МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

На правах рукописи

Иванов Дмитрий Александрович

Нейроморфные методы оптимизации систем искусственного интеллекта для задач обучения с подкреплением

Специальность 2.3.5
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук,
профессор, член-корреспондент РАН
Воеводин Владимир Валентинович

Москва — 2025

Оглавление

Стр.

Стр.

Стр.

Введение

Современные *системы искусственного интеллекта* (системы ИИ), представляющие собой комбинацию алгоритмов ИИ и аппаратного обеспечения, в большинстве случаев построены на основе глубоких нейронных сетей (НС) и компьютеров на архитектуре фон Неймана. При этом, по сравнению с мозгом человека современные системы ИИ значительно менее энергоэффективны: при меньшем количестве нейронов и связей они потребляют значительно больше энергии. Мозг человека, состоящий приблизительно из 80-100 миллиардов нейронов и 1000 триллионов связей (синапсов), потребляет около 20 Вт. Для сравнения, современная высокопроизводительная видеокарта Nvidia H100 с энергопотреблением 700 Вт имеет память объемом 80 ГБ, достаточную для моделирования лишь 80 миллиардов связей. Таким образом, моделирование на четыре порядка меньшего количества связей требует на порядок большего энергопотребления. Как следствие, высокое энергопотребление ограничивает применение нейронных сетей в робототехнике и других встраиваемых системах.

Столь высокое энергопотребление обусловлено значительными энергозатратами на обращение к основной памяти по сравнению с вычислительными операциями, что является следствием разделения памяти и вычислений, характерного для большинства современных вычислительных архитектур. Кроме того, разделение вызывает большие временны́е задержки при обращении в память и ограничивает общую производительность вычислительной системы из-за узкого канала передачи данных между памятью и вычислителем. Данные проблемы часто в совокупности называются *проблемой бутылочного горлышка фон Неймана*. Стоит отметить, что многие алгоритмы, лежащие в основе нейронных сетей, относятся к алгоритмам с низкой вычислительной интенсивностью, что еще больше усиливает проблему бутылочного горлышка фон-неймановской архитектуры в системах ИИ. Особенно остро проблема обращений в память проявляется в ситуациях, когда размер пакета входных данных НС равен единице, что приводит к еще меньшей вычислительной интенсивности. Характерным примером таких ситуаций являются задачи управления, в частности, *задачи обучения с подкреплением*.

Обучение с подкреплением имеет множество приложений в реальном мире, таких как робототехника, управление сложными устройствами (к примеру, удержание плазмы в токамаке в реальном времени), игры, биржевая торговля и

другие. Эти приложения часто накладывают требования малого времени отклика и высокой частоты работы нейронных сетей, тренированных с помощью методов обучения с подкреплением. Применение НС для задач удержания плазмы в токамаке требует частоты их работы в 100 кГц, а при автономном управлении высокоманевренным квадрокоптером необходимы частоты их работы не менее 100 Гц. Время отклика НС в десятки наносекунд необходимо при биржевой торговле. При попытках использования в подобных задачах нейронных сетей с миллиардами связей, их максимальная частота работы существенно снижается (до нескольких Гц), что вынуждает применять либо малые по размерам сети, либо ограничиваться низкой частотой их работы. Таким образом, алгоритмические свойства НС в сочетании с особенностями фон-неймановской архитектуры накладывают серьезные ограничения на применение больших нейронных сетей в задачах обучения с подкреплением.

Для преодоления обозначенных выше ограничений исследователи предлагают как алгоритмические методы оптимизации нейронных сетей, так и принципиально новые аппаратные решения. Существуют различные методы алгоритмической оптимизации нейронных сетей, такие как структурная разреженность (прюнинг), квантование, дистилляция знаний, поиск вычислительно эффективных архитектур НС. К сожалению, эти методы далеко не всегда учитывают взаимосвязь аппаратного обеспечения и алгоритмов, что затрудняет практическое применение данных методов. Одновременно с этим существует крайне малое количество научных работ, посвященных применению методов оптимизации к нейронным сетям, тренированных методами обучения с подкреплением. Также, в последние годы активно развиваются аппаратные методы оптимизации и предлагаются новые вычислительные архитектуры, такие как не фон-неймановские вычислители, направленные на преодоление ограничений современного аппаратного обеспечения для нейронных сетей.

Одним из наиболее перспективных направлений являются нейроморфные (биологически подобные) архитектуры и методы, стремящиеся имитировать алгоритмически или аппаратно некоторые принципы функционирования мозга человека, что может способствовать повышению энергоэффективности, скорости работы и масштабируемости систем искусственного интеллекта. К данному направлению относятся: импульсные нейронные сети, «вычисления в памяти»/«рядом с памятью», асинхронное исполнение нейронных сетей, поддержка разреженных и аналоговых вычислений.

Таким образом, поиск аппаратных и алгоритмических методов для оптимизации инференса систем искусственного интеллекта, основанных на нейронных сетях, тренированных с помощью методов обучения с подкреплением, представляется крайне значимой и перспективной задачей. В данной работе предлагается использовать нейроморфные методы для её решения.

В диссертации делаются акценты на следующих вопросах:

1. Анализ причин низкой эффективности в аспекте энергопотребления и скорости работы существующих систем искусственного интеллекта, сравнение принципов их работы с принципами функционирования мозга человека.
2. Разработка, применение и анализ эффективности нейроморфных методов для оптимизации систем ИИ для задач обучения с подкреплением.

Цели и задачи работы. Целью данной работы является разработка нейроморфных методов, позволяющих существенно ускорить и понизить энергозатратность систем искусственного интеллекта для задач обучения с подкреплением.

Для достижения поставленной цели необходимо было решить следующие задачи:

1. Проанализировать принципы работы современных систем искусственного интеллекта с целью выявления их узких мест и проблем. Исследовать взаимосвязь работы алгоритмов нейронных систем с современными аппаратными платформами. Провести анализ существующих аппаратных и программных методов оптимизации нейронных сетей.
2. Выделить принципы функционирования мозга человека и их связь с системами ИИ. Оценить их преимущества и недостатки для их возможной имплементации в современные системы ИИ.
3. Исследовать особенности работы нейронных сетей, тренированных с помощью методов обучения с подкреплением.
4. Разработать алгоритмы оптимизации нейронных сетей, тренированных методами обучения с подкреплением, на основе нейроморфных методов: с помощью комбинации временно́й и структурной разреженности и на основе комбинации структурной разреженности и квантования.
5. Программно реализовать описанные подходы, проанализировать их эффективность и возможность потенциальной имплементации в аппаратных вычислительных платформах. Провести сравнение с существующими решениями.

Научная новизна:

1. Проведено детальное сравнение принципов работы мозга человека с принципами работы систем ИИ на основе фон-неймановских и на основе не фон-неймановских нейроморфных вычислителей. На основе проведенного сравнения предложена классификация принципов работы мозга человека и проведен анализ на предмет их имплементируемости в современных системах ИИ.
2. Впервые предложен алгоритм на основе комбинации структурной разреженности и квантования для оптимизации нейронных сетей, тренированных методами обучения с подкреплением. Метод на 1 - 2 порядка (вплоть до 400 раз) уменьшает занимаемую нейронными сетями память без потери качества работы, что позволяет размещать нейронные сети в быстрой памяти или снижать число обращений в память.
3. Впервые предложен алгоритм оптимизации нейронных сетей, тренированных методами обучения с подкреплением, на основе комбинации двух видов разреженности: структурной и временно́й. Метод уменьшает на 1 - 2 порядка число обращений в память и число необходимых арифметических операций при инференсе нейронных сетей при сохранении качества работы. Данный метод обладает еще одним преимуществом, позволяя нейронным сетям работать в асинхронном режиме, что повышает их потенциальную масштабируемость.

Теоретическая и практическая значимость.

В диссертационной работе проведено сравнение принципов работы мозга человека и современных систем искусственного интеллекта, построенных на основе как фон-неймановских вычислителей, так и не фон-неймановских нейроморфных архитектур. Показано, что современные системы ИИ на основе фон-неймановских вычислителей не используют ряд ключевых принципов работы мозга человека. Это обуславливает низкую энергоэффективность, масштабируемость и скорость работы современных систем ИИ. Рассмотрены возможности реализации части принципов работы мозга человека в современных системах ИИ для улучшения их энергоэффективности и скорости работы.

Практическая значимость работы связана с повышением эффективности инференса систем ИИ для задач обучения с подкреплением с помощью нейроморфных подходов. Были предложены новые и развиты существующие подходы на основе структурной разреженности, временно́й разреженности и квантования

для оптимизации нейронных сетей, тренированных методами обучения с подкреплением. С помощью данных методов была показана возможность уменьшать размеры нейронных сетей на 1 - 2 порядка и уменьшать на 1 - 2 порядка число арифметических операций. Была показана связь этих методов оптимизации с некоторыми принципами работы мозга человека.

Создан программный комплекс, который позволяет проводить эксперименты для изучения и тестирования методов оптимизации нейронных сетей, тренированных методами обучения с подкреплением.

Методология и методы исследования. При получении основных результатов диссертационной работы использовались методы обучения с подкреплением, глубокое машинное обучение, методы системного и сравнительного анализа. Использовались методы программирования на языке Python и C++.

Основные положения, выносимые на защиту:

1. Нейроморфные методы и подходы к оптимизации систем искусственного интеллекта для задач обучения с подкреплением на основе свойств и принципов функционирования мозга человека с целью повышения энергоэффективности, пропускной способности, масштабируемости и скорости работы современных систем ИИ.
2. Метод оптимизации инференса нейронных сетей, тренированных методами обучения с подкреплением, на основе комбинации структурной разреженности и квантования. Метод уменьшает на 1 - 2 порядка (вплоть до 400 раз) размеры нейронных сетей без потери качества работы, что позволяет размещать нейронные сети в быстрой памяти или уменьшать число обращений в память.
3. Метод оптимизации инференса нейронных сетей, тренированных методами обучения с подкреплением, на основе комбинации временно́й и структурной разреженности. Метод уменьшает на 1 - 2 порядка число обращений в память и число арифметических операций при инференсе нейронных сетей без потери качества работы. Введенная в методе асинхронность нейронов дает возможность обеспечения большей масштабируемости.

Апробация работы. Представленные в работе результаты докладывались на следующих научных конференциях и семинарах:

1. Научная конференция «Тихоновские чтения» 2023, Москва, Россия, 29 октября - 3 ноября 2023.

2. Всероссийская конференция «Ломоносовские чтения» 2022, Москва, Россия, 14 - 22 апреля 2022.
3. Научный семинар кафедры Интеллектуальных информационных технологий ВМК МГУ, 2024.
4. Научный семинар по машинному обучению под руководством проф. А.Г. Дьяконова, Центральный университет, 2024.
5. Научный семинар «Методы машинного обучения в автоматической обработке текстов», НИВЦ МГУ, 2024.

Личный вклад. Все результаты работы получены автором лично под научным руководством д.ф.-м.н., чл.-корр. РАН Воеводина Владимира Валентиновича. В работах, написанных в соавторстве, вклад автора диссертации является определяющим.

В работе [А.1] автором выполнен анализ принципов работы современных вычислительных систем и проведено сравнение с принципами работы мозга человека. На основе этого предложена классификация принципов работы мозга человека и проанализирована их реализация в нейроморфных системах ИИ. Работа опубликована в журнале Frontiers in Neuroscience [А.1].

В работе [А.2] автором предложен метод оптимизации инференса нейронных сетей, тренированных методами обучения с подкреплением, на основе комбинации структурной разреженности и квантования. Все эксперименты выполнены лично автором. Работа опубликована в журнале Scientific Reports [А.2].

В работе [А.3] предложен метод оптимизации инференса нейронных сетей, тренированных методами обучения с подкреплением, на основе комбинации структурной и временно́й разреженности. Все эксперименты выполнены лично автором. Работа опубликована в журнале Scientific Reports [А.3].

Публикации. Основные результаты по теме диссертации изложены в 3 публикациях [А.1—А.3], изданных в рецензируемых научных изданиях, определенных в п. 2.3 Положения о присуждении ученых степеней в Московском государственном университете имени М. В. Ломоносова.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения и 1 приложения. Полный объём диссертации составляет 116 страниц, включая 34 рисунка и 10 таблиц. Список литературы содержит 110 наименований.

Глава 1. Системы искусственного интеллекта для задач обучения с подкреплением

Обучение нейронных сетей (нейросетей) и их последующее использование (*инференс, inference*) требует значительных объемов данных, вычислительных ресурсов, памяти и энергии. Так, обучение алгоритмов для игр Atari требует нескольких миллионов часов игры [1], а для освоения игры go искусственным интеллектом потребовалось проведение 140 миллионов партий [2]. Процесс обучения языковой модели LLAMA-2 [3], содержащей 70 миллиардов параметров, занимает 71680 GPU-дней, что эквивалентно примерно 200 годам вычислений. Финансовые затраты на обучение моделей, исключая оплату труда инженеров, могут достигать десятков и сотен миллионов долларов [4]. Одной из ключевых причин высоких затрат на алгоритмы ИИ является постоянно возрастающий размер моделей (рост числа параметров), обеспечивающий стабильное улучшение результатов качества работы алгоритмов [5].

Однако, если вычислительную сложность определяет только сам алгоритм (для нейронных сетей она зависит от количества параметров и объема обрабатываемых данных), то такие параметры как время работы, скорость отклика и энергопотребление зависят от используемого аппаратного обеспечения, на котором происходит исполнение алгоритма.

Учитывая вышеизложенное, необходимо рассматривать алгоритмы ИИ в сочетании с аппаратным обеспечением, на котором осуществляется их функционирование. Именно аппаратное обеспечение определяет доступность и эффективность алгоритмов ИИ. Совокупность алгоритмов ИИ и аппаратного обеспечения будем называть *системой ИИ*. Рассмотрим подробнее устройство современных систем ИИ.

1.1 Архитектура фон Неймана

Практически все современные компьютеры основаны на архитектуре фон Неймана. Архитектура фон Неймана [6] — это концепция построения компьютеров, которая была описана Джоном фон Нейманом и его коллегами в 1945 году

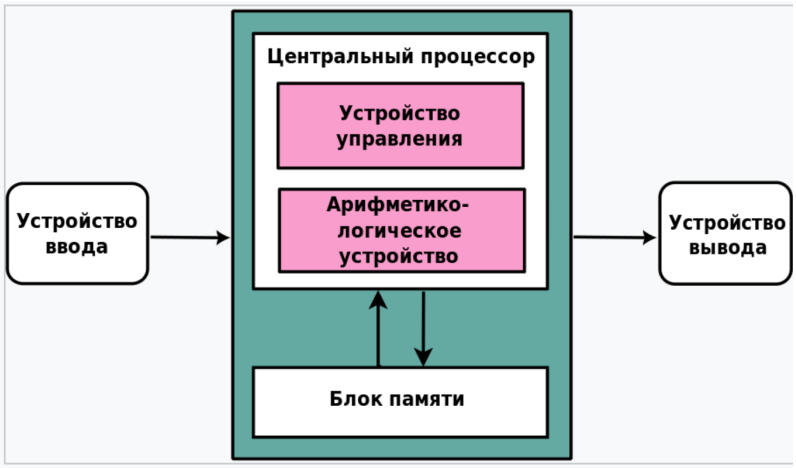


Рисунок 1.1 — Схематическое представление архитектуры фон Неймана. Рисунок взят из сети Интернет.

при работе над компьютером EDVAC. Она предполагает, что все вычисления представляются в виде программ, являющихся последовательностями машинных команд. Команды выполняет процессор. Команды и данные хранятся в общей памяти. Классическая архитектура фон Неймана включает следующие основные компоненты (см рис. 1.1):

1. Процессор, содержащий арифметико-логическое устройство (АЛУ) и регистры.
2. Блок управления, включающий в себя регистр команд и счетчик программ.
3. Память для хранения данных и инструкций.
4. Внешнее запоминающее устройство.
5. Устройство ввода и вывода.

1.1.1 Узкие места архитектуры фон Неймана

Архитектура фон Неймана обладает рядом ограничений (узких мест, бутылочных горлышек). Термин *«бутылочное горлышко фон Неймана» (von Neumann bottleneck, memory bottleneck, memory wall)* был впервые введен Джоном Бэкусом в 1977 году на вручении ему премии Тьюринга [7]. Данный термин описывает ограничение производительности компьютера, вызванное ограниченной пропускной способностью канала между памятью и процессором. Рассмотрим более подробно узкие места архитектуры фон Неймана.

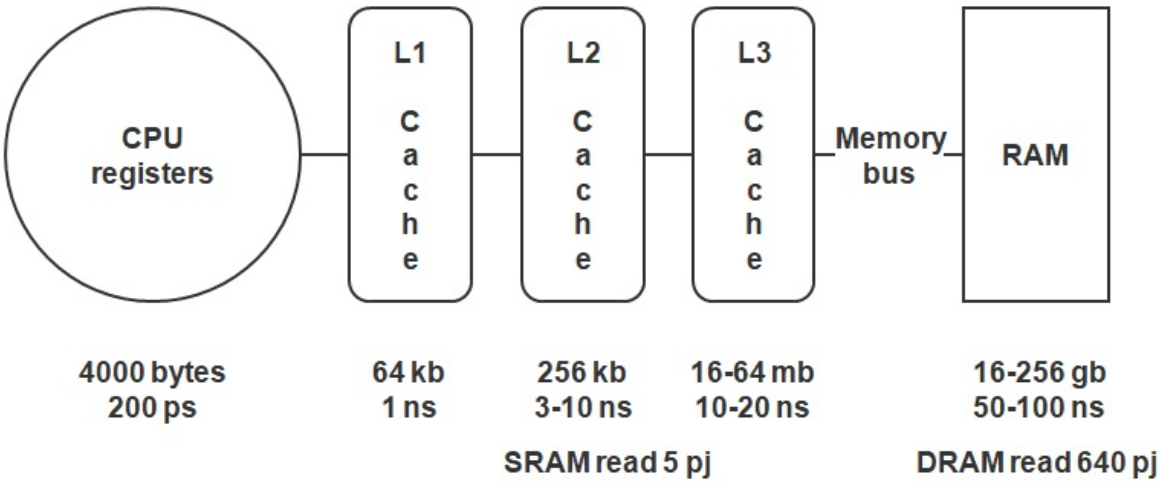


Рисунок 1.2 — Иерархия памяти, типовые скорости доступа и энергопотребление. Рисунок автора из статьи [A.1].

Первым узким местом является *ограниченная пропускная способность шины данных* между памятью и процессором. При выполнении программы шина данных преимущественно загружена передачей данных между оперативным запоминающим устройством (ОЗУ) и процессором. При этом, максимальная пропускная способность шины данных существенно уступает скорости обработки данных процессором [A.1].

Второе существенное ограничение — *значительная разница в скорости работы (времени отклика, латентности) оперативной памяти и регистров процессора* (см. рис. 1.2). Это может приводить к задержке и простою процессора при извлечении данных из памяти [A.1].

Третье ограничение — *энергетическое «узкое место» (energy wall)*, обусловленное большой разницей (на несколько порядков) в энергопотреблении между выполнением вычислений и передачей данных между процессором и памятью (см. рис. 1.2, 1.3). Например, при использовании 45 нм технологии, операция сложения двух 8-битных целых чисел потребляет 0,03 пикоджоулей, тогда как операция чтения из динамической оперативной памяти 32 бит (DRAM) требует 640 пикоджоулей (см. рис. 1.4) [8].

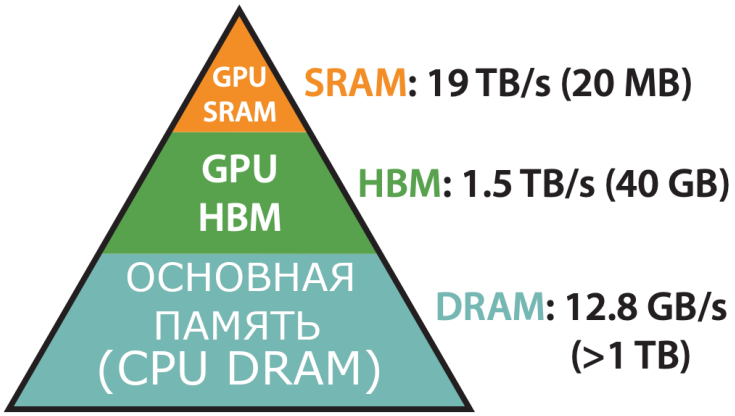
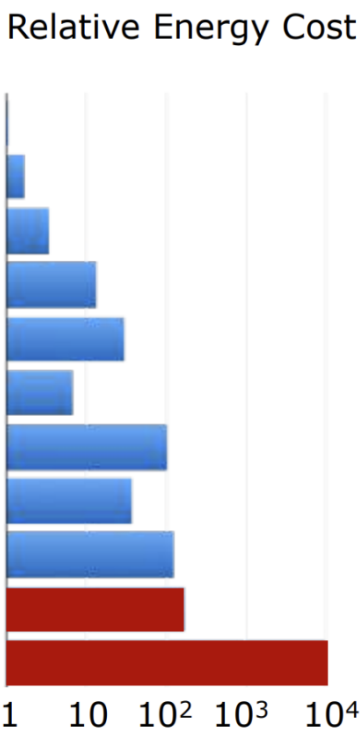


Рисунок 1.3 — Иерархия памяти в GPU и характерные пропускные способности. Рисунок заимствован из статьи [9].

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



[**Horowitz**, *ISSCC* 2014]

Рисунок 1.4 — Энергетические затраты на различные арифметические операции и операция взятия данных их памяти. Рисунок заимствован из статьи [8].

1.1.2 Смягчение проблем архитектуры фон Неймана

Разработчики и инженеры нашли ряд способов минимизировать влияние узких мест архитектуры фон Неймана на производительность:

- *Кэширование (Cache)* — использование высокоскоростной памяти (обычно основанной на технологии статической памяти - SRAM) для временного хранения часто используемых данных и инструкций, чтобы уменьшить время доступа к памяти. Однако эффективность кэширования обеспечивается только при условии соблюдения принципов временной и пространственной локальности.
- *Конвейеризация (instruction pipelining)* — метод параллельного выполнения нескольких инструкций на различных стадиях обработки, повышающий общую производительность системы.
- *Спекулятивные вычисления (Speculative computations)* — подход, основанный на предварительном выполнении операций на основе предсказания ветвей (branch prediction), для минимизации возможных задержек.
- *Мультипоточность (Multithreading)* - механизм выполнения нескольких потоков на одном процессорном ядре и переключения между ними, обеспечивающий возможность эффективно использовать вычислительные ресурсы при ожидании данных из памяти и во время исполнения других медленных операций. Несмотря на то, что мультипоточность существенно повышает утилизацию процессора и общую производительность системы, она не решает проблему высоких задержек доступа к памяти для отдельных потоков.
- *HBM (High Bandwidth Memory)* - технология многослойной памяти с использованием сквозных кремниевых переходов (TSV). HBM подключается к основному кристаллу CPU/GPU/SOC через переходник внутри подложки (interposer) корпуса (package) и обеспечивает высокую пропускную способность (более 7,2 Тбит/с для HBM3) благодаря широкой шине данных.
- *Вычисления в памяти и рядом с памятью (in-memory computations, near-memory computations)* — подход, основанный на переносе вычислений ближе к памяти и направленный на минимизацию использования шины данных. К нему можно отнести добавление логических блоков в DRAM и

перенос памяти ближе к процессору путем хранение всех данных в SRAM памяти.

1.2 Современные нейронные сети

Искусственные нейронные сети, которые используются в машинном обучении и искусственном интеллекте для решения разнообразных задач, от распознавания изображений до автоматического перевода текста, состоят из искусственных нейронов подобно тому как биологические нейронные сети состоят из клеток-нейронов. Данный подход в ИИ получил название *коннекционизма*.

1.2.1 Модель искусственного нейрона

Модель искусственного нейрона (см рис. 1.5), предложенная Розенблаттом [10], является упрощённой абстракцией биологического нейрона. Основными компонентами модели являются входы, веса, сумматор и функция активации.

1. Входы (Input): Входы модели соответствуют дендритам биологического нейрона и представляют собой значения данных, подлежащие обработке. Искусственный нейрон обычно имеет множество входов.
2. Веса (Weights): Каждому входу соответствует вес, определяющий степень влияния входного значения на выход нейрона. Веса аналогичны *синаптической эффективности* в биологических нейронах и могут быть как положительными (усиливающими сигнал), так и отрицательными (ослабляющими сигнал).
3. Сумматор (Adder): Данный компонент осуществляет агрегацию взвешенных входных сигналов. Результатом является суммированный взвешенный вход, который затем передаётся на функцию активации. Этот процесс аналогичен интеграции сигналов в *теле* биологического нейрона.
4. Функция активации (Activation Function): Определяет, будет ли нейрон активирован, и какой сигнал будет передан на выход. Может быть реали-

зована как пороговая функция, активирующая нейрон при превышении входным сигналом определенного порога, или как более сложная функция: сигмоидная, гиперболический тангенс или ReLU (Rectified Linear Unit).

5. Выход (Output): Результат функции активации формирует выходной сигнал нейрона, который может быть передан на входы других нейронов сети.

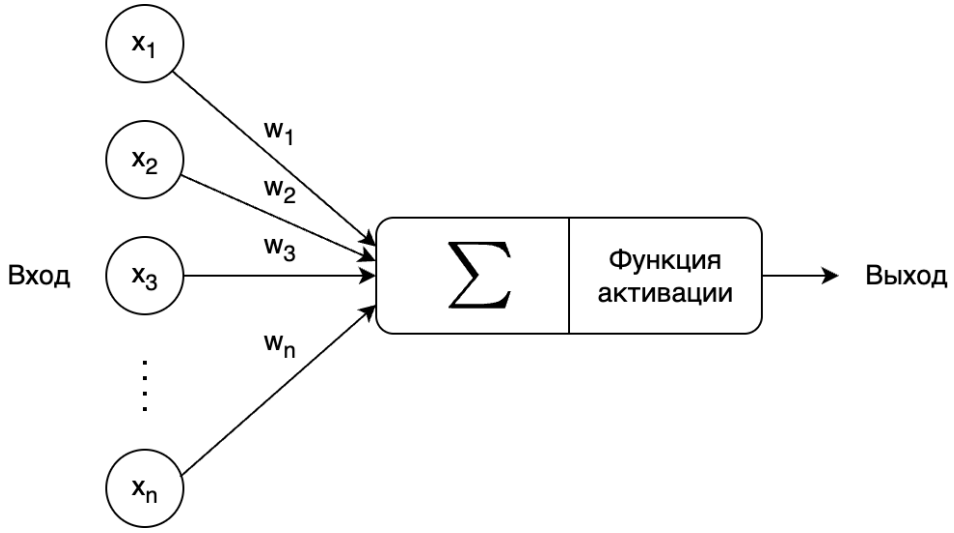


Рисунок 1.5 — Математическая модель нейрона Розенблатта. Рисунок автора.

Модель искусственного нейрона Розенблатта позволяет создавать нейронные сети путём соединения множества таких нейронов в сложные архитектуры, способные выполнять разнообразные задачи, включая классификацию, регрессию и генерацию данных. Обучение нейронных сетей заключается в адаптации весов между нейронами посредством алгоритма обратного распространения ошибки, что позволяет модели оптимизировать свои предсказания в соответствии с предоставленными данными.

На сегодняшний день существует огромное количество различных нейронных сетей, нацеленных на решение тех или иных когнитивных задач. Однако их архитектура базируется на нескольких основных структурных блоках и функциях активации.

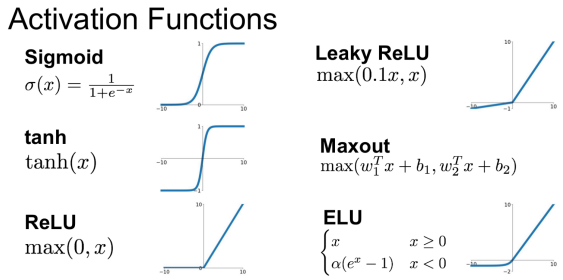
Обычно нейронные сети организованы послойно. Группы нейронов объединяются в слои, идущие друг за другом. А так как каждый нейрон представляет из себя перемножение вектора весов нейрона и вектора входов нейрона с последующим применением функции активации, то каждый слой определяется некоторой матричной операцией и функцией активации, применяющейся к выходам нейронов.

Рассмотрим сначала популярные функции активаций нейронов.

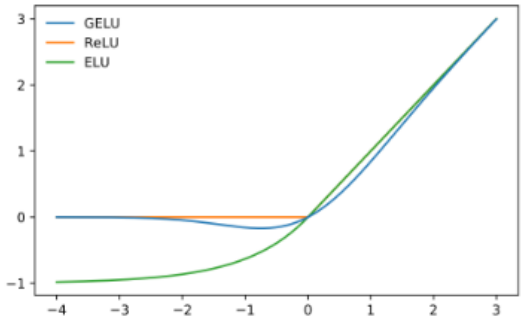
1.2.2 Функции активации нейрона

Функции активации играют центральную роль в архитектуре нейронных сетей, определяя характер активации нейрона в ответ на суммарный входной сигнал. Их существенная роль заключается во введении нелинейности, что позволяет нейронным сетям моделировать сложные абстракции данных и выявлять нелинейные зависимости. Это свойство делает их неотъемлемым инструментом в построении мощных и эффективных моделей глубокого обучения.

Среди основных функций активации выделяют: Logistic Sigmoid, Гиперболический тангенс (tanh), ReLU (Rectified Linear Unit) [11], LeakyReLU [12], ELU (Exponential Linear Unit) [13], GeLU (Gaussian Error Linear Unit) [14], Графики некоторых функций активаций представлены на рис. 1.6.



а) Функции активации 1



б) Функции активации 2

Рисунок 1.6 — Наиболее распространенные функции активации. Рисунки заимствованы из сети Интернет.

Перейдем к рассмотрению типов слоев.

1.2.3 Основные типы слоев нейронной сети с точки зрения вычислений

Полносвязный слой (Fully Connected Layer, FC). В полносвязном слое каждый нейрон входного слоя соединён с каждым нейроном выходного слоя. Вы-

числения в таком слое сводятся к матричному умножению входного вектора признаков на матрицу весов с последующим добавлением вектора смещения (bias).

$$y = W \cdot x + b \tag{1.1}$$

где:

– W - матрица весов,

– x - вектор входных признаков,

– b - вектор смещения.

Вычислительная сложность полносвязного слоя для одного вектора входных данных определяется как $O(n \cdot m)$ где n - размерность входного вектора, m - размерность выходного вектора. Полносвязные слои широко используются в классических нейронных сетях, рекуррентных нейронных сетях, архитектуре трансформер, а также в качестве заключительных слоев в свёрточных сетях.

Свертка, Конволюция (Convolutional Layer). Сверточные слои основаны на операции свёртки и предназначены для автоматического извлечения признаков из входных данных. В отличие от полносвязных слоёв, свёртка сокращает количество параметров за счёт применения одинаковых весов к различным областям входных данных. Это обеспечивает эффективную обработку изображений, видео и других данных с пространственной структурой.

Вычислительная сложность свёрточного слоя не выражается одной простой формулой и зависит от размера входных данных, размера ядра свёртки, количества фильтров, шага свёртки и других характеристик.

В современных библиотеках глубинного обучения реализация сверток обычно осуществляется через матричные вычисления с использованием алгоритма im2col [15]. Данный подход значительно повышает производительность вычислений на GPU благодаря возможности эффективного распараллеливания матричных операций.

Рекуррентный слой и его разновидности (Recurrent Layer, RNN, LSTM, GRU). Рекуррентные нейронные сети (RNN) представляют собой класс нейронных сетей, предназначенных для обработки последовательных данных, таких как текст или временные ряды. В RNN текущие выходные данные слоя используется в качестве части входных данных на следующем шаге, что обеспечивает сохранение информации о предыдущих состояниях.

Простейшая форма RNN имеет следующую формулу:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{1.2}$$

где:

– h_t — новое скрытое состояние на временном шаге t,
– h_{t-1} — предыдущее скрытое состояние,
– x_t - входное значение на временном шаге t,
– W_{hh} - веса между скрытыми состояниями,
– W_{xh} - веса между входным состоянием и скрытым состоянием,
– b_h - смещение скрытого слоя.

То есть, с точки зрения вычислений рекуррентный слой состоит в основном из матричных операций.

Слой внимания (Attention Layer). Слой внимания позволяет модели фокусироваться на определённых частях входных данных. Данный механизм особенно эффективен и полезен в задачах, требующих учета контекста или различной значимости входных данных.

Слой внимания определяется следующей формулой:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.3}$$

где:

– Q - матрица запросов (queries),
– K - матрица ключей (keys),
– V - матрица значений (values),
– d_k размерность ключей,
– softmax — функция softmax, применяемая по строкам результата умножения QK^T, обеспечивающая получение весов внимания, которые в сумме равны 1.

Вычислительная сложность слоя внимания зависит от размера входных данных и составляет $O(N^2 \cdot d)$, где N — число элементов в последовательности, а d — размерность представления каждого элемента.

Каждый из вышеописанных базовых блоков имеет свои особенности и области применения, но вместе они формируют основу для построения сложных и мощных нейронных сетей, способных решать широкий спектр задач. Как мы убедились, в основе всех этих блоков лежат **матричные операции умножения и сложения**.

1.2.4 Размер пакета (batch)

При работе с нейросетями еще одной важной характеристикой является *размер пакета (батча, batch)* входных данных, обрабатываемых сетью за один проход. При размере батча равном 1 сеть принимает на вход единичный объект, что *наиболее характерно* для систем работающих в потоковом режиме (потоковом инференсе): обработка видео, системы управления, робототехника, edge-устройства, системы реального времени т. д. Основным недостатком данного случая является меньшая эффективность использования вычислительных ресурсов, особенно на GPU и других параллельных вычислительных устройствах, поскольку в меньшей степени используются преимущества параллельной обработки больших данных. При размере батча больше 1 на вход сети подаются одновременно несколько объектов, которые сеть обрабатывает параллельно, что повышает общую эффективность использования вычислительных ресурсов, но обычно увеличивает задержку получения результата. Батч больше 1 применяется преимущественно при обучении нейронных сетей и решении задач, не требующих обработки данных в реальном времени: в системах рекомендаций, обработке изображений, переводе, обработке текста и распознавания речи на сервере (когда запросы поступают одновременно от множества пользователей, формируя пакет).

1.2.5 Вычислительный процесс в нейронных сетях

Как было продемонстрировано, в основе работы полносвязных и рекуррентных блоков лежит умножение матриц на вектор, где размерности матриц соответствуют размерам входных векторов. При добавлении пакетной обработки данных (батч больше 1), входные векторы заменяются матрицами, составленными из входных векторов, где вторая размерность матрицы соответствует размеру батча.

В сверточных сетях процесс вычислений имеет свою специфику. Поскольку свертка выполняется многократно над различными участками изображения, даже при обработке единичного изображения данную операцию можно рассматривать

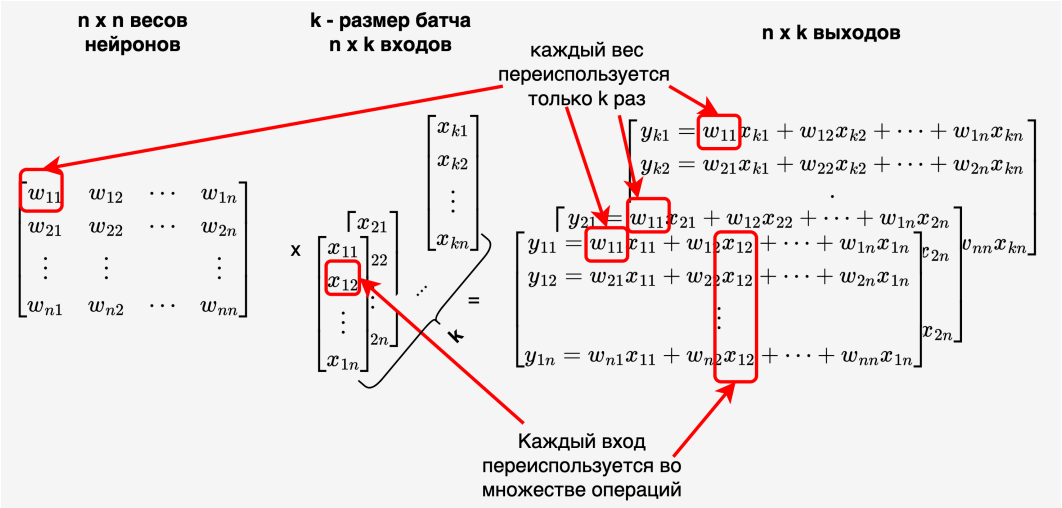


Рисунок 1.7 — Вычислительный процесс в полносвязном слое нейронной сети. Рисунок автора.

как пакетную, где размер пакета равен числу применений свертки. Этим частично объясняется высокая вычислительная эффективность сверточных сетей.

К примеру, на начальных слоях сверточных сетей размеры сверток обычно невелики и, соответственно, число параметров в этих слоях тоже невелико. При этом, размер входного изображения значительно превышает размеры свертки, что обусловливает многократное повторение операции свертки. Например, первый слой ResNet34 [16] имеет $3 \times 64 \times 3 \times 3 = 1\,728$ параметров, в то время как число повторений свертки достигает 112×112 раз. Однако на последних слоях размеры сверток увеличиваются до $512 \times 512 \times 3 \times 3 = 2\,359\,296$, что сопоставимо с размерами других типовых блоков, при этом повторение свертки сокращается до 7×7 раз.

Таким образом для большинства слоев нейронных сетей вычислительный процесс при размере пакета, равном 1, представляет собой перемножение большой матрицы на «узкую» матрицу или вектор, что приводит к низкой вычислительной интенсивности.

1.3 Нейронные сети и архитектура фон Неймана

Продемонстрировав, что исполнение нейронных сетей требует выполнения матричного умножения $Y = WX$, где $W \in \mathbb{R}^{m \times n}$, $X \in \mathbb{R}^{n \times k}$, рассмотрим процесс умножения матрицы на вектор и матрицы на матрицу на архитектуре фон Неймана.

Для выполнения данной операции необходимо первоначально извлечь матрицы весов из памяти. Учитывая типовые размеры современных нейронных сетей, веса, как правило, располагаются в долговременной памяти, поскольку были вытеснены из кэш-памяти при вычислении других слоев.

Рассмотрим предельный случай $k = 1$, характерный для полносвязных слоев при размере батча равном 1. Матрица весов W имеет размерность $m * n$ элементов. При вычислении результирующей матрицы каждый весовой коэффициент используется *однократно* ($k = 1$) (см.рис. 1.7), после чего удаляется из быстрой памяти, и система переходит к вычислению следующего блока. При этом элементы входного вектора используются m раз.

Такая структура вычислений создает существенный дисбаланс в использовании данных: в то время как входные значения многократно переиспользуются, весовые коэффициенты требуют постоянной загрузки из памяти. Учитывая высокие временные и энергетические затраты на извлечение данных из памяти, можно утверждать, что операции загрузки весов являются крайне дорогими по времени и энергии, и становятся узким местом в производительности системы ИИ. Это противоречие между частым доступом к весам и ограничениями архитектуры фон Неймана, где память и вычислительный блок разделены, формирует *фундаментальную проблему* исполнения нейронных сетей на современных вычислительных устройствах.

Данная проблема частично решается при использовании батчей (в сверточных слоях и слоях внимания это происходит неявно). В этом случае каждый весовой коэффициент используется k раз, где k соответствует размеру батча для полносвязных и рекуррентных слоев, и кратен размеру батча для сверточных слоев и слоев внимания. Однако, поскольку часто размер батча при инференсе НС существенно меньше высоты m матрицы W для большинства слоев, это не приводит к принципиальному улучшению ситуации. На основе этого мы приходим к выводу, что нейронные сети принадлежат к классу memory-bound алгоритмов. Отдельно отметим, что при обучении эта проблема не стоит столь остро из-за возможности использовать при тренировке нейронных сетей большой размер пакета.

Наиболее сложной является обработка пакетов размером 1, что характерно для систем искусственного интеллекта, применяющихся для решения задач управления, робототехники и обработки потокового видео, а также других задач, связанных с обработкой данных из реального мира. При этом, подобные задачи

часто предъявляют строгие требования к энергопотреблению, времени отклика и пропускной способности системы ИИ.

Данная проблема не является алгоритмической, поскольку существуют принципиальные ограничения на минимизацию числа обращений к памяти при матричном умножении, и обусловливается именно принципами организации архитектуры фон Неймана. Именно доступ к памяти является *основным препятствием* для достижения эффективности современных систем искусственного интеллекта как по временным, так и по энергетическим показателям. Далее будут рассмотрены основные методы преодоления ограничений («бутылочных горлышек») архитектуры фон Неймана в различных современных вычислительных системах.

1.4 Аппаратное обеспечение ИИ

Данный раздел посвящен анализу современных аппаратных архитектур для выполнения нейросетевых вычислений. Мы рассмотрим принципы работы CPU, GPU и TPU, чтобы выявить их фундаментальные ограничения в контексте задач инференса нейронных сетей при малом размере пакета.

1.4.1 CPU

В центральном процессоре (central processing unit, CPU) проблема задержки памяти традиционно решается посредством сложной многоуровневой системы кэшей (см. рисунок 1.2) [17]. В современных процессорах кэш-память может занимать до 40% площади кристалла, предоставляя десятки мегабайт высокоскоростной памяти. Размеры современных нейронных сетей обычно превышают объем кэш-памяти, что не позволяет разместить все параметры нейросети в кэше. Однако появление процессоров с технологией вертикально расположенного кэша (к примеру, процессор AMD Ryzen 7 5800X3D) способствует увеличению объема кэш-памяти, что может изменить текущую ситуацию.

Другими традиционными подходами к аппаратной оптимизации исполнения кода являются спекулятивные вычисления и предсказание ветвлений и др. [17]. Однако при умножении матриц порядок вычислений определен заранее, что делает данные механизмы нецелесообразными. Это означает, что в области искусственных нейронных сетей CPU эффективен только для вычисления сетей с малым количеством параметров и неприменим для современных крупных архитектур с количеством параметров более нескольких сот миллионов. В последующих разделах будут рассмотрены методы оптимизации нейронных сетей посредством квантования (см. гл. 3.2.1) и обрезания (см. гл. 3.2.2), позволяющие существенно сократить размеры нейронных сетей и обеспечить эффективное выполнение больших нейронных сетей на CPU.

1.4.2 GPU

Графический процессор (graphics processing unit, GPU) представляет собой устройство с массивно-параллельной архитектурой. Его структура включает множество вычислительных ядер. Данная организация обеспечивает выполнение одного потока инструкций над множеством потоков данных (SIMD - Single Instruction Multiple Data). Вычислительные ядра, объединены в потоковые мультипроцессоры (streaming multiprocessors) благодаря чему GPU способен одновременно обрабатывать несколько потоков SIMD.

Графический процессор применяет несколько стратегий для минимизации задержек памяти. Основная стратегия заключается в предоставлении каждому потоковому мультипроцессору большого файла регистров, который сохраняет контекст выполнения множества SIMD-потоков и обеспечивает их быстрое переключение. Планировщик вычислений использует данную функциональность следующим образом: при возникновении длительной задержки в SIMD-потоке 1, например, при получении данных из памяти, происходит немедленное переключение на SIMD-поток 2. В случае возникновения задержки во втором потоке, планировщик инициирует выполнение SIMD-потока 3. После поступления данных первый поток становится готовым к выполнению (см. рис. 1.8). Данный механизм позволяет скрывать задержки памяти [17].

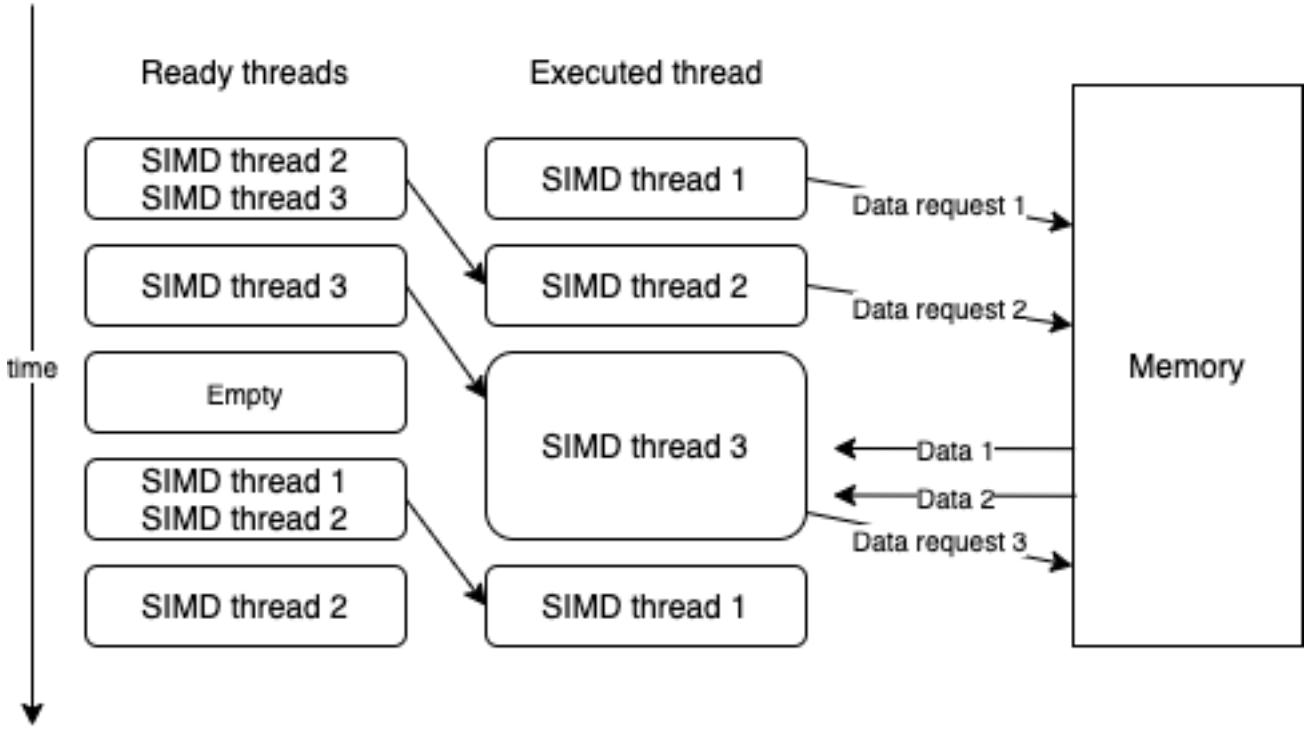


Рисунок 1.8 — Переключение между SIMD-потоками. Рисунок автора из статьи [A.1].

Помимо задержки памяти, существенным фактором для провизодительности вычислительной системы является пропускная способность памяти - максимальный объем данных, передаваемый из памяти в единицу времени. Компания Nvidia решила данную проблему внедрением памяти с высокой пропускной способностью (High Bandwidth Memory, HBM) в графических процессорах, начиная с чипа Nvidia P100 (2016), что значительно повысило их производительность. В последующих архитектурах Volta, Ampere и Hopper компания продолжила увеличивать пропускную способности памяти, достигнув показателей 1,5 ТБ/с на чипе A100 [18] и 3,35 ТБ/с на чипе H100.

Однако данные оптимизации хорошо работают при больших размерах пакетов, что и объясняет высокую популярность GPU при обучении НС. В случае работы с малыми размерами пакетов утилизация GPU может снижаться до 10-20 процентов от пиковой, а исполнение больших НС будет иметь большие задержки и низкую пропускную способность при высоком энергопотреблении.

1.4.3 TPU

Компания Google представила свой первый тензорный процессор (tensor processing unit, TPU) — TPUv1 в 2016 году [19]. Данное устройство оптимизирует задержки и повышает пропускную способность памяти посредством применения *систолических матриц (systolic array)*. Концепция систолических вычислений основана на формировании большой матрицы вычислительных блоков размером 256×256 для TPUv1. Каждый вычислительный блок содержит весовой коэффициент и выполняет две последовательные операции. Первая операция заключается в умножении числа x, поступающего из вышележащего блока, на весовой коэффициент с последующим сложением результата с числом из левого блока. Вторая операция обеспечивает передачу числа x в нижележащий блок и полученной суммы в правый блок. Данная архитектура позволяет TPU осуществлять конвейерное умножение матриц. При достаточном размере пакета отсутствует необходимость в постоянном обращении к весовым коэффициентам в памяти, поскольку они хранятся непосредственно в вычислительных блоках. TPU способен генерировать результат умножения матрицы 256x256 на вектор длиной 256 на каждый цикле. Однако TPU, как и GPU, ориентирован на исполнение задач с большими размерами пакетов.

Таким образом, современные наиболее распространенные аппаратные архитектуры такие как CPU, GPU и TPU плохо подходят для задач инференса НС при малом размере пакета.

1.5 Обучение с подкреплением

Обучение с подкреплением (Reinforcement Learning, RL) представляет собой подраздел машинного обучения, основная задача которого заключается в разработке алгоритмических методов, позволяющих оптимизировать процесс принятия решений путем максимизации *сигнала вознаграждения (reward signal)* на основе эмпирических данных, получаемых в процессе *взаимодействия* с окружающей средой. В данной области обучающаяся программа формализуется как *политика действий (action policy)* или *политика (policy)*, которая реализует отоб-

ражение из пространства состояний в пространство возможных действий [20]. RL играет большую роль в задачах управления и имеет большое количество применений в реальном мире, например, в робототехнике [21], управлении сложными устройствами, такими как токамак [22], играх [2] и биржевой торговле.

1.5.1 Основные понятия обучения с подкреплением

Введем основные понятия обучения с подкреплением.

Агент (agent) - программа, осуществляющая процесс обучения и принятия решений на основе получаемых данных.

Среда (environment) - внешняя система с которой взаимодействует агент для выполнения задач и получения опыта. Она включает в себя набор всех возможных состояний, правила перехода между ними, а также механизмы реагирования на действия агента. Среда обеспечивает обратную связь, предоставляя агенту информацию о текущем состоянии, реагирует на его действия изменением состояний и выдачей сигналов подкрепления (награды или штрафы). Благодаря этой обратной связи агент может *обучаться и адаптироваться* для достижения целей.

Ключевыми характеристиками среды являются:

- **Состояния (States)**. Среда характеризуется множеством возможных состояний, которые могут отражать её физическую конфигурацию, наблюдаемые характеристики или любые другие аспекты, релевантные для задачи.
- **Действия (Actions)**. В каждом состоянии агенту доступен определенный набор действий. Взаимодействие агента с окружающей средой посредством этих действий формирует основу процесса обучения.
- **Переходы (Transitions)**. Окружающая среда определяет правила перехода между состояниями в ответ на действия агента. Эти переходы могут быть детерминированными или стохастическими.
- **Награды (Rewards)**. После каждого действия окружающая среда предоставляет агенту награду (или штраф), которая указывает на успешность или неудачу предпринятого действия в контексте задачи. Функция награды является ключевым механизмом для обучения агента, поскольку

она направляет процесс оптимизации стратегии агента для максимизации суммарной награды.

1.5.2 Математическая постановка задачи обучения с подкреплением

Перейдем к формальному определению задачи обучения с подкреплением. Данный математический аппарат формализует понятие оптимального поведения агента с помощью задачи максимизации дисконтированной кумулятивной награды, и устанавливает уравнения Беллмана, которые связывают ценность текущего состояния с ценностью будущих состояний. Этот подход лежит в основе всех современных алгоритмов обучения с подкреплением.

Марковский процесс принятия решений

В математических терминах задача обучения с подкреплением формулируется через концепцию *Марковского процесса принятия решений (МППР, Markov Decision Process, MDP)* [20], который представляет собой математическую модель для описания взаимодействия агента с окружающей его средой в терминах состояний, действий и наград (см рис. 1.9, 1.10). МППР определяется четверкой основных компонент $< S,A,P,R >$:

1. Множество состояний S.
2. Множество действий A.
3. Функция перехода $P(s' \mid s, a)$ определяет вероятность перехода системы из состояния s в состояние s' при выполнении действия a. При этом выполняется условие нормировки: $\sum_{s' \in S} P(s', \mid s, a) = 1, \quad \forall s, s' \in S, a \in A$. Данная функция отражает динамику среды и является основополагающей для понимания изменений в среде в ответ на действия агента.
4. Функция награды $R(s',a,s)$ определяет численное значение вознаграждения или штрафа, получаемое агентом при переходе из состояния s в состояние s' в результате выполнения действия a.

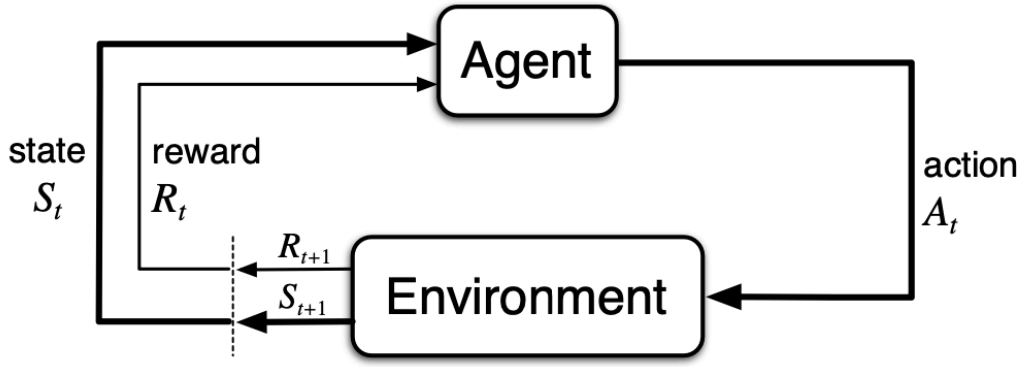


Рисунок 1.9 — Взаимодействие среды и агента в Марковском процессе принятия решений. Рисунок заимствован из сети Интернет.

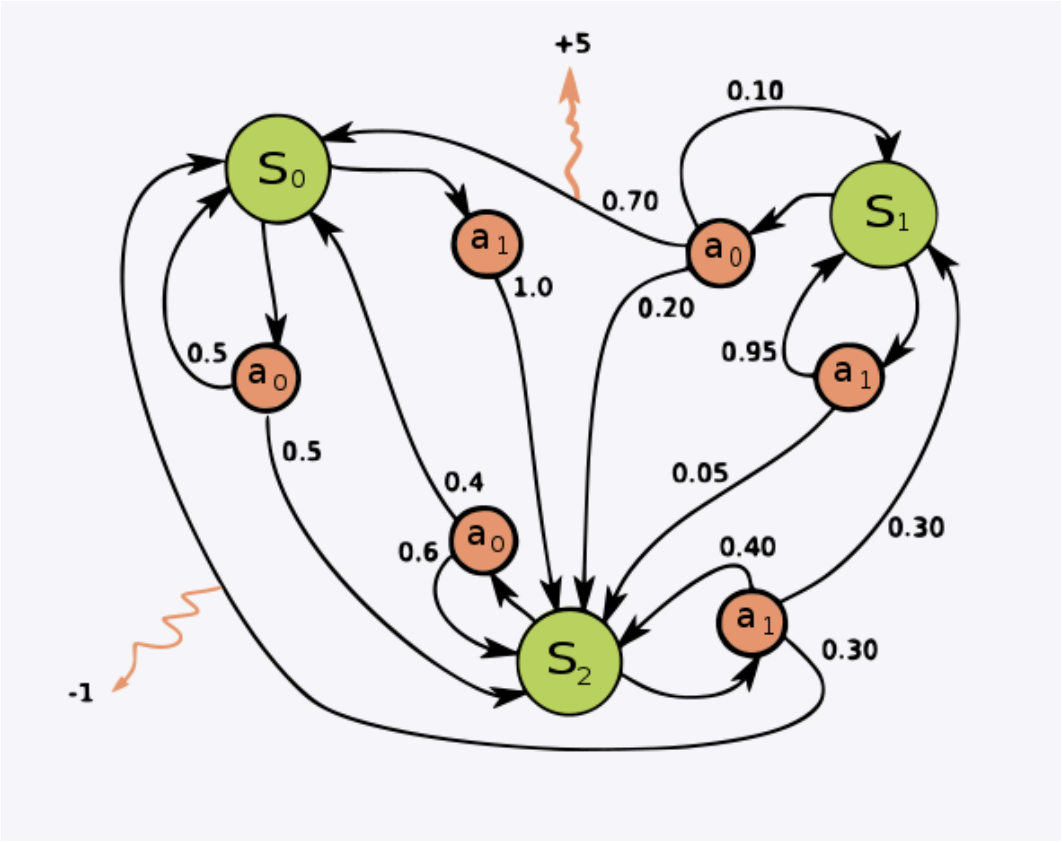


Рисунок 1.10 — Пример МППР представленного в виде графа с тремя состояниями и двумя действиями. Рисунок заимствован из сети Интернет.

Агент в рамках МППР осуществляет выбор действий на основе политики, $\pi(a|s)$, которая для каждого состояния s определяет вероятность выбора каждого возможного действия a.

Целью агента в рамках МППР является нахождение *оптимальной политики* π^*, максимизирующей значения функции (суммы дисконтированных будущих наград) $v_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right]$, где $\mathbb{E}$ это математическое ожидание суммы случайных величин, получаемых при условии, что агент действует в соответствии с политикой π из состояния s. Функция $v_\pi(s)$ называется *функцией ценности (value function)* и играет центральную роль в обучении с подкреплением наряду с *функцией Q (Q-function)*, которая определяется как $q_\pi(s,a) \doteq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right]$.

Оптимальной политикой π^* называется такая политика, что

$$v_*(s) \doteq \max_\pi v_\pi(s) \quad \forall s \in S$$

Аналогично выглядит для оптимальных политик функция Q:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a) \quad \forall s \in S \quad \forall a \in A$$

Уравнения Беллмана

Для функции v_π верно следующее соотношение [20]:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s' \mid s,a) \left[R(s,a,s') + \gamma v_\pi(s') \right], \quad \forall s \in S, \tag{1.4}$$

Данное уравнение называется *уравнением Беллмана (Bellman equation)* для политики v_π. Оно связывает значение (ценность) состояния со значениями (ценностями) последующих состояний.

Поскольку v_* является функцией ценности политики, она должна удовлетворять уравнению Беллмана 1.4 для функции ценности. В силу того, что v_* представляет собой функцию оптимального значения, её условие согласованности может быть записано в специальной форме без привязки к конкретной

политике. Это уравнение Беллмана для v_* или *уравнение оптимальности Беллмана*.

$$v_*(s) = \max_{a \in A(s)} q_{\pi_*}(s,a) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]. \tag{1.5}$$

В свою очередь для q_* уравнение оптимальности Беллмана будет выглядеть как:

$$q_*(s,a) = \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]. \tag{1.6}$$

Явное решение уравнения оптимальности Беллмана является одним из способов нахождения оптимальной стратегии и, таким образом, решения проблемы обучения с подкреплением. Однако данный подход редко бывает применим на практике. Метод основан на полном переборе всех возможных состояний системы с последующим вычислением вероятностей возникновения различных ситуаций и их оценкой с точки зрения ожидаемого вознаграждения. Возможность реализации данного метода базируется, как минимум, на трех предположениях, которые редко выполняются на практике: (1) наличия полной информации о динамике окружающей среды; (2) достаточности вычислительных ресурсов для получения решения; (3) выполнения марковского свойства. В большинстве практических задач точная реализация данного метода невозможна ввиду нарушения одного или нескольких вышеуказанных предположений.

В связи с этим, при обучении с подкреплением обычно приходится довольствоваться приближенными решениями [20]. Одним из вариантов получения приближенных решения являются алгоритмы обучения с подкреплением, основанные на нейронных сетях. В этих алгоритмах, обычно, происходит аппроксимация функций v_π, q_π посредством нейронных сетей.

Особого внимания заслуживает специфика процесса инференса в задачах обучения с подкреплением. Характерной особенностью данного процесса является то, что размер пакета равен единице, поскольку агент получает и обрабатывает состояния среды последовательно, по одному в каждый момент времени. То есть при инференсе в задачах RL наиболее сильно проявляются вычислительные проблемы нейронных сетей, описанные в разделе 1.3. Основная цель данной работы заключается в нахождении способов оптимизации систем ИИ для задач обучения с подкреплением.

1.6 Нейроморфные подходы

Несмотря на значительный прогресс аппаратного обеспечения, описанного выше, и его доминирование на рынке, системы искусственного интеллекта на его основе все еще далеки от своих биологических аналогов. Существует большое различие в уровне энергопотребления, гибкости и масштабируемости. Более того, существующие системы, прежде всего, нацелены на решение задач с батчем большем единицы, что кардинально их отличает от мозга человека, который функционирует в режиме потоковой обработки информации.

Мозг человека — это пример принципиально иного, не фон-неймановского вычислителя. В отличие от классических нейронных сетей, реализованных в современных вычислительных системах, в мозге человека:

- Энергопотребление составляет всего десятки ватт. Это на порядки меньше потребления современных систем искусственного интеллекта.
- Нейроны обмениваются информацией с помощью дискретных импульсов, т. е. спайков, а не чисел.
- Все события передаются и принимаются асинхронно — не существует единого процесса, явно синхронизирующего работу всех нейронов.
- Не существует общей памяти, подобной той, с которой работают современные вычислительные устройства. Вместо этого большое количество простых вычислительных ячеек (нейронов), являющихся одновременно и памятью, и вычислителем, функционируют самоорганизующимся образом.

В связи с этим можно предположить, что реализация важнейших свойств и принципов работы мозга человека могла бы помочь сократить отставание современных систем ИИ в энергоэффективности, гибкости и масштабируемости. *Нейроморфный подход* к разработке систем искусственного интеллекта является ответом на этот запрос и пытается использовать принципы организации и функционирования мозга человека в вычислительных системах.

1.7 Выводы по главе 1

В данной главе были рассмотрены основные принципы работы современных систем ИИ, построенных на основе вычислителей на архитектуре фон Неймана и искусственных нейронных сетей. Недостатки современных систем ИИ объясняются свойствами искусственных нейронных сетей и свойствами аппаратного обеспечения. Нейронные сети относятся к memory-bound алгоритмам. Получение данных из памяти на архитектуре фон Неймана - это крайне дорогая операция с точки зрения времени и энергии, при этом шина данных имеет ограниченную пропускную способность. Именно эта проблема ведет в современных системах ИИ к низкой скорости работы, ограничениям по пропускной способности и большим энергозатратам, что делает их использование проблематичным.

Было показано, что особенно остро эта проблема проявляется при инференсе в задачах управления, в частности RL, в которых размер пакета обычно равен 1, что еще больше усугубляет свойство memory-bound из-за низкой вычислительной интенсивности.

Предлагается выделить принципы функционирования мозга человека, чтобы имплементировать их в системы ИИ для задач RL для повышения их энергоэффективности, скорости работы, масштабируемости и оптимизации размера сети.

Глава 2. Принципы работы мозга человека для конструирования биологически подобных систем ИИ

Мозг человека, является самым совершенным когнитивным устройством из известных человечеству. Он легко способен проводить обобщения, показывает быструю обучаемость, адаптивность и низкое энергопотребление. Он представляет из себя огромную нейронную сеть, состоящую примерно из 85 млрд нейронов с общим числом синапсов (связей) примерно в 850 - 1000 триллионов [23].

Современные системы ИИ отстают по многим из этих признаков от человеческого мозга. К примеру, по своему энергопотреблению (см таблицу 1), системы ИИ на порядки превышают энергопотребление мозга человека, при этом число связей в современных нейронных сетях отстает на несколько порядков от мозга человека. Причем оценки для запуска нейронных сетей на нескольких GPU не учитывает затраты энергии на обмен данными между видеокартами (как известно, сетевой обмен более дорог чем взятие данных из памяти).

Система	Число параметров (связей)	Энергопотребление
GPT-3 + Nvidia H100	175 млрд	3600 вт (нижняя оценка)
Мозг человека	850-1000 трлн	20 вт

Таблица 1 — Сравнение числа параметров системы интеллекта и энергопотребления.

Возникает вопрос: в чем причины столь высокой эффективности мозга человека и каковы принципы его устройства? Также поднимается вопрос о возможности создания биоподобных искусственных интеллектуальных систем, которые могли бы сравняться с мозгом человека в энергоэффективности и масштабируемости. Ответы на эти вопросы ищет область нейроморфных технологий [A.1], [24], [25], которая стремительно развивается в последние годы.

Чтобы найти ответы на вышепоставленные вопросы, нужно понять принципы работы мозга человека. Начнем с описания биологического нейрона и принципов его работы.

2.1 Биологический нейрон

В основе работы мозга человека лежат нейроны – специализированные клетки, которые обрабатывают и передают информацию с помощью электрических и химических сигналов.

Нейрон состоит из трех основных частей: тела, дендритов и аксона.

- Тело нейрона (сома) содержит ядро и является центральной частью нейрона, где происходит обработка поступающей информации.
- Дендриты представляют собой множество отростков, которые распространяются от тела нейрона и принимают сигналы от других нейронов через синапсы.
- Аксон – это длинный отросток, который передает электрические сигналы от тела нейрона к другим нейронам или эффекторным клеткам.

Соединение между нейронами происходит через синапсы – специализированные структуры, которые позволяют передавать сигналы от одного нейрона к другому. В синапсе сигнал обычно передается химическим путем с помощью нейромедиаторов.

Основой функционирования нейрона является способность генерировать и передавать электрические сигналы. Эти сигналы возникают благодаря изменениям мембранного потенциала – разности потенциалов между внутренней и внешней сторонами нейронной мембраны, который является фундаментальным свойством всех живых клеток и, в частности, нейронов (см рис. 2.1). Мембранный потенциал возникает из-за разности концентраций ионов внутри и снаружи клетки (в основном натрия (Na+) и калия (K+)) а также из-за различий в проницаемости мембраны для этих ионов. Мембрана нейрона обладает ионными каналами и насосами, которые позволяют ионам перемещаться внутрь и наружу клетки, следуя их концентрационным и электрическим градиентами.

Обычно, нейрон стремится к состоянию покоя, т.е. к мембранному потенциалу равному *потенциалу покоя*, лежащему в диапазоне от -60 до -70 мВ. Потенциал покоя необходим для поддержания готовности нейрона к активации, то есть к генерации потенциала действия в ответ на стимуляцию.

Потенциал действия – это резкое, кратковременное изменение мембранного потенциала, во время которого он становится положительным (деполяризация), а затем возвращается к отрицательному значению (реполяризация). Потенциал

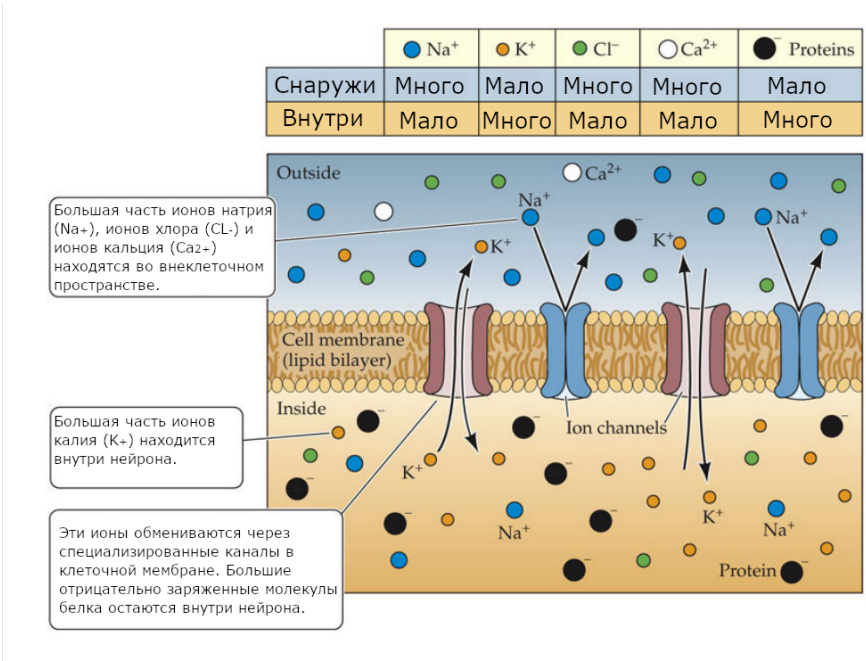


Рисунок 2.1 — Мембрана нейрона и поток ионов через нее. Рисунок заимствован из [26].

действия инициируется, когда достигается *пороговое значение*(примерно -55мВ) мембранного потенциала.

Изменение поляризации происходит из-за получения сигналов от других нейронов. Они могут быть как возбуждающими, так и тормозящими. Возбуждающие сигналы вызывают деполяризацию мембраны - процесс, при котором мембранный потенциал становится менее отрицательным (приближается к нулю) относительно потенциала покоя. Тормозящие ведут, наоборот, к реполяризации мембраны. Если потенциал мембраны, вызванный этими сигналами, достигает порогового уровня деполяризации, то нейрон генерирует потенциал действия. Принципы работы вышеописанного процесса изображены на рис. 2.2

Мембранный потенциал имеет ключевое значение для функционирования нейронов, поскольку он обеспечивает основу для генерации потенциалов действия, которые являются **основным механизмом передачи информации** в нервной системе. Контроль над мембранным потенциалом и его изменениями позволяет нейронам обрабатывать информацию, интегрировать сигналы и коммуницировать с другими нейронами, формируя основу для всех нейронных функций, от рефлексов до мышления.

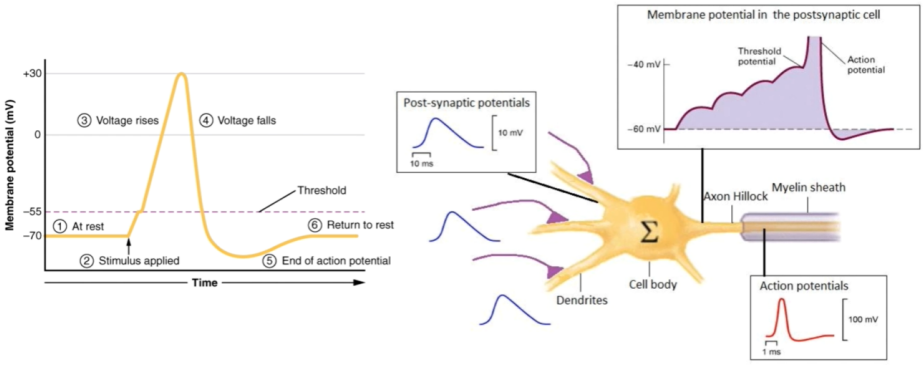


Рисунок 2.2 — Мембрана и мембранный потенциал, образуемый потоком ионов. Рисунок заимствован из [27].

2.1.1 Математические модели биологического нейрона

Очевидно, что модель нейрона Розенблатта 1.2.1 крайне далека от принципов работы биологического нейрона. Однако существует большое количество более биоподобных моделей нейронов. Вычислительная нейробиология [28] как раз и занимается разработкой таких моделей. Здесь мы ограничимся описанием одной простой, но более биоподобной модели нейрона, чем нейрон Розенблатта. Она является одной из наиболее распространенных и часто применяется в компьютерных симуляциях.

Модель LIF нейрона

Модель LIF (Leaky Integrate-and-Fire) нейрона описывает динамику мембранного потенциала нейрона как процесс интегрирования входных сигналов с последующим пороговым возбуждением. В этой модели мембранный потенциал $V(t)$ нейрона изменяется согласно следующим уравнениям:

$$\tau \frac{dV}{dt} = -(V(t) - V_{rest}) + RI_{ext}(t) \tag{2.1}$$

$$\text{if V(t)} \geqslant threshold => V(T) = V_0 \tag{2.2}$$

где:

– $V(t)$ - потенциал нейрона,

– V_{rest} - потенциал покоя,
– R - мембранное сопротивление,
– τ - мембранная константа.

Когда мембранный потенциал достигает определенного порогового значения $threshold$, нейрон генерирует потенциал действия, и мембранный потенциал сбрасывается до начального уровня V_0, после чего может начаться новый цикл интеграции.

Мембранный потенциал изменяется под влиянием I_{ext}. Этим слагаемым могут моделироваться различные виды внешней стимуляции, включая синаптическую активность от других нейронов.

Семейство LIF моделей нейронов несовершенно и, как любая модель, имеет свои ограничения. К примеру, в ней не описывается механика возникновения потенциала действия, которая просто заменяется условным выражением.

Ограниченность вычислительных ресурсов очень редко позволяет использовать сложные модели нейронов при моделировании больших сетей нейронов. Этим отчасти объясняется столь высокая популярность модели Розенблатта. Она очень «удобна» и проста для вычислительной техники (хотя несомненно есть более значимый фактор - её дифференцируемость).

2.2 Вычислительные принципы работы мозга человека

Рассмотрев некоторые базовые принципы функционирования мозга человека, попробуем взглянуть на него более системно и выделить вычислительные принципы его работы, которые могли бы быть перенесены в современные системы ИИ. В данной главе мы постараемся выделить наиболее важные вычислительные принципы функционирования мозга человека [A.1], а также рассмотрим их сходства и различия с механизмами функционирования современных систем ИИ. Кроме того, мы обсудим проекты вычислительных систем, в которых предпринимаются попытки использовать эти принципы. Представленная классификация вычислительных принципов работы мозга человека была изложена автором в статье «Neuromorphic Artificial Intelligence Systems» [A.1].

2.2.1 Коннекционизм

Центральная идея коннекционизма заключается в том, что ментальные феномены могут быть описаны через призму нейронных сетей [29].

Компьютерные нейронные сети представляют собой вычислительные системы, вдохновлённые биологическими нейронными сетями. Они состоят из большого количества простых элементов, каждый из которых моделирует биологический нейрон с различной степенью точности. Эти элементы соединены между собой весами - искусственными аналогами синапсов, связывающих нейроны друг с другом. Любое ментальное состояние в рамках нейронной сети представляется в виде вектора активаций нейронов.

Обучение нейронной сети происходит через подбор оптимальных весовых коэффициентов, что позволяет эффективно решать разнообразные задачи. В ходе множества экспериментальных исследований было доказано, что нейронные сети способны осваивать различные умения, включая распознавание образов, моделирование языковых структур, компьютерные игры, ведение диалогов и другие [30].

Способность решать конкретную проблему и качество её решения могут определяться как моделью нейрона, так и топологией сети. Так как самые передовые алгоритмы ИИ используют данный принцип, то можно говорить, что он уже имплементирован в системы ИИ.

2.2.2 Параллелизм

Каждый нейрон в мозге человека является автономным вычислительным устройством, однако его скорость значительно уступает современным кремниевым процессорам (по оценкам частота работы нейрона не превышает 1кГц [28]). Тем не менее, количество нейронов в мозге человека, выполняющих скоординированную работу, достигает 87 миллиардов. Ещё в конце 1980-х годов исследователи [31] пришли к выводу, что для эффективного функционирования нейронных сетей требуются массивно-параллельные архитектуры. В поддержку этой идеи служит тот факт, что именно массовое использование высокопараллель-

ных архитектур (в основном GPU) обеспечило современный успех нейронных сетей.

Практически все аппаратные платформы, использующиеся для работы нейронных сетей, поддерживают параллелизм или на векторном уровне, и/или на уровне вычислительных ядер (параллелизм потоков управления), и/или на уровне вычислительных узлов (параллелизм задач).

2.2.3 Импульсный характер передачи информации

Как уже было сказано, нейроны способны индуцировать потенциал действия (спайк), который затем распространяется по аксону к синапсам. Таким образом, информация в мозге человека практически всегда передается в виде бинарных нервных импульсов [28], [32], которые распространяются вдоль нервных волокон и имеют примерно одинаковую продолжительность и амплитуду. Исключением у млекопитающих являются клетки сетчатки, которые передают информацию не в виде бинарных, а в виде континуальных импульсов.

Идея использовать модель нейрона, оперирующую спайками для создания сетей нейронов для решения задач распознавания образов, была предложена в 1997 [33]. Она получила название *импульсных нейронных сетей (ИНС, Spiking Neural Networks, SNN)* [34]. В основе SNN (см. рис. 2.3) лежат нейроны, основанные на LIF модели нейрона (см. гл. 2.1.1) или её вариаций, и соединенные связями, которым приписаны определенные веса. В SNN нейроны асинхронно обмениваются спайками, то есть элементарными событиями, не имеющими других атрибутов, кроме времени их генерации.

Стоит отметить что в мозге человека передача спайка от нейрона к нейрону не происходит мгновенно, но требует некоторого времени, которое варьируется для различных пар нейронов. Таким образом, каждый синапс можно характеризовать не только весом, но и временем задержки. В связи с этим в SNN, иногда, также добавляют время задержки связи. Время спайков и задержки служат механизмом для явного введения времени в вычислительную модель.

Исполнение импульсных нейронных сетей обычно сводится к пошаговой численной симуляции системы уравнений [28], описывающих нейроны с помощью метода Эйлера [35]. Шаг симуляции для модели LIF нейрона 2.1.1

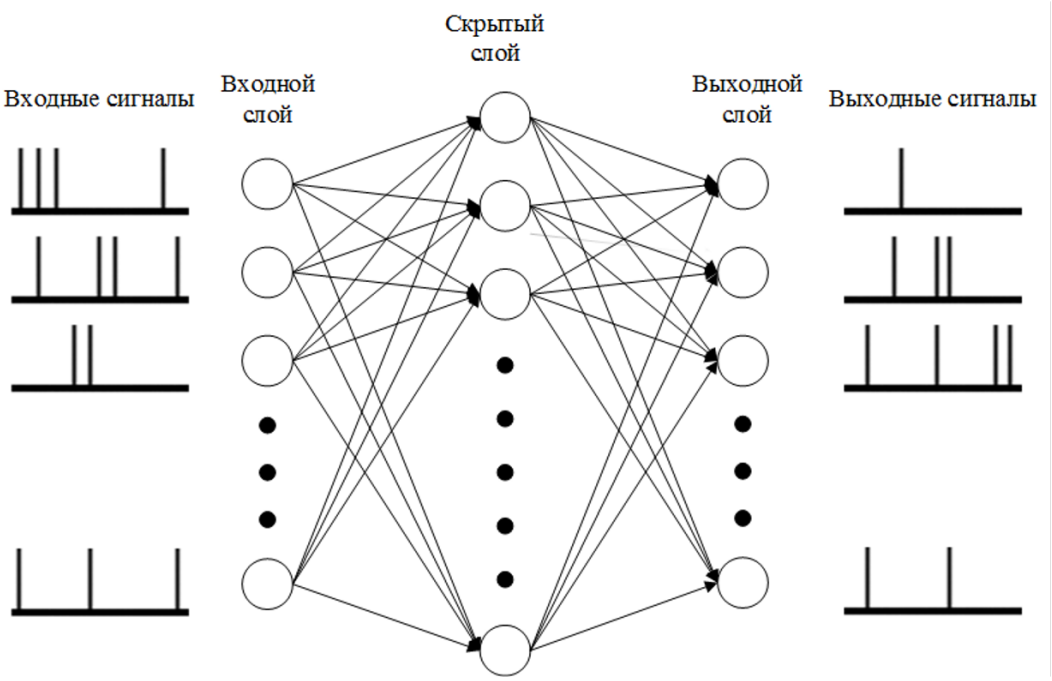


Рисунок 2.3 — Пример импульсной нейронной сети. Рисунок заимствован из сети Интернет.

может быть реализован исключительно с помощью операций сложения и вычитания. Особую проблему вызывает кодирование численных данных, которое может происходить как частотным способом (т.е. число импульсов в окне пропорционально/log-пропорционально численному значению), так и позиционным способом (к примеру, каждый нейрон отвечает за то или иное значение в дискретном множестве).

Рассмотрим теперь сильные и слабые стороны импульсных нейронных сетей. К несомненным преимуществам импульсных нейронных сетей можно отнести следующие факты:

– Данные могут передаваться между нейронами простым асинхронным способом;
– SNN позволяют работать с динамическими данными, поскольку в них явно включен временной компонент;
– SNN могут быть энергоэффективными. Активность нейрона сводится к его реакции на прибывающий спайк, после чего нейрон переходит в неактивное состояние, потребляющее лишь небольшое количество энергии [36]. Таким образом, в каждый момент времени только небольшая часть нейронов в сети находится в «активном режиме» и потребляет энергию.

Однако они обладают большим количеством недостатков:

– Сложность обучения. Одной из главных проблем ИНС является сложность их обучения. Методы обратного распространения ошибки, хорошо зарекомендовавшие себя в области традиционных нейронных сетей, не всегда подходят для ИНС из-за их временной динамики и специфики передачи информации через импульсы (невозможно дифференцировать). Существуют разные подходы к решению данной проблемы такие как создание специализированных вариаций алгоритмов обратного распространения ошибки [37], обучения обычной сети, с последующей её конвертацией в импульсную форму [38], переход к методам локального обучения [39]. Разработка эффективных алгоритмов обучения, которые могли бы учитывать особенности импульсной активности, остаётся актуальной задачей.
– Более низкое качество работы. На классических тестах проверки качеств работы систем глубокого машинного обучения, импульсные нейронные сети отстают от своих классических, менее биоподобных собратьев-аналогов. Тем не менее стоит отметить существенный прогресс в сокращении отставания в последние годы. Также важным моментом может быть тот факт, что SNN, из-за своей природы, будут себя лучше показывать на каких-то более специфических задачах, где их свойства могут проявить себя с сильной стороны [34].
– Вычислительная сложность. Вопреки тому что SNN потенциально могут быть более энергоэффективными, реализация таких сетей на стандартном оборудовании часто приводит к значительной вычислительной сложности, так как необходимо численно симулировать систему нейронов, а численные данные кодировать частотно. То есть вместо операции умножения входа на вес в персептроне Розенблатта в SNN необходимо будет сделать на несколько порядков больше операций, так как необходимо просимулировать несколько сот шагов чтобы принять импульсы, кодирующие вход. Несмотря на то, что шаг симуляции SNN может быть реализован только с помощью операций сложения и вычитания [34], учитывая современные вычислительные системы, это не выглядит значимым преимуществом.

Учитывая все недостатки SNN пока нет никакого смысла их использовать на практике в качестве замены классических нейронных сетей для задач распознавания образов, обработки языка и обучения с подкреплением.

Причины энергоэффективности импульсных чипов

Сторонники SNN обещают решить как минимум одну из вышеперечисленных проблем импульсных сетей, а именно вычислительную сложность, с помощью использования специализированных чипов, адаптированных для работы с подобными сетями. Такие проекты как TrueNorth [40], Loihi [41], [42], NeuronFlow [43] показывают крайне низкое энергопотребление и высокую скорость работы при работе с алгоритмами импульсных нейронных сетей. Однако при пристальном анализе оказывается, что все эти чипы имеют большой объем быстрой и энергоэффективной SRAM памяти, что и дает вышеописанные преимущества, а не сам алгоритм импульсных нейронных сетей.

Доказательством этого является чип NorthPole [44], который также использует большой объем SRAM-памяти (см. таблицу 2), но не работает с импульсными нейронными сетями и показывает высокую энергоэффективность и низкое время отклика (см. рис. 2.6). Другими примерами чипов, использующих большой объем SRAM-памяти и работающих с классическими нейронными сетями, являются Groq [45] и Cerebras.

Передача информации импульсами, отличными от бинарных

Отдельный вопрос вызывает необходимость работать исключительно с бинарными спайками. В биологических нейронах это скорее является необходимостью для *передачи импульсов на большое расстояние*. Действительно, потенциал действия, как уже было упомянуто распространяется по аксону. Однако со временем он затухает , что является неизбежным следствием устройства мембраны клеток. Однако существуют перехваты Ранвье, которые заново восстанавливают сигнал. Происходит это за счет электрического поля, которое деполяризует мембрану соседних перехватов до критического уровня, что приводит к возникновению в них новых потенциалов действия, то есть возбуждение переходит скачкообразно, от одного перехвата к другому. Такой способ проведения информации называется сальтаторной проводимостью. Таким образом, если бы информация передавалась бы не бинарным способом, то перехват Ранвье дол-

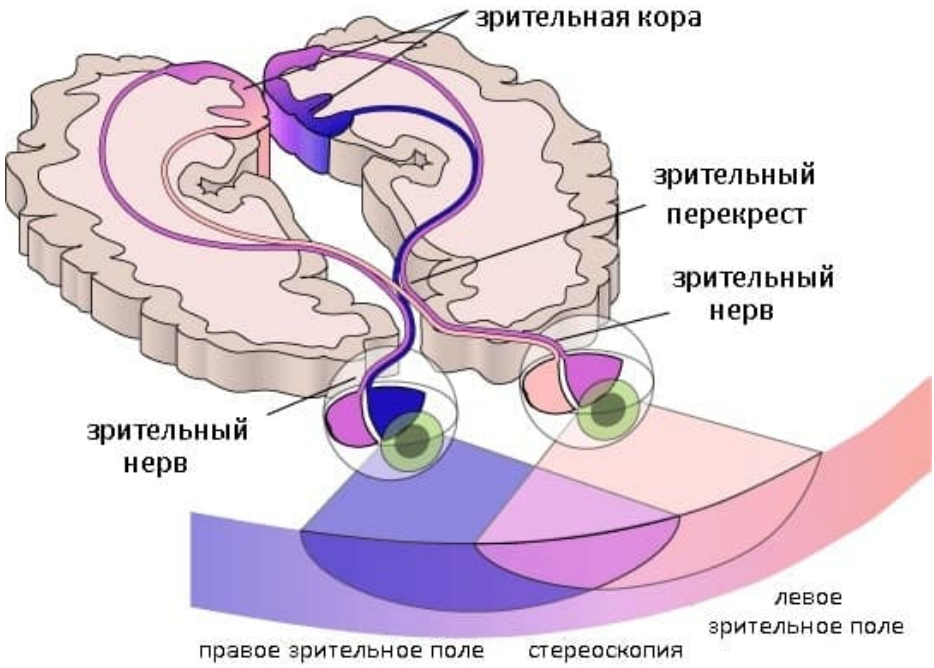


Рисунок 2.4 — Зрительная система человека. Рисунок заимствован из сети Интернет.

жен был как-то узнавать значение амплитуды исходного потенциала действия. Так как распад потенциала и расположение перехватов не являются жестко детерминированными явлениями, мы приходим к выводу, что точное восстановление невозможно.

Дополнительным аргументом, подтверждающим прошлые рассуждения, является тот факт, что обработка информации в сетчатке происходит как раз без помощи бинарных импульсов [32]. Визуальная информация становится бинарно импульсной только при её отправке по зрительному нерву в зрительную зону, что является примером передачи импульса на большое расстояние (см. рис. 2.4).

Также необходимо учитывать, что передача информации с помощью потенциала действия является энергозатратным процессом [46], [47] и при этом позволяет передавать исключительно сигнал 0/1. В то время как небинарной передачей информации можно передавать значительно больше информации при схожих затратах энергии. Алгоритмической попыткой реализовать небинарные импульсы в нейронных сетях являются работы команды чипа NeuronFlow. Алгоритм подробно разобран в гл. 3.2.3. Данный подход способен обеспечивать на ряде задач качество, сравнимое с классическими нейронными сетями. Данный тип импульсных нейронных сетей поддерживается процессором NeuronFlow [43].

2.2.4 Асинхронность

Когда требуется синхронизация между вычислительными узлами, сам по себе параллелизм не всегда обеспечивает желаемый вычислительный эффект. Одним из следствий из закона Амдала [48], является тот факт, что накладные расходы на синхронизацию растут нелинейно по мере увеличения числа вычислительных узлов, тем самым ограничивая прирост производительности от параллелизма. Действительно, закон Амдала определяется формулой:

$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}} \tag{2.3}$$

где:

– S_p - ускорение, которое может быть получено на системе из p процессоров;
– α - доля от общего объёма вычислений, которая может быть получена только последовательными расчётами (к примеру, синхронизация);
– p - число процессоров.

Учитывая частоту работы нейрона в 1кГц, получается, что если бы «программа» нейронов проводила бы явную синхронизацию между собой, то есть α был бы хоть сколько-то значимым числом, то не было бы смысла с точки зрения производительности в таком количестве нейронов, как есть в мозге человека [49].

Более того, синхронизация потребляет энергию. Например, в современных синхронных цифровых схемах древовидная структура распределения тактового сигнала внутри схемы потребляет 20–45% от общего энергопотребления чипа [50]. Но мозг человека, по-видимому, не имеет механизма, который бы явно синхронизировал работу всех нейронов. Биологические нейроны работают асинхронно, что позволяет обойти ограничения закона Амдала и избежать дополнительного энергопотребления на распространение сигнала синхронизации. Тем не менее, это не означает, что нейроны в мозге человека работают хаотично, определенная синхронизация возникает с помощью волн активности мозга, но явного механизма синхронизации в мозге не наблюдается.

В современных системах ИИ сама структура вычислений классических нейронных сетей налагает необходимость синхронизации, и, как следствие, закон

Амдала вносит серьезные ограничения на ускорение работы сетей [49] с помощью параллелизма.

Примером нейроморфного компьютера, который пытается обойти закон Амдала, для симуляции нейронных сетей является компьютер SpiNNaker [51], [52]. Работа узлов внутри всей вычислительной системы асинхронна по отношению друг к другу. Именно благодаря его асинхронным принципам функционирования у исследователей появилась возможность масштабировать моделирование мозга человека и млекопитающих практически без ограничений. Так, моделирование одной кортикальной колонки мозга (примерно 1 mm^2 поверхности мозга) можно реализовать на одном GPU [53], однако, из-за необходимости синхронизации вычислительных устройств, возникнет проблема масштабирования процесса моделирования на ситуацию с большим количеством колонок. В то же время, SpiNNaker, благодаря принципам асинхронности, заложенным в его архитектуре, способен масштабироваться с меньшими ограничениями. Другими примерами асинхронной архитектуры являются нейроморфные чипы Loihi [42], NeuronFlow [43] и другие.

Стоит отметить, что несмотря на все преимущества данного типа архитектур и потенциал их практически безграничного масштабирования, они не подходят для запуска современных нейронных сетей, которые в своей основе требуют синхронизации выполнения слоев. При этом импульсные нейронные сети, для запуска которых они подходят, и которые способны работать в асинхронном режиме, как было упомянуто выше, часто не способны достигать сравнимого качества на задачах машинного обучения. Исключением являются импульсные сети на основе небинарных активаций, которые были разобраны выше (см. 2.2.3). Они способны иметь качество, сравнимое с классическими нейросетями, но также поддерживают асинхронность исполнения.

2.2.5 Активационная разреженность

Хорошо известно [54], [55], что в мозге человека обычно одновременно активны менее 10% нейронов. Причем активность нейронов распределена между нейронами неравномерно, то есть существуют крайне редко активирующиеся нейроны. Они даже получили название «темных нейронов» по аналогии

с «темных веществом». Это явление можно объяснить тем, что большая часть нейронов работает по принципу активен/не активен. Нейрон может молчать даже при наличии входных сигналов, если суммарный потенциал, вызванный ими, не превосходит порог активации. Возникающая в результате разреженность потоков данных называется *активационной разреженностью (activation sparsity)*.

Это сильно отличается от режима вывода классических нейронных сетей, в которых в вычислениях участвуют все нейроны. Единственным исключением являются нейроны, в которых функцией активации является ReLU-функция, которая реализует идею похожую на принцип активации нейрона. То есть, активируется только в том случае, если сумма входных сигналов превысит порог активации (обычно 0 для ReLU). При использовании ReLU-активации значительное количество нейронов имеют выходной сигнал, равный нулю. Однако при вычислениях на графическом процессоре эти нули все равно будут умножаться, как и другие числа.

Также существует класс работ в которых пытаются поддерживать регулярный уровень разреженности на слоях нейронной сети (например должны активироваться только 10 процентов нейронов). Обычно это реализуется с помощью подхода на основе k-winner (т.е. k-победителей) [56], [57]. В этом случае из всех нейронов слоя будут активироваться k нейронов с наибольшей суммой взвешенных входов. В отличие от подхода на основе ReLU, в котором заранее неизвестен процент нулевых активаций, данный подход позволяет стабилизировать число необходимых вычислений на каждом шаге нейронов.

Стоит отметить ещё один важный факт об активационной разреженности, связанный с кодированием информации. Одной из важных гипотез в современной нейронауке является гипотеза о том, что нейроны используют разреженное кодирование информации, схожее с математическим аппаратом разреженной распределенной памяти (Sparse Distributed Memory), разработанного P. Kanerva [58]. Кодирование высокоразреженными кодами информации (образов, воспоминаний) является устойчивой к сбоям (потери части информации об адресе), а вероятность пересечения сигналов и, как следствие, неправильного восстановления данных, является крайне малой.

Активационная разреженность поддерживается практически всеми процессорами, работающими с импульсными нейронными сетями (в силу особенностей их работы), к примеру Loihi [42], TrueNorth [40], Tianjic [59] и Neuronflow [43]. Для

активационной разреженности в классических нейросетях был предложен экспериментальный чип EIE [60].

2.2.6 Временна́я разреженность

Мозг человека не стремится обрабатывать всю поступающую информацию заново с нуля, а обычно фокусируется на изменениях, произошедших с предыдущего момента времени. Так зрение больше фокусируется именно на измененных областях видимого изображения, что помогает быстрее обрабатывать информацию и экономить ресурсы на её обработку. Этому способствует высокая корреляция изображений в моменты времени t и $t-1$ Данный эффект мы будем в дальнейшем называть *временно́й разреженностью*.

Временна́я разреженность находит свое применение в задачах сжатия видеофайлов в группе алгоритмов, основанных на компенсации движения, которые являются одним из наиболее эффективных технологий в области кодирования видеоданных с потерями. Основная идея подобных алгоритмов заключается в том, что вместо того чтобы хранить каждый кадр целиком, система хранит только первый кадр и информацию о движении объектов. Для остальных кадров сохраняется информация только о величине и направлении движения, что позволяет существенно сократить объём необходимой для хранения информации. Данный подход позволяет значительно уменьшить объем данных без потери качества изображения, что делает его неотъемлемой частью современных стандартов видеосжатия, таких как H.264 и MPEG-4.

Вместе с небинарными импульсами, описанными выше, явление временно́й разреженности находит свое применение в нейронных сетях. В их основе лежит применение алгоритма SpArNet [61], который позволяет эффективно использовать временну́ю разреженность в нейронных сетях. Данный алгоритм был разработан командой чипа NeuronFlow [43], который поддерживает временну́ю разреженность, и подробно разобран в гл. 3.2.3.

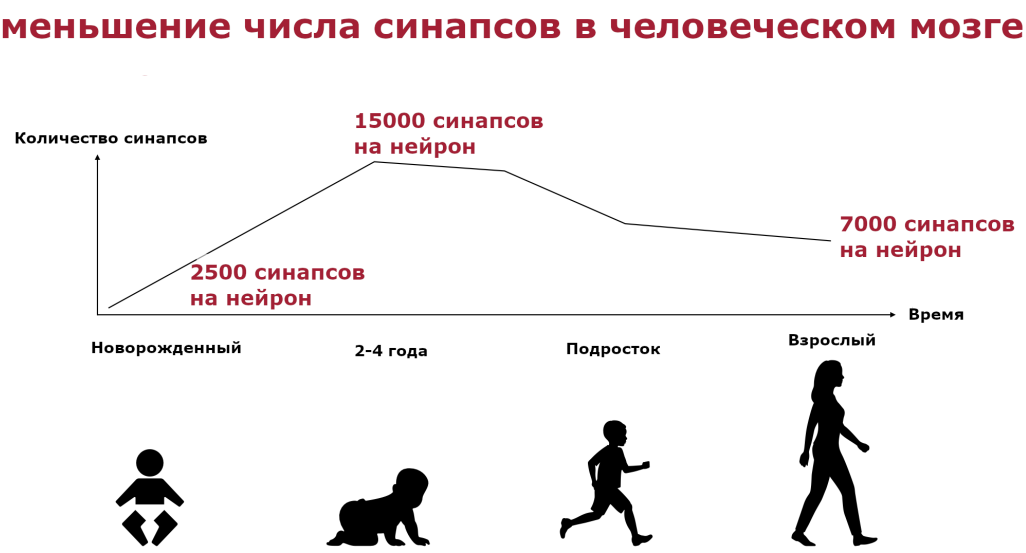


Рисунок 2.5 — Изменение среднего числа синапсов у человека во время его жизни. Рисунок заимствован из сети Интернет.

2.2.7 Структурная разреженность

В отличие от классических нейронных сетей, в мозге человека отсутствуют полносвязные слои. Каждый нейрон имеет ограниченное число соединений. Более того, во время жизни человека/животного среднее число синапсов приходящееся на один нейрон меняется (см. рис. 2.5), достигая своего пика в детском возрасте и затем снижается. Считается, что процесс перенастройки весов (удаления старых и создание новых) играет важнейшую роль в процессах обучения и в работе памяти [23].

Существует множество работ, по глубокому машинному обучению, в которых пытаются добавить разреженность в нейронные сети. Эксперименты показывают, что можно без потерь качества убрать 90 - 95 процентов связей сети. Однако применение на практике данных подходов ограничивается аппаратным обеспечением, которое спроектировано под операции с плотными матрицами. Разреженность топологии нейронной сети называется *структурной разреженностью (structural sparsity)*.

Аналогично активационной разреженности структурная разреженность поддерживается практически всеми процессорами, работающими с импульсными нейронными сетями (в силу особенностей их работы), к примеру Loihi [42],

TrueNorth [40], Tianjic [59] и Neuronflow [43]. Для структурной разреженности в классических нейросетях был предложен экспериментальный чип EIE [60].

2.2.8 Квантованность

Квантованием называется процесс отображения большого (обычно непрерывного) множества значений в меньшее по размеру конечное множество значений. В ряде работ по нейробиологии утверждается, что мозг человека представляет и обрабатывает информацию в дискретной/квантованной форме [62] [63]. В [64] авторы оценивают примерную емкость синапсов в 4.7 бита (в то время как в искусственных нейронных сетях обычно используется 16/32 бита). Дискретность мозга человека можно объяснить, как минимум, тем, что информация, хранящаяся в непрерывной форме, неизбежно будет испорчена шумом, присутствующим в любой физической системе [65]. Более того, с точки зрения байесовской структуры квантование приводит к стабильности представления информации и устойчивости к аддитивному шуму [66].

Большое число работ в области глубокого обучения пытаются заменить стандартные представления чисел в формате float point 32 [67], использующиеся в нейронных сетях, на более компактные. Квантованные веса нейронных сетей могут использоваться не только в режиме вывода сети, но и при её обучении.

Работу с квантованными нейронными сетями поддерживает большое количество аппаратных платформ для ИИ, начиная от чипов Nvidia с форматами fp16, bf16, int8 и int4 [18] и Google TPU [68] с форматом bf16, заканчивая нейроморфными чипами IBM NorthPole [44] и Neuronflow [43]. Также многие чипы для edge и мобильных применений поддерживают квантованные сети (а зачастую работают только с ними).

2.2.9 Аналоговость вычислений

Нейрон не является цифровым вычислительным устройством. Свою функцию он реализует с помощью работы ионных насосов и других аналоговых устройств.

На компьютере поведение биологических нейронов обычно моделируют системой дифференциальных уравнений, описывающих динамику мембранного потенциала. При отсутствии аналитического решения численное решение такой системы уравнений может оказаться очень затратным. Однако существуют и другие физические объекты, демонстрирующие аналогичную динамику (например, RC-цепь). Таким образом, биологический нейрон можно моделировать не только с помощью численного моделирования систем дифференциальных уравнений, но и с помощью подходящей аналоговой схемы, описываемой такими уравнениями. Аналоговые нейроны могут быть в 10 000 раз быстрее и энергоэффективнее [69], а также естественным образом поддерживать параллелизм. Принципиальным недостатком аналоговых нейронов является сложность их настройки и отладки, а также размер и отсутствие гибкости (после создания устройства его крайне тяжело перенастроить).

Другая область применения аналоговых схем — реализация синаптических операций. Например, классическая модель нейрона требует выполнения операций умножения и накопления (MAC), которые имеют вид: $sum = W_1 * X_1 + ... + W_n * X_n$. Её можно представить как комбинацию законов Ома и Кирхгофа: $= I_1 * R_1 + ... + I_n * R_n$, где ток I играет роль сигнала X, а сопротивление R выражает значение веса W. В такой схеме, все элементы операции умножения-накопления (MAC) выполняются параллельно за один шаг.

Аналоговые вычисления для ИИ поддерживают такие чипы, как BrainScaleS [69] и многочисленные мемристорные проекты [A.1]. Однако стоит отметить, что данные проекты не выходят за пределы исследовательских лабораторий и на данный момент пока не используются в индустрии.

2.2.10 Вычисления в памяти

Обычно, при эмуляции нейронной сети на ЭВМ одно ядро моделирует большое количество нейронов, последовательно переключая контекст между ними. Это создает значительные затраты времени и энергии на передачу значений контекста нейрона в память и обратно. В биологических нейронах подобные механизмы не наблюдаются.

Биологический нейрон реализует *принцип вычислений в памяти (in-memory computations)*. Он одновременно является устройством, хранящим свое состояние (память, представленная мембранным потенциалом и силой синаптических связей), и устройством, выполняющим вычисления [32]. Этот подход свободен от ограничений фон Неймана, основанных на физическом разделении общей памяти и процессоров, и реализует концепцию «один нейрон — один вычислитель», тогда как в цифровых устройствах один вычислительный блок обычно моделирует множество нейронов путем переключения контекста между ними.

Однако реализовывать принцип «один нейрон — один вычислитель» в цифровых схемах крайне расточительно, поскольку теряется возможность использования общего АЛУ для моделирования множества нейронов. В связи с этим используют гибридный подход. Данный подход заключается в использовании быстрой памяти, физически расположенной рядом с вычислительным ядром, моделирующем группу нейронов. Такой подход часто называют *вычислениями рядом с памятью (near-memory computations)*. Память статического произвольного доступа (SRAM), используемая для таких решений, дороже по сравнению с DRAM, и это ограничивает разработку микросхем на основе данного подхода.

В последние годы появилось большое количество чипов как для классических сетей, так и для импульсных, которые используют вычисления рядом с памятью. К ним можно отнести Neuronflow [43], IBM NorthPole [44], Intel Loihi [42], Cerebras, Groq и EIE [60]. Именно благодаря большому объему SRAM памяти они достигают высокой энергоэффективности и низких задержек. Особенно сильно это проявляется для задач, в которых размер пакета равен 1. Графики с замерами времени работы, задержек и затрат энергии для чипа NorthPole показан на рис. 2.6.

В таблице 2 перечислены некоторые вычислители с указанием типа вычислителя, объема их SRAM памяти и нормированной плотности SRAM памяти на

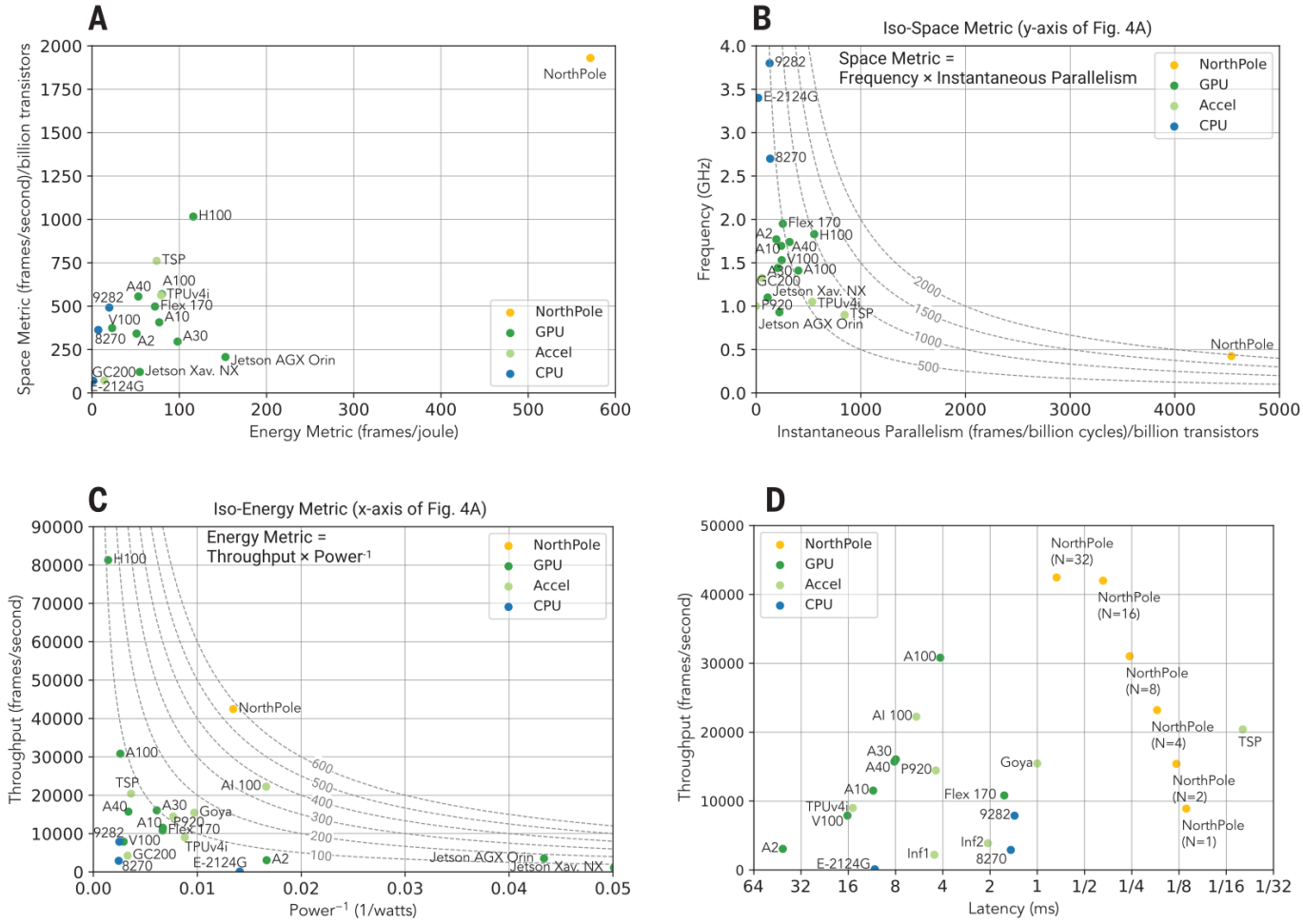


Рисунок 2.6 — Графики с замерами времени работы, задержек и затрат энергии для чипа NortPole. Рисунок взят из статьи [44].

.

чипе. Как можно легко увидеть все нейроморфные чипы имеют плотность SRAM памяти в разы выше, чем у CPU (кроме Xeon серверного класса, стоящего более 11 000 долларов), и на порядок выше, чем у GPU. Именно этот факт во многом объясняет энергоэффективность и меньшее время отклика нейроморфных чипов.

Чип	Тип	Объем SRAM	Площадь чипа	Техпроцесс	Плотность SRAM	Энергопотребление
Nvidia A100	GPU	60.25 Мб	826 мм^2	7 нм	3 574 кбайт/мм^2	400 ватт
IBM NorthPole	нейроморфный	224 Мб	790 мм^2	12 нм	40 830 кбайт/мм^2	74 ватт
Intel Loihi	нейроморфный	24.5 Мб	31 мм^2	7 нм	38 725 кбайт/мм^2	1 ватт
Cerebras WSE-3	нейроускоритель	44 Гб	46 225 мм^2	5 нм	23 796 кбайт/мм^2	23 000 ватт
AMD Ryzen7 5700	ЦПУ	16.5 Мб	180 мм^2	7 нм	4 491 кбайт/мм^2	65 ватт
Intel Core i5-14400	ЦПУ	21.25 Мб	215 мм^2	10 нм	9 883 кбайт / мм^2	65 ватт
Intel Xeon 8592	Серверный ЦПУ	320 Мб	2x763 мм^2	10 нм	20 969 кбайт/мм^2	350 ватт

Таблица 2 — Плотность SRAM памяти в различных чипах.

2.3 Выводы об использовании биологически подобных методов для систем ИИ

Рассмотрев некоторые принципы работы мозга человека, можно уже сделать выводы о том, какие принципы на текущем этапе развития электроники и алгоритмов ИИ будут полезны на практике, и реализации каких из них возможны в современных электронных устройствах.

Вычисления в памяти полностью решают проблему бутылочного горлышка фон Неймана, однако текущие принципы работы электроники не позволят эффективно использовать данный принцип. Это связано либо с негибкостью, как в случае с массивами мемристоров для выполнения MAC операций, либо с необходимостью создания большого количества вычислительных устройств, которые, несомненно, будут избыточны, поскольку частоты современных вычислителей составляют гигагерцы и позволяют эффективно разделять АЛУ между моделируемыми нейронами. В связи с этим использование компромиссного варианта в виде вычислений рядом с памятью на основе SRAM выглядит привлекательно и реализуется на большом количестве современных вычислителей, таких как Cerebras, NeuronFlow [43], Loihi [42], NorthPole [44], EIE [60] и др.

Тем не менее, расположение памяти на чипе, рядом с вычислителем, дорого. В связи с этим крайне желательно уменьшать размеры нейросетей. Для этого хорошо подходят вышеупомянутые принципы квантования и принцип структурной разреженности. Эти методы, как мы увидим далее, позволяют значительно сокращать размеры нейронных сетей (в 40-200 раз), практически без потери качества их работы. Такой подход благоприятно отражается на энергоэффективности, пропускной способности и времени отклика систем ИИ за счет уменьшения обмена данными между АЛУ и памятью.

Таким образом, объединение принципов обработки информации рядом с памятью и оптимизация размеров нейронных сетей дают синергетический эффект, который позволяет реализовывать энергоэффективные и быстрые системы ИИ, в частности для инференса нейронных сетей в задачах обучения с подкреплением. Интересным выглядит тот факт, что все эти принципы, как было показано выше, реализованы в мозге человека и отчасти объясняют его эффективность.

Многие упомянутые аппаратные решения как раз и используют этот синергетический эффекти и объединяют в себе реализацию нескольких из

вышеперечисленных подходов. К примеру NorthPole [44] от компании IBM, представленный в 2023 году, объединяет вычисления рядом с памятью (в SRAM) и фокусируется на поддержке квантованных нейронных сетей. Графики с замерами его времени работы, задержек и затрат энергии показаны на рис. 2.6.

Исследовательский чип EIE [60], созданный в 2016 году, объединяет в себе вычисления рядом с памятью, поддержку структурно и активационно разреженных вычислений и квантованные нейронные сети. Он показывал для задач вывода нейронных сетей при размере пакета равном 1 энергоэффективность в 24000 раз выше и быстродействие в 189 раз выше, чем CPU того времени и в 3400 раз выше энергоэффективность и в 13 раз выше быстродействие, чем GPU того времени. Это было достигнуто за счет комбинации быстрой памяти (120x к энергоэффективности), структурной разреженности (x10), квантования (x8) и активационной разреженности (x3).

Также, если возникает необходимость сокращения количества арифметических операций, то можно использовать принципы временно́й и активационной разреженности. Их применение вместе с вычислениями рядом с памятью было продемонстрировано в чипах NeuronFlow [43], Loihi [41] и EIE [60].

Глава 3. Нейроморфные методы оптимизации систем ИИ для задач обучения с подкреплением

Данная глава посвящена разработке нейроморфных методов оптимизации систем ИИ для задач обучения с подкреплением на основе предложенных в предыдущей главе принципов работы мозга человека. Как было выяснено в разделе 1.3 именно доступ к памяти является основной проблемой при инференсе в задачах RL. Её можно решать либо путем уменьшения числа обращений к памяти, либо путем расположения нейронной сети в быстрой памяти.

Хотя методы оптимизации сетей (в частности, на основе структурной разреженности и квантования) хорошо изучены в контексте классического глубокого обучения, их применение к обучению с подкреплением сопряжено со специфическими трудностями и нюансами. Обучение с подкреплением предполагает динамическое взаимодействие со средой и нестационарное распределение данных, обусловоенное влиянием изменяющейся политики агента на стратегию исследования среды и на процесс получения данных для обучения. Эти особенности предметной области затрудняют перенос классических методов оптимизации нейронных сетей в область обучения с подкреплением, поскольку они оказывают влияние на то, как разреженность влияет на процесс исследования, и как ошибки квантования изменяют распространение сигнала вознаграждения.

Далее будут описаны два алгоритма обучения с подкреплением и алгоритмы оптимизации нейронных сетей на основе структурной разреженности, временно́й разреженности и квантования, которые будут использоваться в методах, предложенных автором и описанных в последних двух разделах этой главы.

3.1 Алгоритмы глубокого обучения с подкреплением

3.1.1 DQN

Одним из наиболее значимых прорывов в мире искусственного интеллекта является публикация в 2015 году статьи [1], в которой была представлена си-

стема Deep Q-Network (DQN), комбинирующая глубокое машинное обучение с обучением с подкреплением, способная решать большую часть игр Atari [70] на уровне равном или выше человеческого. Идея метода заключается в использовании глубокой нейронной сети для аппроксимации Q-функции, которая оценивает ожидаемое будущее вознаграждение, получаемое агентом, выбравшим определенное действие в определенном состоянии. Замена Q-функции универсальным аппроксиматором в виде нейронных сетей является логичным развитием классических табличных Q-методов, которые раньше использовались для оценки Q-функции [20]. Именно нейронные сети позволяют применять данные методы в задачах, в которых число состояний становится крайне большим. К примеру, число потенциальных состояний в игре, отображаемой на мониторе 210x110 с тремя цветными каналами (типичное разрешение игр Atari) составляет $|S| = 2^{210\times110\times3\times8}$, что больше числа атомов во Вселенной. Помимо очевидной проблемы с обходом всех элементов подобной таблицы, также возникает проблема её хранения. Перейдем к рассмотрению алгоритма DQN.

Функция ошибки

Так как Q-функция является веществennозначной функцией, то обучение нейронной сети будет представлять задачу регрессии. Будем обозначать обучаемую сеть DQN как $Q_\theta(s,a)$. В качестве целевых значений возьмем значения оптимальной Q-функции $Q^*(s,a)$. Тогда функция ошибки для одного примера примет вид:

$$L(\theta) = (Q^*(s,a) - Q_\theta(s,a))^2 \tag{3.1}$$

Используя уравнение оптимальности Беллмана 1.6 и подставив его в первый член, получим:

$$L(\theta) = (r + \gamma \max_{a'} Q^*(s',a') - Q_\theta(s,a))^2 \tag{3.2}$$

Остается выяснить откуда будут браться значения оптимальной функции Беллмана Q^* под знаком max. Её заменит та же самая обучаемая функция Q_θ. В

итоге получим формулу, в которой и целевое и предсказываемое значение рассчитывается с помощью функции-аппроксиматора Q_θ:

$$L(\theta) = (r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a))^2 \tag{3.3}$$

Таким образом один шаг обучения будет принимать форму:

$$\theta_{t+1} = \theta_t - \alpha \frac{\delta L}{\delta \theta} \tag{3.4}$$

Буффер данных

Агент делает переходы из состояния s в следующее состояние s', выполняя действие a и получая награду r. Мы можем сохранить эту четверку $< s, a, r, s' >$, содержащую информацию о переходе, в *буфере опыта (буфер примеров, experience replay, replay buffer)*, который обычно обозначается буквой D. Эта информация о переходе, по сути, является опытом агента. Мы сохраняем опыт агента, полученный им в течении множества эпизодов. Во время обучения данные из буфера семплируются случайным образом. Полученные четвёрки образуют минипакет (minibatch), используемый на текущем шаге обучения сети. При этом, размер буффера ограничен, и старые / ненужные данные будут из него выбрасываться.

Целевая сеть

Для стабилизации процесса обучения используются отдельные *целевые сети (target networks)* для оценки целевого Q-значения. Обновление весов целевой сети происходит реже, чем обучение основной сети, что помогает предотвратить расхождение в процессе обучения. Таким образом вводится вторая сеть $Q_{\theta'}(s, a)$, и функция ошибки 3.3 принимает вид:

$$L(\theta) = (r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a))^2 \tag{3.5}$$

Алгоритм

Финальный алгоритм принимает следующий вид:

1. Инициализировать параметры θ основной сети случайными значениями.
2. Инициализировать параметры θ' целевой сети путем копирования весов основной сети θ.
3. Создать буфер опыта $\mathcal{D}$.
4. Для N эпизодов повторять шаг 5
5. Для каждого шага эпизода до окончания:
    7 Получить состояние s, выбрать случайное действие с вероятностью ε, и с вероятностью $1 - \varepsilon$ выбрать действие $a = \arg\max_a Q_\theta(s, a)$
    8 Выполнить выбранное действие a, получить награду r.
    9 Положить полученную четверку $< s,a,r,s' >$ в буфер опыта $\mathcal{D}$.
    10 Семплировать минибатч размером K из буфера опыта.
    11 Посчитать функцию ошибки: $L(\theta) = \frac{1}{K}\sum_{i=1}^{K}(r+\gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a))^2$
    12 Посчитать градиент параметров основной сети θ: $\theta_{t+1} = \theta_t - \alpha\frac{\delta L}{\delta\theta}$.
    13 Если значение целевой сети с параметрами θ' не обновлялось M шагов, то скопировать в параметры целевой сети θ' значения параметров основной сети θ.

3.1.2 SAC

Актор-Критик

Алгоритм SAC (Soft Actor-Critic) [71] — это метод обучения с подкреплением, относящийся к семейству алгоритмов *actor-critic (актор-критик)*. Суть данного класса алгоритмов заключается в параллельной работе двух компонентов: актора, который отвечает за выбор действий в данной среде, и критика,

оценивающего эффективность данных действий с точки зрения получаемой награды.

Идея SAC

Особенность SAC заключается в том, что алгоритм не просто стремится максимизировать суммарный выигрыш, как многие другие алгоритмы обучения с подкреплением, но и учитывает энтропию политики — меру случайности выбора действий. Благодаря этому, агенты, обученные с использованием SAC, достигают не только высокой производительности, но и устойчивости в разнообразных средах. SAC был впервые представлен в 2018 году группой исследователей под руководством Сергея Левина из UC Berkeley [71].

В SAC политика является стохастической и выражается нейронной сетью с параметрами φ и обозначается π_φ. Сам алгоритм состоит из трех компонентов: актор, группа для оценки Q-функции и группа для оценки V-функции. Ключевая идея SAC заключается в дополнительном использовании максимизации энтропии для обучения политик, которые не только максимизируют ожидаемую награду, но и поддерживают высокий уровень случайности в выборе действий. Последнее помогает лучше исследовать среду и находить более стабильные политики. Целевая функция оптимизации выглядит следующим образом:

$$J(\varphi) = E_{\tau\sim\pi_\varphi}\left[\sum_{t=0}^{T-1} r_t + \alpha H(\pi(\cdot|s_t))\right]$$

Для оценки V-функции используется нейронная сеть с параметрами ψ, а также таргетная к ней сеть ψ'. Для дальнейшего повышения стабильности алгоритма вместо одной Q сети используется две, из которых выбирается наименьшая оценка:

$$y_v = \min\left(Q_{\theta_1}(s,a), Q_{\theta_2}(s,a)\right) - \alpha \log \pi_\varphi(a|s) \tag{3.6}$$

Для оценки Q-функции используется нейронная сеть с параметрами θ. Для оценки целевых значений Q-функции будет применяться V-функция в соответствии с 3.7. При этом в формуле будет использоваться таргетная сеть φ':

$$y_q = r + \gamma V_{\varphi'}(s') \tag{3.7}$$

Алгоритм

Объединяя все вышеперечисленное алгоритм Soft Actor-Critic принимает следующий вид:

1. Инициализировать параметры ψ основной V-сети, параметры двух Q-сетей θ_1, θ_2 и параметры actor-сети φ случайными значениями.
2. Инициализировать параметры ψ' целевой V-сети путем копирования весов основной V-сети ψ.
3. Создать буфер опыта D.
4. Для N эпизодов повторять шаг 5
5. Выполнять для каждого шага эпизода до окончания:
    1 Выбрать действие a на основе политики $\pi_\varphi(S)$, то есть, $a = \pi_\varphi(s)$
    2 Выполнить выбранное действие a, перейти к следующему состоянию s', получить награду r, сохранить переход в буфере памяти $\mathcal{D}$
    3 Семплировать минибатч из K переходов из буфера памяти $\mathcal{D}$.
    4 Вычислить целевую оценку состояния (target state value):
    $$y_i = \min_{j=1,2} Q_{\theta_j}(s_i, a_i) - \alpha \log \pi_\varphi(a_i \mid s_i)$$
    5 Вычислить функцию ошибки для целевой сети V $J_V(\psi)$
    $$J_V(\psi) = \frac{1}{K} \sum_i (y_i - V_\psi(s_i))^2$$
    6 Обновить её параметры ψ с помощью градиентного спуска, $\psi = \psi - \nabla_\psi J_V(\psi)$
    7 Вычислить целевое значение Q (Q-value): y_{q_i}
    $$y_{q_i} = r_i + \gamma V'_\psi(s'_i)$$
    8 Вычислить ошибки для Q-сетей
    $$J_Q(\theta_j) = \frac{1}{K} \sum_i \left(y_{qi} - Q_{\theta_j}(s_i, a_i)\right)^2 \quad \text{for } j = 1, 2$$

9 Обновить значения параметров Q-сетей,

$$\theta_j = \theta_j - \lambda \nabla_{\theta_j} J(\theta_j) \quad \text{for } j = 1, 2$$

10 Вычислить градиенты целевой функции актора (actor objective function) $\nabla_\varphi J(\varphi)$ и обновить параметры, используя градиентные подъем $\varphi = \varphi + \lambda \nabla_\varphi J(\varphi)$

11 Обновить значения параметров таргетной V-сети

$$\psi' = \tau\psi + (1 - \tau)\psi'$$

3.2 Методы оптимизации нейронных сетей

На сегодняшний день существует острая потребность в методах оптимизации нейросетевых моделей для уменьшения их размера, ускорения их работы и снижения энергопотребления. Существует большое количество методов, для оптимизации нейронных сетей и уменьшение вычислительных затрат. В основном они могут быть отнесены к одной из следующих категорий: разработка вычислительно эффективных архитектур НС (NAS) [72], подрезание (обрезание, разреженность, pruning) [73], квантование [74], дистилляция [74]. Далее мы опишем методы оптимизации нейронных сетей с помощью квантования и структурной разреженности.

3.2.1 Квантование нейронных сетей

Квантованием называется процесс отображения большого (обычно непрерывного) множества значений в меньшее по размеру конечное множество значений.

Одним из применений квантования является *квантование нейронных сетей*. В нейронных сетях квантование может быть использована как для *квантования весов*, так и/или для *квантования активаций*.

Положительные эффекты данного процесса очевидны - уменьшение размера весов ведет к уменьшению переноса данных между памятью и вычислителем,

а иногда может привести к возможности умещать квантованную модель в более быстрой памяти [60]. Также квантование может быть полезно для упрощения вычислений путем использования целочисленной арифметики вместо арифметики с плавающей точкой.

Известно, что в мозге человека синапсы не хранят свои веса с разнообразием, присущим числам с плавающей точкой, а ограничиваются несколькими десятками значений. В связи с этим квантование нейронных сетей приводит нас к более биоподобным нейронным сетям [74].

Целочисленное линейное квантование

Одним из наиболее популярных методов оптимизации НС является квантование НС в целочисленную форму, самым распространенным вариантом которого является *целочисленное линейное квантование* [75], которое, в отличие от представления с плавающей точкой, имеет линейный шаг.

Квантование происходит по следующей формуле [74]:

$$q = Q(r) = round(\frac{r}{S}) + Z \tag{3.8}$$

где $Q : \mathbb{R} \to \mathbb{Z}$ - оператор квантования, $r \in \mathbb{R}$ - оригинальное значение в формате числа с плавающей точкой, $q \in \mathbb{Z}$ - квантованное целочисленное значение, $S \in \mathbb{R}$ - коэффициент масштабирования, $Z \in \mathbb{Z}$ - центральная точка (целочисленное), $round$ - функция округления.

Для определения значений S и Z используется формулы 3.9, 3.10. Они получаются из следующих соображений: определяются минимально и максимально возможные значения чисел с плавающей точкой из квантуемого множества: r_{min} и r_{max}. При этом необходимо, чтобы r_{min} отображался в q_{min} (минимальное целочисленное значение), ровно как r_{max} в q_{max} (максимальное целочисленное значение). Тогда:

$$S = \frac{r_{\text{max}} - r_{\text{min}}}{q_{\text{max}} - q_{\text{min}}} \tag{3.9}$$

$$Z = \text{round}\left(q_{\text{min}} - \frac{r_{\text{min}}}{S}\right) \tag{3.10}$$

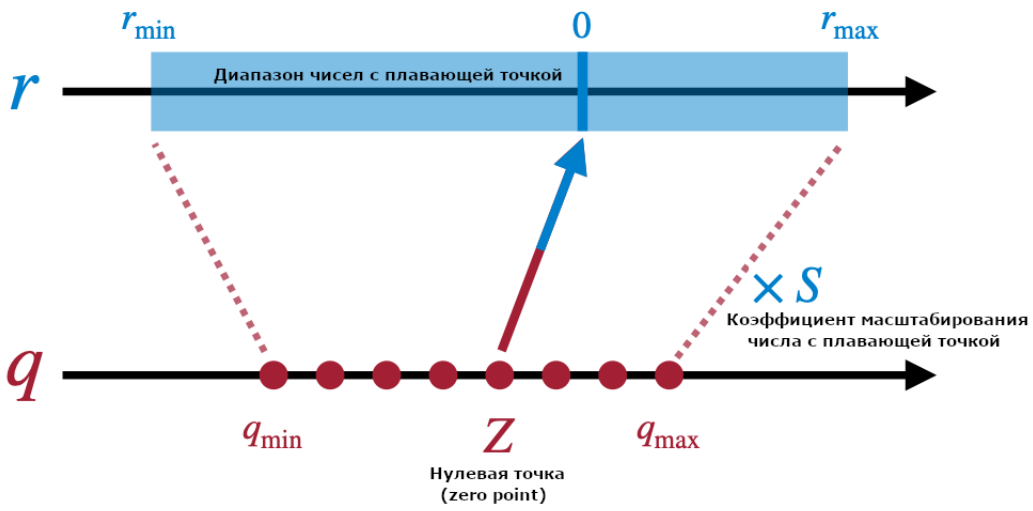


Рисунок 3.1 — Схема линейного квантования. На верхней оси показан диапазон значений квантуемого множества, состоящего их чисел с плавающей точкой. На нижней показан диапазон значений оператора квантования. По формулам 3.9, 3.10 находятся значения коэффициента масштабирования и сопоставление центральной точки. Рисунок взят из сети Интернет.

На изображении 3.1 показана схема линейного квантования.

Гранулярность квантования

Еще одним важным признаком является *гранулярность квантования (qunatization granurality)*. Для какой группы весов необходимо определять диапазоны значений и проводить квантование? Для всего *тензора (per-Tensor)*, для *каналов (per-Channel)* или для *групп (per-Group)*? Часто разные каналы в одном слое сети имеют разные диапазоны значений, и в случае квантования по всему тензору ошибка квантования будет выше. Как следствие, будут либо потеряны больши́е, но редкие веса, либо будет потеряна детализацию в малых весах. В целом, чем больше детализация используется, тем к меньшим потерям приводит квантование, но это вызывает накладные расходы. На сегодняшний день стандартом де-факто при квантовании сверточных слоев является квантование по каналам, а при квантовании полносвязных слоев чаще используется квантование по всему тензору.

Алгоритмы квантования

Существует два основных способа квантования предобученной модели: *квантование после обучения (Post Training Quantization, PTQ)* и *обучением с учетом квантования (Quantization Aware Training, QAT)*.

PTQ

Алгоритм квантования после обучение (PTQ) выполняет квантование и корректировку весов без какого-либо дообучения самой нейронной сети (finetuning) [74]. Благодаря этому, PTQ имеет низкие накладные расходы и может быть применен, когда доступна только небольшая часть данных для калибровки и даже, когда она не размечена. Однако, данный подход обычно показывает более низкие показатели по точности (особенно при квантовании с низкой битностью), чем алгоритмы типа QAT.

QAT

Процесс квантования (особенно агрессивный до 4 бит и менее) может вносить существенные изменения в тонко настроенные веса модели и уводить модель от точки, к которой она сошлась при обучении с плавающей точкой. Эту проблему можно решить путем повторного обучения/дообучения модели с использованием квантованных параметров. Благодаря этому, модель, обученная таким образом, имеет более высокое качество работы. Одним из подобных и наиболее популярных подходов является обучение с учетом квантования (QAT), при котором обычные проходы вперед (forward pass) и назад (backward pass) выполняются для квантованной модели с плавающей точкой, но параметры модели квантуются после каждого обновления градиента.

Квантование в задачах обучения с подкреплением

При квантовании нейронных сетей, тренированных с помощью методов обучения с подкреплением, помимо обычных проблем и сложностей, присущих квантованию нейронных сетей, появляются проблемы, связанные с тем, что в обучении с подкреплением отсутствует как таковая выборка обучающих данных, и получаемые данные, зависят от самого агента. Следовательно, квантуя нейрон-

ные связи в агенте (по факту изменяя их), мы рискуем потерять часть данных о поведении, что будет влиять на все последующие шаги.

В целом, квантованию RL сетей уделялось не очень много внимания. Существует всего несколько работ, которые исследуют данную область. Во-первых, работа по квантованию политик (policy quantization) [76]. В данной работе авторы успешно применили PTQ и QAT, используя стандартное однородное аффинное квантование, к политикам, обученным с использованием различных алгоритмов глубокого обучения с подкреплением, таких как A2C, DDPG, DQN, D4PG, PPO. Это позволило квантовать политики плоть до 6 бит с помощью алгоритма QAT. Также следует отметить работу [77], в которой смешанная 16 битная точность использовалась в процессе обучения и работы сети. Подытоживая, можно сказать, что данная область остается слабо изученной, несмотря на свой большой практический потенциал.

3.2.2 Структурная разреженность

Еще одним популярным способом оптимизации нейронных сетей является метод *прюнинга (прореживания, подрезания, обрезания, pruning)*, который приводит к структурной разреженности нейронных сетей. Первые работы по данному направлению были опубликованы в 90-х годах [78], в частности, в знаменитых статьях Optimal Brain Damage [79] и Optimal Brain Surgeon [80]. Как уже упоминалось в разделе 2.2.7 мозг человека также проявляет феномен перенастройки связей (новые синапсы могут появляться, а старые отмирать). Также ему присуща структурная разреженность – в мозге отсутствуют регулярные полносвязные слои. В связи с этим использование структурной разреженности приводит к более биоподобным нейронным сетям.

Следует также отметить, что в [81], [82] было убедительно эмпирические показано одно замечательное свойство разреженных сетей: оказывается, что разреженная сеть с тем же числом параметром, что и плотная сеть почти всегда будет превосходить по качеству работы плотную сеть.

Критерии прюнинга

Существуют разные подходы к выбору весов для удаления.

1. *На основе Гессианов*. Ранние работы 1990-х годов [79], [80] использовали метод обрезания весов на основе аппроксимации второго порядка по Тейлору для оценки увеличения функции потерь сети при обнулении того или иного веса.
2. *Обрезание по модулю (magnitude based pruning)* [83], [84], при котором удаляются веса наименьшие по модулю, является самым известным и чаще всего используемым на практике. Алгоритмы основанные на данном подходе, показывают высокие коэффициенты сжатия при минимальной потере точности.
3. *Подходы основанные на байесовской статистике и теории информации* [85], [86], [87], [88]. Эти методы также достигают высоких коэффициентов сжатия и имеют математическое теоретическое обоснование, однако редко применяются на практике.

Алгоритмы прюнинга

Существует несколько подходов к алгоритму прюнинга:

1. *Прюнинг предобученной модели*. Базовый вариант. Обычно приводит к серьезной деградации качества работы сети при уровне разреженности выше 50 процентов.
2. *Прюнинг предобученной модели и её донастройка*. Обычно, приводит к более высоким показателям качества работы в сравнении с первым методом.
3. *Итеративный прюнинг*. Основная идея этой группы методов является итеративное повторение процесса удаления части весов и донастройки получающихся моделей. Благодаря не разовому удалению большой части весов, «урон», наносимый сети остается приемлемым и сеть способна восстановить качество своей работы.

4. *Градуальный прюнинг*. Идея [81] заключается в постепенном, плавном обрезании сети каждые Δt шагов во время обучения в соответствии с некоторым расписанием (обычно нелинейным). В отличие от итеративного прюнинга градуальный прюнинг уделяет большое внимание плавности перехода между уровнями разреженности сети.
5. *Обучение заранее обрезанной модели с сохранением паттерна разреженности (one-shot pruning)*. В работах [89], [90], [91] производится обрезание сети перед обучением на шаге 0. Полученная разреженная сеть используется в качестве инициализации и паттерн разреженности сохраняется на протяжении всего обучении. Несмотря на превосходство над случайным обрезанием на шаге 0, эти методы сильно уступают даже обрезанию после обучения без дообучения. Особенно сильно это проявляется на больших датасетах, таких как ImageNet [92].
6. *Обучение заранее обрезанной модели с возможностью удалять и добавлять веса (rewiring)*. В данном подходе [93—95] обучение также начинается с разреженной сети, уровень разреженности которой мы поддерживаем во время обучения. Однако мы не фиксируем на все время обучения один паттерн разреженности. Сеть имеет возможность перенастраивать свои соединения, удаляя ненужные и создавая новые.

Распределение обрезанных весов в сети

Другим важным аспектом алгоритмов обрезания сети является распределение обрезаемых весов по сети. Эмпирически, было показано, что обрезание первых слоев сверточных сетей ведет к более значительным потерям в качестве работы, чем последующих. В самом деле, обычно на первых слоях число весов невелико и многие выученные паттерны крайне важны.

Существуют следующие подходы к распределению обрезаемых весов:

– *Глобальный (global)*. В данном подходе все веса сети рассматриваются в совокупности и веса для обрезания выбираются из всей совокупности весов модели.
– *Локально равномерный (Local uniform)*. В данном подходе на каждом слое обрезается одинаковая доля весов.

– *Локальный Эрдеш-Реньи (Local Erdos-Renyi)* [82; 94; 95].
  В данном случае вводится неравномерное распределение обрезаемых весов по слоям в соответствии с формулой:
  $s^l = \varepsilon * \frac{n^l + n^{l+1}}{n^l * n^{l+1}}$ - для полносвязных слоев, где s^l доля остающихся весов на слое l, n^l размерность слоя l, ε коэффициент, контролирующий глобальный уровень разреженности.
  $s^l = \varepsilon * \frac{n^l + n^{l+1} + w^l + h^l}{n^l * n^{l+1} * w^l * h^l}$ - для сверточных слоев, где s^l доля остающихся весов на слое l, n^l число каналов в слое l, w^l - ширина сверточного ядра, h^l высота сверточного ядра, ε коэффициент, контролирующий глобальный уровень разреженности.
  Основная идея заключается в том, что у первых и последних слоев сети, у которых размерности входов/выходов n^l будут меньше, будет обрезаться меньшая доля весов. В работах [94], [95], [82] было показано, что ERK приводит к слегка более высоким показателям качества при обрезании весов в сравнении с равномерным распределением.

Гипотеза Лотерейного билета

Гипотеза лотерейного билета (Lottery Ticket Hypothesis, LTH) была предложена Джонатаном Франклом и Майклом Карабином в 2018 году [96]. Суть этой гипотезы заключается в предположении, что внутри обширных нейронных сетей существуют значительно меньшие подсети, которые могут быть обучены с той же скоростью, что и оригинальная большая сеть, и достигать сопоставимой (или даже лучшей) производительности. Эти подсети были названы “лотерейными билетами“. То есть, гипотеза утверждает, что можно заранее выбрать довольно маленькую подсеть исходной сети и обучать её до сопоставимой, а иногда и более высокой точности, по сравнению с оригинальною точностью плотной сети.

Авторы предложили алгоритм поиска подобных сетей, который заключается в повторяющемся прореживании сети и возвращению оставшихся значений в исходные значения, как показано на рис. 3.2.

В работах [97; 98] был показан данный эффект и для RL алгоритмов.

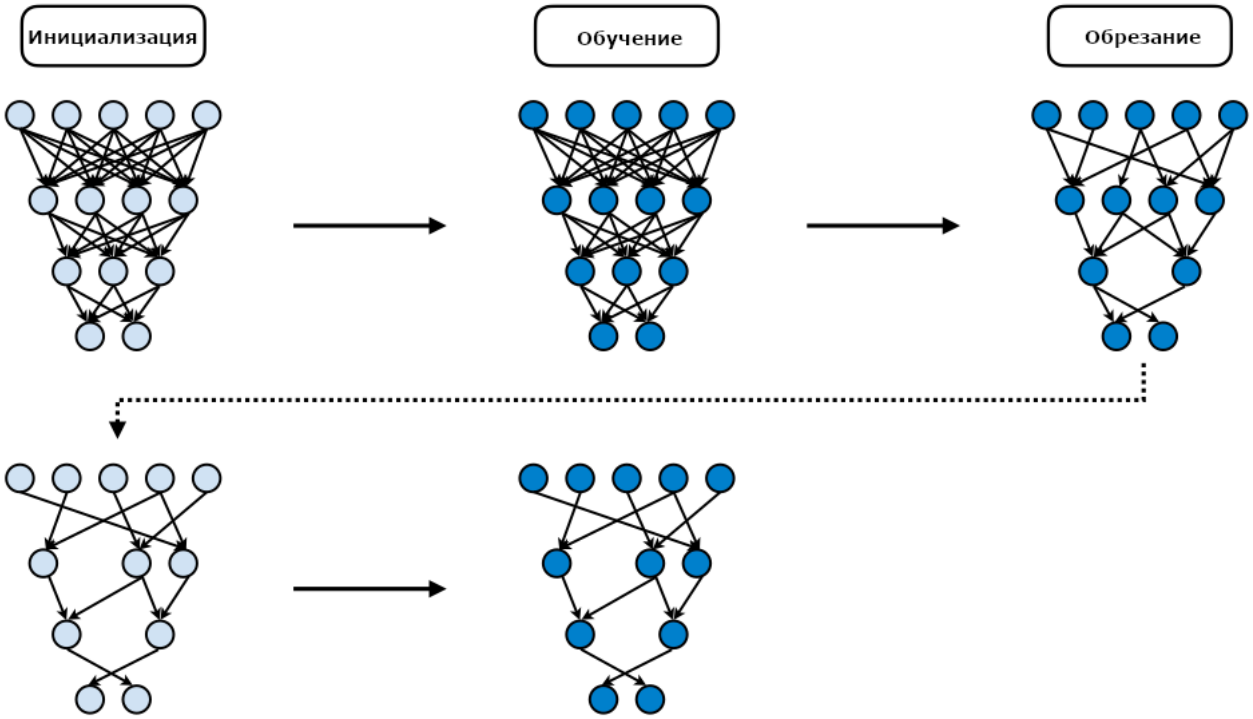


Рисунок 3.2 — Алгоритм Лотерейного билета. Рисунок заимствован из сети Интернет.

Структурная разреженность в задачах обучения с подкреплением

Обрезание нейронных сетей, обученных с помощью методов обучения с подкреплением, является отдельным поднаправлением исследований. Помимо обычных проблем и сложностей, присущих обрезанию нейронных сетей, здесь появляются проблемы, связанные с тем, что в обучении с подкреплением отсутствует как таковая выборка обучающих данных, и получаемые данные, зависят от самого агента. Следовательно, обрезая нейронные связи в агенте мы рискуем потерять часть данных о поведении, что будет влиять на все последующие шаги.

В целом, разреженным RL сетям уделялось не очень много внимания, вероятно, из-за убеждения, что чрезмерная параметризация сети помогает в обучении. Однако в работе [99] авторы предполагают, что на самом деле агенты RL, наоборот, могут страдать от неявной недостаточной параметризации, и выразительность сети на самом деле используется недостаточно. Кроме того, в [100] предполагают, что агенты глубокого RL могут иметь тенденцию к переобучению на данных, получаемых на ранних этапах обучения. Учитывая это, можно было бы ожидать, что существует значительная возможность сжатия агентов RL. Кроме

того, разреженные RL сети могут снижать стоимость обучения или помогать использовать их в условиях, где требуется очень высокая пропускная способность или малое время отклика, к примеру, в управлении токамаком [22] и высокоманевренными дронами [21].

Одной из первых работ в этом направлении являлась работа [101], которая использовала итеративный прюнинг сети [83] и дистилляцию. Исследование гипотезы лотерейных билетов в глубоком обучении с подкреплением было проведено в [97], а затем в [98], которые продемонстрировали, что гипотеза лотерейного билета также верна и для задач глубокого обучения с подкреплением. В [82] было показано, что аналогично результатам из других областей глубокого машинного обучения [95], разреженные сети в RL домене также будут иметь почти всегда более высокое вознаграждение, чем плотные сети с тем же числом параметров. Также было показано, что почти всегда можно без потери качества работы обрезать 80-95 процентов весов сети.

3.2.3 Временна́я разреженность

Как уже обсуждалось в гл. 2.2.6 природа может использовать явление временно́й разреженности. То что мы видели мгновенье назад скорей всего очень похоже на то, что мы видим сейчас. Эти идеи используются, к примеру, при сжатии видео в алгоритмах на основе компенсации движения [102]. Попробуем внедрить их в нейронную сеть.

Выход $k+1$ слоя нейронной сети может быть записан следующим образом:

$$o^{k+1} = W^k x^k + b^k \tag{3.11}$$

$$x^{k+1} = f(o^{k+1}) \tag{3.12}$$

где $x^{k+1} \in R^n$ является выходом $(k+1)$-го слоя нейронной сети, $x^k \in R^n$ представляет собой вход $(k+1)$-го слоя нейронной сети (выход k-го слоя), $W^k \in R^{nxn}$ обозначает матрицу весов, а $b^k \in R^n$ является вектором смещения. В традиционной нейронной сети для каждого нового входного вектора $x^k(t)$ в момент

времени t требуется полный перерасчёт выходного значения $x^{k+1}(t)$, что потребует n^2 умножений. Однако следует отметить следующее:

$$\Delta x^k(t) = x^k(t) - x^k(t-1) \tag{3.13}$$

$$o^{k+1}(t) = W^k \Delta x^k(t) + o^{k+1}(t-1) \tag{3.14}$$

$$\Delta x^{k+1}(t) = f(o^{k+1}(t)) - f(o^{k+1}(t-1)) \tag{3.15}$$

Таким образом, возможно пересчитать выходные значения слоя в момент времени t с использованием уравнений 3.13, 3.14, 3.15, опираясь на изменения входных данных слоя, произошедшие относительно состояния в момент $t-1$.

Это замечание само по себе не приводит к оптимизации нейронной сети. Однако можно ввести порог D для изменений выходного значения $\Delta x^k(t)$ так, что перерасчёт последующих нейронов начинается только тогда, когда выходное значение превышает этот порог.

Авторы [61; 103] называют этот подход *«Гистерезисным квантованием» (Hysteresis Quantizer)*. Для его реализации необходимо ввести дополнительные переменные $x\_prev(t)$ и o в каждый нейрон. $x\_prev(t)$ будет использоваться для записи последнего переданного значения, а o - для сохранения текущего состояния нейрона, в которое будут накапливаться входящие сигналы. Такой алгоритм будет выполняться на каждом нейроне. Будем называть данный метод оптимизации *дельта-алгоритмом*, а оптимизированные нейроны - *дельта-нейронами*.

На рис. 3.3 показана схематичная визуализация работы данной сети. На ней видно, что при использовании временно́й разреженности большая часть нейронов молчит и только небольшая часть нейронов активна и распространяет сигнал от себя последующим нейронам, причем не все из нейронов, принимающих сигнал, в дальнейшем активируются. Аналогичная идея была применена для рекуррентных сетей в работе [104].

Данный подход приводит к существенному уменьшению объема арифметических операций и обращений к памяти (для полносвязных слоев) при работе с последовательными данными, имеющими высокую корреляцию в соседние моменты времени. Также благодаря переходу к небинарной импульсной форме передачи сигнала и наличию памяти в нейронах появляется возможность уйти

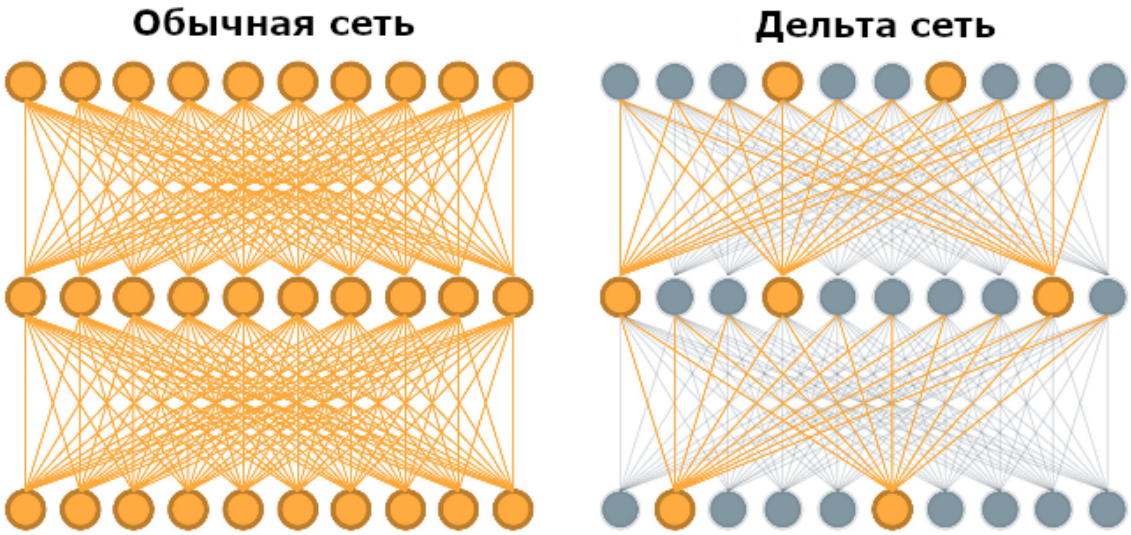


Рисунок 3.3 — Сравнение принципов работы обычной сети (слева) и сети, к которой применили дельта-алгоритм (справа). На правом рисунке видно, что только часть нейронов активируется и передает свой сигнал дальше. При этом не все принимаюшие нейроны активируются. Рисунок взят из сети Интернет.

от необходимости жесткой синхронизации вычислений, присущих обычным нейронным сетям, и перейти к асинхронной работе при условии гарантированной доставки сигнала. Стоит отдельно отметить, что применение данного алгоритма не требует заново переобучать обученную сеть. Наоборот, данный алгоритм может быть имплементирован в любую нейронную сеть, работающую с данными, имеющими временную зависимость.

Временна́я разреженность в задачах обучения с подкреплением

Автором не было обнаружено попыток применения временно́й разреженности в задачах обучения с подкреплением. Наиболее близкой работой является применение временно́й разреженности для вывода нейронной сети, используемой для управления автомобилем PilotNet [105]. Данная работа была выполнена авторами, предложившими гистерезисное квантование [103], [61], [43], которое было описано выше. Эффективность применения временно́й разреженности растет с увеличением частоты фреймов (FPS), что убедительно показано на рис. 3.4.

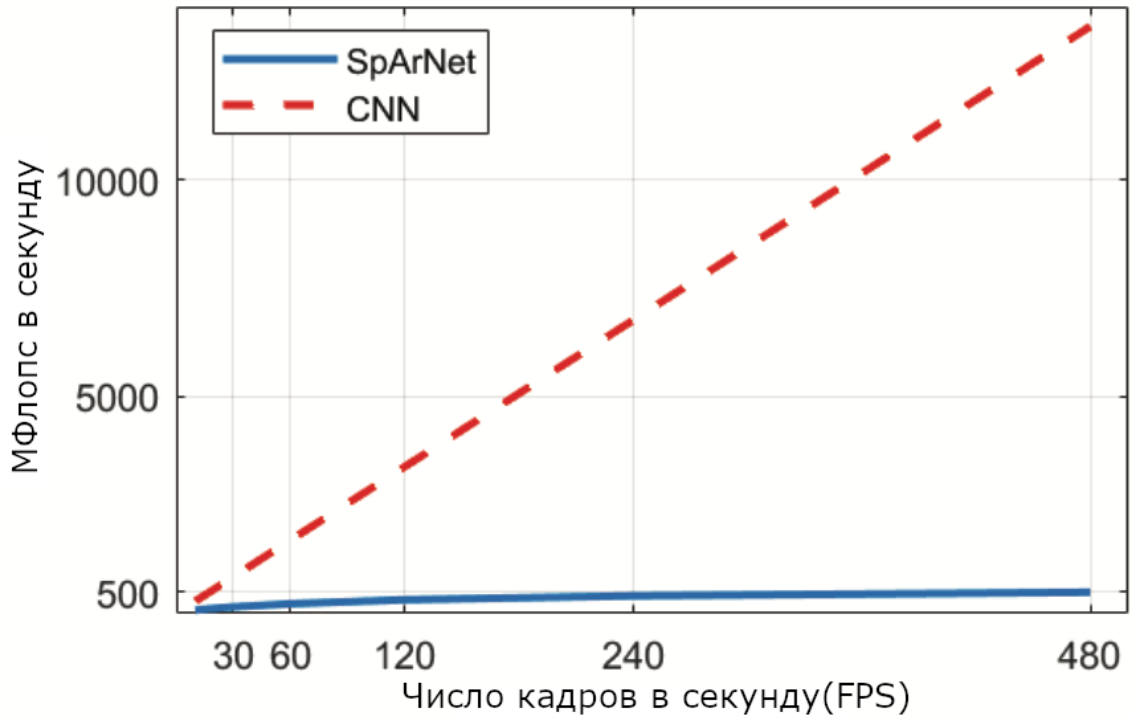


Рисунок 3.4 — Среднее число операций в секунду для инференса сети PilotNet в зависимости от частоты фреймов. Видно, что число операций растет линейно для обычной сети, и сублинейно с насыщением для сети с поддержкой временно́й разреженности. Взято из [43].

При этом, обычная сеть при увеличении частоты работы показывает линейный рост числа операций.

Автор данной диссертации применил данную методику совместно со структурной разреженностью для задач обучения с подкреплением. Методика и результаты описаны в разделе 3.4.

Недостатки дельта-алгоритма

При работе со сверточными нейронными сетями, дельта-алгоритм не всегда может показать свои сильные стороны. Действительно, для каждого элемента (пикселя) входного тензора сверточного слоя необходимо ввести отдельную ячейку памяти. Однако, может оказаться, особенно на первых слоях, что число таких ячеек памяти значительно превосходит число самих весов на этом слое. И несо-

мненно, в теории он будет уменьшать число умножений, однако необходимость расположения в памяти большего числа элементов лишает этот подход всякого смысла. В связи с этим, этот алгоритм может себя показывать с лучшей стороны на полносвязных слоях/на глубоких слоях сверточных сетей.

Перейдем теперь к описанию методов, предложенных автором.

3.3 Алгоритм оптимизации инференса нейронных сетей на основе комбинации структурной разреженности и квантования для задач обучения с подкреплением

Целью данного алгоритма является оптимизация обращений к памяти при инференсе нейронных сетей, тренированных методами RL. Оптимизация выполняется путем уменьшения обращений к памяти и/или расположения нейронных сетей в быстрой памяти. Для достижения этой цели предлагается сократить размеры нейронной сети с помощью структурной разреженности и квантования. Стоит отметить, что на текущий момент количество работ, применяющих квантование или разреженность к обучению с подкреплением, крайне ограничено [76], [82]. Исследования, посвященные комбинированию данных подходов для задач обучения с подкреплением, насколько известно автору, раньше не проводились и были впервые им проведены и описаны в статье [A.2].

Ключевым вопросом при применении структурной разреженности для оптимизации инференса нейронных сетей в задачах RL является определение оптимального момента выполнения прунинга (обрезания) нейронной сети в процессе обучения. В отличие от классических задач машинного обучения, где модель обучается на статическом наборе данных, в RL среда и данные для обучения формируются динамически в процессе взаимодействия агента с окружением. Это создает дополнительные сложности при внедрении методов компрессии.

Для достижения эффективной структурной разреженности в предложенном алгоритме используется подход на основе градуального прюнинга (см. гл. 3.2.2), который позволяет постепенно адаптировать структуру сети в процессе обучения, минимизируя негативное влияние на способность агента оптимально взаимодействовать со средой.

3.3.1 Описание алгоритма

Предлагаемый алгоритм оптимизации (см. рис. 3.5, 3.6) состоит из двух частей: обучения с одновременным прореживанием нейронной сети и последующего квантования сети. Полученный алгоритм позволяет значительно уменьшить число обращений к памяти и/или разместить нейронную сеть в быстрой памяти благодаря сжатию модели в десятки, а иногда в сотни раз при сохранении качества работы.

Обрезание сети будет выполняться в процессе обучения агента на основе модуля значения параметров 3.2.2 с помощью градуального прюнинга на шагах обучения, лежащих в интервале $[t_s, t_f]$, где $t_s = 0.2 \cdot T$ и $t_f = 0.8 \cdot T$, а T - число шагов обучения, в соответствии с расписанием 3.16:

$$s_t = s_f * \left(1 - \left(1 - \frac{t - t_0}{n\Delta t}\right)^3\right) for\ t \in \{t_0, t_0 + \Delta t, ..., t_0 + n\Delta t\} \tag{3.16}$$

где t - текущий момент времени, Δt - частота обрезания весов, n - общее число итераций обрезания (обычно несколько сотен), s_0 - начальный уровень разреженности, s_f - конечный (целевой) уровень разреженности.

Во время работы данного алгоритма разреженность постепенно увеличивается с s_0 до целевого значения s_f на шагах $[t_s, t_f]$, причем количество обрезаемых весов на каждом шаге со временем плавно уменьшается. График разреженности алгоритма представлен на рис. 3.7. Интуитивно смысл этого уменьшения можно объяснить тем, что в начальной фазе обучения можно более агрессивно обрезать веса, так как имеется их избыток (из-за сверхпараметризованности модели) и желательно постепенно снижать число обрезаемых весов с уменьшением числа оставшихся весов [81]. Несмотря на простоту данного метода он превосходит или равен по качеству многим более сложным подходам [73], [82].

После завершения основного цикла обучения с прореживанием сети, в течение еще $0.2 \cdot T$ шагов выполняется процедура квантования параметров сети с донастройкой с помощью алгоритма QAT (см. гл. 3.2.1). Для сверточных слоев будет выполняться асимметричное поканальное квантование, что позволяет сохранить точность свертки при значительном сокращении битности представления весов. Для полносвязных слоев используется асимметричное тензорное квантование, эффективно сжимающее веса с учетом их статистических характеристик.

Псевдокод предложенного алгоритма оптимизации нейронных сетей представлен ниже:

Алгоритм 1 Алгоритм оптимизации инференса нейронных сетей на основе комбинации структурной разреженности и квантования для задач обучения с подкреплением.

Входные данные:

$f_\theta(x)$ – необученная сеть с 32-битными параметрами

s_{final} – целевое значение разреженности

T – число шагов обучения

n – число итераций обрезания

L – частота измерения качества

Выходные данные:

$f_{\theta''}(x)$ – обученная квантованная сеть, M – маска весов

1: // *Этап 1: Обрезание*
2: $M \leftarrow \mathbf{1}(|\theta|)$ ▷ Инициализация маски, $|\theta|$ - количество параметров
3: **for** $t = 1$ to $T \cdot 0.2$ **do** ▷ Начальное обучение
4: train_RL_step(f_θ)
5: **end for**
6: **for** $i = 1$ to n **do** ▷ Фаза обрезания
7: $t \leftarrow t_{start} + i \cdot \Delta t$
8: $s_t \leftarrow s_{final} \cdot (1 - (1 - \frac{t - t_{start}}{n \cdot \Delta t})^3)$
9: **active_weights** $\leftarrow \theta[M \neq 0]$ ▷ Необрезанные веса
10: $k \leftarrow |\theta| \cdot (s_t - |\textbf{active\_weights}|/|\theta|)$ ▷ k - количество обрезаемых весов
11: threshold $\leftarrow$ kabsmin(**active_weights**, k) ▷ Поиск k-го по модулю веса
12: $M[|\theta| \leqslant \text{threshold}] \leftarrow 0$
13: **for** step = 1 to Δt **do**
14: train_RL_step($f_\theta \odot M$) ▷ $\odot$ - поэлементное умножение
15: **end for**
16: **end for**

Рисунок 3.5 — Псевдокод алгоритма оптимизации инференса нейронных сетей на основе комбинации структурной разреженности и квантования для задач обучения с подкреплением. Часть 1.

```
θ_best ← θ
R_best ← −∞
for t = t_final to T do                ▷ Финальное обучение и поиск наилучшей сети
    train_RL_step(f_θ ⊙ M)
    if t mod L = 0 then
        R_current ← evaluate_performance(f_θ ⊙ M)
        if R_current > R_best then
            R_best ← R_current
            θ_best ← θ
        end if
    end if
end for
// Этап 2: Квантование
θ_best_quantized ← θ
R_best_quantized ← −∞
for t = t_final to T do        ▷ Поиск наилучшей квантованной обрезанной сети
    train_RL_step_with_QAT(f_θ ⊙ M)
    if t mod L = 0 then
        R_current ← evaluate_performance(f_θ ⊙ M)
        if R_current > R_best_quantized then
            R_best_quantized ← R_current
            θ_best_quantized ← θ
        end if
    end if
end for
θ_result ← θ_best_quantized
return f_θ_result, M
```

Рисунок 3.6 — Псевдокод алгоритма оптимизации инференса нейронных сетей на основе комбинации структурной разреженности и квантования для задач обучения с подкреплением. Часть 2.

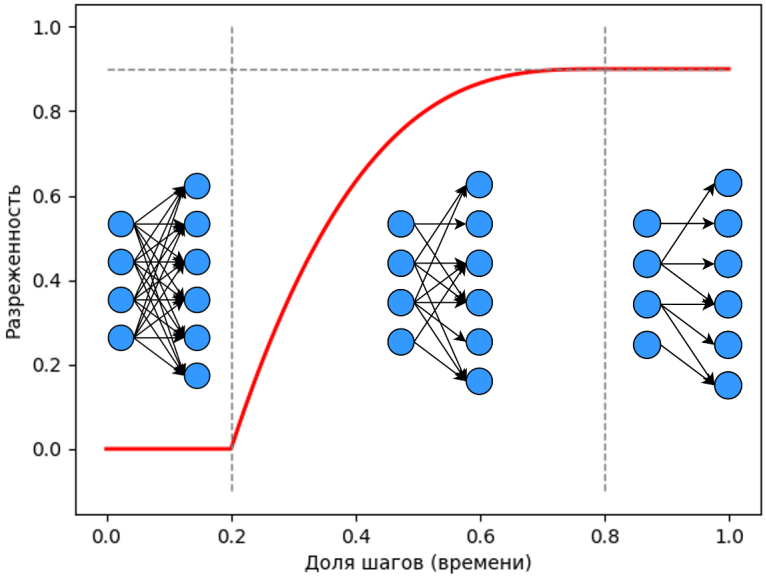


Рисунок 3.7 — График разреженности алгоритма. Рисунок автора.

3.3.2 Свойства алгоритма

1. Алгоритм совмещает структурную разреженность и квантование для задач обучения с подкреплением.
2. Алгоритм уменьшает число обращений в память за счет удаления весов и их сжатия, что приводит к увеличению пропускной способности системы ИИ, снижению времени отклика и понижению энергопотребления. Благодаря сжатия многие НС, оптимизированные алгоритмом, могут быть расположены в быстрой памяти.
3. Появляется возможность использовать целочисленную арифметику, что положительно влияет на производительность.
4. Алгоритм позволяет сжимать сети практически без потери качества их работы.
5. Метод может быть применен к любой нейронной сети из класса полносвязных и сверточных сетей.
6. Алгоритм может быть применен к большинству современных алгоритмов обучения с подкреплением.

3.4 Алгоритм оптимизации инференса нейронных сетей на основе комбинации структурной и временно́й разреженностей для задач обучения с подкреплением

В предыдущих разделах мы описали два нейроморфных подхода к оптимизации нейронных сетей: структурную разреженность (см. гл. 2.2.7) и временну́ю разреженность (см. гл. 2.2.6). Возникает логичный вопрос о возможности их объединения и применения для оптимизации инференса систем ИИ для задач обучения с подкреплением. Данный вопрос был подробно исследован автором в работе [А.3].

Целью данного алгоритма является уменьшение числа обращений к памяти и сокращение объема вычислений при инференсе НС, тренированных методами RL. Для этого предлагается обрабатывать только измененные области изображения и активировать только необходимые нейроны, а также сжать размер сети с помощью структурной разреженности.

Так как задачи RL обычно имеют высокую корреляцию между входными данными в соседние моменты времени (то есть обладают свойством временно́й разреженности), то применение временно́й разреженности путем обработки только измененных областей изображения и активации только необходимых нейронов является логичным шагом. Для получения структурной разреженности используется метод на основе Lottery Ticket Hypothesis (см. гл. 3.2.2).

3.4.1 Описание алгоритма

Алгоритм оптимизации состоит из двух частей: обучения разреженной нейронной сети на основе гипотезы лотерейного билета (см. гл. 3.2.2) и применения при выводе нейронной сети дельта-алгоритма (см. рис. 3.9) [61; 103]. Полученный алгоритм позволяет уменьшить число обращений к памяти и значительно снизить число *значимых операций умножения (операции умножения, в которых один из операндов не ноль)* при инференсе нейронной сети.

Этап 1. Получение обученной структурно разреженной нейронной сети с помощью алгоритма LTH - Lottery Ticker Hypothesis

На первом этапе (см. рис. 3.8), используя алгоритм Лотерейного Билета 3.2.2, мы получим набор структурно разреженных нейронных сетей с различной степенью разреженности. Количество нейронных сетей в наборе соответствует количеству итераций алгоритма обрезания.

Алгоритм 2 Этап 1: Структурная оптимизация (LTH)

Входные данные:

$f_\theta(x)$ – необученная нейронная сеть с параметрами θ

n – число итераций обрезания сети

p – доля удаляемых весов на итерации

T – число шагов обучения на итерации

D – пороговое значение для дельта-алгоритма

Выходные данные:

Кортеж $\langle (f_{\theta_1}(x), M_1), (f_{\theta_2}(x), M_2), \ldots, (f_{\theta_n}(x), M_n) \rangle$, где $M_i \in \{0,1\}^{|\theta|}$

$M \leftarrow \mathbf{1}^{|\theta|}$ ▷ Инициализация маски единицами

for $i \leftarrow 1$ to n **do**

  $\theta_i \leftarrow \theta$ ▷ Копирование исходных весов

  $f_{\theta_i}(x) \leftarrow \text{RL\_train}(f_{\theta_i}(x), T, M)$ ▷ Обучение с учетом маски

  $k \leftarrow p \cdot |\theta|$ ▷ Число весов для обрезания

  active_weights $\leftarrow \theta[M \neq 0]$ ▷ Необрезанные веса

  threshold $\leftarrow$ kabsmin(**active_weights**, k) ▷ Поиск k-го минимального по модулю необрезанного веса

  $M[|\theta| \leqslant \text{threshold}] \leftarrow 0$ ▷ Удаление k наименее значимых весов

  $M_i \leftarrow M$ ▷ Сохранение текущей маски

end for

return $\langle (f_{\theta_1}(x), M_1), (f_{\theta_2}(x), M_2), \ldots, (f_{\theta_n}(x), M_n) \rangle$ ▷ Кортеж обрезанных сетей

Рисунок 3.8 — Псевдокод алгоритма оптимизации инференса нейронных сетей на основе комбинации структурной и временно́й разреженностей для задач обучения с подкреплением. Этап 1. Получение структурной разреженности.

Этап 2. Применение к обученной сети алгоритма разреженности по времени (дельта-алгоритм)

На этапе вывода обученных и обрезанных нейронных сетей для дальнейшей оптимизации применяется дельта-алгоритм (см. гл. 3.2.3) с пороговым значением $D = 0.01$. Стоит отдельно отметить, что применение дельта-алгоритма не требует повторного обучения уже обученной сети. Напротив, данный алгоритм может быть имплементирован в любую нейронную сеть, работающую с данными, имеющими временну́ю зависимость.

В результате мы получаем набор новых нейронных сетей, использующих как структурную, так и временну́ю разреженность. Выбор конкретной нейронной сети из такого набора зависит от желаемого баланса между количеством значимых умножений и производительностью нейронной сети. Таким образом алгоритм 3 (см. рис. 3.9) будет исполняться для каждого нейрона.

Алгоритм 3 Этап 2: Применение дельта-алгоритма для получения временно́й разреженности

1: // *Выполняем алгоритм для каждого нейрона в сети*
2: **while** True **do**
3: $x_i^k(t) = Wait(predecessors)$ ▷ Получаем сигнал от предшественника i с предыдущего слоя k
4: $o_j^{k+1}(t) \leftarrow o_j^{k+1}(t-1) + W_{ij} \times \Delta x_i^k(t)$ ▷ Пересчет внутреннего состояния нейрона
5: $\Delta x_j^{k+1}(t) \leftarrow f(o_j^{k+1}(t)) - x\_prev_j^{k+1}(t)$ ▷ Пересчет изменения активации нейрона
6: **if** $|\Delta x_j^{k+1}(t)| \geqslant D$ **then** ▷ Проверка превышения активации порога
7: $x\_prev_j^{k+1}(t) = f(o_j(t))$ ▷ Обновление значения активации
8: $Send(\Delta x_j^{k+1}(t), successors)$ ▷ Посылка изменения активации $\Delta x_j^{k+1}(t)$ из нейрона последующим нейронам
9: **end if**
10: **end while**

Рисунок 3.9 — Псевдокод алгоритма оптимизации инференса нейронных сетей на основе комбинации структурной и временно́й разреженностей для задач обучения с подкреплением. Этап 2. Получение временно́й разреженности.

Обозначения:

- $\Delta x_i^k(t)$ – изменение выходного значения i-го нейрона на слое k в момент времени t
- D – пороговое значение для активации пересчета

3.4.2 Свойства алгоритма

1. Алгоритм совмещает структурную оптимизацию на основе гипотезы лотерейного билета с временно́й оптимизацией на основе дельта-алгоритма.

2. Алгоритм уменьшает число обращений в память за счет удаления весов и редкой активации нейронов (нет необходимости получать выходные веса значительной части нейронов), что приводит к увеличению пропускной способности системы ИИ, снижению времени отклика и понижению энергопотребления.
3. Уменьшение размеров нейронной сети делает потенциально возможным размещение в быстрой SRAM памяти.
4. Введение дельта-алгоритма позволяет использовать небинарные импульсы при выводе нейронной сети, что порождает возможность асинхронной работы нейронной сети.
5. Применение дельта-алгоритма не требует переобучения сети.
6. Уменьшение числа арифметических операций достигается за счет удаления весов и редкой активации нейронов (для молчащего нейрона нет смысла умножать его активацию на исходящие веса).
7. Хорошо подходит для сред с высоким уровнем корреляции кадров.
8. Может быть применен к любой нейронной сети из класса сверточных и реккурентных сетей.
9. Может быть применен к большинству современных алгоритмов обучения с подкреплением.

На рис. 3.10 показана схематичная визуализация работы данной сети. На ней видно, что при использовании временно́й разреженности большая часть нейронов молчит и только небольшая часть нейронов активна и распространяет сигнал последующим нейронам, причем не все из нейронов, принимающих сигнал, в дальнейшем активируются. Также видно, что часть связей между нейронами отсутствует.

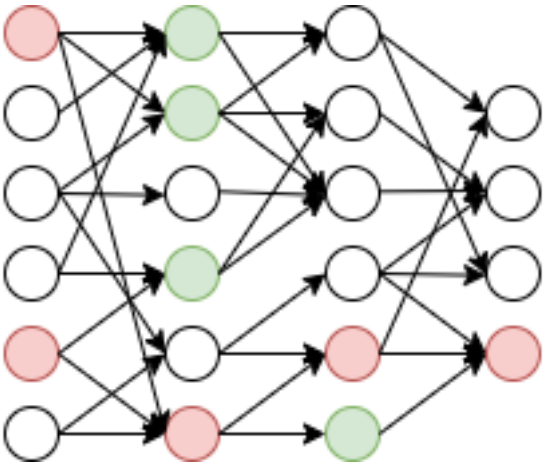


Рисунок 3.10 — Схема работы нейронной сети, оптимизированной предложенным методом на основе структурной и временно́й разреженностей. Зеленые нейроны – получившие сигнал, но не активированные. Красные нейроны – получившие сигнал и активированные. Белые нейроны - спящие. Только активированные нейроны распространяют сигнал. При этом не все принимающие нейроны активируются. Часть соединений между нейронами отсутствует из-за структурной разреженности. Рисунок автора.

Глава 4. Анализ эффективности предложенных методов оптимизации

Описанные далее эксперименты реализованы на языке Python с использованием библиотеки глубокого обучения PyTorch и библиотеки глубокого обучения с подкреплением StableBaselines3 [106]. Все эксперименты были проведены на вычислительных мощностях Московского государственного университета имени М.В. Ломоносова [107] (в частности, на платформах DGX), а также на вычислительных мощностях автора.

4.1 Тестовые среды для алгоритмов обучения с подкреплением

В данном разделе мы рассмотрим среды (окружения) Atari и MuJoCo, которые будут использоваться для оценки эффективности алгоритмов, предложенных в предыдущей части исследования (см гл. 3.3, 3.4). Эти среды являются стандартными бенчмарками для оценки качества работы алгоритмов обучения с подкреплением. Использование стандартных окружений позволяет объективно сравнивать результаты различных исследований и оценивать прогресс в области. Кроме того, широкое распространение этих бенчмарков облегчает воспроизводимость экспериментов и верификацию полученных результатов.

4.1.1 Среды Atari games

Игровые среды Atari [70] представляют собой набор классических видеоигр, адаптированных для тестирования алгоритмов обучения с подкреплением. Эти среды отличаются от традиционных тестовых платформ следующими ключевыми характеристиками:

– Высокоразмерное (210x160x3) пространство наблюдений s : в отличие от сред с низкоразмерными представлениями состояний, игры Atari оперируют непосредственно пиксельными данными игрового экрана, что значительно усложняет задачу обработки информации.

Рисунок 4.1 — Кадры из Atari окружений Breakout, SpaceInvaders, Robotank, Enduro, Krull, Freeway.

– Дискретное пространство действий a: возможные действия агента соответствуют движениям джойстика и нажатиям кнопок, доступным на оригинальной консоли Atari 2600. Это создает четко определенный, но ограниченный набор возможных взаимодействий со средой (от 1 до 18 дискретных действий).
– Система вознаграждений с задержкой: агент получает вознаграждения или штрафы за свои действия, однако эффект от действий может проявляться не мгновенно, а с некоторой задержкой. Это добавляет сложность в процесс обучения, так как требует от алгоритма способности связывать текущие действия с отложенными последствиями.
– Визуальное разнообразие: каждая игра Atari предоставляет уникальный визуальный контекст, что позволяет тестировать гибкость и адаптивность алгоритмов к различным визуальным паттернам и игровым механикам.

Для иллюстрации разнообразия и сложности тестовых сред на рис. 4.1 представлены примеры кадров из различных игр Atari. Эти изображения демонстрируют широкий спектр визуальных стилей и игровых элементов, с которыми придется работать алгоритмам обучения с подкреплением.

4.1.2 Среды MuJoCo

Среды *MuJoCo (Multi-Joint dynamics with Contact)* [108] относятся к классу окружений с континуальным управлением. В этих средах алгоритм должен генерировать команды в виде векторов вещественных чисел. Как правило, в средах MuJoCo необходимо контролировать поведение (например, ходьбу) биоподобных механизмов, сформированных из твердых тел, соединенных суставами. Состояниями s этих окружений являются векторы действительных чисел размерностью

от 8 до 376, которые включают информацию о состоянии агента и мира (например, положения, скорости и углы сочленений). Действия a для этих сред также являются векторами вещественных значений с размерностями от 1 до 17. Они определяют, как агент может взаимодействовать с окружающей средой (например, прикладывать силы или крутящие моменты к своим суставам). Примеры механизмов для управления из среды MuJoCo представлены на рис. 4.2.

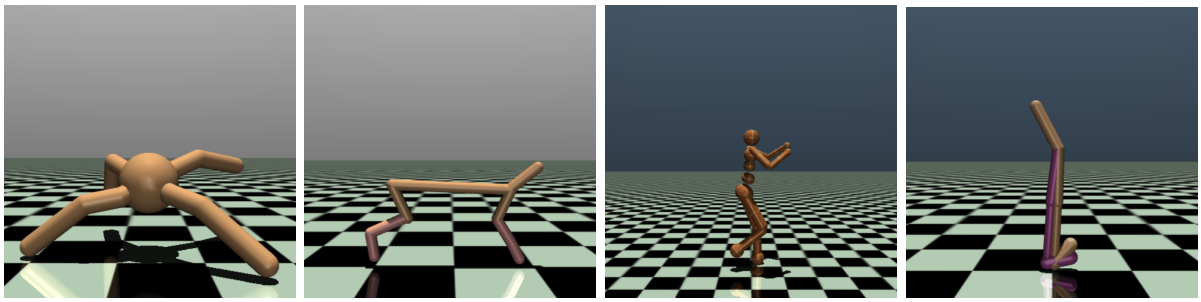


Рисунок 4.2 — Визуализация MuJoCo окружений Ant, HalfCheetah, Humanoid, Walker2d

4.2 Анализ эффективности алгоритма оптимизации на основе комбинации структурной разреженности и квантования в задачах обучения с подкреплением

Алгоритм оптимизации инференса нейронных сетей, полученный на основе объединения структурной разреженности и квантования, применялся для задач обучения с подкреплением. В качестве методов RL были выбраны SAC (см. 3.1.2) и DQN (см. 3.1.1). Под понятием качества работы алгоритма будем понимать сумму наград набранных алгоритмом при запуске в конкретном окружении.

Для метода SAC были выбраны среды MuJoCo: Ant, HalfCheetah, Hopper, Humanoid, Swimmer и Walker2d. В качестве архитектуры обучаемой сети был выбран многослойный персептрон (MLP-сеть) с двумя скрытыми слоями по 256 нейронов каждый.

В качестве модельных сред для метода DQN были выбраны среды игры Atari: Pong, Boxing, Tutankham, CrazyClimber. В качестве архитектур обучаемых сетей были использованы как классическая сверточная сеть размером в 1 млн параметров из работы DQN [1], так и сеть ResNet размером в 10 млн параметров, эквивалентная сети представленной в [109].

На рис. 4.3, 4.4, 4.5 представлена производительность обрезанных и/или квантованных нейронных сетей в различных средах.

На рис. 4.3 представлены результаты для MLP-сетей, обученных с помощью алгоритма SAC, для сред MuJoCo. Мы видим, что для всех сред MuJoCo (за исключением HalfCheetah) можно без потери качества работы сети обрезать до 98% весов и квантовать оставшиеся, что приводит к 200-кратному уменьшению размера нейронных сетей: 4-кратное уменьшение за счет квантования и 50-кратное за счет прореживания. Даже для HalfCheetah можно без потери качества работы сети обрезать 80 % весов и квантовать оставшиеся, что приводит к 20-кратному уменьшению размера нейронной сети. Для некоторых сред, таких как Hopper и Swimmer, можно обрезать без потери качества 99 % весов и квантовать оставшиеся, что приводит к 400-кратному уменьшению размера нейронной сети. Более того, качество работы обрезанных и квантованных сетей обычно немного превосходит качество работы только обрезанных сетей, что приводит к лучшим результатам даже в сравнении с неоптимизированными моделями. Данный феномен можно объяснить тем, что квантование действует как дополнительный регуляризатор, ограничивая пространство возможных значений весов и тем самым снижая риск переобучения модели.

По сравнению с результатами, представленными в [82], мы достигли высоких уровней разреженности (до 99 процентов) без потери качества для алгоритма SAC. Это можно объяснить стратегией прореживания только модели актора, в отличие от прореживания как актора, так и критика в [82]. Авторы [82] проводили эксперименты, чтобы определить оптимальное соотношение параметров между актором и критиком для заданного количества параметров. Они пришли к выводу, что параметры актора менее значимы, чем параметры критика, что согласуется с нашими результатами. Мы выбрали прореживание только актора, поскольку *только его разреженность* важна для эффективного инференса.

На рис. 4.4 представлены результаты для классической DQN-сети на основе CNN для сред Atari. Мы видим, что для всех сред мы можем обрезать без потери качества до 80 процентов и квантовать оставшиеся веса, что приводит к 20-кратному уменьшению размера оптимизированных нейронных сетей. Для Pong и Tutankham мы можем обрезать и квантовать до 95 процентов весов, что приводит к общему 80-кратному уменьшению размера нейронных сетей.

На рис. 4.5 представлены результаты для DQN-сетей на основе ResNet для сред Atari. Для сред Boxing и CrazyClimber можно обрезать до 90 процентов весов

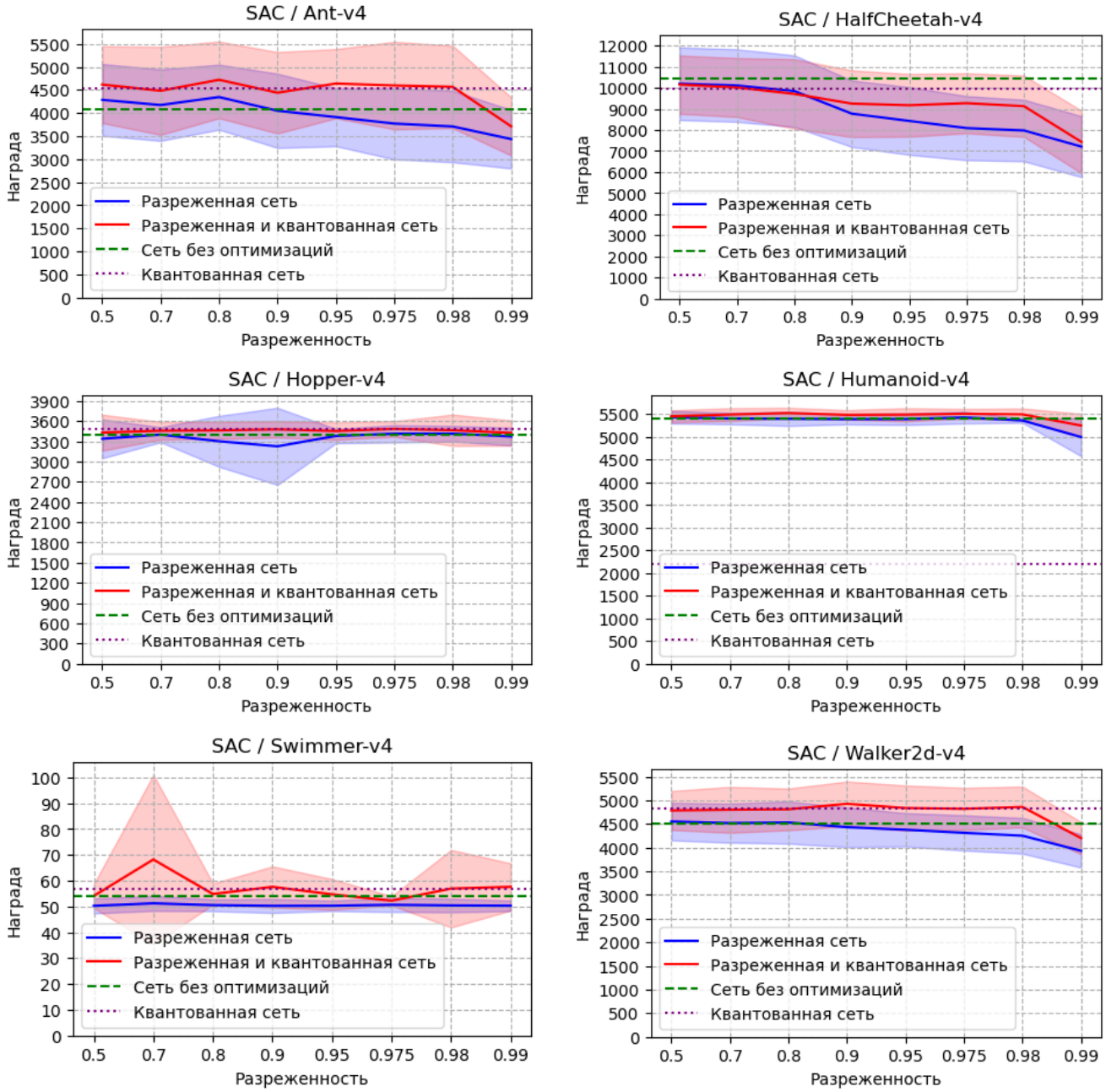


Рисунок 4.3 — Результаты оптимизации для алгоритма SAC применительно к окружениям MuJoCo. Оси x на графиках отображают степень разреженности нейронной сети; оси y обозначают производительность — награду, полученную агентом (чем больше тем лучше). Синяя линия показывает производительность обрезанной сети [82], красная линия показывает производительность обрезанной и квантованной сети (результат автора) [A.2]. Пунктирная фиолетовая линия показывает производительность только квантованной сети [76]. Зелёная пунктирная линия показывает производительность сети по умолчанию (без оптимизаций) [71].

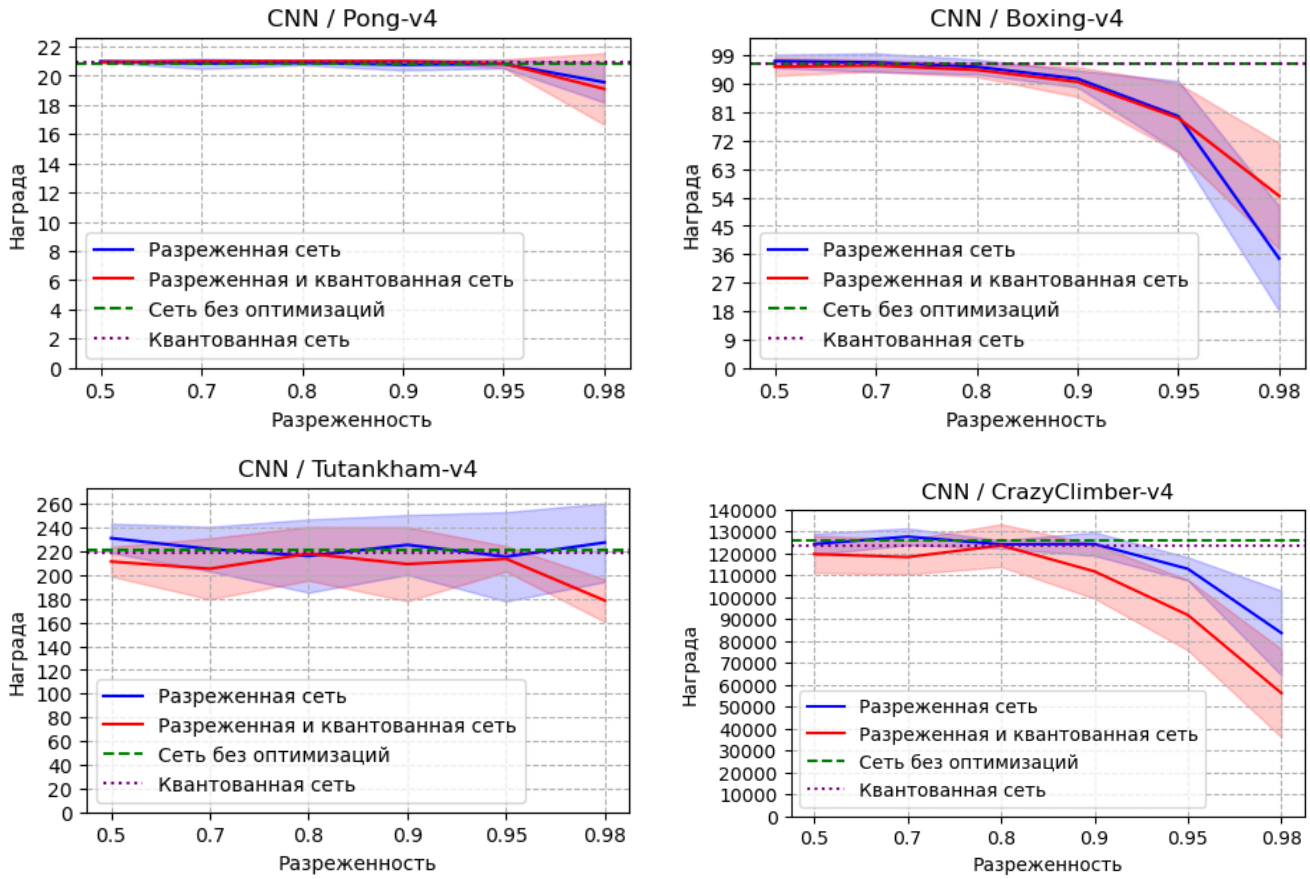


Рисунок 4.4 — Результаты оптимизации для алгоритма DQN и сверточной нейронной сети (CNN), применительно к Atari средам. Оси x на графиках отображают степень разреженности нейронной сети; оси y обозначают производительность — награду, полученную агентом (чем больше тем лучше). Синяя линия показывает производительность обрезанной сети [82], а красная линия показывает производительность обрезанной и квантованной сети (результат автора) [A.2]. Пурпурная пунктирная линия показывает производительность только квантованной сети [76]. Зеленая пунктирная линия показывает производительность сети по умолчанию (без оптимизаций) [1].

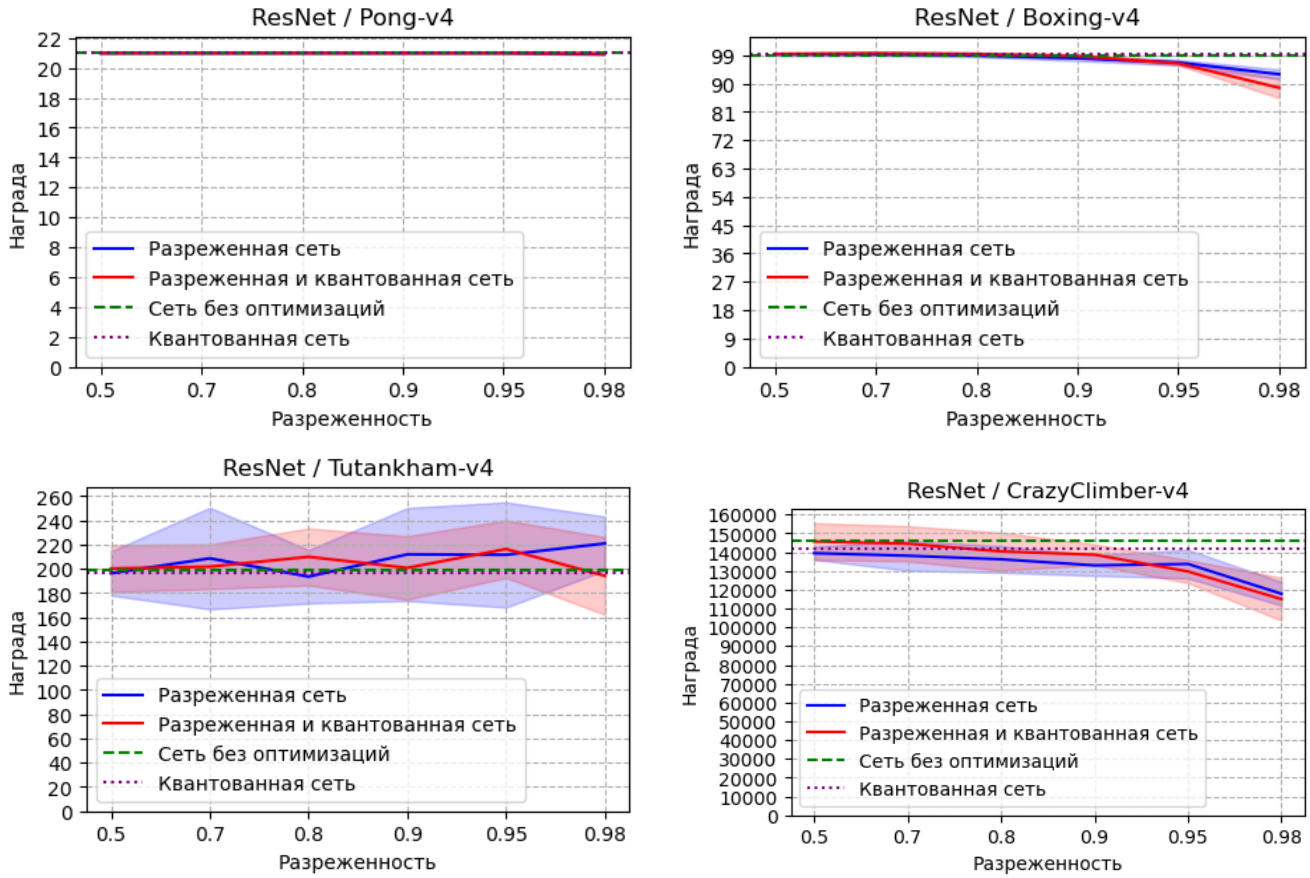


Рисунок 4.5 — Результаты оптимизации для алгоритма DQN и сети ResNet, применительно к Atari средам. Ось x на графиках обозначает степень разреженности нейронной сети; ось y обозначает производительность - награду, полученную агентом (чем больше тем лучше). Синяя линия показывает производительность обрезанной сети [82], красная линия показывает производительность обрезанной и квантованной сети (результат автора) [A.2]. Пунктирная фиолетовая линия показывает производительность только квантованной сети [76]. Зеленая пунктирная линия показывает производительность сети по умолчанию (без оптимизаций) [109].

НС и квантовать оставшиеся веса с потерей качества не более чем в три процента, что приводит к 40-кратному уменьшению размера нейронных сетей. Для окружений Pong и Tutankham можно обрезать без потери качества до 98 процентов весов НС и квантовать оставшиеся, что приводит к 200-кратному уменьшению размера нейронных сетей. Стоит отметить, что нейронные сети на основе ResNet намного более пригодны для прюнинга и квантования, что совпадает с выводами в [82].

4.2.1 Выводы

Проведенный анализ эффективности алгоритма оптимизации на основе комбинации структурной разреженности и квантования показал значительное сокращение размера нейронных сетей (до 200-400 раз) без потери качества работы в задачах обучения с подкреплением. Для большинства исследуемых сред MuJoCo удалось обрезать до 98-99% весов и квантовать оставшиеся. В экспериментах с DQN на средах Atari на классических CNN-архитектурах была показана возможность обрезать до 80% весов без потери качества для большинства игр и до 95% для Pong и Tutankham, с последующим квантованием весов. В то же время ResNet-архитектуры показали большую устойчивость к оптимизации, допуская обрезание до 98% весов с последующим квантованием, что обеспечило 200-кратное уменьшение размера моделей. Эти результаты подтверждают высокую эффективность предложенного метода оптимизации для различных архитектур нейронных сетей и алгоритмов обучения с подкреплением. Вышеописанные результаты были опубликованы автором в статье [A.2].

4.3 Анализ эффективности алгоритма оптимизации на основе комбинации структурной и временной разреженностей в задачах обучения с подкреплением

Алгоритм, полученный путем объединения структурной и временной разреженностей (см. гл. 3.4), был применен к задачам обучения с подкреплением. В качестве модельного алгоритма обучения с подкреплением был использован алгоритм DQN (см. гл. 3.1.1), а в качестве модельной среды для обучения были использованы окружения из семейства Atari games: Freeway, Enduro, Krull, Robotank, Breakout, SpaceInvaders.

На рис. 4.6 представлены показатели производительности и вознаграждения нейронной сети при различных уровнях разреженности и для различных сред.

Результаты показателей вознаграждения разреженных сетей без применения дельта-алгоритма (синяя линия на графике) согласуются с результатами, полученными в [97; 98], где наибольшее снижение вознаграждения при обрезании

наблюдалось в средах SpaceInvaders и Breakout. Оранжевая линия демонстрирует вознаграждение для разреженных нейронных сетей с дельта-нейронами (т.е. после применения дельта-алгоритма). Их производительность сопоставима (иногда незначительно выше или ниже) с производительностью сети без дельта-нейронов, что свидетельствует о том, что дельта-алгоритм *не оказывает* существенного влияния на вознаграждение.

4.3.1 Анализ числа обращений к памяти

Структурная разреженность уменьшает необходимое количество передаваемых параметров из памяти за счет сжатия размеров слоев. Дельта-нейрон в полносвязном слое при отсутствии активации не требует получения значения выходных весов. Комбинация этих двух факторов позволяет существенно уменьшить количество передаваемых данных из памяти при инференсе сети, оптимизированной с помощью предложенного алгоритма. Следует отметить, что в зависимости от обрабатываемых входных данных на разных шагах дельта-алгоритм будет демонстрировать различные уровни временной разреженности, поэтому целесообразно рассматривать усредненные показатели числа обращений в память. На рис. 4.6 зеленой пунктирной линией показаны усредненные доли числа обращений в память оптимизированного алгоритма от числа обращений в память алгоритма без оптимизаций при различных уровнях разреженности и для различных сред.

Число обращений к памяти зависит от игры, в которую играет агент, и от уровня разреженности его сети. Предложенный алгоритм оптимизации 3.4 обеспечивает различные уровни структурной и временной разреженностей в зависимости от выбранного порога, слоя и входных данных (окружения). Дельта-алгоритм (без обрезания весов) приводит к уменьшению количества обращений к памяти от 3,7 раз для Robotank до 14,3 раз для Breakout. Одновременное использование с дельта-алгоритмом структурной разреженности еще больше уменьшает количество обращений к памяти. Например, для Freeway доля обращений к памяти уменьшается при обрезании с 0,33 до 0,026 (уменьшение в 12,6 раз). Однако для Krull она уменьшается с 0,07 лишь до 0,033 (уменьшение в 2,2 раза). Такая вариабельность объясняется тем, что структурная разреженность весов статически

влияет на количество обращений в память, в то время как дельта-алгоритм обеспечивает разные уровни временной разреженности в зависимости от выбранного порога, слоя, разреженности и входных данных (окружения). Таким образом проявляется сложное внутреннее взаимодействие и взаимная интерференция обоих алгоритмов оптимизации.

4.3.2 Анализ числа значимых операций умножения

Будем называть операцию умножения *значимой* (ненулевой) если хотя бы один её операндов не равен нулю. Несмотря на то, что сокращение числа операций умножения намного меньше влияет на скорость работы и энергопотребление системы ИИ, этот фактор также может положительно повлиять на производительность системы.

В данном алгоритме нулевые операнды могут появляться либо из-за неактивных дельта-нейронов, либо из-за структурной разреженности. В первом случае все выходные веса нейрона будут умножаться на нулевое значение, что делает эту операцию избыточной. Во втором случае часть весов будет равна нулю, и эти операции всегда могут быть пропущены (например, с помощью применения масок и вычислителей, поддерживающих разреженные вычисления). Следует отметить, что в зависимости от обрабатываемых входных данных на различных этапах дельта-алгоритм будет демонстрировать различные уровни разреженности, поэтому целесообразно рассматривать усредненные показатели числа значимых операций умножения.

На рис. 4.6 красной пунктирной линией показаны усредненные доли ненулевых операций умножений в оптимизированном алгоритме от числа операций умножения алгоритма без оптимизаций при различных уровнях разреженности и для различных сред.

Число ненулевых умножений зависит от игры, в которую играет агент. Степень структурной разреженности весов статически влияет на количество ненулевых умножений. Однако, как уже упоминалось, дельта-алгоритм обеспечивает различные уровни временной разреженности, что и объясняется вышеупомянутую вариабельность.

Слой	Умножения	Ненулевые умножения	Доля нулевых умножений	Разреженность весов	Дельта-разреженность
Input	0	0	0.0	0.0	0.994
Conv2d-1	3,276,800	6554	0.998	0.600	0.974
Conv2d-2	2,654,208	13271	0.995	0.778	0.927
Conv2d-4	1,806,144	19868	0.989	0.843	0.949
Dense-1	1,605,632	80281	0.950	0.79	0.815
Dense-2	2,048	379	0.815	0.79	0.412
Общее	9344832	121483	0.987	0.79	0.977

Таблица 3 — Число умножений в среде Breakout с уровнем структурной разреженности равным 0.79 и порогом D равным 0.01.

Слой	Умножения	Ненулевые умножения	Доля нулевых умножений	Разреженность весов	Дельта-разреженность
Input	0	0	0.0	0.0	0.986
Conv2d-1	3,276,800	16384	0.995	0.693	0.935
Conv2d-2	2,654,208	45122	0.983	0.764	0.802
Conv2d-4	1,806,336	70447	0.961	0.835	0.857
Dense-1	1,605,632	228000	0.858	0.79	0.756
Dense-2	2,048	498	0.757	0.79	0.005
Общее	9344832	364448	0.961	0.79	0.943

Таблица 4 — Число умножений в среде SpaceInvaders с уровнем структурной разреженности равным 0.79 и порогом D равным 0.01.

Один только дельта-алгоритм (без обрезания весов) приводит к уменьшению количества ненулевых операций умножения от 5,5 раз для Robotank до 55 раз для Breakout. Для некоторых сред доля ненулевых операций умножения уменьшается в результате обрезания больше, чем для других. Например, для Robotank доля ненулевых операций умножения уменьшается при обрезании с 0,184 до 0,059 (около 3,11 раз). В то же время для Breakout она уменьшается всего лишь с 0,018 до 0,015 (1,2 раза). В любом случае очевидно, что выигрыш от дельта-алгоритма с точки зрения числа значимых операций умножения значительно выше, чем от обрезания весов. Более того, прюнинг иногда приводит к ухудшению производительности. Более подробные замеры числа значимых операций умножения приведены в таблицах 3 и 4.

4.3.3 Зависимость числа значимых операций умножения в зависимости от среды и порога

Различия в приросте производительности между различными средами можно объяснить тем фактом, что в некоторых средах (например, SpaceInvaders)

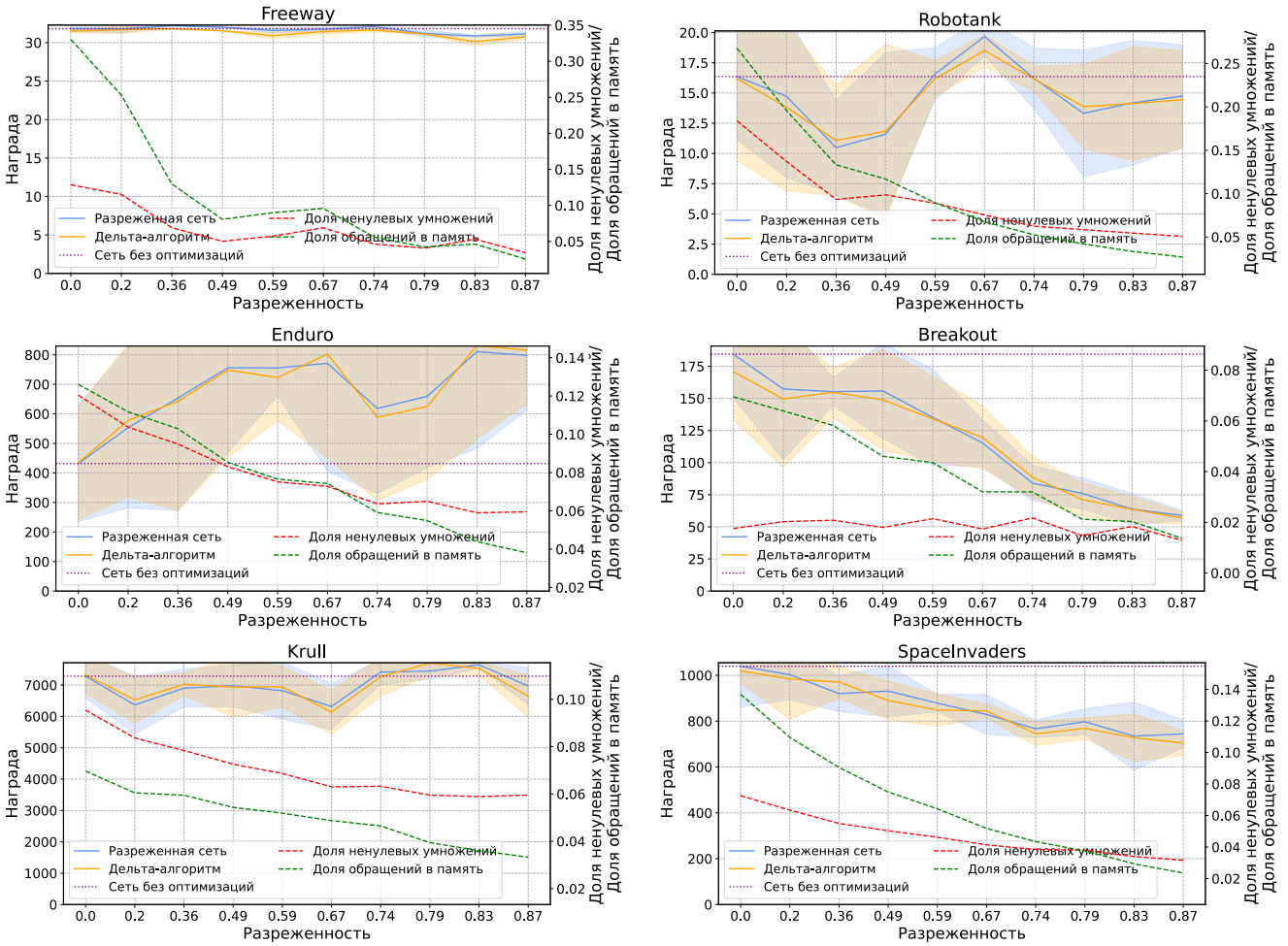


Рисунок 4.6 — Результаты оптимизации для сред Freeway, Robotank, Enduro, Breakout, Krull and SpaceInvaders. На всех графиках ось x отображает степень разреженности нейронной сети. Левая ось y обозначает вознаграждения (качество работы), полученные агентом; правая ось y представляет долю значимых операций умножения и долю обращений к памяти, усредненную по запускам среды. Синяя линия демонстрирует производительность обрезанной сети [98], а оранжевая - производительность обрезанной сети с дополнительным применением дельта-алгоритма (результат автора) [A.3]. Красная пунктирная линия отображает долю значимых умножений обрезанной нейронной сети, дополненной дельта-алгоритмом от числа значимых умножений неоптимизированной сети (меньшее значение предпочтительнее, результат автора) [A.3]. Зеленая пунктирная линия показывает долю необходимых обращений к памяти обрезанной нейронной сети, дополненной дельта-алгоритмом от числа обращений неоптимизированной сети (меньшее значение предпочтительнее, результат автора) [A.3]. Фиолетовая пунктирная линия иллюстрирует качество работы нейронной сети без какой-либо оптимизации [1].

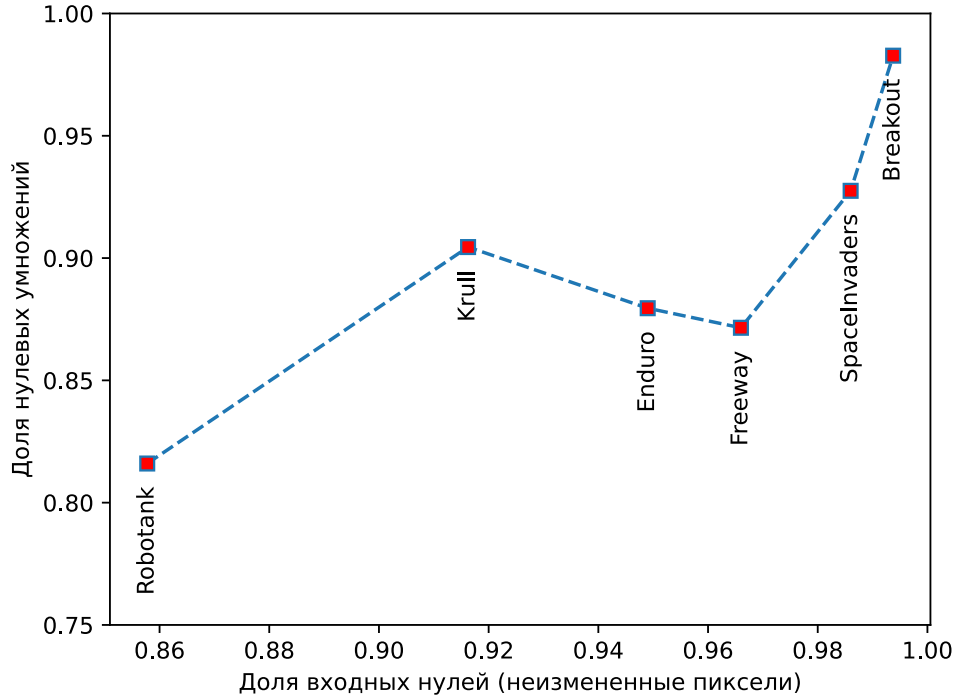


Рисунок 4.7 — Корреляция входной разреженности (ось X) и доли умножений с нулевыми операндами (ось Y). Каждый красный квадрат соответствует определенной среде.

на каждом временном шаге меняется гораздо больше пикселей, чем в других (например, Breakout). В Breakout движутся только игровая площадка и мяч, а в SpaceInvaders могут двигаться сразу несколько объектов — выстрелы, корабль и инопланетяне. Это хорошо подтверждается разницей в уровне дельта-разреженности (доли активаций нейронов при выполнении дельта-алгоритма) при игре в Breakout и SpaceInvaders (см. табл. 2 и 3). На рис. 4.7 показан корреляция между уровнем входной разреженности, вызванной дельта-алгоритмом, с долей нулевых умножений. Прослеживается тенденция увеличения доли нулевых умножений во время вывода нейронной сети с увеличением доли входных нулей.

Кроме того, были проведены исследования того, как параметр D (порог) влияет на производительность и выигрыш в числе значимых операций умножения. Производительность оценивалась для сред Krull и SpaceInvaders, используя различные значения T, а именно 0,001, 0,005, 0,01 и 0,05. Результаты представлены на рис. 4.8. Они наглядно иллюстрируют, что 0,01 является наиболее предпочтительным значением порога среди протестированных. Более высокие значения, несмотря на выигрыш в вариантах умножения, приводят к значительному уменьшению вознаграждения. В то время как более низкие пороговые

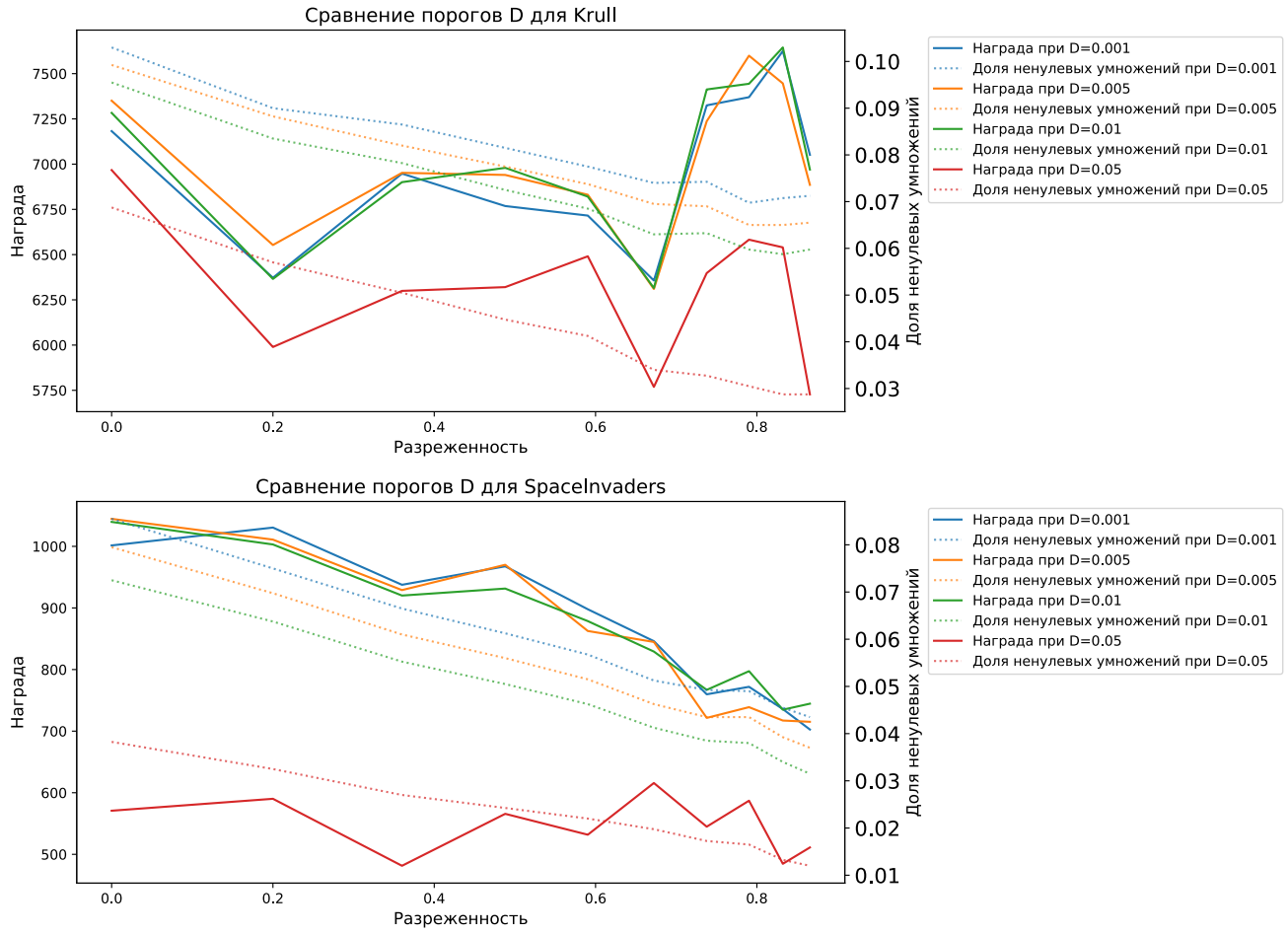


Рисунок 4.8 — Сравнение влияния значений порога D для Krull и SpaceInvaders: ось x представляет степень разреженности нейронной сети. Левые оси y соответствуют вознаграждениям, полученных агентом, а правые оси y представляют долю значимых умножений. Пунктирные линии представляют процент значимых умножений (чем меньше, тем лучше) в обрезанных нейронных сетях, усиленных дельта-алгоритмом. Сплошные линии обозначают вознаграждения агентов. Разные цвета соответствуют разным порогам.

значения дают почти идентичные вознаграждения с более скромным повышением производительности.

4.3.4 Выводы: уменьшение числа операций и обращений в память, возможность асинхронной работы

Умножение — дорогостоящая вычислительная операция с точки зрения энергии и времени [8]. Предоставленный метод с помощью дельта-алгоритма

уменьшает количество значимых умножений. Также он позволяет уменьшить обращения в память, которые еще более затратны по энергии и времени [8], благодаря «молчанию» нейронов и, как следствие, не получать веса выходных соединений «молчащих» нейронов. Более того, из-за структурной разреженности такой подход еще больше уменьшает количество обращений к памяти и количество значимых операций умножения. Вышеописанные результаты были опубликованы автором в статье [А.3].

Стоит отметить что в настоящее время имеется крайне мало возможностей использовать существующее оборудование для эффективной реализации данного алгоритма. Это связано с тем, что современные графические процессоры предназначены для работы с «плотными» матрицами. Тем не менее, попытки изменить ситуацию предпринимаются. Авторы вышеупомянутого дельта-алгоритма представили архитектуру NeuronFlow [43; 110], поддерживающую дельта-алгоритм. Процессор Loihi2, который Intel представила [42] в сентябре 2021 года, также имеет многоядерную асинхронную архитектуру, способную запускать разреженные по структуре и времени нейронные сети на основе дельта-нейронов.

Заключение

Основные результаты работы заключаются в следующем:

1. На основе детального анализа и классификации вычислительных принципов работы мозга человека показано, что отсутствие ряда ключевых принципов в современных системах ИИ, построенных на основе вычислителей с фон-неймановской архитектурой, определяет их низкую энергоэффективность, масштабируемость и скорость работы. Указанные ключевые принципы легли в основу предложенных в работе методов, направленных на создание быстрых и энергоэффективных систем ИИ для инференса задач глубокого обучения с подкреплением.
2. Впервые предложен метод оптимизации инференса нейронных сетей, тренированных алгоритмами обучения с подкреплением, на основе комбинации структурной разреженности и квантования. Метод существенно уменьшает размеры нейронных сетей (на 1 – 2 порядка, вплоть до 400 раз) без потери качества работы, что позволяет размещать нейронные сети в быстрой памяти или уменьшать число обращений в память, а также уменьшает на порядок число необходимых арифметических операций для инференса.
3. Впервые предложен метод оптимизации инференса нейронных сетей, тренированных алгоритмами обучения с подкреплением, на основе комбинации временно́й и структурной разреженности. Метод уменьшает на порядок (до 25 раз) число обращений в память и число необходимых арифметических операций при инференсе нейронных сетей без потери качества работы.
4. Предложенные методы оптимизации программно реализованы и прошли апробацию на тестовых окружениях Atari и MuJoCo, являющихся стандартными бенчмарками для задач обучения с подкреплением и хорошо отражающих типовые случаи входных данных в них, подтвердив эффективность предложенных методов.

В заключение автор выражает благодарность и большую признательность научному руководителю Воеводину Владимиру Валентиновичу за помощь, обсуждение результатов и научное руководство. Автор также выражает благодарность своей матери Ивановой Ольге Петровне и своему дедушке Иванову Петру

Васильевичу (1930–2025) за воспитание, образование и поддержку. Автор глубоко признателен своей жене Кузнецовой Дарье Владимировне за её неоценимую поддержку, помощь, терпение и понимание.

Публикации автора по теме диссертации

Научные статьи, опубликованные в изданиях, рекомендованных для защиты в диссертационном совете МГУ имени М.В. Ломоносова по специальности и отрасли наук:

A.1. Neuromorphic artificial intelligence systems / D. Ivanov [et al.] // Frontiers in Neuroscience. — 2022. — Vol. 16. — P. 959626. — EDN: HZGYDI — (WoS Q2, Импакт-фактор 3.2 (JIF)) [1.25/0.75].

A.2. Neural network compression for reinforcement learning tasks / D. A. Ivanov [et al.] // Scientific Reports. — 2025. — Vol. 15, no. 1. — P. 9718. — EDN: PGLCHZ — (WoS Q1, Импакт-фактор 3.8 (JIF)) [0.68/0.5].

A.3. Deep reinforcement learning with significant multiplications inference / D. A. Ivanov [et al.] // Scientific Reports. — 2023. — Vol. 13, no. 1. — P. 20865. — EDN: PAYIZL — (WoS Q1, Импакт-фактор 3.8 (JIF)) [0.625/0.45].

Список литературы

1. Human-level control through deep reinforcement learning / V. Mnih [et al.] // nature. — 2015. — Vol. 518, no. 7540. — P. 529—533.
2. Mastering the game of go without human knowledge / D. Silver [et al.] // nature. — 2017. — Vol. 550, no. 7676. — P. 354—359.
3. Llama 2: Open foundation and fine-tuned chat models / H. Touvron [et al.] // arXiv preprint arXiv:2307.09288. — 2023.
4. *Perrault*, *R.* Artificial Intelligence Index Report 2024 / R. Perrault, J. Clark // https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI$_A I - Index - Report - 2024.pdf$. — 2024.
5. Training compute-optimal large language models / J. Hoffmann [et al.] // arXiv preprint arXiv:2203.15556. — 2022.
6. *Von Neumann*, *J.* First Draft of a Report on the EDVAC / J. Von Neumann // IEEE Annals of the History of Computing. — 1993. — Vol. 15, no. 4. — P. 27—75.
7. *Backus*, *J.* Can programming be liberated from the von Neumann style? A functional style and its algebra of programs / J. Backus // Communications of the ACM. — 1978. — Vol. 21, no. 8. — P. 613—641.
8. *Horowitz*, *M.* 1.1 computing's energy problem (and what we can do about it) / M. Horowitz // 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). — IEEE. 2014. — P. 10—14.
9. Flashattention: Fast and memory-efficient exact attention with io-awareness / T. Dao [и др.] // Advances in neural information processing systems. — 2022. — Т. 35. — С. 16344—16359.
10. *Rosenblatt*, *F.* The perceptron, a perceiving and recognizing automaton Project Para / F. Rosenblatt. — Cornell Aeronautical Laboratory, 1957.
11. *Nair*, *V.* Rectified linear units improve restricted boltzmann machines / V. Nair, G. E. Hinton // Proceedings of the 27th international conference on machine learning (ICML-10). — 2010. — P. 807—814.

12. Rectifier nonlinearities improve neural network acoustic models / A. L. Maas, A. Y. Hannun, A. Y. Ng, [et al.] // Proc. icml. Vol. 30. — Atlanta, GA. 2013. — P. 3.

13. *Clevert*, *D.-A.* Fast and accurate deep network learning by exponential linear units (elus) / D.-A. Clevert, T. Unterthiner, S. Hochreiter // arXiv preprint arXiv:1511.07289. — 2015.

14. *Hendrycks*, *D.* Gaussian error linear units (gelus) / D. Hendrycks, K. Gimpel // arXiv preprint arXiv:1606.08415. — 2016.

15. *Rodriguez*, *A.* Deep learning systems: algorithms, compilers, and processors for large-scale production / A. Rodriguez. — Morgan & Claypool Publishers, 2020.

16. Deep residual learning for image recognition / K. He [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — P. 770—778.

17. *Hennessy*, *J.* Computer Organization and Design. A Quantitative Approach / J. Hennessy, D. Patterson. — Morgan Kaufmann, 2017.

18. Nvidia ampere architecture in-depth / R. Krashinsky [et al.] // NVIDIA blog: https://devblogs. nvidia. com/nvidia-ampere-architecture-in-depth. — 2020.

19. In-datacenter performance analysis of a tensor processing unit / N. P. Jouppi [et al.] // Proceedings of the 44th annual international symposium on computer architecture. — 2017. — P. 1—12.

20. *Sutton*, *R. S.* Reinforcement learning: An introduction / R. S. Sutton, A. G. Barto. — MIT press, 2018.

21. Champion-level drone racing using deep reinforcement learning / E. Kaufmann [et al.] // Nature. — 2023. — Vol. 620, no. 7976. — P. 982—987.

22. Magnetic control of tokamak plasmas through deep reinforcement learning / J. Degrave [et al.] // Nature. — 2022. — Vol. 602, no. 7897. — P. 414—419.

23. Principles of neural science / A. J. Hudspeth [et al.]. — McGraw-Hill, Health Professions Division, 2013.

24. *Furber*, *S.* Large-scale neuromorphic computing systems / S. Furber // Journal of neural engineering. — 2016. — Vol. 13, no. 5. — P. 051001.

25. Opportunities for neuromorphic computing algorithms and applications / C. D. Schuman [et al.] // Nature Computational Science. — 2022. — Vol. 2, no. 1. — P. 10—19.

26. *Watson*, *N. V.* The mind's machine: Foundations of brain and behavior. / N. V. Watson, S. M. Breedlove. — Sinauer Associates, 2012.

27. *LIZARAZU UGALDE*, *M.* Speech-brain synchronization: a possible cause for developmental dyslexia : дис. … канд. / LIZARAZU UGALDE MIKEL. — Universidad del País Vasco-Euskal Herriko Unibertsitatea, 2017.

28. *Miller*, *P.* An introductory course in computational neuroscience / P. Miller. — MIT Press, 2018.

29. *Buckner*, *C.* Connectionism / C. Buckner, J. Garson // The Stanford Encyclopedia of Philosophy / под ред. E. N. Zalta. — Fall 2019. — Metaphysics Research Lab, Stanford University, 2019.

30. *Goodfellow*, *I.* Deep learning / I. Goodfellow, Y. Bengio, A. Courville. — MIT press, 2016.

31. Parallel distributed processing. Vol. 1 / D. E. Rumelhart, J. L. McClelland, P. R. Group, [et al.]. — IEEE New York, 1988.

32. *Sterling*, *P.* Principles of neural design / P. Sterling, S. Laughlin. — MIT press, 2015.

33. *Maass*, *W.* Networks of spiking neurons: the third generation of neural network models / W. Maass // Neural networks. — 1997. — Vol. 10, no. 9. — P. 1659—1671.

34. *Киселев*, *М.* Импульсные нейронные сети: Представление информации, обучение, память / М. Киселев. — Palmarium Academic Publishing, 2020.

35. *Самарский*, *А. А.* Численные методы / А. А. Самарский, А. В. Гулин. — 1989.

36. *Niven*, *J. E.* Neuronal energy consumption: biophysics, efficiency and evolution / J. E. Niven // Current opinion in neurobiology. — 2016. — Vol. 41. — P. 129—135.

37. *Lee*, *J. H.* Training deep spiking neural networks using backpropagation / J. H. Lee, T. Delbruck, M. Pfeiffer // Frontiers in neuroscience. — 2016. — Vol. 10. — P. 228000.

38. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation / N. Rathi [et al.] // arXiv preprint arXiv:2005.01807. — 2020.

39. Spike-timing dependent plasticity / J. Sjöström, W. Gerstner, [et al.] // Spike-timing dependent plasticity. — 2010. — Vol. 35, no. 0. — P. 0–0.

40. A million spiking-neuron integrated circuit with a scalable communication network and interface / P. A. Merolla [et al.] // Science. — 2014. — Vol. 345, no. 6197. — P. 668—673.

41. Loihi: A neuromorphic manycore processor with on-chip learning / M. Davies [et al.] // Ieee Micro. — 2018. — Vol. 38, no. 1. — P. 82—99.

42. *Intel*. Taking Neuromorphic Computing to the Next Level with Loihi 2 / Intel // Technology Brief. — 2021.

43. NeuronFlow: a neuromorphic processor architecture for Live AI applications / O. Moreira [et al.] // 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE). — IEEE. 2020. — P. 840—845.

44. IBM NorthPole Neural Inference Machine / D. S. Modha [et al.] // 2023 IEEE Hot Chips 35 Symposium (HCS). — IEEE Computer Society. 2023. — P. 1—58.

45. A software-defined tensor streaming multiprocessor for large-scale machine learning / D. Abts [и др.] // Proceedings of the 49th Annual International Symposium on Computer Architecture. — 2022. — С. 567—580.

46. *Laughlin*, *S. B.* The metabolic cost of neural information / S. B. Laughlin, R. R. de Ruyter van Steveninck, J. C. Anderson // Nature neuroscience. — 1998. — Vol. 1, no. 1. — P. 36—41.

47. *Attwell*, *D.* An energy budget for signaling in the grey matter of the brain / D. Attwell, S. B. Laughlin // Journal of Cerebral Blood Flow & Metabolism. — 2001. — Vol. 21, no. 10. — P. 1133—1145.

48. *Amdahl*, *G. M.* Validity of the single processor approach to achieving large scale computing capabilities / G. M. Amdahl // Proceedings of the April 18-20, 1967, spring joint computer conference. — 1967. — P. 483—485.

49. *Végh, J.* How Amdahl's Law limits the performance of large artificial neural networks: why the functionality of full-scale brain simulation on processor-based simulators is limited / J. Végh // Brain informatics. — 2019. — Vol. 6, no. 1. — P. 4.

50. *Frenkel, C. P.* Bottom-up and top-down neural processing systems design: Neuromorphic intelligence as the convergence of natural and artificial intelligence / C. P. Frenkel, D. Bol, G. Indiveri // arXiv. org. — 2021. — No. 2106.01288.

51. The spinnaker project / S. B. Furber [et al.] // Proceedings of the IEEE. — 2014. — Vol. 102, no. 5. — P. 652—665.

52. The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing / S. Höppner [et al.] // arXiv preprint arXiv:2103.08392. — 2021.

53. Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model / S. J. Van Albada [et al.] // Frontiers in neuroscience. — 2018. — P. 291.

54. *Quian Quiroga, R.* Measuring sparseness in the brain: comment on Bowers (2009). / R. Quian Quiroga, G. Kreiman. — 2010.

55. *Shoham, S.* How silent is the brain: is there a "dark matter" problem in neuroscience? / S. Shoham, D. H. O'Connor, R. Segev // Journal of Comparative Physiology A. — 2006. — Vol. 192, no. 8. — P. 777—784.

56. *Makhzani, A.* Winner-take-all autoencoders / A. Makhzani, B. J. Frey // Advances in neural information processing systems. — 2015. — Vol. 28.

57. *Ahmad, S.* How can we be so dense? The benefits of using highly sparse representations / S. Ahmad, L. Scheinkman // arXiv preprint arXiv:1903.11257. — 2019.

58. *Kanerva, P.* Sparse distributed memory / P. Kanerva. — MIT press, 1988.

59. Towards artificial general intelligence with hybrid Tianjic chip architecture / J. Pei [et al.] // Nature. — 2019. — Vol. 572, no. 7767. — P. 106—111.

60. EIE: Efficient inference engine on compressed deep neural network / S. Han [et al.] // ACM SIGARCH Computer Architecture News. — 2016. — Vol. 44, no. 3. — P. 243—254.

61. SpArNet: Sparse Asynchronous Neural Network execution for energy efficient inference / M. A. Khoei [et al.] // 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). — IEEE. 2020. — P. 256—260.

62. *Tee*, *J.* Is information in the brain represented in continuous or discrete form? / J. Tee, D. P. Taylor // IEEE Transactions on Molecular, Biological and Multi-Scale Communications. — 2020. — Vol. 6, no. 3. — P. 199—209.

63. *VanRullen*, *R.* Is perception discrete or continuous? / R. VanRullen, C. Koch // Trends in cognitive sciences. — 2003. — Vol. 7, no. 5. — P. 207—213.

64. Nanoconnectomic upper bound on the variability of synaptic plasticity / T. M. Bartol Jr [et al.] // elife. — 2015. — Vol. 4. — e10778.

65. *Faisal*, *A. A.* Noise in the nervous system / A. A. Faisal, L. P. Selen, D. M. Wolpert // Nature reviews neuroscience. — 2008. — Vol. 9, no. 4. — P. 292—303.

66. A framework for Bayesian optimality of psychophysical laws / J. Z. Sun [et al.] // Journal of Mathematical Psychology. — 2012. — Vol. 56, no. 6. — P. 495—501.

67. *Kahan*, *W.* IEEE standard 754 for binary floating-point arithmetic / W. Kahan // Lecture Notes on the Status of IEEE. — 1996. — Vol. 754, no. 94720—1776. — P. 11.

68. A domain-specific supercomputer for training deep neural networks / N. P. Jouppi [et al.] // Communications of the ACM. — 2020. — Vol. 63, no. 7. — P. 67—78.

69. Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system / S. Schmitt [et al.] // 2017 international joint conference on neural networks (IJCNN). — IEEE. 2017. — P. 2227—2234.

70. *Bellemare*, *M.* Investigating contingency awareness using Atari 2600 games / M. Bellemare, J. Veness, M. Bowling // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 26. — 2012. — P. 864—871.

71. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor / T. Haarnoja [et al.] // International conference on machine learning. — PMLR. 2018. — P. 1861—1870.

72. A comprehensive survey of neural architecture search: Challenges and solutions / P. Ren [et al.] // ACM Computing Surveys (CSUR). — 2021. — Vol. 54, no. 4. — P. 1—34.

73. *Gale*, *T.* The state of sparsity in deep neural networks / T. Gale, E. Elsen, S. Hooker // arXiv preprint arXiv:1902.09574. — 2019.

74. A survey of quantization methods for efficient neural network inference / A. Gholami [et al.] // arXiv preprint arXiv:2103.13630. — 2021.

75. Quantization and training of neural networks for efficient integer-arithmetic-only inference / B. Jacob [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2018. — P. 2704—2713.

76. QuaRL: Quantization for fast and environmentally sustainable reinforcement learning / S. Krishnan [et al.] // arXiv preprint arXiv:1910.01055. — 2019.

77. Low-precision reinforcement learning: running soft actor-critic in half precision / J. Björck [et al.] // International Conference on Machine Learning. — PMLR. 2021. — P. 980—991.

78. *Thimm*, *G.* Evaluating pruning methods / G. Thimm, E. Fiesler // Proceedings of the International Symposium on Artificial neural networks. — 1995. — P. 20—25.

79. *LeCun*, *Y.* Optimal brain damage / Y. LeCun, J. S. Denker, S. A. Solla // Advances in neural information processing systems. — 1990. — P. 598—605.

80. *Hassibi*, *B.* Second order derivatives for network pruning: Optimal brain surgeon / B. Hassibi, D. G. Stork. — Morgan Kaufmann, 1993.

81. *Zhu*, *M.* To prune, or not to prune: exploring the efficacy of pruning for model compression / M. Zhu, S. Gupta // arXiv preprint arXiv:1710.01878. — 2017.

82. The State of Sparse Training in Deep Reinforcement Learning / L. Graesser [et al.] // International Conference on Machine Learning. — PMLR. 2022. — P. 7766—7792.

83. Learning both weights and connections for efficient neural network / S. Han [et al.] // Advances in neural information processing systems. — 2015. — Vol. 28.

84. *See*, *A.* Compression of neural machine translation models via pruning / A. See, M.-T. Luong, C. D. Manning // arXiv preprint arXiv:1606.09274. — 2016.

85. *Molchanov, D.* Variational dropout sparsifies deep neural networks / D. Molchanov, A. Ashukha, D. Vetrov // International Conference on Machine Learning. — PMLR. 2017. — P. 2498—2507.

86. *Ullrich, K.* Soft weight-sharing for neural network compression / K. Ullrich, E. Meeds, M. Welling // arXiv preprint arXiv:1702.04008. — 2017.

87. Compressing neural networks using the variational information bottleneck / B. Dai [et al.] // International Conference on Machine Learning. — PMLR. 2018. — P. 1135—1144.

88. *Louizos, C.* Bayesian compression for deep learning / C. Louizos, K. Ullrich, M. Welling // Advances in neural information processing systems. — 2017. — Vol. 30.

89. *Lee, N.* Snip: Single-shot network pruning based on connection sensitivity / N. Lee, T. Ajanthan, P. H. Torr // arXiv preprint arXiv:1810.02340. — 2018.

90. *Wang, C.* Picking winning tickets before training by preserving gradient flow / C. Wang, G. Zhang, R. Grosse // arXiv preprint arXiv:2002.07376. — 2020.

91. Pruning neural networks without any data by iteratively conserving synaptic flow / H. Tanaka [et al.] // Advances in neural information processing systems. — 2020. — Vol. 33. — P. 6377—6389.

92. Pruning neural networks at initialization: Why are we missing the mark? / J. Frankle [et al.] // arXiv preprint arXiv:2009.08576. — 2020.

93. Deep rewiring: Training very sparse deep networks / G. Bellec [et al.] // arXiv preprint arXiv:1711.05136. — 2017.

94. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science / D. C. Mocanu [et al.] // Nature communications. — 2018. — Vol. 9, no. 1. — P. 2383.

95. Rigging the lottery: Making all tickets winners / U. Evci [et al.] // International Conference on Machine Learning. — PMLR. 2020. — P. 2943—2952.

96. *Frankle, J.* The lottery ticket hypothesis: Finding sparse, trainable neural networks / J. Frankle, M. Carbin // arXiv preprint arXiv:1803.03635. — 2018.

97. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp / H. Yu [et al.] // arXiv preprint arXiv:1906.02768. — 2019.

98. *Vischer, M. A.* On Lottery Tickets and Minimal Task Representations in Deep Reinforcement Learning / M. A. Vischer, R. T. Lange, H. Sprekeler // arXiv preprint arXiv:2105.01648. — 2021.

99. Implicit under-parameterization inhibits data-efficient deep reinforcement learning / A. Kumar [et al.] // arXiv preprint arXiv:2010.14498. — 2020.

100. Deep reinforcement learning with plasticity injection / E. Nikishin [et al.] // Advances in Neural Information Processing Systems. — 2024. — Vol. 36.

101. *Livne, D.* Pops: Policy pruning and shrinking for deep reinforcement learning / D. Livne, K. Cohen // IEEE Journal of Selected Topics in Signal Processing. — 2020. — Vol. 14, no. 4. — P. 789—801.

102. *Chen, J.* Design of digital video coding systems: a complete compressed domain approach / J. Chen, U.-V. Koc, K. R. Liu. — CRC Press, 2001.

103. Asynchronous spiking neurons, the natural key to exploit temporal sparsity / A. Yousefzadeh [et al.] // IEEE Journal on Emerging and Selected Topics in Circuits and Systems. — 2019. — Vol. 9, no. 4. — P. 668—678.

104. Delta networks for optimized recurrent network computation / D. Neil [et al.] // International conference on machine learning. — PMLR. 2017. — P. 2584—2593.

105. Explaining how a deep neural network trained with end-to-end learning steers a car / M. Bojarski [et al.] // arXiv preprint arXiv:1704.07911. — 2017.

106. Stable-baselines3: Reliable reinforcement learning implementations / A. Raffin [et al.] // Journal of Machine Learning Research. — 2021. — Vol. 22, no. 268. — P. 1—8.

107. Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community / V. V. Voevodin [et al.] // Supercomputing Frontiers and Innovations. — 2019. — Vol. 6, no. 2. — P. 4—11.

108. *Todorov, E.* Mujoco: A physics engine for model-based control / E. Todorov, T. Erez, Y. Tassa // 2012 IEEE/RSJ international conference on intelligent robots and systems. — IEEE. 2012. — P. 5026—5033.

109. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures / L. Espeholt [et al.] // International conference on machine learning. — PMLR. 2018. — P. 1407—1416.

110. NeuronFlow: A Hybrid Neuromorphic–Dataflow Processor Architecture for AI Workloads / O. Moreira [et al.] // 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). — IEEE. 2020. — P. 01—05.

Приложение А

Таблицы 5, 6, 7, 8, 9, 10 содержат систематизированную информацию о количестве значимых операций умножения, требуемых для инференса нейросетей в различных средах, с детальной разбивкой по слоям.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	183500	131072	104038	77004	58163	44236	31948	24576
Conv2d-2	2654208	331112	261439	230916	180486	150626	118112	88915	72990
Conv2d-3	1806144	475467	419025	358068	315172	255569	212673	170229	144943
Dense-1	1605632	134873	157753	193478	203513	237633	305070	303063	364478
Dense-2	2048	175	180	205	204	227	222	226	205
Total NonZero	9344832	1125127	969469	886705	776379	702218	680313	594381	607192
NonZero frac	1.000	0.120	0.104	0.095	0.083	0.075	0.073	0.064	0.065

Таблица 5 — Число значимых операций умножения в Enduro.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	117964	101580	86016	72089	59801	52428	44236	37683
Conv2d-2	2654208	199729	187121	120102	89579	68345	59719	37822	29859
Conv2d-3	1806144	405027	332330	174744	118753	114690	116496	67730	50120
Dense-1	1605632	478478	455196	264126	188260	292225	415457	283795	262119
Dense-2	2048	553	529	259	175	279	431	281	278
Total NonZero	9344832	1201751	1076756	645247	468856	535340	644531	433864	380059
NonZero frac	1.000	0.129	0.115	0.069	0.050	0.057	0.069	0.046	0.041

Таблица 6 — Число значимых операций умножения в Freeway.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	49971	42598	36044	29491	26214	22118	18841	16384
Conv2d-2	2654208	204374	168542	130719	109486	90243	72990	56401	45785
Conv2d-3	1806144	271824	228928	187387	163004	129590	103853	83082	69536
Dense-1	1605632	152133	152133	159358	167788	185450	187457	200302	222781
Dense-2	2048	394	402	461	452	485	510	477	513
Total NonZero	9344832	678696	592603	513969	470221	431982	386928	359103	354999
NonZero frac	1.000	0.073	0.063	0.055	0.050	0.046	0.041	0.038	0.038

Таблица 7 — Число значимых операций умножения в SpaceInvaders.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	299827	232652	180224	136806	103219	78643	63897	50790
Conv2d-2	2654208	368271	305233	272719	242196	223617	188448	177168	159252
Conv2d-3	1806144	185581	195515	201836	199127	179711	149458	129139	107465
Dense-1	1605632	37732	47767	77070	99549	134471	171802	220372	238436
Dense-2	2048	488	489	494	470	450	461	447	430
Total NonZero	9344832	891899	781656	732343	678148	641468	588812	591023	556373
NonZero frac	1.000	0.095	0.084	0.078	0.073	0.069	0.063	0.063	0.060

Таблица 8 — Число значимых операций умножения в Krull.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	22118	18022	17203	15564	11468	12288	9830	8192
Conv2d-2	2654208	38486	41803	39813	30523	33841	25878	26542	15261
Conv2d-3	1806144	64118	72697	62311	48314	54184	37929	39735	22576
Dense-1	1605632	39337	56598	75063	73457	100352	85901	126443	91521
Dense-2	2048	114	184	223	201	369	349	492	415
Total NonZero	9344832	164173	189304	194613	168059	200214	162345	203042	137965
NonZero frac	1.000	0.018	0.020	0.021	0.018	0.021	0.017	0.022	0.015

Таблица 9 — Число значимых операций умножения в Breakout.

Степень разреженности	Без оптимизаций	0%	20%	36%	49%	60%	68%	74%	80%
Conv2d-1	3276800	497254	367820	232652	261324	244121	207257	154828	144998
Conv2d-2	2654208	439271	299925	187785	184467	154607	120766	86925	74981
Conv2d-3	1806144	408188	277243	178356	167519	141782	106562	80373	66827
Dense-1	1605632	374112	335175	273760	307478	291020	270147	260513	259710
Dense-2	2048	252	255	243	270	306	303	285	292
Total NonZero	9344832	1719077	1280418	872796	921058	831836	705035	582924	546808
NonZero frac	1.000	0.184	0.137	0.093	0.099	0.089	0.075	0.062	0.059

Таблица 10 — Число значимых операций умножения в Robotank.