, Florida, United States, June 19–23, 2023. New York, ACM, 2023 P. 21–22. DOI: 10.1145/3578338.3593568.
29. Кучумов Р.И. Реализация и анализ Work-Stealing планировщика задач // Стохастическая оптимизация в информатике. 2016. Т. 12, № 1. С. 20–39.
30. Sokolov A.V., Drac A.V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory // Information Processing Letters. 2013. Vol. 113, no. 19–21. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.
31. Аксёнова Е.А., Лазутина А.А., Соколов А.В. Об оптимальных методах представления динамических структур данных // Обзорение прикладной и промышленной математики. 2003. Т. 10, № 2. С. 375–376.
32. Hendler D., Lev Y., Moir M., Shavit N. A Dynamic-Sized Nonblocking Work Stealing Deque // Distributed Computing. 2006. Vol. 18. P. 189–207. DOI: 10.1007/s00446-005-0144-5.
33. Соколов А.В. Математические модели и алгоритмы оптимального управления динамическими структурами данных. Петрозаводск: Изд-во ПетрГУ, 2002. 215 с.
34. Драц А.В., Соколов А.В. Управление двумя FIFO-очередями в случае их движения друг за другом по кругу // Математические методы распознавания образов. 2011. Т. 15, № 1. С. 315–317.
35. Drac A.V., Sokolov A.V. The Circular Representation of 2 FIFO-Queues in Single Level Memory // Proceedings of the International Conference on Numerical Analysis and Applied Mathematics, ICNAAM-2014, Rhodes, Greece, September 22–28, 2014. AIP Publishing LLC, 2015. Vol. 1648. P. 520002. DOI: 10.1063/1.4912732.
36. Барковский Е.А., Соколов А.В. Способ управления памятью компьютерной системы (RU 2647627 C1). URL: http://www1.fips.ru/registers-doc-view/fips_servlet?DB=RUPAT&DocNumber=2647627&TypeFile=html (дата обращения: 18.07.2023).

37. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques // *Parallel Computing Technologies (PaCT 2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015*. Springer, 2015. Lecture Notes in Computer Science. Vol. 9251, P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.
38. Барковский Е.А., Кучумов Р.И., Соколов А.В. Оптимальное управление двумя work-stealing деками в общей памяти при различных стратегиях перехвата работы // *Программные системы: теория и приложения*. 2017. Т. 8, № 1. С. 83–103. DOI: 10.25209/2079-3316-2017-8-1-83-103.
39. Aksenova E.A., Sokolov A.V. Modeling of the Memory Management Process for Dynamic Work-Stealing Schedulers // *Proceedings of Ivannikov ISPRAS Open Conference, ISPRAS 2017, Moscow, Russia, November 30 – December 1, 2017*. IEEE, 2017. P. 12–15. DOI: 10.1109/ISPRAS.2017.00009.
40. Барковский Е.А., Лазутина А.А., Соколов А.В. Построение и анализ модели процесса работы с двумя деками, двигающимися друг за другом в общей памяти // *Программные системы: теория и приложения*. 2019. Т. 10, № 1. С. 3–17. DOI: 10.25209/2079-3316-2019-10-1-3-17.
41. Aksenova E.A., Barkovsky E.A., Sokolov A.V. The Models and Methods of Optimal Control of Three Work-Stealing Deques Located in a Shared Memory // *Lobachevskii Journal of Mathematics*. 2019. Vol. 40. P. 1763–1770. DOI: 10.1134/S1995080219110052.
42. Kuchumov R., Sokolov A., Korkhov V. Staccato: Shared-Memory Work-Stealing Task Scheduler with Cache-Aware Memory Management // *International Journal of Web and Grid Services*. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.
43. Work-Stealing Task Scheduler. URL: <https://github.com/rkuchumov/staccato>.
44. CPP Distributed Scheduler. URL: <https://gitlab.com/mildlyparallel/cpp-distributed-scheduler>.
45. Хайдарова Р.Р., Муромцев Д.И., Лапаев М.В., Фищенко В.Д. Модель распределенной сверточной нейронной сети на кластере компьютеров с ограниченными вычислительными ресурсами // *Научно-технический вестник информационных технологий, механики и оптики*. 2020. Т. 20. № 5. С. 739–746. DOI: 10.17586/2226-1494-2020-20-5-739-746.
46. Saddik A., Latif R., Ouardi A.E., *et al.* Computer development based embedded systems in precision agriculture: tools and application // *Acta Agriculturae Scandinavica, Section B – Soil & Plant Science*. 2022. Vol. 72, no. 1. P. 589–611. DOI: 10.1080/09064710.2021.2024874.
47. Лазутина А.А., Соколов А.В. Об оптимальном управлении Work-Stealing деками в двухуровневой памяти // *Вестник компьютерных и информационных технологий*. 2020. Т. 17, № 4. С. 51–60. DOI: 10.14489/vkit.2020.04.pp.051-060.
48. Аксёнова Е. А., Лазутина А. А., Соколов А. В. Минимизация средних затрат на перераспределение при работе с work-stealing деком в двухуровневой памяти // *Программные системы: теория и приложения*. 2021. Т. 12, № 2. С. 53–71. DOI: 10.25209/2079-3316-2021-12-2-53-71.
49. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory // *Lobachevskii Journal of Mathematics*. 2021. Vol. 42. P. 1475–1482. DOI: 10.1134/S1995080221070027.

50. Aksenova E., Sokolov A. Minimizing the Average Cost of Redistribution when Working with Two Work-Stealing Deques in Two-Level Memory. Russian Supercomputing Days: Proceedings of Russian Supercomputing Days, Moscow, Russia, September 26–27, 2022. Moscow, MAKS Press, 2022. P. 4–12. URL: https://russianscdays.org/files/2022/RuSCDays22_Proceedings.pdf

Аксёнова Елена Алексеевна, к.ф.-м.н., н.с., Институт прикладных математических исследований КарНЦ РАН (Петрозаводск, Российская Федерация)

Соколов Андрей Владимирович, д.ф.-м.н., в.н.с., Институт прикладных математических исследований КарНЦ РАН (Петрозаводск, Российская Федерация)

DOI: 10.14529/cmse230403

CONTROL METHODS OF WORK-STEALING DEQUES IN DYNAMIC SCHEDULERS OF MULTIPROCESSOR PARALLEL COMPUTATIONS

© 2023 E.A. Aksenova, A.V. Sokolov

Institute of Applied Mathematical Research

of the Karelian Research Centre of the Russian Academy of Sciences

(Pushkinskaya str. 11, Petrozavodsk, 185910 Russia)

E-mail: aksenova@krc.karelia.ru, sokavs@gmail.com

Received: 21.07.2023

In parallel task schedulers, which are using the work-stealing strategy, each processor has own task deque. One end of the deque is used for insertion and deletion of tasks only by the owner, and the other is used for stealing of tasks by other processors. The article offers an overview of work-stealing deque's description of the deque's optimal management problems, which our team had solved for the work-stealing strategy. The idea of the algorithm for deque's managing in two-level memory is that if the memory allocated to the deques becomes overflow, elements are redistributed between memory levels. Elements from the deque's ends are stored in fast memory, since they will be worked with in the near time, and elements from the deque's middle part are stored in slow memory. In this case, it is necessary to determine the required number of elements that need to be left in fast memory, depending on the optimal criteria and system parameters.

Keywords: controlled random walks, optimal control of work-stealing deques, optimal deque caching, optimization of work-stealing load schedulers, simulation and Markov models of optimal control of data structures.

FOR CITATION

Aksenova E.A., Sokolov A.V. Control Methods of Work-stealing Deques in Dynamic Schedulers of Multiprocessor Parallel Computations. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 76–93. (in Russian) DOI: 10.14529/cmse230403.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Herlihy M., Shavit N. The Art of Multiprocessor Programming. Elsevier, 2008. 508 p.
2. Yu-Kwong K., Ishfaq A. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. ACM Computing Surveys. 1999. Vol. 31, no. 4. P. 406–471. DOI: 10.1145/344588.344618.
3. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems. International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.
4. Beaumont O., Carter L., Ferrante J., *et al.* Centralized versus Distributed Schedulers for Bag-of-Tasks Applications. IEEE Transactions on Parallel and Distributed Systems. 2008. Vol. 19, no. 5. P. 698–709. DOI: 10.1109/TPDS.2007.70747.
5. Xia Y., Prasanna V. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping. Job Scheduling Strategies for Parallel Processing. Springer, 2010. Lecture Notes in Computer Science. Vol. 6253, P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.
6. Hendler D., Shavit N. Non-Blocking Steal-Half Work Queues. Proceedings of the 21st annual symposium on Principles of distributed computing, PODC '02, Monterey, California, July 21–24, 2002. New York, ACM, 2002. P. 280–289. DOI: 10.1145/571825.571876.
7. Hendler D., Shavit N. Work Dealing. Proceedings of the 14th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '02, Winnipeg, Canada, August 10–13, 2002. New York, ACM, 2002. P. 164–172. DOI: 10.1145/564870.564900.
8. Acar U.A., Chargueraud A., Rainey M. Scheduling Parallel Programs by Work Stealing with Private Deques. ACM SIGPLAN Notices. 2013. Vol. 48, no. 8. P. 219–228. DOI: 10.1145/2517327.2442538.
9. Arora N.S., Blumof R.D., Plaxton C.G. Thread Scheduling for Multiprogrammed Multiprocessors. Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '98, Puerto Vallarta, Mexico, June 28 – July 2, 1998. New York, ACM, 1998. P. 119–129. DOI: 10.1145/277651.277678.
10. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey. International Journal of Parallel Programming. 2018. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
11. Blumof R.D., Joerg C.F., Kuszmaul B.C., *et al.* Cilk: An Efficient Multithreaded Runtime System. ACM SIGPLAN Notices. 1995. Vol. 30, no. 8. P. 207–216. DOI: 10.1145/209937.209958.
12. Leijen D., Schulte W., Burckhardt S. The Design of a Task Parallel Library. ACM SIGPLAN Notices. 2009. Vol. 44, no. 10. P. 227–242. DOI: 10.1145/1639949.1640106.
13. Tardieu O., Wang H., Lin H. A Work-Stealing Scheduler for X10's Task Parallelism with Suspension. Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP '12, New Orleans, Louisiana, USA, February 25–29, 2012. New York, ACM, 2012. P. 267–276. DOI: 10.1145/2145816.2145850.
14. Knuth D. The Art of Computer Programming. Volume 1. Addison-Wesley Professional, 1997. 672 p.

15. Blumofe R.D., Leiserson C.E. Scheduling Multithreaded Computations by Work Stealing. *Journal of the ACM*. 1999. Vol. 46, no. 5. P. 720–748. DOI: 10.1145/324133.324234.
16. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System. *International Journal of Applied Evolutionary Computation*. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.
17. Amelina N., Fradkov A., Jiang Y., Vergados D.J. Approximate Consensus in Stochastic Networks With Application to Load Balancing. *IEEE Transactions on Information Theory*. 2015. Vol. 61, no. 4. P. 1739–1752. DOI: 10.1109/TIT.2015.2406323.
18. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency. *ACM SIGPLAN Notices*. 2016. Vol. 51, no. 8. P. 1–13. DOI: 10.1145/3016078.2851151.
19. Robison A., Voss M., Kukanov A. Optimization via Reflection on Work Stealing in TBB. *Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '08, Miami, FL, USA, April 14–18, 2008*. IEEE, 2008. P. 1–8. DOI: 10.1109/IPDPS.2008.4536188.
20. Dijk T., Pol J.C. *Lace: Non-blocking Split Deque for Work-Stealing*. *Parallel Processing Workshops*. Springer, 2014. *Lecture Notes in Computer Science*. Vol. 8806, P. 206–217. DOI: 10.1007/978-3-319-14313-2_18.
21. Faxén K.-F. Wool-A work stealing library. *ACM SIGARCH Computer Architecture News*. 2008. Vol. 36, no. 5. P. 93–100. DOI: 10.1145/1556444.1556457.
22. Gmys J., Leroy R., Mezmaž M., *et al.* Work Stealing with Private Integer-Vector-Matrix Data Structure for Multi-core Branch-and-Bound Algorithms. *Concurrency and Computation Practice and Experience*. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: 10.1002/cpe.3771.
23. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-based Priority Scheduling. *ACM SIGPLAN Notices*. 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.
24. Chen Q., Guo M., Guan H. LAWS: Locality-Aware Work-Stealing for Multi-socket Multi-core Architectures. *Proceedings of the 28th ACM International Conference on Supercomputing, ICS'14, Munich, Germany, June 10–13, 2014*. New York, ACM, 2014. P. 3–12. DOI: 10.1145/2597652.2597665.
25. Guo H., Chen Q., Guo M., Xu L. SAWS: Selective Asymmetry-Aware Work-Stealing for Asymmetric Multi-core Architectures. *Proceedings of the IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12–14, 2016*. IEEE, 2016. P. 116–123. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0027.
26. Guo Y., Zhao J., Cave V., Sarkar V. SLAW: A Scalable Locality-aware Adaptive Work-stealing Scheduler for Multi-core Systems. *ACM SIGPLAN Notices*. 2010. Vol. 45, no. 5. P. 341–342. DOI: 10.1145/1837853.1693504.
27. Mitzenmacher M. Analyses of Load Stealing Models Based on Differential Equations. *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA/PODC98, Puerto Vallarta, Mexico, June 28 – July 2, 1998*. New York, ACM, 1998. P. 212–221. DOI: 10.1145/277651.277687.

28. Wangkai J., Xiangjun P. SLITS: Sparsity-Lightened Intelligent Thread Scheduling. Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '23, Orlando, Florida, United States, June 19–23, 2023. New York, ACM, 2023 P. 21–22. DOI: 10.1145/3578338.3593568.
29. Kuchumov R.I. Implementation and Analysis of the Work-Stealing Task Scheduler. Stochastic Optimization in Computer Science. 2016. Vol. 12, no. 1. P. 20–39. (in Russian).
30. Sokolov A.V., Drac A.V. The Linked List Representation of n LIFO-Stacks and/or FIFO-Queues in the Single-Level Memory. Information Processing Letters. 2013. Vol. 113, no. 19–21. P. 832–835. DOI: 10.1016/j.ipl.2013.07.021.
31. Aksenova E.A., Lazutina A.A., Sokolov A.V. On Optimal Methods of Representing Dynamic Data Structures. Review of Applied and Industrial Mathematics. 2003. Vol. 10, no. 2. P. 375–376. (in Russian).
32. Hendler D., Lev Y., Moir M., Shavit N. A Dynamic-Sized Nonblocking Work Stealing Deque. Distributed Computing. 2006. Vol. 18. P. 189–207. DOI: 10.1007/s00446-005-0144-5.
33. Sokolov A.V. Mathematical Models and Algorithms of Optimal Control of Dynamic Data Structures. PetrSU Publishing, 2002. 215 p. (in Russian).
34. Drac A.V., Sokolov A.V. Control of Two FIFO-Queues in the Case of Their Movement One After Another in a Circle. Mathematical Methods of Pattern Recognition. 2011. Vol. 15, no. 1. P. 315–317. (in Russian).
35. Drac A.V., Sokolov A.V. The Circular Representation of 2 FIFO-Queues in Single Level Memory. Proceedings of the International Conference on Numerical Analysis and Applied Mathematics, ICNAAM-2014, Rhodes, Greece, September 22–28, 2014. AIP Publishing LLC, 2015. Vol. 1648. P. 520002. DOI: 10.1063/1.4912732.
36. Barkovsky E.A., Sokolov A.V. A Method of Memory Control of a Computer System (RU 2647627 C1). URL: http://www1.fips.ru/registers-doc-view/fips_servlet?DB=RUPAT&DocNumber=2647627&TypeFile=html (in Russian).
37. Sokolov A., Barkovsky E. The Mathematical Model and the Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques. Parallel Computing Technologies (PaCT 2015): Proceedings of the 13th International Conference, Petrozavodsk, Russia, August 31 – September 4, 2015. Springer, 2015. Lecture Notes in Computer Science. Vol. 9251, P. 102–106. DOI: 10.1007/978-3-319-21909-7_11.
38. Barkovksy E., Kuchumov R., Sokolov A. Optimal Control of Two Deques in Shared Memory with Various Work-Stealing Strategies. Program Systems: Theory and Applications. 2017. Vol. 8, no. 1. P. 83–103. DOI: 10.25209/2079-3316-2017-8-1-83-103. (in Russian).
39. Aksenova E.A., Sokolov A.V. Modeling of the Memory Management Process for Dynamic Work-Stealing Schedulers. Proceedings of Ivannikov ISPRAS Open Conference, ISPRAS 2017, Moscow, Russian Federation, November 30 – December 1, 2017. IEEE, 2017. P. 12–15. DOI: 10.1109/ISPRAS.2017.00009.
40. Barkovsky E., Lazutina A., Sokolov A. The Optimal Control of Two Work-Stealing Deques, Moving One After Another in a Shared Memory. Program Systems: Theory and Applications. 2019. Vol. 10, no. 1. P. 19–32. DOI: 10.25209/2079-3316-2019-10-1-3-17.

41. Aksenova E.A., Barkovsky E.A., Sokolov A.V. The Models and Methods of Optimal Control of Three Work-Stealing Deques Located in a Shared Memory. *Lobachevskii Journal of Mathematics*. 2019. Vol. 40. P. 1763–1770. DOI: 10.1134/S1995080219110052.
42. Kuchumov R., Sokolov A., Korkhov V. Staccato: Shared-Memory Work-Stealing Task Scheduler with Cache-Aware Memory Management. *International Journal of Web and Grid Services*. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.
43. Work-Stealing Task Scheduler. URL: <https://github.com/rkuchumov/staccato>.
44. CPP Distributed Scheduler. URL: <https://gitlab.com/mildlyparallel/cpp-distributed-scheduler>.
45. Khaydarova R.R., Mouromtsev D.I., Lapaev M.V., Fishchenko V.D. Distributed convolutional neural network model on resource-constrained cluster. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2020, Vol. 20, no. 5, P. 739–746. DOI: 10.17586/2226-1494-2020-20-5-739-746.
46. Saddik A., Latif R., Ouardi A.E., *et al.* Computer development based embedded systems in precision agriculture: tools and application. *Acta Agriculturae Scandinavica, Section B – Soil & Plant Science*. 2022. Vol. 72, no. 1. P. 589–611. DOI: 10.1080/09064710.2021.2024874.
47. Lazutina A.A., Sokolov A.V. On Optimal Control of Work-Stealing Deques in Two-Level Memory. *Bulletin of Computer and Information Technologies*. 2020. Vol. 17, no. 4. P. 51–60. DOI: 10.14489/vkit.2020.04.pp.051-060. (in Russian).
48. Aksenova E.A., Lazutina A.A., Sokolov A.V. Minimizing the average cost of redistribution when working with work-stealing deques in two-level memory. *Program Systems: Theory and Applications*. 2021. Vol. 12, no. 2. P. 53–71. DOI: 10.25209/2079-3316-2021-12-2-53-71. (in Russian).
49. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. *Lobachevskii Journal of Mathematics*. 2021. Vol. 42. P. 1475–1482. DOI: 10.1134/S1995080221070027.
50. Aksenova E., Sokolov A. Minimizing the Average Cost of Redistribution when Working with Two Work-Stealing Deques in Two-Level Memory. *Russian Supercomputing Days: Proceedings of Russian Supercomputing Days, Moscow, Russia, September 26–27, 2022*. Moscow, MAKS Press, 2022. P. 4–12. URL: https://russianscdays.org/files/2022/RuSCDays22_Proceedings.pdf

ВОЗМОЖНОСТИ ПАРАЛЛЕЛИЗМА ПРИ ИДЕНТИФИКАЦИИ КВАЗИЛИНЕЙНОГО РЕКУРРЕНТНОГО УРАВНЕНИЯ

© 2023 М.С.А. Аботалеб, Т.А. Макаровских, А.В. Панюков

Южно-Уральский государственный университет

(454080 Челябинск, пр. Ленина, д. 76)

E-mail: abotalebmostafa@bk.ru, Makarovskikh.T.A@susu.ru, paniukovav@susu.ru

Поступила в редакцию: 12.08.2022

Анализ временных рядов и прогнозирование являются одной из широко исследуемых областей. Идентификация с помощью различных статистических методов, нейронных сетей или математических моделей уже давно используется в различных областях исследований от промышленности, до медицины, социальной сферы, аграрной среды. В статье рассматривается параллельный вариант алгоритма идентификации параметров квазилинейного рекуррентного уравнения для решения задачи регрессионного анализа с взаимозависимыми наблюдаемыми переменными, основанный на обобщенном методе наименьших модулей (GLDM). В отличие от нейронных сетей, широко используемых в настоящее время в различных системах прогнозирования, данный подход позволяет в явном виде получать качественные квазилинейные разностные уравнения, адекватно описывающие рассматриваемый процесс. Это позволяет повысить качество анализа изучаемых процессов. Существенным преимуществом модели, использующей обобщенный метод наименьших модулей, по сравнению с многочисленными нейросетевыми подходами является возможность интерпретации коэффициентов модели с точки зрения задачи исследования и использование полученного уравнения в качестве модели динамического процесса. Проведенные вычислительные эксперименты с использованием временных рядов показывают, что максимальное ускорение алгоритма происходит при использовании количества потоков, равного половине возможных потоков для данного устройства.

Ключевые слова: параллелизм, квазилинейное рекуррентное уравнение, прогнозирование, моделирование, авторегрессионная модель.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аботалеб М.С.А., Макаровских Т.А., Панюков А.В. Исследование возможностей параллелизма для прогнозирования с использованием квазилинейного рекуррентного уравнения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 94–109. DOI: 10.14529/cmse230404.

Введение

Анализ временных рядов и прогнозирование в настоящее время являются одной из широко исследуемых областей. Данные подходы используются как при анализе и моделировании экономических процессов, транспортных систем [1], так и для исследования процессов, происходящих в технических системах [2], при анализе климатических процессов и прогнозировании погодных явлений [1], для прогнозирования состояния сельскохозяйственных угодий [3] и др. В 2020–2022 годах к разработке методов прогнозирования добавилось тысячи исследований о распространении заболеваемости Covid-19 по всему миру [4–6].

В настоящее время накоплен большой опыт измерения вибрационных сигналов, разработки методов вибродиагностики и прогнозирования состояния и ресурса механических систем. Одним из наиболее актуальных направлений остается повышение точности и скорости определения диагностических признаков. Это относится, прежде всего, к уникальным высоконагруженным механическим системам, рассмотренным, например, в работе [7]. Решение

этой проблемы во многих случаях можно получить через динамические характеристики механических систем. Определению этих характеристик в значительной степени способствует правильный выбор диагностической математической модели, устанавливающей связь между пространством состояний объекта и пространством диагностических признаков. К ним относятся динамические модели, представленные в виде разностных уравнений, феноменологические, структурные, регрессионные модели и т.п. Выбор той или иной модели зависит от определяемых характеристик и характера анализируемого процесса.

Идентификация с помощью различных статистических методов, нейронных сетей или математических моделей уже давно используется в различных областях исследований. В настоящее время эти методы применяются не только в промышленности, но и при попытках прогнозирования развития пандемии Covid-19. Например, в работе [8] сравнивается качество прогнозирования процесса распространения инфекции с помощью разных классических моделей, описывается разработанное программное обеспечение для всех этих методов и проводятся вычислительные эксперименты с использованием временных рядов по распространению заболеваемости Covid-19 в различных регионах. Авторы приходят к выводу, что разработанную систему прогнозирования можно использовать для анализа временных рядов, описывающих и другие процессы.

Большинство прогнозов, особенно при большом количестве данных, зачастую проводится с использованием различных моделей нейронных сетей. Например, в статье [9] рассматривается нейросетевая модель, с помощью которой можно прогнозировать изменение цены ферросилиция на внутреннем рынке РФ в краткосрочной перспективе. Приведенная в статье модель отличается высокой точностью прогнозирования и может быть полезна при обосновании стратегических решений в деятельности отраслевых НИИ и металлургических предприятий. В работе [10] описаны эконометрические модели для определения качественного экономического показателя металлургической отрасли, производства, применимые для оценки статистических характеристик производства черных металлов и перспектив развития черной металлургии. Тем не менее, все подобные модели выглядят как волшебный черный ящик, позволяющий получить некий адекватный ответ для каких-то входных данных. Некоторые исследователи используют так называемое когнитивное моделирование для повышения качества прогнозирования с помощью нейронных сетей. Так, статья [11] направлена на сравнение эффективности когнитивных и математических предикторов временных рядов с точки зрения их точности. Эксперимент, проведенный авторами статьи, показывает, что когнитивные модели имеют по крайней мере эквивалентную точность по сравнению с моделями ARIMA. Большинство этих подходов используются для прогнозирования некоторых экономических единиц, объема производства, некоторых параметров логистики и т. д.

Поскольку все перечисленные выше модели дают хорошую точность при краткосрочном прогнозировании, актуальна задача разработки математического подхода, позволяющего в явном виде получать качественные квазилинейные разностные уравнения (адекватно описывающие рассматриваемый процесс) в долгосрочной перспективе. Известны некоторые исследования в этой области, такие как [12], где предлагаемая модель включает очистку данных, сглаживание данных и окончательные данные после предварительной обработки, которые вводятся в регрессионную модель для прогнозирования промышленного потребления электроэнергии. Но в этой статье, как и во многих других, снова рассматриваются только известные статистические методы.

В данной работе для решения задачи регрессионного анализа с взаимозависимыми наблюдаемыми переменными рассматривается алгоритм идентификации параметров квазилинейного рекуррентного уравнения, основанный на обобщенном методе наименьших отклонений (General Least Deviation Method, GLDM). В отличие от нейронных сетей (например, [2]) этот подход позволяет в явном виде получать качественные квазилинейные разностные уравнения (адекватно описывающие рассматриваемый процесс).

В работе использован алгоритм из [13], реализованный в виде комплекса программ [14] для определения коэффициентов $a_1, a_2, a_3 \dots, a_m \in \mathbb{R}$ квазилинейной авторегрессионной модели m -го порядка

$$y_t = \sum_{j=1}^{n(m)} a_j g_j(\{y_{t-k}\}_{k=1}^m) + \varepsilon_t, \quad t = 1, 2, \dots, T \quad (1)$$

по актуальной информации о значениях переменных состояния $\{y_t \in \mathbb{R}\}_{t=1-m}^T$ в моменты времени t . Здесь $g_j : (\{y_{t-k}\}_{k=1}^m) \rightarrow \mathbb{R}, j = 1, 2, \dots, n(m)$ — заданные $n(m)$ модельных функций; $\{\varepsilon_t \in \mathbb{R}\}_{t=1}^T$ — неизвестные ошибки.

Найденные коэффициенты $a_1, a_2, a_3 \dots, a_m \in \mathbb{R}$ в рассматриваемом случае используются для анализа количества кумулятивных случаев заражения Covid-19 в Челябинской области с 22 марта 2020 г. по 06 января 2022 г. (всего 655 дней распространения инфекции), прогнозирование дальнейшего развития процесса и обсуждение эффективности использования данной модели для прогнозирования событий, связанных с распространением Covid-19. В своем исследовании мы используем временной ряд, начиная с 22 марта 2020 г., поскольку до этой даты для рассматриваемого региона были нулевые значения. Тем не менее, данный подход можно масштабировать и на другие прикладные задачи.

Поскольку при работе с длинными временными рядами потребуются матрицы высокого порядка, для чего необходимы большие затраты памяти и времени, в работе исследована реализация [14] алгоритма прогнозирования, использующая параллелизм. Для исследования возможности многопоточного программирования разработанного алгоритма прогнозирования проведены вычислительные эксперименты на двух разных ПК.

Статья организована следующим образом. В разделе 1 введены основные обозначения, описана схема реализации модели, подробно расписан ход решения поставленной задачи, приведен алгоритм прогнозирования. В разделе 2 описываются особенности программной реализации представленных алгоритмов с использованием распараллеливания. В разделе 3 приводятся результаты вычислительных экспериментов для анализа ускорения алгоритма для разных наборов данных и разных вычислителей. В заключении перечислены полученные в работе результаты, отмечены направления дальнейших исследований.

1. Обозначения и описание алгоритмов

Рассматриваемый алгоритм работает следующим образом (см. рис. 1). Алгоритм GLDM [13] на входе получает временной ряд $\{y_t \in \mathbb{R}\}_{t=-1-m}^T$ длины $T + m \geq (1 + 3m + m^2)$ и определяет коэффициенты $a_1, a_2, a_3 \dots, a_m$, решая задачу оптимизации

$$\{a_j^*\}_{j=1}^{n(m)} = \arg \min_{\{a_j\}_{j=1}^{n(m)} \subset \mathbb{R}} \left[\sum_{t=1}^T \arctan \left| \sum_{j=1}^{n(m)} a_j g_j(\{y_{t-k}\}_{k=1}^m) - y_t \right| \right], \quad (2)$$



Рис. 1. Схема реализации модели

с использованием в функции потерь распределения Коши

$$F(\xi) = \frac{1}{\pi} \arctan(\xi) + \frac{1}{2},$$

которое имеет максимальную энтропию среди распределений случайных величин, не имеющих математического ожидания и дисперсии.

Ниже приведен пример полного набора квадратичных модельных функций для разностного уравнения порядка m .

$$g_j(\{y_{t-k}\}_{k=1}^m) = y_{t-j}, \quad j = 1, 2, \dots, m$$

$$g_j(\{y_{t-k}\}_{k=1}^m) = y_{t-r} \cdot y_{t-s}, \quad 1 \leq r \leq s \leq m,$$

$$j = m + 1, m + 2, \dots, n(m) = m + C_{m+1}^2 = m(m + 3)/2.$$

Предиктор формирует индексированное $t = 1, 2, \dots, T-1, T$ семейство разностных уравнений m -го порядка

$$\overline{y[t]}_\tau = \sum_{j=1}^{n(m)} a_j^* g_j(\{\overline{y[t]}_{\tau-k}\}_{k=1}^m), \quad \tau = t, t + 1, t + 2, t + 3, \dots, T - 1, T, T + 1, \dots \quad (3)$$

для решетчатых функций $\overline{y[t]}$ со значениями $\overline{y[t]}_\tau$, которые интерпретируются как построенный в момент времени t прогноз для y_τ . Воспользуемся решением задачи Коши для разностного уравнения (3) при начальных условиях

$$\overline{y[t]}_{t-1} = y_{t-1}, \overline{y[t]}_{t-2} = y_{t-2}, \dots, \overline{y[t]}_{t-m} = y_{t-m}, \quad t = 1, 2, \dots, T - 1, T \quad (4)$$

и определим значения функции $\overline{y[t]}$.

Итак, имеем множество $\overline{Y}_\tau = \{\overline{y[t]}_\tau\}_{t=1}^T$ возможного прогнозного значения y_τ . Далее используем этот набор для оценки вероятностных характеристик величины y_τ .

1.1. Оценка по обобщенному методу наименьших отклонений

Задача (2), т.е. GLDM-оценка, представляет собой задачу многоэкстремальной оптимизации. GLDM-оценки устойчивы к наличию корреляции значений в $\{y_t \in \mathbb{R}\}_{t=-1-m}^T$ и (при соответствующих параметрах) лучше всего подходят для вероятностных распределений ошибок с более тяжелыми (чем у нормального распределения) хвостами [7]. Все вышесказанное показывает возможность решения задачи идентификации (1) с использованием решения (2).

Используем взаимосвязь между GLDM-оценками и оценками взвешенным методом наименьших отклонений [15] (WLDM-оценки) для решения задач (2) более высокой размерности.

В данной статье рассмотрим алгоритм оценивания GLDM [16]. Начнем с алгоритма оценки взвешенного метода наименьших отклонений (Weighted Least Deviation Method, WLDM), используемого в алгоритме GLDM.

1.2. Оценка по взвешенному методу наименьших отклонений

Алгоритм WLDM-оценки [13] в качестве входных данных получает временной ряд $\{y_t \in \mathbb{R}\}_{t=1-m}^T$ и весовые коэффициенты $\{p_t \in \mathbb{R}^+\}_{t=1}^T$ и вычисляет множители

$$a_1, a_2, a_3 \dots, a_{n(m)} \in \mathbb{R}$$

путем решения задачи оптимизации

$$\{a_j^*\}_{j=1}^{n(m)} = \arg \min_{\{a_j\}_{j=1}^{n(m)} \in \mathbb{R}^{n(m)}} \left[\sum_{t=1}^T p_t \cdot \left| \sum_{j=1}^{n(m)} a_j g_j(\{y_{t-k}\}_{k=1}^m) - y_t \right| \right]. \quad (5)$$

Эта задача представляет собой задачу выпуклой кусочно-линейной оптимизации, а введение дополнительных переменных сводит ее к задаче линейного программирования

$$\sum_{t=1}^T p_t z_t \rightarrow \min_{\substack{(a_1, a_2, \dots, a_{n(m)}) \in \mathbb{R}^m, \\ (z_1, z_2, \dots, z_T) \in \mathbb{R}^T}} \quad (6)$$

$$-z_t \leq \sum_{j=1}^{n(m)} [a_j g_j(\{y_{t-k}\}_{k=1}^m)] - y_t \leq z_t, \quad t = 1, 2, \dots, T, \quad (7)$$

$$z_t \geq 0, \quad t = 1, 2, \dots, T. \quad (8)$$

Задача (6)–(8) имеет каноническую форму $n(m) + T$ с переменными и $3n$ ограничениями-неравенствами, включая условия неотрицательности для переменных z_j , $j = 1, 2, \dots, T$.

Двойственная задаче (6) является задача

$$\sum_{t=1}^T (u_t - v_t) y_t \rightarrow \max_{u, v \in \mathbb{R}^T}, \quad (9)$$

$$\sum_{t=1}^T a_j g_j(\{y_{t-k}\}_{k=1}^m) (u_t - v_t) = 0, \quad j = 1, 2, \dots, n(m), \quad (10)$$

$$u_t + v_t = p_t, \quad u_t, v_t \geq 0, \quad t = 1, 2, \dots, T. \quad (11)$$

Введем переменные $w_t = u_t - v_t$, $t = 1, 2, \dots, T$. Условия (11) подразумевают

$$u_t = \frac{p_t + w_t}{2}, \quad v_t = \frac{p_t - w_t}{2}, \quad -p_t \leq w_t \leq p_t, \quad t = 1, 2, \dots, T.$$

Поэтому оптимальное значение задачи (9)–(11) равно оптимальному решению задачи

$$\sum_{t=1}^T w_t \cdot y_t \rightarrow \max_{w \in \mathbb{R}^T}, \quad (12)$$

$$\sum_{t=1}^T g_j(\{y_{t-k}\}_{k=1}^m) \cdot w_t = 0, \quad j = 1, 2, \dots, n(m), \quad (13)$$

$$-p_t \leq w_t \leq p_t, \quad t = 1, 2, \dots, T. \quad (14)$$

Ограничения (13) определяют $(T - n(m))$ -мерное линейное подпространство \mathcal{L} с матрицей

$$S = \begin{bmatrix} g_1(\{y_{1-k}\}_{k=1}^m) & g_1(\{y_{2-k}\}_{k=1}^m) & \cdots & g_1(\{y_{T+1-k}\}_{k=1}^m) \\ g_2(\{y_{1-k}\}_{k=1}^m) & g_2(\{y_{2-k}\}_{k=1}^m) & \cdots & g_2(\{y_{T+1-k}\}_{k=1}^m) \\ \vdots & \vdots & \ddots & \vdots \\ g_{n(m)}(\{y_{1-k}\}_{k=1}^m) & g_{n(m)}(\{y_{2-k}\}_{k=1}^m) & \cdots & g_{n(m)}(\{y_{T+1-k}\}_{k=1}^m) \end{bmatrix}$$

Ограничения (14) определяют T -мерный брус \mathcal{T} . Простая структура допустимого множества задачи (12)–(14): пересечение $(T - n(m))$ -мерного линейного подпространства \mathcal{L} (13) и T -мерного бруса \mathcal{T} (14), — позволяют найти решение алгоритмом, использующим проекцию градиента целевой функции (12) (т.е. вектор $\nabla = \{y_t\}_{t=1}^T$) на допустимой области $\mathcal{L} \cap \mathcal{T}$, которая определяется формулой ограничения (13)–(14). Матрица оператора проектирования на \mathcal{L} имеет вид

$$S_{\mathcal{L}} = E - S^T \cdot (S \cdot S^T)^{-1} \cdot S,$$

а проекция градиента на \mathcal{L} равна $\nabla_{\mathcal{L}} = S_{\mathcal{L}} \cdot \nabla$. Кроме того, если внешняя нормаль на какой-либо грани бруса образует острый угол с проекцией градиента $\nabla_{\mathcal{L}}$, то перемещение по этой грани равно нулю.

Алгоритм **DualWLDMSolver** решает задачу (12)–(14), он начинает поиск оптимального решения с нуля, двигаясь по направлению $\nabla_{\mathcal{L}}$. Если текущая точка попадает на грань бруса \mathcal{T} , то соответствующая координата в направлении движения принимается равной нулю. Вычислительная сложность такого алгоритма не превышает величины $O(T^2)$ благодаря простой структуре допустимого множества: пересечение T -мерного кубоида (14) и $(T - n(m))$ -мерного линейного подпространства (13).

Если (w^*, R^*) — результат выполнения алгоритма проектирования градиента, то w^* — оптимальное решение задачи (12)–(14), а оптимальное решение задачи 9–11 равно

$$u_t^* = \frac{p_t + w_t^*}{2}, \quad v_t^* = \frac{p_t - w_t^*}{2}, \quad t = 1, 2, \dots, T.$$

1.3. Решение прямой задачи

Решение прямой задачи следует из условия дополненности для пары взаимно двойственных задач (6)–(8) и (9)–(11)

$$y_t = \sum_{j=1}^{n(m)} [a_j g_j(\{y_{t-k}\}_{k=1}^m)] \quad \forall t \notin R^*, \quad (15)$$

$$y_t = \sum_{j=1}^{n(m)} [a_j g_j(\{y_{t-k}\}_{k=1}^m)] + z_t^*, \quad \forall t \in R^* : w_t^* = p_t, \quad (16)$$

$$y_t = \sum_{j=1}^{n(m)} [a_j g_j(\{y_{t-k}\}_{k=1}^m)] - z_t^*, \quad \forall t \in R^* : w_t^* = -p_t. \quad (17)$$

Фактически решение $(\{a_j^*\}_{j=1}^{n(m)}, z^*)$ системы линейных алгебраических уравнений (15)–(17) представляет собой оптимальное решение двойственной задачи (12)–(14) и оптимальное решение задачи (5), что доказывает справедливость следующей теоремы.

Теорема 1. Пусть w^* — оптимальное решение задачи (12)–(14), $(\{a_j^*\}_{j=1}^{n(m)}, z^*)$ — решение системы линейных алгебраических уравнений (15)–(17), тогда $(\{a_j^*\}_{j=1}^{n(m)})$ — оптимальное решение задачи (5).

Вышеизложенное позволяет нам предложить алгоритм WLDM-оценки. Основной проблемой при использовании этого алгоритма является отсутствие общих формальных правил выбора весовых коэффициентов. Следовательно, такой подход требует дополнительных исследований.

Установленные в [16] результаты позволяют свести задачу определения GLDM-оценки к итерационной процедуре с WLDM-оценками.

1.4. Алгоритм оценки по обобщенному методу наименьших отклонений

Задача (2) оценки GLDM является задачей вогнутой оптимизации. GLDM-оценки устойчивы к наличию корреляции значений в $\{S_{jt} : t = 1, 2, \dots, T; j = 1, 2, \dots, N\}$, и (с соответствующими настройками) как лучший для вероятностных распределений ошибок с более тяжелыми (чем у нормального распределения) хвостами [7]. Вышеизложенное показывает возможность решения задачи идентификации (1) путем решения последовательности задач (15) с соответствующими весами. Установленные в [16] результаты позволяют свести задачу определения GLDM-оценки к итерационной процедуре с WLDM-оценками.

Теорема 2. Последовательность $\{(A^{(k)}, z^{(k)})\}_{k=1}^{\infty}$, построенная по алгоритму GLDM-оценки, сходится к глобальному минимуму (a^*, z^*) задачи (2).

Описание алгоритма **GLDM-оценка** показывает, что его вычислительная сложность пропорциональна вычислительной сложности алгоритма решения простых и/или двойственных задач WLDM (5). Многочисленные вычислительные эксперименты показывают, что среднее число итераций алгоритма **GLDM-оценки** равно количеству коэффициентов в идентифицированном уравнении. Если эта гипотеза верна, то вычислительная сложность решения практических задач не превышает $O((n(m))^3 T + n(m) \cdot T^2)$. Необходимо учитывать, что поиск и нахождение уравнения авторегрессии высокого порядка имеют свои специфи-

ческие условия. Одним из таких условий, в частности, является высокая чувствительность алгоритма к ошибкам округления. Для исключения возможности ошибки в вычислениях необходимо точно выполнять основные арифметические операции над полем рациональных чисел [17] и дополнять их параллельным программным кодом.

1.5. Алгоритм прогнозирования

Предиктор формирует индексированное $t = 1, 2, \dots, T-1, T$ семейство разностных уравнений m -го порядка (3) для решетчатых функций $\overline{y[t]}$ со значениями $\overline{y[t]}_\tau$ и интерпретируется как построенный на момент времени t прогноз для y_τ . Воспользуемся решением задачи Коши для ее разностного уравнения (3) при начальных условиях (4), чтобы найти значения функции $\overline{y[t]}$.

Итак, у нас есть множество $\overline{Y}_\tau = \{\overline{y[t]}_\tau\}_{t=1}^T$ возможного предсказания значения y_τ . Далее мы используем этот набор для оценки вероятностных характеристик величины y_τ .

Схема алгоритма приведена на рис. 2.

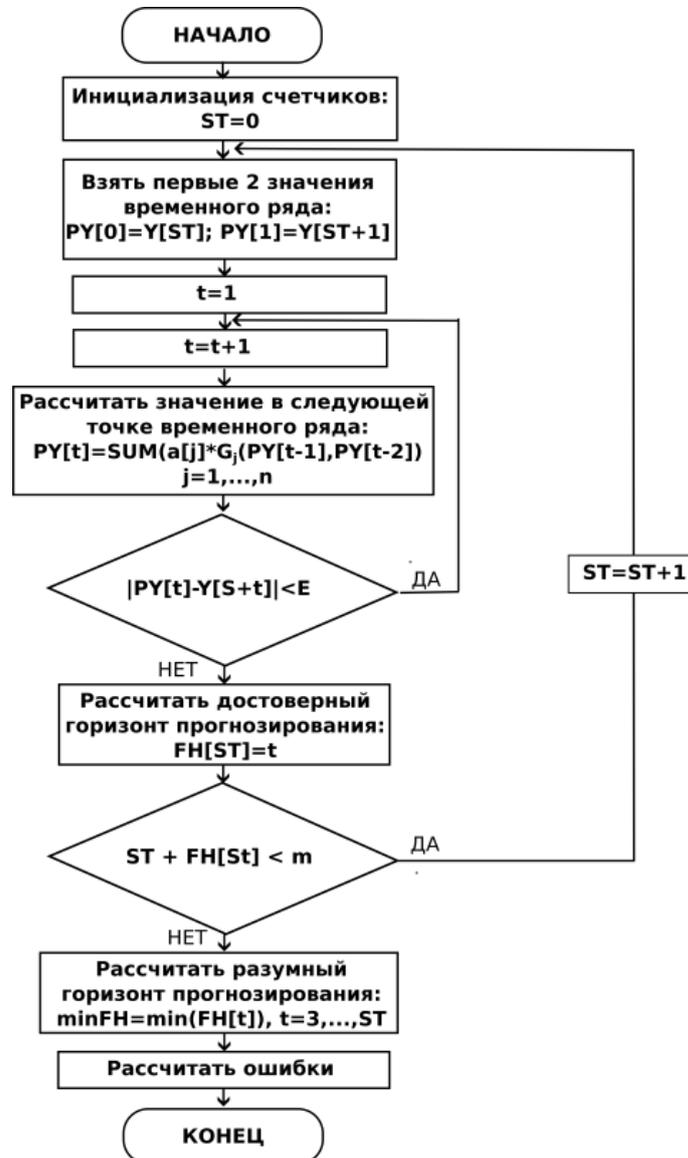


Рис. 2. Схема алгоритма прогнозирования



Рис. 3. Последовательность вызова функций из основной функции программы

2. Программная реализация

В рассматриваемой программе используется распараллеливание средствами OpenMP, что предполагает модель параллельного программирования SPMD. В этом случае для всех параллельных потоков используется один и тот же код, а в программу вставляются специальные директивы. Программа имеет последовательные области, когда выполняется один процесс (поток), а при входе в параллельную область порождается определенное количество процессов, между которыми в дальнейшем распределяются части кода. Максимально возможное количество потоков в рассматриваемой программе определяется с помощью функции `omp_get_max_threads()` в зависимости от ресурсов компьютера, на котором запущена программа. В вычислительном эксперименте, представленном в следующем разделе, программа запускается с количеством потоков от 1 до максимально возможного значения для конкретного компьютера.

На рис. 3 приведена последовательность вызова разработанных процедур из функции `void main()`, используемая для запуска алгоритма. Листинги всех функций, вызываемых из `main()`, приведены в [14]. В разработанной программе распараллелено выполнение всех циклов один за другим. Циклы, которые не могут быть адекватно распараллелены и остаются последовательными областями: (1) первый цикл функции решения прямой задачи WLDM; (2) базовый цикл `do...while` алгоритма оценки GLDM (вложенные циклы распараллелены). Все остальные циклы могут выполняться как параллельные участки.

Функция прогнозирования `ForecastingEst()` (схема алгоритма приведена на рис. 2) не может быть адекватно распараллелена, поскольку в ней осуществляется расчет рекуррентной последовательности значений. Возможности параллельных вычислений могут быть использованы только во вложенном цикле, где осуществляется вычисление значения в за-

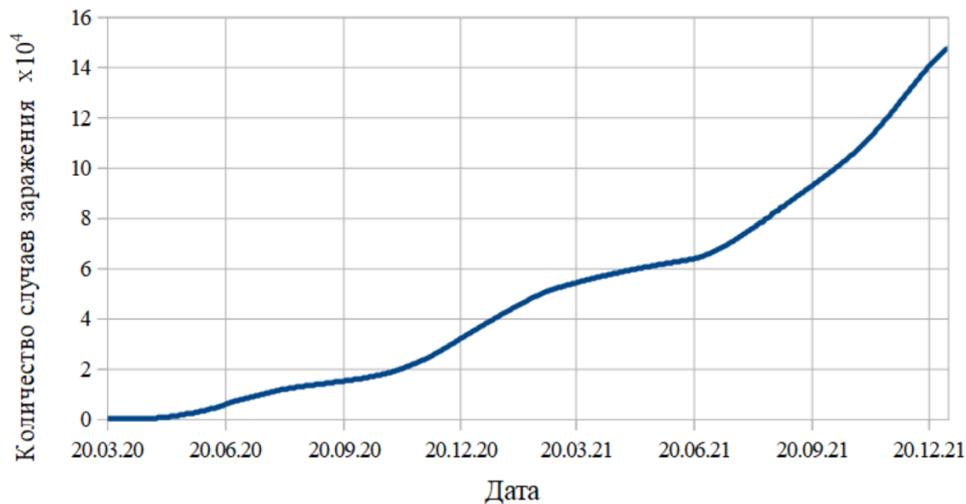


Рис. 4. Прирост числа кумулятивных случаев Covid-19 по Челябинской области (данные с 23.03.2020 по 06.01.2022)

данной точке временного ряда. Тем не менее, такое использование не представляется разумным, поскольку число коэффициентов модели не велико. Потому единственный цикл, в котором использованы директивы OpenMP, — это цикл, отвечающий за поиск надежного горизонта прогнозирования, который заключается в поиске минимального значения в массиве. В результате работы этой функции формируется массив $\overline{y}[t]$ каждый элемент которого равен значениям решеточных функций $\overline{y}[t]_\tau$ семейства разностных уравнений m -го порядка (3). Используя этот массив, можно получить значения ошибок MBE (равна сумме ошибок предсказания $Y[i]$ по значениям $Y[i-T-1]$ и $Y[i-T]$) и MAE (равна сумме абсолютных ошибок предсказания $Y[i]$ по значениям $Y[i-T-1]$ и $Y[i-T]$):

$$MBE = \sum_{t=3}^{T_R} (\overline{y}[t] - \overline{y}[t]_\tau), \quad MAE = \sum_{t=3}^{T_R} |\overline{y}[t] - \overline{y}[t]_\tau|,$$

где T_R — надежный горизонт прогнозирования, т.е.

$$T_R = \min \left\{ \tau : z > \left| \overline{y}[t] - \overline{y}[t]_\tau \right| \right\}.$$

3. Вычислительные эксперименты. Анализ ускорения работы алгоритма при распараллеливании процесса вычислений

При проведении экспериментов были использованы два временных ряда.

1. Временной ряд, содержащий кумулятивные данные по случаям заражения вирусом Covid-19 в Челябинской области. В эксперименте рассматриваются векторы разной длины, самый длинный из которых имеет 655 значений. График процесса показан на рис. 4.
2. Один из рядов лаборатории OikoLab (<https://oikolab.com>), который содержит почасовые климатические данные рядом с университетом Монаш, Клейтон, Виктория, Австралия, с 01 января 2010 г. по 31 мая 2021 г. Длина ряда порядка 10^5 значений.

Полученные средние ошибки прогноза разработанной модели для ряда (1) составляют $MBE = 1.615210 \cdot 10^{-1}$ и $MAE = 9.852680 \cdot 10^{-1}$ при доверительном горизонте реализации

Таблица 1. Время вычислений для разного количества потоков и длин векторов

N	Количество нитей							
	1	2	3	4	5	6	7	8
Intel Core i7-4770, 8 Gb ОЗУ, 8 ядер								
150	0.0970	0.0690	0.0490	0.0420	0.0320	0.0300	0.0420	0.0300
300	0.236	0.164	0.131	0.102	0.114	0.099	0.091	0.107
500	0.624	0.411	0.353	0.347	0.317	0.249	0.232	0.229
655	1.05	0.76	0.63	0.57	0.45	0.41	0.38	0.38
Intel Core i5-2430M, 4 Gb ОЗУ, 4 ядра								
150	0.642	0.461	0.449	0.421	0.424	0.421	0.419	0.417
300	2.46	1.77	1.73	1.64	1.64	1.63	1.63	1.62
500	6.67	4.85	4.72	4.49	4.51	4.49	4.46	4.50
655	11.47	8.33	8.15	7.71	8.30	7.76	7.68	7.62

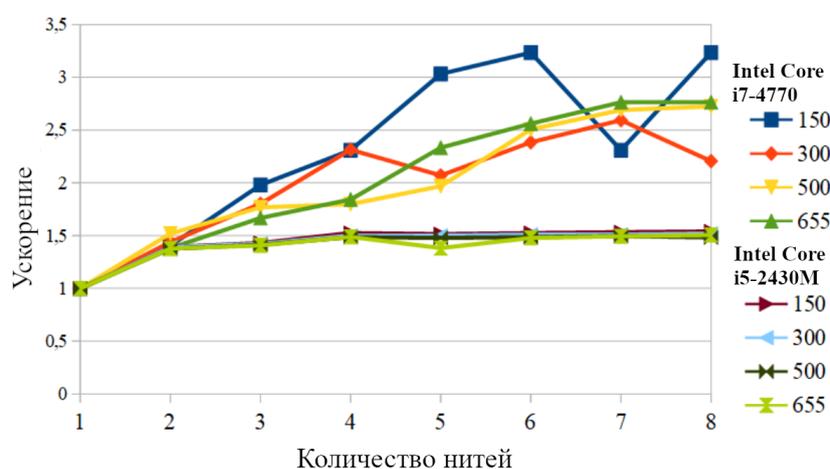


Рис. 5. График ускорения работы параллельной версии программы для временных рядов на исследуемых вычислительных ресурсах

645 дней. Этот результат лучше результатов, полученных с помощью классической статистической модели и модели нейронных сетей, поскольку наша модель позволяет получить долгосрочный прогноз (тем не менее, ARIMA, линейная модель Хольта—Уинтерса, NNAR, LSTM позволяют получать хорошие прогнозы только на 30–60 дней, т.к. это было показано в вычислительных экспериментах, описанных в [18]).

Исследуем время выполнения и ускорение при распараллеливании для векторов длиной 150, 300, 500 и 655 элементов. Эксперименты проводились на двух персональных компьютерах. Для компьютера Intel Core i7-4770, 3.4GHz, RAM 16Gb программа определила максимальное количество потоков равное 8, а для компьютера Intel Core i5-2430M, 2.4 GHz, RAM 4 Gb — 4 потока. Для нашего эксперимента зафиксируем количество потоков, равное 8 для обоих компьютеров. Время выполнения разработанной программы для разного количества N потоков показано в табл. 1. На рис. 5 приводится полученное ускорение при вычислении рассматриваемых временных рядов для прогнозирования заболеваемости Covid-19 на двух исследуемых персональных компьютерах. При рассмотрении более длинных временных рядов, например, ряда (2), получим график, приведенный на рис. 6. Эксперименты

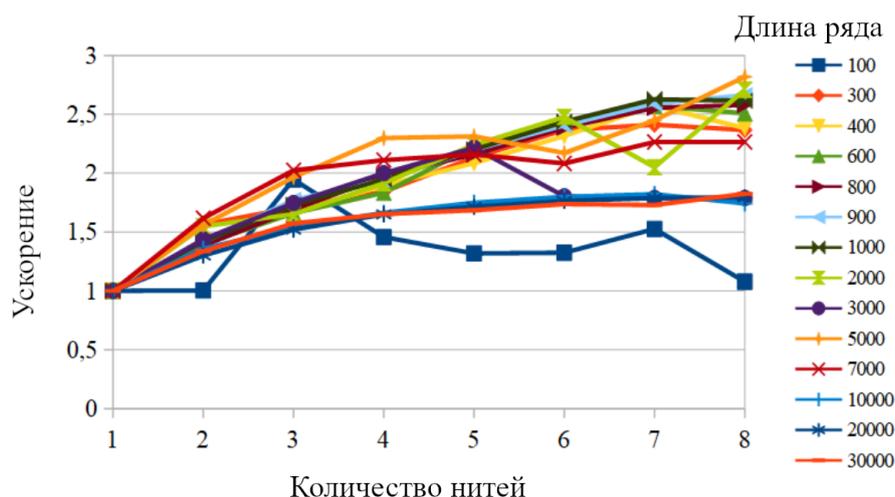


Рис. 6. График ускорения работы параллельной версии программы для временных рядов разной длины на компьютере с процессором Intel Core i7-4770

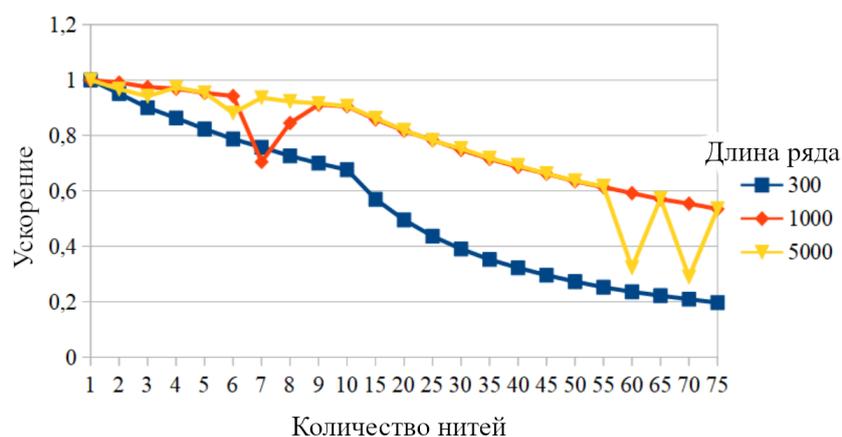


Рис. 7. Графики ускорения вычислений при использовании вычислительного комплекса «Нейрокомпьютер ЮУрГУ» для анализа временных рядов разной длины

показывают, что в лучшем случае имеем ускорение в 3.2 раза на персональном компьютере, позволяющем получить 8 нитей. В среднем ускорение составляет величину 2.7 раз. Проведение экспериментов на более мощных вычислителях не оправдано, поскольку для временных рядов даже размерности порядка 10^3 – 10^5 элементов возникают значительные накладные расходы (см. рис. 7). Еще одним из возможных применений данного алгоритма является анализ урожайности сельскохозяйственных угодий на основании снимков, полученных с помощью аэрофотосъемки. В данном случае будем за один посевной сезон иметь относительно короткие временные ряды (порядка 50–60 точек), тем не менее, количество таких рядов будет равно числу участков, обладающих определенными свойствами, описывающих отдельно взятое поле. Тогда параллелизм можно использовать не для самого алгоритма получения коэффициентов модели и построения прогнозов, а для разделения по потокам различных временных рядов. Такое распараллеливание, очевидно, будет более эффективным при использовании не только персональных компьютеров, но и более мощных вычислительных ресурсов. Данный вопрос является темой отдельного исследования.

Заключение

У модели, использующей обобщенный метод наименьших модулей (GLDM) есть одно существенное преимущество по сравнению с многочисленными нейросетевыми подходами, заключающееся в возможности интерпретировать коэффициенты модели с точки зрения задачи исследования. Поскольку для запуска разработанного алгоритма необходимо использовать матрицы высокого порядка, полезно распараллелить алгоритм с целью ускорения процесса получения результата. Проведенные вычислительные эксперименты показывают, что при простейших способах распараллеливания рассмотренных алгоритмов имеем ускорение алгоритма в три раза на современных персональных компьютерах при использовании половины из возможных потоков. Тестирование алгоритма на более мощных вычислителях показало не только отсутствие эффекта от распараллеливания для временных рядов длиной до 100 000 значений, но и значительное замедление вычислений за счет накладных расходов на организацию параллельных областей в каждой из функций. Таким образом, для реализации алгоритма идентификации параметров одного квазилинейного рекуррентного соотношения нецелесообразно использовать возможности параллельных вычислений.

Дальнейшие исследования возможностей использования параллелизма для алгоритма идентификации коэффициентов модели с помощью обобщенного метода наименьших модулей будут посвящены использованию алгоритма для решения прикладных задач (прогнозирование урожайности поля), где будет присутствовать достаточно большое число коротких временных рядов, и потребуется распределение отдельных подзадач по разным вычислительным процессам.

Литература

1. Li Q., Wang J., Zhang H. A wind speed interval forecasting system based on constrained lower upper bound estimation and parallel feature selection // *Knowl. Based Syst.* 2021. Vol. 231. DOI: 10.1016/j.knosys.2021.107435.
2. Khashei M., Chahkoutahi F. A comprehensive low-risk and cost parallel hybrid method for electricity load forecasting // *Comput. Ind. Eng.* 2021. Vol. 155. DOI: 10.1016/j.cie.2021.107182.
3. Supuwiningasih N.N., Kadeksukerti N., Putra A., Dewanti P. Forecasting of Agricultural Production Results in South Denpasar Using Quadratic Trend Method Based GIS // *International Journal of Engineering Technologies and Management Research.* 2018. Vol. 5, no. 2. DOI: 10.5281/zenodo.1186523.
4. Hamdi F., Raby H., Hakim G., *et al.* A Generalized Mechanistic Model for Assessing and Forecasting the Spread of the COVID-19 Pandemic // *IEEE Access.* 2021. Vol. 9. P. 13266–13285. DOI: 10.1109/ACCESS.2021.3051929.
5. Dash S., Chakraborty C., Giri S.K., *et al.* BIFM: Big-Data Driven Intelligent Forecasting Model for COVID-19 // *IEEE Access.* 2021. Vol. 9. P. 97505–97517. DOI: 10.1109/ACCESS.2021.3094658.
6. Corpas-Burgos F., Martinez-Beneito M.A. An Autoregressive Disease Mapping Model for Spatio-Temporal Forecasting // *Mathematics.* 2021. Vol. 9, no. 4. Article 384. DOI: 10.3390/math9040384.

7. Panyukov A.V., Tyrsin A.N. Stable Parametric Identification of Vibratory Diagnostics Objects // Journal of Vibroengineering. 2008. Vol. 10, no. 2. P. 142–146. URL: <http://elibrary.ru/item.asp?id=14876532>.
8. Makarovskikh T., Abotaleb M. Comparison Between Two Systems for Forecasting Covid-19 Infected Cases // IFIP Advances in Information and Communication Technology. 2021. Vol. 616. P. 107–114. DOI: 10.1007/978-3-030-86582-5_10.
9. Sirotin D.V. Neural network approach to forecasting the cost of ferroalloy products // Izvestiya. Ferrous Metallurgy. 2020. Vol. 63, no. 1. P. 78–83. DOI: 10.17073/0368-0797-2020-1-78-83.
10. Yakubova D.M. Econometric models of development and forecasting of black metallurgy of Uzbekistan // Asian Journal of Multidimensional Research (AJMR). 2019. Vol. 8, no. 5. P. 310–314. DOI: 10.5958/2278-4853.2019.00205.2.
11. Neto A.B.S., Ferreira T.A.E., Batista M.C.M., Firmino P.R.A. Studying the Performance of Cognitive Models in Time Series Forecasting // Revista de Informatica Teorica e Aplicada. 2020. Vol. 27, no. 1. P. 83–91. DOI: 10.22456/2175-2745.96181.
12. Panchal R., Kumar B. Forecasting industrial electric power consumption using regression based predictive model // Recent Trends in Communication and Electronics. 2021. DOI: 10.1201/9781003193838-26.
13. Panyukov A.V., Mezaal Y.A. Improving of the Identification Algorithm for a Quasilinear Recurrence Equation // Advances in Optimization and Applications. Vol. 1340 / ed. by N. Olenov, Y. Evtushenko, M. Khachay, V. Malkova. Springer, 2020. P. 15–26. DOI: 10.1007/978-3-030-65739-0_2.
14. Makarovskikh T., Panyukov A., Abotaleb M. Generalized least deviation method for identification of quasi-linear autoregressive model. URL: <https://github.com/tmakarovskikh/GLDMPredictor.git> (дата обращения: 27.07.2022).
15. Pan J., Wang H., Qiwei Y. Weighted Least Absolute Deviations Estimation for ARMA Models with Infinite Variance // Econometric Theory. 2007. Vol. 23, no. 3. P. 852–879.
16. Panyukov A.V., Mezaal Y.A. Stable estimation of autoregressive model parameters with exogenous variables on the basis of the generalized least absolute deviation method // IFAC-PapersOnLine. 2018. Vol. 51, no. 11. P. 1666–1669. DOI: 10.1016/j.ifacol.2018.08.217.
17. Panyukov A.V. Scalability of Algorithms for Arithmetic Operations in Radix Notation // Reliable Computing. 2015. Vol. 19. P. 417–434. URL: <http://interval.louisiana.edu/reliable-computing-journal/volume-19/reliable-computing-19-pp-417-434.pdf>.
18. Abotaleb M.S.A., Makarovskikh T. Analysis of Neural Network and Statistical Models Used for Forecasting of a Disease Infection Cases // 2021 Int. Conf. on Information Technology and Nanotechnology (ITNT). 2021. P. 1–7. DOI: 10.1109/ITNT52450.2021.9649126.

Аботалев Мостафа Салахелдин Абделсалам, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Макаровских Татьяна Анатольевна, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Панюков Анатолий Васильевич, д.ф.-м.н., профессор, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

POSSIBILITIES OF PARALLELISM UNDER IDENTIFYING A QUASI-LINEAR RECURRENT EQUATION

© 2023 M.S.A. Abotaleb, T.A. Makarovskikh, A.V. Panyukov

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: abotalebmostafa@bk.ru, Makarovskikh.T.A@susu.ru, paniukovav@susu.ru

Received: 12.08.2022

Time series analysis and forecasting are one of the widely researched areas nowadays. Identification using various statistical methods, neural networks or mathematical models has long been used in various fields of research from industry, to medicine, the social sphere, and the agricultural researches. The article considers a parallel version of the algorithm for identifying the parameters of a quasi-linear recurrent equation for solving the task of regression analysis with interdependent observable variables, based on the generalized least modules method (GLDM). Unlike neural networks, which are widely used nowadays in various forecasting systems, this approach allows us to explicitly obtain qualitative quasi-linear difference equations that adequately describe the considered process. This makes it possible to improve the quality of the studied processes analysis. A significant advantage of the model using the generalized least deviation method, in comparison with numerous neural network approaches, is the possibility of interpreting the coefficients of the model from the point of view of the research task and using the resulting equation as a model of a dynamic process. The conducted computational experiments using time series show that the maximum acceleration of the algorithm occurs when using the number of threads equal to half of the possible threads for a given device.

Keywords: parallelism, quasi-linear recurrent equation, forecasting, simulation, autoregressive model.

FOR CITATION

Abotaleb M.S.A., Makarovskikh T.A., Panyukov A.V. Investigation of the Possibilities of Parallelism for Forecasting Using a Quasi-linear Recurrent Equation. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 94–109. (in Russian) DOI: 10.14529/cmse230404.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Li Q., Wang J., Zhang H. A wind speed interval forecasting system based on constrained lower upper bound estimation and parallel feature selection. Knowl. Based Syst. 2021. Vol. 231. DOI: 10.1016/j.knosys.2021.107435.
2. Khashei M., Chahkoutahi F. A comprehensive low-risk and cost parallel hybrid method for electricity load forecasting. Comput. Ind. Eng. 2021. Vol. 155. DOI: 10.1016/j.cie.2021.107182.
3. Supuwingsih N.N., Kadeksukerti N., Putra A., Dewanti P. Forecasting of Agricultural Production Results in South Denpasar Using Quadratic Trend Method Based GIS. International Journal of Engineering Technologies and Management Research. 2018. Vol. 5, no. 2. DOI: 10.5281/zenodo.1186523.
4. Hamdi F., Raby H., Hakim G., *et al.* A Generalized Mechanistic Model for Assessing and Forecasting the Spread of the COVID-19 Pandemic. IEEE Access. 2021. Vol. 9. P. 13266–13285. DOI: 10.1109/ACCESS.2021.3051929.

5. Dash S., Chakraborty C., Giri S.K., *et al.* BIFM: Big-Data Driven Intelligent Forecasting Model for COVID-19. *IEEE Access*. 2021. Vol. 9. P. 97505–97517. DOI: 10.1109/ACCESS.2021.3094658.
6. Corpas-Burgos F., Martinez-Beneito M.A. An Autoregressive Disease Mapping Model for Spatio-Temporal Forecasting. *Mathematics*. 2021. Vol. 9, no. 4. Article 384. DOI: 10.3390/math9040384.
7. Panyukov A.V., Tyrsin A.N. Stable Parametric Identification of Vibratory Diagnostics Objects. *Journal of Vibroengineering*. 2008. Vol. 10, no. 2. P. 142–146. URL: <https://www.extrica.com/article/10181>.
8. Makarovskikh T., Abotaleb M. Comparison Between Two Systems for Forecasting Covid-19 Infected Cases. *IFIP Advances in Information and Communication Technology*. 2021. Vol. 616. P. 107–114. DOI: 10.1007/978-3-030-86582-5_10.
9. Sirotin D.V. Neural network approach to forecasting the cost of ferroalloy products. *Izvestiya. Ferrous Metallurgy*. 2020. Vol. 63, no. 1. P. 78–83. DOI: 10.17073/0368-0797-2020-1-78-83.
10. Yakubova D.M. Econometric models of development and forecasting of black metallurgy of Uzbekistan. *Asian Journal of Multidimensional Research (AJMR)*. 2019. Vol. 8, no. 5. P. 310–314. DOI: 10.5958/2278-4853.2019.00205.2.
11. Neto A.B.S., Ferreira T.A.E., Batista M.C.M., Firmino P.R.A. Studying the Performance of Cognitive Models in Time Series Forecasting. *Revista de Informatica Teorica e Aplicada*. 2020. Vol. 27, no. 1. P. 83–91. DOI: 10.22456/2175-2745.96181.
12. Panchal R., Kumar B. Forecasting industrial electric power consumption using regression based predictive model. *Recent Trends in Communication and Electronics*. 2021. DOI: 10.1201/9781003193838-26.
13. Panyukov A.V., Mezaal Y.A. Improving of the Identification Algorithm for a Quasilinear Recurrence Equation. *Cham*, 2020. DOI: 10.1007/978-3-030-65739-0_2.
14. Makarovskikh T., Panyukov A., Abotaleb M. Generalized least deviation method for identification of quasi-linear autoregressive model. URL: <https://github.com/tmakarovskikh/GLDMPredictor.git> (accessed: 27.07.2022).
15. Pan J., Wang H., Qiwei Y. Weighted Least Absolute Deviations Estimation for ARMA Models with Infinite Variance. *Econometric Theory*. 2007. Vol. 23, no. 3. P. 852–879.
16. Panyukov A.V., Mezaal Y.A. Stable estimation of autoregressive model parameters with exogenous variables on the basis of the generalized least absolute deviation method. *IFAC-PapersOnLine*. 2018. Vol. 51, no. 11. P. 1666–1669. DOI: 10.1016/j.ifacol.2018.08.217.
17. Panyukov A.V. Scalability of Algorithms for Arithmetic Operations in Radix Notation. *Reliable Computing*. 2015. Vol. 19. P. 417–434. URL: <http://interval.louisiana.edu/reliable-computing-journal/volume-19/reliable-computing-19-pp-417-434.pdf>.
18. Abotaleb M.S.A., Makarovskikh T. Analysis of Neural Network and Statistical Models Used for Forecasting of a Disease Infection Cases. 2021 *Int. Conf. on Information Technology and Nanotechnology (ITNT)*. 2021. P. 1–7. DOI: 10.1109/ITNT52450.2021.9649126.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии <https://vestnikvmi.susu.ru>. Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:
Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,
зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: vestnikvmi@susu.ru
4. Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
2023 Том 12, № 4

16+

Техн. редактор А.В. Миних

Издательский центр Южно-Уральского государственного университета

Подписано в печать 30.11.2023. Дата выхода в свет 11.12.2023. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 13,02. Тираж 500 экз. Заказ 363/403. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.