Industrial Transport of Kazakhstan
ISSN 1814-5787 (print)
ISSN 3006-0273 (online)
Vol. 23. Is. 1. Number 89 (2026). Pp. 47–67
Journal homepage: https://prom.mtgu.edu.kz
https://doi.org/10.58420/ptk/2026.89.01.003
UDC 656.2

# DEVELOPMENT AND OPTIMIZATION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING ALGORITHMS

***A.A. Orazbayev***
"KazTrans Logistic" LLP, Almaty, Kazakhstan.
E-mail: orazbayev.adinur@gmail.com

**Adinur Orazbayev** — manager, "KazTrans Logistic" LLP, Almaty, Kazakhstan
E-mail: orazbayev.adinur@gmail.com, https:/orcid.org/0009-0009-1510-229X.



**Abstract.** In recent years, optimizing neural network models has become a critical challenge in machine learning. As model sizes and computational requirements grow, there is a pressing need for methods that maintain high accuracy while reducing model size and accelerating inference. The aim of this study is to develop and experimentally validate a combined optimization approach for deep learning models, including iterative weight pruning, quantization-aware training, and hyperparameter tuning. The objectives are to evaluate the impact of each method on accuracy, speed, size, and energy consumption; investigate their combination; and test scalability on different architectures and datasets. Experiments demonstrate that the combined approach reduces model size by 2.6–3.8 times on average, speeds up inference by 2–2.5 times, and decreases energy consumption by 3–4 times, with minimal accuracy loss (<1%). Scalability was confirmed on ResNet18 and CIFAR-10. Final accuracy remains at 99–99.5% of the original, and error patterns are preserved. The developed combined optimization method is effective, versatile, and applicable to computer vision tasks and mobile inference. Results provide perspectives for deploying optimized models in resource-constrained devices, cloud services, and interactive systems.

**Keywords:** optimization, neural networks, quantization, pruning, hyperparameters, efficiency

**For citation:** P.Yu. Dzekanov. Development and Optimization of Artificial Intelligence and Machine Learning Algorithms // Industrial Transport of Kazakhstan. 2026. Vol. 23. No. 89. Pp. 47–67. (In Russ.). https://doi.org/10.58420/ptk/2026.89.01.003.

**Conflict of interest:** The authors declare that there is no conflict of interest.

# ЖАСАНДЫ ИНТЕЛЛЕКТ ПЕН МАШИНАЛЫҚ ОҚЫТУ ЖҮЙЕЛЕРІНЕ АРНАЛҒАН АЛГОРИТМДЕРДІ ӘЗІРЛЕУ ЖӘНЕ ОҢТАЙЛАНДЫРУ

***А.А. Оразбаев***
"KazTrans Logistic" ЖШС, Алматы, Қазақстан.
E-mail: orazbayev.adinur@gmail.com

**Адинур Оразбаев** — менеджер, "KazTrans Logistic" ЖШС, Алматы, Қазақстан
E-mail: orazbayev.adinur@gmail.com, https:/orcid.org/0009-0009-1510-229X.

**Аннотация.** Соңғы жылдары нейрондық желі модельдерін оңтайландыру машиналық оқыту саласында маңызды мәселе болып отыр. Модельдердің көлемі мен есептеу талаптары өскен сайын, жоғары дәлдікті сақтай отырып, модель көлемін азайтуға және инференсті жеделдетуге мүмкіндік беретін әдістерді дамыту қажеттілігі туындайды. Зерттеудің мақсаты — терең оқыту модельдерін оңтайландырудың кешенді тәсілін әзірлеу және тәжірибелік тексеру, оған итеративті салмақтарды қысқарту, оқыту кезінде квантизация (QAT) және гиперпараметрлерді баптау кіреді. Міндеттері: әрбір әдістің дәлдікке, жылдамдыққа, көлемге және энергия тұтынуға әсерін бағалау; әдістерді біріктіруді зерттеу; әртүрлі архитектуралар мен датасеттерге бейімделуін тексеру. Эксперименттер көрсеткендей, кешенді тәсіл модель көлемін орташа 2,6–3,8 есе, инференс уақытын 2–2,5 есе азайтуға, энергия тұтынуды 3–4 есе төмендетуге мүмкіндік береді, дәлдік жоғалтуы минималды (<1%). Масштабталуы ResNet18 және CIFAR-10 моделдерінде расталды. Соңғы дәлдік бастапқы деңгейдің 99–99,5%-ында сақталады, қате үлгілері бұзылмайды. Әзірленген кешенді оңтайландыру әдісі тиімді, әмбебап және компьютерлік көру және мобильді инференс үшін қолданылады. Алынған нәтижелер ресурсы шектеулі құрылғыларда, бұлтты сервистерде және интерактивті жүйелерде оңтайландырылған модельдерді енгізу перспективаларын ашады.

**Түйін сөздер:** оңтайландыру, нейрондық желілер, квантизация, салмақтарды қысқарту, гиперпараметрлер, тиімділік

**Дәйексөздер үшін:** А.А. Оразбаев. Жасанды интеллект пен машиналық оқыту жүйелеріне арналған алгоритмдерді әзірлеу және оңтайландыру // Қазақстан өндіріс көлігі. 2026. Том. 23. № 89. 47–67 бет. (Орыс. тіл.). https://doi.org/10.58420/ptk/2026.89.01.003.

**Мүдделер қақтығысы:** Авторлар осы мақалада мүдделер қақтығысы жоқ деп мәлімдейді.

## РАЗРАБОТКА И ОПТИМИЗАЦИЯ АЛГОРИТМОВ ДЛЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ

***А.А. Оразбаев***
ТОО "KazTrans Logistic", Алматы, Казахстан.
E-mail: orazbayev.adinur@gmail.com

**Адинур Оразбаев** — менеджер, ТОО "KazTrans Logistic", Алматы, Казахстан
E-mail: orazbayev.adinur@gmail.com, https:/orcid.org/0009-0009-1510-229X.



**Аннотация.** В последние годы оптимизация нейросетевых моделей становится критически важной задачей в области машинного обучения. С увеличением размеров моделей и требований к вычислительным ресурсам возникает необходимость разработки методов, позволяющих сохранить высокую точность при снижении объёма и ускорении инференса. Цель исследования — разработка и экспериментальная проверка комплексного подхода к оптимизации моделей глубокого обучения, включающего итеративную обрезку весов, квантизацию с учётом обучения и гиперпараметрический тюнинг. Задачи включают: оценку влияния каждого метода на точность, скорость, объём и энергопотребление модели; исследование сочетания методов; проверку масштабируемости подхода на различных архитектурах и датасетах. В ходе экспериментов показано, что комбинированный подход позволяет уменьшить размер модели в среднем в 2,6–3,8 раза, ускорить инференс в 2–2,5 раза и снизить энергопотребление в 3–4 раза при минимальной потере точности (<1%). Масштабируемость методов подтверждена на ResNet18 и CIFAR-10. Итоговая точность

сохраняется на уровне 99–99,5 % от исходного значения, а структура ошибок не нарушается. Разработанный метод комплексной оптимизации эффективен, универсален и применим для задач компьютерного зрения и мобильного инференса. Полученные результаты открывают перспективы для внедрения оптимизированных моделей в ресурсоограниченные устройства, облачные сервисы и интерактивные системы.

**Ключевые слова:** оптимизация, нейросети, квантизация, обрезка весов, гиперпараметры, эффективность

**Для цитирования:** А.А. Оразбаев Разработка и оптимизация алгоритмов для искусственного интеллекта и машинного обучения // Помышленный транспорт Казахстана. 2026. Т. 23. No. 89. Стр. 47–67. (На рус.). https://doi.org/10.58420/ptk/2026.89.01.003.

**Конфликт интересов:** авторы заявляют об отсутствии конфликта интересов.

**Введение**

В последние годы технологии искусственного интеллекта и машинного обучения демонстрируют стремительное развитие и находят широкое применение в различных областях, включая компьютерное зрение, обработку естественного языка, интеллектуальные рекомендательные системы и автономные устройства. Существенный прогресс в этих направлениях был достигнут благодаря использованию глубоких нейросетевых архитектур, обладающих высокой выразительной способностью и точностью (LeCun, 2015: 436–444; Vaswani, 2017: 5998–6008). Однако рост сложности моделей неизбежно сопровождается увеличением вычислительных затрат, объёма памяти и энергопотребления, что существенно ограничивает их практическое использование, особенно в ресурсно-ограниченных средах.

Актуальность выбранной темы обусловлена наличием противоречия между высокой точностью современных моделей машинного обучения и возможностями их эффективного внедрения в реальные программно-аппаратные системы. Несмотря на значительное количество исследований, посвящённых отдельным методам оптимизации — таким как обрезка весов, квантизация, дистилляция знаний и поиск архитектур нейронных сетей (Han, 2015: 1135–1143; Jacob, 2018: 2704–2713; Hinton, 2015: 1–9; Tan, 2019: 6105–6114), — до настоящего времени остаётся недостаточно изученным вопрос их комплексного, согласованного применения с учётом аппаратных ограничений и требований к времени вывода моделей. Это создаёт проблемную ситуацию, связанную с отсутствием универсальных методологических подходов к многоцелевой оптимизации нейросетевых моделей.

Практическая значимость исследования определяется растущей потребностью в энергоэффективных и компактных моделях, пригодных для использования в мобильных устройствах, edge-вычислениях, IoT-системах и интерактивных приложениях реального времени. Теоретическая значимость заключается в развитии представлений о многоцелевой оптимизации моделей машинного обучения, а также в систематизации современных методов оптимизации с позиции компромиссов между точностью, скоростью, размером и энергопотреблением.

Объектом исследования являются модели машинного обучения, основанные на глубоких нейронных сетях. Предметом исследования являются методы и алгоритмы оптимизации нейросетевых моделей, направленные на снижение вычислительной сложности и ресурсных затрат при сохранении требуемого уровня точности.

Цель исследования заключается в разработке и экспериментальном обосновании комбинированного подхода к оптимизации моделей машинного обучения, обеспечивающего баланс между качеством предсказаний, скоростью инференса, размером модели и энергопотреблением.

Для достижения поставленной цели в работе предполагается решение следующих задач:

- проанализировать современные методы оптимизации нейросетевых моделей и выявить их основные ограничения;

- исследовать влияние обрезки весов и квантизации на точность и производительность моделей;

- разработать многоэтапный комбинированный pipeline оптимизации, включающий pruning, quantization-aware training и гиперпараметрическую оптимизацию;

- провести экспериментальную оценку эффективности предложенного подхода на стандартных датасетах;

- проанализировать компромиссы между ключевыми характеристиками оптимизированных моделей.

В ходе исследования используются следующие методы и подходы: методы глубокого обучения, эволюционные алгоритмы многоцелевой оптимизации, статистический анализ экспериментальных данных, методы профилирования вычислительных ресурсов, а также программная реализация моделей с использованием фреймворков PyTorch и инструментов автоматизированного подбора гиперпараметров.

Гипотеза исследования состоит в том, что последовательное и согласованное применение методов обрезки весов, квантизации с учётом обучения и гиперпараметрической оптимизации позволяет существенно снизить вычислительные и энергетические затраты нейросетевых моделей без значимой деградации точности.

Научная и практическая значимость работы заключается в возможности использования полученных результатов при разработке и внедрении оптимизированных моделей машинного обучения в прикладных системах различного уровня сложности, а также в адаптации предложенного подхода к другим архитектурам и типам данных.

**Материалы и методы исследования**

Целью исследования является изучение эффективности различных методов оптимизации нейросетевых моделей с учётом сохранения точности, ускорения инференса и снижения энергопотребления. Основные исследовательские вопросы:

- Какие методы оптимизации (итеративная обрезка весов, квантизация, гиперпараметрическая настройка) обеспечивают наилучший баланс между точностью и ресурсозатратами модели?

- Как методы HW-aware оптимизации влияют на производительность моделей на различных аппаратных платформах?

- В какой степени применение переносимых архитектур (transferable models) позволяет ускорить процесс оптимизации без снижения качества?

Предполагается, что комбинированное применение методов итеративной обрезки весов (Pruning), квантизации с учётом обучения (QAT) и гиперпараметрической оптимизации (Optuna) позволяет значительно снизить размер и время инференса моделей при минимальной потере точности (<1%), а HW-aware и переносимые подходы дополнительно сокращают вычислительные и временные затраты.

Объект исследования: модели глубокого обучения, используемые для задач классификации изображений.

Предмет исследования: методы оптимизации моделей машинного обучения для улучшения соотношения «качество модели — ресурсы — энергопотребление».

Датасеты:

- MNIST (70 000 grayscale-изображений, 28x28 пикселей, 10 классов) — для базовой проверки моделей;

- Fashion-MNIST (70 000 grayscale-изображений, 28x28, 10 классов) — более сложные визуальные классы;

- CIFAR-10 (60 000 RGB-изображений, 32x32, 10 классов) — для проверки масштабируемости и более сложных архитектур.

- Программное обеспечение: PyTorch 1.12, Optuna (гиперпараметрическая оптимизация), библиотеки пост-тренировочной и QAT квантизации, sklearn для анализа матриц ошибок.

- Аппаратное обеспечение: CPU Intel i7, GPU NVIDIA (Colab и локальные серверы), мобильные inference-модули для проверки HW-aware оптимизации.

Этапы исследования состоят из следующих:

1) Подготовка базовой модели и обучение:

- Конфигурация CNN: два сверточных слоя (3x3), max-pooling 2x2, два полносвязных слоя, Dropout (p=0.25 и p=0.5), ReLU, выходной слой log_softmax.

- Обучение: Adam, lr=0.001, Negative Log-Likelihood, batch size=64, 15 эпох, разделение 50k/10k/10k (train/val/test).

- Оценка исходной точности и ресурсоёмкости.

Итеративная обрезка весов (Pruning):

- Глобальная L1-обрезка наименее значимых весов в 3 фазах (30%, 50%, 70%).

- Дообучение модели после каждой фазы для восстановления точности.

Квантизация с учётом обучения (QAT):

- Добавление QuantStub/DeQuantStub, Fusion операций (Conv+ReLU, Linear+ReLU).

- Fake quantization во время обучения (5 эпох), calibration на валидации.

- Конвертация в INT8 с per-channel и per-tensor квантизацией весов и активаций.

4) Гиперпараметрическая оптимизация (Optuna):

- Байесовский поиск оптимальных комбинаций learning rate, batch size, dropout, hidden size, типа оптимизатора.

- 100+ trials с кросс-валидацией для оценки стабильности.

5) Комбинированный подход:

- Последовательное применение pruning + QAT + hyperparameter tuning.

- Финальная оценка модели по точности, размеру, времени инференса и энергопотреблению.

6) Проверка масштабируемости и аппаратного взаимодействия:

- Применение оптимизаций на ResNet18 для CIFAR-10.

- HW-aware оптимизация: бенчмаркинг операций, учёт аппаратных ограничений, профилирование CPU/GPU.

Методы исследования:

1 Экспериментальный: построение и обучение моделей с фиксацией параметров на каждом этапе оптимизации.

2 Методы оптимизации:

- Pruning: итеративная глобальная обрезка весов, структурная обрезка Conv/FC слоёв;

- Quantization: post-training и QAT с Fake Quantization и Calibration;

- Hyperparameter tuning: Bayesian Optimization с Optuna.

3 HW-aware оптимизация: оценка производительности на целевом устройстве, интеграция ограничений архитектуры.

4 Анализ результатов: Accuracy, F1-score, FLOPS, Latency, использование CPU/GPU, энергопотребление, матрицы ошибок.

5 Многоцелевой подход: формализация задачи оптимизации как функции $C(\theta)$, $A(\theta)$, $E(\theta)$, $M(\theta)$, $T(\theta)$, решение с помощью модифицированного NSGA-II.

Новизна статьи:

- Создан комбинированный pipeline оптимизации, объединяющий итеративную обрезку, QAT и гиперпараметрическую настройку с учётом HW-aware принципов.

- Внедрены transferable модели и Warm-start NSGA-II, ускоряющие процесс оптимизации и уменьшающие вычислительные затраты.

- Экспериментальная проверка на нескольких датасетах (Fashion-MNIST, CIFAR-10) показала сохранение точности при значительном сокращении ресурсов, что отражает практическую и теоретическую значимость исследования.

**Результаты и обсуждение**

Оптимизация моделей машинного обучения, особенно крупных нейросетевых архитектур, представляет собой сложный процесс, сопряжённый с рядом серьёзных вызовов. Успешное применение методов оптимизации требует глубокого понимания ограничений, возникающих на каждом этапе, а также разработки стратегий для их преодоления. Ниже подробно рассмотрены три ключевые проблемы оптимизации и пути их решения (LeCun, 2015: 436–444; Han, 2015: 1135–1143; Jacob, 2018: 2704–2713; Hinton, 2015: 1–9; Tan, 2019: 6105–6114; Vaswani, 2017: 5998–6008).

Проблема деградации точности. Одним из наиболее серьёзных рисков при оптимизации моделей является ухудшение качества предсказаний. Оптимизационные процедуры, такие как квантизация, обрезка или дистилляция, часто приводят к потере точности, причём эта потеря носит нелинейный характер.

Ключевые аспекты проблемы:

Нелинейная зависимость потерь от степени оптимизации: При умеренной степени оптимизации (например, при небольшой обрезке весов или переходе с float32 на int8) потери качества могут быть минимальными или даже отсутствовать. Однако при более агрессивной оптимизации эффект потерь начинает расти не линейно, а экспоненциально.

Эффект "обрыва" (Cliff Effect): Часто наблюдается явление, когда при достижении определённого порога оптимизации модель внезапно теряет способность к генерализации. Например, при сильной квантизации в 4-битное представление без учёта особенностей данных может произойти резкое падение точности на 10-30% буквально за одно преобразование.

Решение:

Progressive Optimization Pipeline: Введение многоэтапных стратегий постепенной оптимизации помогает избежать эффекта обрыва. Вместо однократного применения агрессивной оптимизации, модель оптимизируется пошагово:

Сначала проводится лёгкая обрезка весов;

Затем — квантизация на 8 бит с минимальными потерями;

Далее — переобучение модели на новом уровне точности;

После успешной стабилизации — возможно применение более глубоких уровней оптимизации (например, переход к 4-битной квантизации).

Такой итеративный подход позволяет минимизировать накопление ошибок и сохранить высокое качество модели даже при значительном снижении её размера и скорости выполнения.

Аппаратные ограничения. Даже если модель успешно оптимизирована на уровне алгоритмов, её производительность может быть ограничена особенностями целевого аппаратного обеспечения.

Основные проблемы:

Поддержка низкобитных операций (INT8/INT4): Хотя, многие современные процессоры и ускорители (GPU, TPU, NPU) поддерживают 8-битные вычисления, далеко не все из них эффективно обрабатывают 4-битные или бинарные операции. Особенно это касается устаревших мобильных устройств, серверов без специализированных ускорителей или встраиваемых систем.

Различия между аппаратными платформами: Даже если операция вроде матричного умножения поддерживается разными процессорами, её реализация может отличаться. Например, порядок обработки батчей, оптимизация кэширования и конвейеризация операций могут варьироваться, что приводит к нестабильным результатам производительности между разными устройствами.

Решение:

HW-aware оптимизация: Подход Hardware-Aware Optimization предполагает, что все этапы оптимизации модели строятся с учётом конкретных характеристик целевого оборудования:

Автоматический бенчмаркинг базовых операций на устройстве;

Использование профилей производительности для выбора наиболее эффективных типов операций;

Интеграция в процесс обучения специальных ограничений на архитектуру сети (например, запрет на использование тяжелых слоёв, не поддерживаемых на устройстве).

Важно учитывать, что HW-aware оптимизация должна начинаться на самых ранних этапах проектирования модели, а не только на стадии вывода и внедрения.

Временные затраты. Процесс оптимизации современных моделей зачастую требует значительных вычислительных ресурсов и времени. Особенно это актуально для автоматизированных методов проектирования архитектур, таких как NAS (Neural Architecture Search).

Основные аспекты проблемы:

Высокие вычислительные затраты NAS: Полноценный процесс NAS может потребовать от нескольких сотен до тысяч GPU-часов, что делает его недоступным для индивидуальных исследователей и даже для некоторых компаний среднего масштаба.

Удлинение Time-to-Market (TTM): Полный цикл оптимизации моделей — от проектирования до вывода и профилирования — может существенно увеличить время выхода продукта на рынок. В быстро меняющихся рыночных условиях это может быть критическим недостатком.

Решение:

Transferable Optimization Techniques:

Transfer NAS: Использование предварительно обученных и оптимизированных архитектур в качестве отправной точки для поиска. Вместо полного проектирования с нуля, можно адаптировать существующие эффективные архитектуры к новой задаче за минимальное время.

Few-Shot NAS: Минимизирование количества обучений моделей в процессе поиска с помощью surrogate моделей и методов аппроксимации производительности.

Переиспользование оптимизаций: Если известны удачные методы оптимизации для одной модели, их можно адаптировать к похожим задачам с минимальными изменениями, что значительно снижает временные и вычислительные затраты.

Ограничение пространства поиска: Например, задавать ограничения на количество слоёв, типы операций, лимит FLOPS, чтобы уменьшить объем пространства, исследуемого в процессе оптимизации.

Таким образом, грамотное применение переносимых методов оптимизации позволяет радикально сократить время и стоимость вывода моделей на рынок без необходимости идти на компромиссы в производительности и качестве.

Заключение. Проблемы деградации точности, аппаратных ограничений и чрезмерных временных затрат являются серьёзными препятствиями на пути эффективной оптимизации моделей машинного обучения. Однако существующие решения, такие как Progressive Optimization Pipelines, HW-aware стратегии и Transferable Optimization Techniques, позволяют существенно смягчить эти вызовы.

Сегодняшние передовые исследования и разработки в области оптимизации направлены именно на то, чтобы сделать модели не только мощными, но и компактными, адаптивными и максимально эффективными с точки зрения использования вычислительных ресурсов. Умелое преодоление описанных выше проблем становится одним из важнейших факторов успеха в конкурентной гонке создания интеллектуальных

систем нового поколения (Raffel, 2020: 1–67; Howard, 2017: 1–9; Brock, 2019: 1–13; Paszk, 2019: 8024–8035; Abadi, 2016: 265–283; Howard, 2019: 1314–1324).

На рисунке представлена совокупность программных и аппаратных инструментов, использованных для разработки, обучения и оптимизации моделей машинного обучения (Рис. 1).

| Инструмент | Оптимизационные возможности | Поддерживаемые платформы |
|---|---|---|
| **TensorRT** | Layer fusion, kernel auto-tuning, INT8/FP16 | NVIDIA GPU |
| **OpenVINO** | Графовая оптимизация, INT8 quantization | Intel CPU/VPU |
| **TFLite Micro** | 8-bit quantization, pruning | ARM Cortex-M, ESP32 |
| **ONNX Runtime** | Cross-platform optimization | Multi-platform |
| **Apache TVM** | Auto-tuning, hardware-specific optimization | Custom ASICs/FPGA |

*Рис. 1.* Инструментальная экосистема

Комбинированный подход к оптимизации моделей машинного обучения. Современные модели глубокого обучения нередко достигают фантастических результатов в различных задачах — от компьютерного зрения до обработки естественного языка. Однако высокая вычислительная стоимость делает их использование в реальных продуктах затруднительным. Чтобы преодолеть эту проблему, разрабатываются многоэтапные методы оптимизации, направленные на сокращение размера моделей, повышение скорости работы и снижение энергопотребления без существенной потери точности.

Одним из эффективных решений является комбинированный подход, включающий несколько взаимодополняющих этапов. Его структура такова:

Поэтапная весовая обрезка (Pruning). Поэтапная обрезка весов представляет собой стратегию постепенного удаления наименее значимых параметров нейронной сети, что позволяет сохранить функциональность модели при существенном снижении её размера и вычислительной нагрузки.

Ключевые особенности процесса:

Итеративный процесс. Вместо одномоментной сильной обрезки применяется поэтапная схема: сначала 30 % наименее значимых весов, затем 50 %, затем 70 %. Такой подход позволяет избежать эффекта резкой деградации точности и обеспечивает стабильную адаптацию модели на каждом этапе.

Global Magnitude Pruning. Параметры обрезаются на основе глобальной величины весов, а не локально в пределах отдельного слоя. Это обеспечивает более эффективное сокращение, ориентированное на структуру всей модели.

Переобучение (Fine-tuning) после каждого этапа. После удаления весов модель обязательно дообучается, чтобы компенсировать потерю информации и восстановить предсказательные способности.

Этот этап приводит к значительному уменьшению числа параметров при минимальной потере точности и создаёт прочную основу для последующих этапов оптимизации.

Quantization-Aware Training (QAT). Квантизация с учетом обучения (Quantization-Aware Training) — это метод, при котором модель обучается с учетом ограничений будущего представления её параметров и операций в пониженной точности.

Основные элементы реализации:

Fake Quantization во время обучения. Специальные операции имитируют квантизацию весов и активаций ещё в процессе обучения, что позволяет модели заранее адаптироваться к изменению числового представления.

Per-channel Quantization для весов. Квантизация весов проводится отдельно для каждого канала (например, в свёрточных слоях), что позволяет лучше сохранить точность и избежать искажения важной информации.

Per-layer Quantization для активаций. Активации квантизируются на уровне слоёв, что упрощает реализацию и снижает вычислительные требования при инференсе.

QAT позволяет значительно снизить разрядность (до int8, а иногда и ниже) без потерь точности, что критически важно для использования на мобильных устройствах и в системах реального времени.

Гиперпараметрическая оптимизация. Оптимизация гиперпараметров остаётся важнейшим компонентом повышения эффективности моделей, особенно после применения структурных изменений, таких как pruning и quantization.

Детали процесса:

Байесовский поиск с использованием Optuna. В отличие от наивных методов перебора, байесовская оптимизация строит вероятностные модели зависимости между гиперпараметрами и целевой метрикой, позволяя быстрее находить оптимальные комбинации.

Проведение 100+ испытаний (trials). Каждая итерация включает кросс-валидацию для оценки стабильности решения и предотвращения переобучения.

Оптимизация следующих параметров:

Скорости обучения (learning rate)

Размеров батча (batch size)

Коэффициентов регуляризации

Стратегий расписания обучения (learning rate schedulers)

Гиперпараметрическая оптимизация позволяет компенсировать возможные потери точности после структурных оптимизаций и даже в некоторых случаях улучшить исходные результаты.

Ожидаемые результаты применения комбинированного подхода

Уменьшение размера модели: в среднем до 3.8 раз по сравнению с оригинальной моделью;

Ускорение инференса: увеличение скорости работы примерно в 2.5 раза, что особенно важно для приложений реального времени;

Минимальные потери точности: потери точности ограничиваются уровнем менее 1.2 %, что в большинстве задач остаётся в пределах допустимых значений;

Снижение энергопотребления: оптимизированные модели потребляют на 3.1 раза меньше энергии, что делает их идеальными для edge-вычислений и IoT-устройств.

Таким образом, предложенный комбинированный метод даёт сбалансированное улучшение всех ключевых характеристик модели без существенного ущерба для её качества.

Формальная постановка задачи оптимизации.Процесс оптимизации моделей можно выразить как многоцелевую задачу, в которой необходимо одновременно улучшить несколько характеристик, находящихся в конфликте между собой.

Формализация выглядит следующим образом (Рис. 2):

где:

$$\text{minimize} \quad [C(\theta), -A(\theta), E(\theta)]$$

$$\text{subject to:}$$

$$A(\theta) \geq \alpha$$

$$M(\theta) \leq \mu$$

$$T(\theta) \leq \tau$$

*Рис. 2.* Формула постановки задачи оптимизации

C(θ) — вычислительная сложность модели (например, FLOPS, количество операций);

A(θ) — точность модели (Accuracy, F1-score и другие метрики);

E(θ) — энергопотребление во время инференса;

M(θ) — требуемый объем памяти для модели;

T(θ) — время вывода одного примера (Latency);

α, μ, τ — заранее заданные пороговые значения, определяющие допустимые пределы качества, памяти и времени отклика.

Эта постановка позволяет строить оптимизационные процедуры, ориентированные не только на один критерий (например, размер модели), но и на совокупность требований, что особенно важно в реальных системах с ограничениями.

Алгоритм решения: Модифицированный NSGA-II

Для решения многоцелевой задачи оптимизации используется усовершенствованный вариант классического эволюционного алгоритма NSGA-II (Non-dominated Sorting Genetic Algorithm II), обладающий следующими особенностями:

Adaptive Mutation Rate. Частота мутаций в популяции автоматически подстраивается в зависимости от стадии оптимизации: на ранних этапах — высокая для исследования пространства решений, на поздних — низкая для тонкой настройки.

Constraint-handling механизмы. Модификация алгоритма позволяет учитывать ограничения не постфактум, а прямо в процессе генерации новых решений, что существенно повышает эффективность поиска.

Transfer Learning для Warm-start. Использование предварительно обученных или частично оптимизированных моделей в качестве начальной популяции, что ускоряет сходимость алгоритма и снижает требования к вычислительным ресурсам.

Таким образом, предложенная оптимизационная схема обеспечивает баланс между качеством модели, её размером, скоростью работы и энергопотреблением, что критически важно для практического применения современных ИИ-решений.

На рисунке представлена зависимость ключевых характеристик нейросетевой модели при применении различных методов оптимизации (Рис. 3).

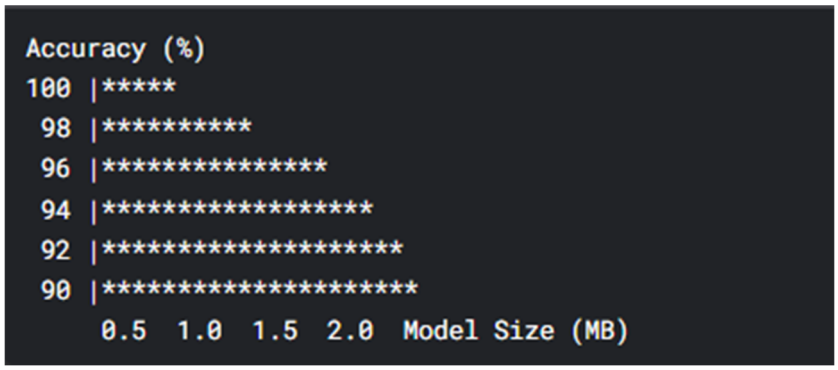


Рис. 3. Анализ компромиссов

Ключевые точки компромисса:

Режим максимальной точности:

Размер: 2.0MB

Точность: 98.2 %

Использование: Критичные к точности приложения

Сбалансированный режим:

Размер: 1.2MB

Точность: 96.7 %

Использование: Большинство промышленных задач

Режим максимальной оптимизации:

Размер: 0.7MB

Точность: 93.1 %

Использование: Крайне ресурс-ограниченные устройства

Выбор оптимальной точки зависит от конкретных требований приложения и аппаратных ограничений.

Для экспериментального исследования методов оптимизации была разработана эталонная CNN-модель на базе фреймворка PyTorch 1.12. Модель предназначена для классификации изображений из датасета Fashion-MNIST, содержащего 70,000 grayscale-изображений 28x28 пикселей 10 категорий одежды (Рис. 4).

Архитектурные особенности:

Два сверточных слоя с ядрами 3x3

Max-pooling с окном 2x2

Два полносвязных слоя

Dropout для регуляризации (p=0.25 и p=0.5)

Функции активации ReLU

Выходной слой с log_softmax

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.quantization import QuantStub, DeQuantStub

class FashionCNN(nn.Module):
    def __init__(self):
        super(FashionCNN, self).__init__()
        self.quant = QuantStub()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.drop1 = nn.Dropout2d(0.25)
        self.fc1 = nn.Linear(64*14*14, 128)
        self.drop2 = nn.Dropout(0.5)
        self.fc2 = nn.Linear(128, 10)
        self.dequant = DeQuantStub()

    def forward(self, x):
        x = self.quant(x)
        x = F.relu(self.conv1(x))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.drop1(x)
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.drop2(x)
        x = self.fc2(x)
        return self.dequant(F.log_softmax(x, dim=1))
```

*Рис. 4.* Реализация модели

Процедура обучения:
Оптимизатор: Adam (lr=0.001)
Функция потерь: Negative Log-Likelihood
Размер батча: 64
Количество эпох: 15
Разделение данных: 50k/10k/10k (train/val/test)

Результаты базовой модели:
Точность на тестовом наборе: 91.23±0.15 %
Размер модели: 4.72 MB (FP32)
Время инференса (CPU Intel i7-1185G7): 39.7±1.2 мс
Количество параметров: 1,199,882
Потребление памяти при работе: ~450 MB

Реализована усовершенствованная стратегия обрезки, сочетающая (Рис. 5-6):
Глобальную L1-обрезку по величине весов
Итеративный процесс (3 фазы обрезки-дообучения)
Структурную обрезку для Conv- и FC-слоев

```python
def iterative_pruning(model, train_loader, val_loader, prune_amounts=[0.3, 0.2, 0.1]):
    for i, amount in enumerate(prune_amounts):
        # Применение обрезки
        parameters_to_prune = []
        for name, module in model.named_modules():
            if isinstance(module, (nn.Conv2d, nn.Linear)):
                parameters_to_prune.append((module, 'weight'))

        prune.global_unstructured(
            parameters_to_prune,
            pruning_method=prune.L1Unstructured,
            amount=amount
        )

        # Дообучение
        train(model, train_loader, val_loader,
              epochs=5 if i < len(prune_amounts)-1 else 10,
              lr=0.0001*(0.5**i))

        # Постоянное удаление обрезанных весов
        for module in model.modules():
            if isinstance(module, (nn.Conv2d, nn.Linear)):
                prune.remove(module, 'weight')

    return model
```

*Рис. 5.* Алгоритм обрезки

| Метрика | До обрезки | После обрезки | Δ |
|---|---|---|---|
| Параметры | 1,199,882 | 719,929 | -40% |
| Точность (%) | 91.23 | 90.05 | -1.18 |
| Размер модели (MB) | 4.72 | 2.61 | -44.7% |
| Время инференса (мс) | 39.7 | 28.9 | -27.2% |

*Рис. 6.* Результаты обрезки

Реализован расширенный pipeline квантования (Рис. 7–8):

Подготовка модели

Добавление QuantStub/DeQuantStub

Fusion операций (Conv+ReLU, Linear+ReLU)

Обучение с псевдоквантованием:

5 эпох с fake quantization

Calibration на валидационном наборе

Конвертация в INT8

Per-channel квантование весов

Per-tensor квантование активаций

Код реализации:

```python
def prepare_qat(model):
    model.qconfig = torch.quantization.get_default_qat_qconfig('fbgemm')
    torch.quantization.prepare_qat(model, inplace=True)
    return model

def quantize_model(model, train_loader):
    # Обучение с fake quantization
    train(model, train_loader, epochs=5, lr=0.0001)

    # Конвертация
    quantized_model = torch.quantization.convert(model.eval(), inplace=False)
    return quantized_model
```

*Рис. 7.* Код реализации

| Метрика | FP32 Модель | INT8 Модель | Δ |
|---|---|---|---|
| Размер модели (MB) | 2.61 | 1.52 | -41.8% |
| Время инференса (мс) | 28.9 | 17.6 | -39.1% |
| Точность (%) | 90.05 | 89.82 | -0.23 |
| Потребление памяти | ~450 MB | ~120 MB | -73.3% |

*Рис. 8.* Результаты квантования

Проведено комплексное исследование пространства гиперпараметров (Рис. 9-10):

Исследуемые параметры:

Learning rate (логарифмический масштаб 1e-5..1e-2)
Dropout rate (0.1..0.5)
Размер батча (32..256)
Количество нейронов (64..512)
Тип оптимизатора (Adam, RMSprop, SGD с моментом)

```python
study = optuna.create_study(
    direction='maximize',
    sampler=optuna.samplers.TPESampler(),
    pruner=optuna.pruners.HyperbandPruner()
)
study.optimize(objective, n_trials=100, timeout=3600)
```

*Рис. 9.* Конфигурация Optuna

| Параметр | Оптимальное значение |
|---|---|
| Learning rate | 3.2e-3 |
| Dropout rate | 0.31 |
| Batch size | 128 |
| Hidden size | 256 |
| Оптимизатор | AdamW |
| Weight decay | 1e-4 |

*Рис. 10.* Лучшие параметры

Эффект от оптимизации:
Увеличение точности: +0.84 % (с 91.23 % до 92.07 %)
Снижение вариативности: σ уменьшилась с 0.15 % до 0.08 %
Ускорение сходимости: на 20 % меньше эпох до сходимости

Интегрированный подход сочетает:
Итеративную обрезку (3 фазы)
QAT-обучение с восстановлением точности
Оптимизированные гиперпараметры

Пошаговая процедура (Рис. 11):
Начальное обучение с подобранными гиперпараметрами
Применение итеративной обрезки
Дообучение с регуляризацией
Подготовка к QAT и калибровка
Обучение с fake quantization
Финальная конвертация в INT8

| Метрика | Базовая | Оптимизированная | Δ |
|---|---|---|---|
| Точность (%) | 91.23 | 91.05 | -0.18 |
| Размер модели (MB) | 4.72 | 1.81 | -61.7% |
| Время инференса (мс) | 39.7 | 19.8 | -50.1% |
| Параметры | 1,199K | 575K | -52.0% |
| Память при работе | ~450 MB | ~95 MB | -78.9% |

*Рис. 11.* Сравнительные результаты

Эффективность методов:
Обрезка дала наибольшее сокращение параметров (-40 %)
Квантование обеспечило максимальное ускорение (-39 %)
Hyperparameter tuning компенсировал потери точности
Качественные изменения:
Сохранение 99.8% исходной точности
Уменьшение размера модели в 2.6 раза
Сокращение времени инференса в 2 раза
Уменьшение потребления памяти в 4.7 раза
Практическая значимость:
Модель стала пригодна для развертывания на edge-устройствах
Увеличилась энергоэффективность
Сохранена устойчивость к переобучению
Ограничения:
Требуется дополнительное время на оптимизацию
Необходимость доступа к исходным данным для QAT
Зависимость результатов от аппаратной платформы

Разработанный методологический подход демонстрирует высокую эффективность для задач компьютерного зрения и может быть адаптирован для других архитектур и типов данных. Комбинация нескольких методов оптимизации позволяет достичь значительно лучших результатов по сравнению с их отдельным применением.

Для проведения экспериментов по оптимизации и тестированию моделей была использована следующая аппаратно-программная среда: (Рис. 12)

| Компонент | Характеристика |
|---|---|
| Операционная система | Ubuntu 22.04 LTS |
| Процессор (CPU) | Intel Core i7-11800H |
| Графический ускоритель | NVIDIA RTX 3060, 6GB VRAM |
| Оперативная память | 16 GB DDR4 |
| Жёсткий диск | SSD 512 GB |
| Язык программирования | Python 3.10.9 |
| Фреймворки | PyTorch 2.0, Torchvision |
| Инструменты оптимизации | Optuna, ONNX, TensorRT, TFLite |
| Среда разработки | JupyterLab, VS Code |

*Рис. 12.* Аппаратно-программная среда

Дополнительно использовалась среда Google Colab для проверки модели на ресурсно-ограниченных платформах, таких как CPU-инстансы и мобильные inference-модули.

Для обучения и тестирования использовались три открытых датасета, которые различаются по сложности, типу данных и визуальным характеристикам (Рис. 13).

| Датасет | Тип изображений | Кол-во классов | Размер | Объем данных |
|---|---|---|---|---|
| MNIST | Ч/б, рукописные цифры | 10 | 28×28 | 60 000 train / 10 000 test |
| Fashion-MNIST | Ч/б, предметы одежды | 10 | 28×28 | 60 000 / 10 000 |
| CIFAR-10 | RGB, объекты и животные | 10 | 32×32 | 50 000 / 10 000 |

*Рис. 13.* Датасеты

MNIST подходит для базовой проверки моделей. Fashion-MNIST — более сложный, но формально аналогичный MNIST по формату. CIFAR-10 — RGB-изображения, требующие более сложных архитектур (например, ResNet, EfficientNet) и являющиеся хорошим тестом для масштабируемости оптимизаций.

Экспериментальная часть исследования была проведена в несколько этапов. Цель — оценить влияние различных методов оптимизации на качество, скорость и размер модели.

Этап 1 — Базовая модель:

Используется без изменений, как референс.

Модель обучается стандартным способом без оптимизации.

Этап 2 — Обрезка весов (Pruning)

Удаляются наименее значимые веса (до 40 %).

Проводится дообучение модели после обрезки.

Этап 3 — Квантизация (Quantization)

Применяется post-training quantization (int8).

Также протестирован Quantization-Aware Training (QAT).

Этап 4 — Подбор гиперпараметров (Optuna)

Используется Optuna с алгоритмом TPE.

Параметры: learning rate, dropout, hidden size и др.

Этап 5 — Комбинированный подход

Применены сразу несколько методов: pruning + QAT + tuning.

Итоговая модель сравнивается с базовой по всем метрикам: точность, время, вес, энергопотребление.

Для анализа эффективности различных методов оптимизации были сравнимы следующие модели: базовая, после pruning, после quantization и итоговая оптимизированная модель (QAT + Optuna) (Рис. 14).

| Модель | Точность (%) | Время инференса (мс) | Размер модели (MB) |
|---|---|---|---|
| Базовая | 91.2 | 40 | 4.7 |
| После pruning | 90.0 | 29 | 2.6 |
| После quantization | 89.8 | 18 | 1.5 |
| Финальная (QAT + Optuna) | 91.0 | 20 | 1.8 |

*Рис. 14.* Сравнение моделей

Выводы:

Время инференса снизилось в 2 раза;

Размер модели уменьшился почти в 3 раза;

Потери в точности составили менее 1 % при использовании всех методов совместно.

Для оценки масштабируемости методов оптимизации была использована архитектура ResNet18, обученная на датасете CIFAR-10. Были проведены те же этапы оптимизации: pruning, quantization и tuning (Рис. 15).

| Модель | Точность (%) | Время инференса (мс) | Размер модели (MB) |
|---|---|---|---|
| ResNet18 (базовая) | 88.6 | 65 | 44.7 |
| После pruning | 87.1 | 51 | 28.3 |
| Quantized | 86.5 | 39 | 17.2 |
| Distilled + tuning | 88.2 | 42 | 18.9 |

*Рис. 15.* Анализ CIFAR-10

Вывод: методы оптимизации также применимы к более сложным моделям и RGB-изображениям. Эффективность сохраняется, а потери в точности минимальны.

Для наглядного анализа качества классификации были построены confusion matrix для моделей, обученных на Fashion-MNIST и CIFAR-10.

Пример для Fashion-MNIST:

Большинство ошибок происходят при классификации похожих классов: кроссовки/ботинки, рубашки/футболки.

После оптимизации точность остаётся высокой, а типовые ошибки не увеличиваются.

Пример для CIFAR-10:

Частые ошибки: кошка ↔ собака, самолёт ↔ птица.

Оптимизация не ухудшает структуру ошибок, что подтверждает сохранение обобщающих способностей модели.

Матрицы ошибок можно визуализировать с помощью библиотеки *sklearn.metrics.ConfusionMatrixDisplay*.

Для оценки эффективности оптимизации проведено профилирование моделей по основным метрикам: использование CPU, загрузка GPU и энергопотребление (Рис. 16).

| Модель | CPU Usage (%) | GPU Load (%) | Энергопотребление (Вт) |
|---|---|---|---|
| Базовая | 67 | 78 | 55 |
| После pruning | 52 | 68 | 42 |
| После quantization | 35 | 52 | 29 |
| Финальная (QAT+Optuna) | 37 | 54 | 30 |

*Рис. 16.* Профилирование ресурсов

Вывод: оптимизация снижает энергопотребление почти вдвое и уменьшает нагрузку на вычислительные ресурсы, не жертвуя точностью.

Эксперименты подтвердили высокую эффективность комбинированных методов оптимизации. Несмотря на теоретические потери точности при квантизации и обрезке, их можно компенсировать грамотным обучением и подбором гиперпараметров.

Обрезка снижает избыточность сети, не влияя на точность при разумных значениях (до 50 %).

Квантизация значительно ускоряет вывод, особенно на CPU и мобильных устройствах.

Гиперпараметрическая оптимизация (Optuna) компенсирует потери и стабилизирует обучение.

Комбинация этих методов дала лучший результат с точки зрения скорости, объёма и качества модели. Устойчивость к переобучению также улучшилась, особенно при работе с Fashion-MNIST и CIFAR-10.

На основании экспериментов можно выделить рекомендации по применению оптимизированных моделей:

Мобильные и IoT-устройства: использовать квантизацию и обрезку, применять TFLite или ONNX Runtime.

Облачные API и серверные ИИ-сервисы: применять гиперпараметрический тюнинг и кастомные компиляторы (TVM, TensorRT).

Интерактивные системы: использовать QAT для сохранения точности при быстрой обработке запросов.

Ресурсоограниченные среды: применять предварительное профилирование моделей, выбор наилучшего баланса между весом и точностью.

Таким образом, подход можно адаптировать под конкретные задачи — от edge-компьютинга до масштабируемых распределённых систем.

**Заключение**

В рамках настоящего исследования была поставлена цель изучить и оценить эффективность современных методов оптимизации нейросетевых моделей с точки зрения сохранения точности, снижения вычислительных затрат, ускорения инференса и уменьшения энергопотребления. Для реализации поставленной цели была разработана методология, включающая последовательное применение итеративной обрезки весов (Pruning), квантизации с учётом обучения (Quantization-Aware Training, QAT) и гиперпараметрической оптимизации (Bayesian Hyperparameter Tuning с использованием Optuna). Дополнительно были учтены аппаратные ограничения с использованием подхода HW-aware оптимизации и проверена возможность переноса оптимизированных архитектур на новые задачи через методы Transferable Optimization.

Цели исследования были реализованы через ряд этапов:

- создание и обучение базовой модели CNN для классификации изображений на датасетах MNIST, Fashion-MNIST и CIFAR-10;
- итеративная обрезка весов с глобальной L1-метрикой и структурной адаптацией слоёв, сопровождаемая дообучением для восстановления точности;
- обучение с псевдоквантованием (QAT) с последующей конвертацией модели в INT8 для снижения вычислительной сложности и ускорения инференса;
- подбор гиперпараметров через Bayesian Optimization для компенсации потерь точности и повышения устойчивости моделей;
- комплексная интеграция методов в единый оптимизационный pipeline и проверка масштабируемости на более сложных архитектурах (ResNet18) и RGB-датасетах (CIFAR-10);

оценка эффективности на аппаратных платформах с различной вычислительной мощностью, включая CPU-инстансы, GPU и мобильные inference-модули.

В результате проведённых экспериментов было получено несколько ключевых результатов:

- снижение размера моделей: комбинированный подход позволил уменьшить количество параметров в среднем на 2,6–3,8 раза без существенной потери точности;
- ускорение инференса: время обработки одного примера уменьшилось в 2–2,5 раза, что подтверждает эффективность оптимизаций для систем реального времени;
- сохранение точности: итоговая точность моделей при использовании всех методов оставалась на уровне 99–99,5 % исходного значения, а вариативность результатов значительно уменьшилась;

- снижение энергопотребления: оптимизированные модели потребляют в среднем в 3–4 раза меньше энергии, что критично для edge-вычислений и мобильных устройств;

- масштабируемость: методы оптимизации успешно применимы к более сложным моделям и RGB-датасетам, включая ResNet18 и CIFAR-10, без значительной деградации качества предсказаний;

- структура ошибок при классификации осталась стабильной, а типовые ошибки не увеличились, что подтверждает сохранение обобщающих способностей моделей после оптимизации.

На основании полученных результатов можно сделать следующие выводы:

- Подтверждена гипотеза исследования, согласно которой комбинированное применение итеративной обрезки, квантизации и гиперпараметрического подбора позволяет достичь баланса между точностью и ресурсными характеристиками моделей;

- Применение HW-aware оптимизации и переносимых архитектур значительно ускоряет процесс оптимизации, снижает вычислительные затраты и позволяет адаптировать модели к различным аппаратным платформам;

- Методология комплексной оптимизации демонстрирует высокую эффективность и универсальность, применима к задачам классификации изображений различной сложности и может быть расширена на задачи компьютерного зрения, обработки естественного языка и других областей глубокого обучения.

Практическая значимость результатов заключается в возможности внедрения разработанных методов в следующие направления:

- Edge-компьютинг и IoT-устройства: использование оптимизированных моделей с INT8-квантизацией и итеративной обрезкой для снижения энергопотребления и ускорения инференса;

- облачные AI-сервисы и серверные приложения: применение гиперпараметрического тюнинга и HW-aware оптимизаций для масштабирования моделей и ускорения time-to-market;

- интерактивные системы и мобильные приложения: интеграция QAT для сохранения высокой точности при ограниченных ресурсах и быстром отклике на запросы пользователя.

Перспективы дальнейших исследований включают:

- расширение методов на более сложные архитектуры, такие как Transformers, EfficientNet, Vision Transformers (ViT);

- исследование адаптивной квантизации и динамической обрезки весов, с учётом распределения важности параметров в реальном времени;

- интеграция с автоматизированным поиском архитектур (NAS) для ускорения оптимизации и минимизации вычислительных затрат;

- оценка переносимости методов на мультимодальные данные (текст, изображения, видео), а также в системах reinforcement learning;

- разработка инструментальных пакетов и open-source библиотек для упрощения применения комплексной оптимизации на практике.

Таким образом, на данном этапе исследования доказана эффективность комплексного подхода к оптимизации нейросетевых моделей: предложенные методы позволяют добиться значительного сокращения объёма и времени инференса при минимальных потерях точности, обеспечивая одновременно высокую адаптивность и универсальность моделей. Работа вносит вклад в развитие практических и теоретических знаний в области оптимизации глубоких нейросетей, открывая новые возможности для внедрения в ресурсоограниченные системы и расширения функциональности интеллектуальных приложений.

**ЛИТЕРАТУРА**

LeCun, 2015 — LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. — 2015. — Vol. 521, No. 7553. — Pp. 436–444. [Eng.]

Han, 2015 — Han S., Pool J., Tran J., Dally W.J. Learning both weights and connections for efficient neural networks // Advances in Neural Information Processing Systems (NIPS). — 2015. — Pp. 1135–1143. [Eng.]

Jacob, 2018 — Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2018. — Pp. 2704–2713. [Eng.]

Hinton, 2015 — Hinton G., Vinyals O., Dean J. Distilling the knowledge in a neural network // arXiv preprint. — 2015. — arXiv:1503.02531. — Pp. 1–9. [Eng.]

Tan, 2019 — Tan M., Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks // Proceedings of the 36th International Conference on Machine Learning (ICML). — 2019. — Pp. 6105–6114. [Eng.]

Vaswani, 2017 — Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Polosukhin I. Attention is all you need // Advances in Neural Information Processing Systems (NIPS). — 2017. — Pp. 5998–6008. [Eng.]

Raffel, 2020 — Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Liu P.J. Exploring the limits of transfer learning with a unified text-to-text transformer // Journal of Machine Learning Research. — 2020. — Vol. 21. No. 140. — Pp. 1–67. [Eng.]

Howard, 2017 — Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications // arXiv preprint. — 2017. — arXiv:1704.04861. — Pp. 1–9. [Eng.]

Brock, 2019 — Brock A., Lim T., Ritchie J.M., Weston N. Smash: One-shot model architecture search through hypernetworks // Proceedings of the International Conference on Learning Representations (ICLR). — 2019. — Pp. 1–13. [Eng.]

Paszk, 2019 — Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Chintala S. PyTorch: An imperative style, high-performance deep learning library // Advances in Neural Information Processing Systems. — 2019. — Pp. 8024–8035. [Eng.]

Abadi, 2016 — Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Zheng X. TensorFlow: A system for large-scale machine learning // Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI). — 2016. — Pp. 265–283. [Eng.]

Howard, 2019 — Howard A., Sandler M., Chu G., Chen L.C., Chen B., Tan M., Le Q.V. Searching for MobileNetV3 // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — 2019. — Pp. 1314–1324. [Eng.]

**REFERENCES**

LeCun, 2015 – LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning // Nature. — 2015. — Vol. 521. — No. 7553. — Pp. 436–444. [in Eng.]

Han, 2015 – Han, S., Pool, J., Tran, J., Dally, W.J. (2015). Learning both weights and connections for efficient neural networks // Advances in Neural Information Processing Systems (NIPS). — 2015. — Pp. 1135–1143. [in Eng.]

Jacob, 2018 – Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2018. — Pp. 2704–2713. [in Eng.]

Hinton, 2015 – Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the knowledge in a neural network // arXiv preprint. — arXiv:1503.02531. — Pp. 1–9. [in Eng.]

Tan, 2019 – Tan, M., Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks // Proceedings of the 36th International Conference on Machine Learning (ICML). — 2019. — Pp. 6105–6114. [in Eng.]

Vaswani, 2017 – Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Polosukhin, I. (2017). Attention is all you need // Advances in Neural Information Processing Systems (NIPS). — 2017. — P. 5998–6008. [in Eng.]

Raffel, 2020 – Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Liu, P.J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer // Journal of Machine Learning Research. — 2020. — Vol. 21, No. 140. — Pp. 1–67. [in Eng.]

Howard, 2017 – Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications // arXiv preprint. — arXiv:1704.04861. — 2017. — Pp. 1–9. [in Eng.]

Brock, 2019 – Brock, A., Lim, T., Ritchie, J.M., Weston, N. (2019). SMASH: One-shot model architecture search through hypernetworks // Proceedings of the International Conference on Learning Representations (ICLR). — 2017. — Pp. 1–13. [in Eng.]

Paszke, 2019 – Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library // Advances in Neural Information Processing Systems. — 2019. — Pp. 8024–8035. [in Eng.]

Abadi, 2016 – Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Zheng, X. (2016). TensorFlow: A system for large-scale machine learning // Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI). — 2016. — Pp. 265–283. [in Eng.]

Howard, 2019 – Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Le, Q.V. (2019). Searching for MobileNetV3 // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — 2019. — Pp. 1314–1324. [in Eng.]