Научная статья
УДК 004.852
URL: https://trudymai.ru/published.php?ID=187466
EDN: https://www.elibrary.ru/OVUCFG

ДИСТИЛЛЯЦИЯ ЗНАНИЙ БОЛЬШОЙ ЯЗЫКОВОЙ МОДЕЛИ В КОМПАКТНУЮ МОДЕЛЬ ДЛЯ ГЕНЕРАЦИИ ПЕРСОНАЛИЗИРОВАННЫХ ТРЕНИРОВОЧНЫХ ПРОГРАММ ДЛЯ ПИЛОТОВ И БОРТПРОВОДНИКОВ ГРАЖДАНСКОЙ АВИАЦИИ

М.Н. Татаренко ✉, **В.Н. Лукин**

Московский авиационный институт (национальный исследовательский университет),

г. Москва, Россия

✉ michaeltatarenko@gmail.com

Цитирование: Татаренко М.Н., Лукин В.Н. Дистилляция знаний большой языковой модели в компактную модель для генерации персонализированных тренировочных программ для пилотов и бортпроводников гражданской авиации // Труды МАИ. 2026. № 146. URL: https://trudymai.ru/published.php?ID=187466

Аннотация. Современные большие языковые модели, основанные на архитектуре трансформер, демонстрируют высокую эффективность в задачах генерации персонализированного контента. Однако их развертывание на устройствах с ограниченными вычислительными ресурсами остаётся сложной задачей, что ограничивает возможности автономного применения в системах подготовки лётного и кабинного экипажа гражданской авиации. В работе предложен подход к дистилляции знаний из модели Qwen 3 4B в компактную модель Gemma 3 270M для генерации персонализированных тренировочных программ физической подготовки пилотов и бортпроводников гражданской авиации. Для обучения использован метод низкоранговой адаптации LoRA, синтетический набор данных из 67 392 примеров, сгенерированный по методологии самообучения (Self-Instruct), а также специализированные программные библиотеки. Достигнуто сжатие модели в 27 раз (с 8 ГБ до 300 МБ в квантованном формате) при финальном значении функции потерь 0,029, что свидетельствует об успешной аппроксимации

поведения модели-учителя. Результирующая модель может быть развёрнута на потребительском графическом процессоре с 6 ГБ памяти, что делает её пригодной для автономного использования в учебных центрах гражданской авиации и обеспечивает возможность одновременного обслуживания множества пользователей.

Ключевые слова: дистилляция знаний, компактные языковые модели, дообучение, LoRA, персонализация, гражданская авиация

KNOWLEDGE DISTILLATION FROM LARGE LANGUAGE MODEL TO COMPACT MODEL FOR PERSONALIZED TRAINING PROGRAM GENERATION FOR PILOTS AND FLIGHT CREW MEMBERS OF CIVIL AVIATION

M.N. Tatarenko✉, V.N.Lukin

Moscow Aviation Institute (National Research University), Moscow, Russia

✉ michaeltatarenko@gmail.com

Citation: Tatarenko M.N., Lukin V.N. Knowledge distillation from large language model to compact model for personalized training program generation for pilots and flight crew members of civil aviation// Trudy MAI. 2026. No. 146. (In Russ.). URL: https://trudymai.ru/published.php?ID=187466

Abstract. Modern large language models based on the Transformer architecture demonstrate high effectiveness in personalized content generation tasks. However, their deployment on resource-constrained devices remains a challenging problem, limiting the possibilities of autonomous application in training systems for flight crew and cabin crew of civil aviation. This paper proposes an approach to knowledge distillation from the Qwen 3 4B model into the compact Gemma 3 270M model for generating personalized physical training programs for pilots and flight crew members of civil aviation. The model was trained using the Low-Rank Adaptation (LoRA) method on a synthetic dataset of 67,392 samples generated via the Self-Instruct methodology, along with specialized software libraries. A 27-fold model compression was achieved (from 8 GB to 300 MB in quantized format) with a final loss value of 0.029, indicating successful approximation of the teacher model's behavior. The resulting model can be deployed on a consumer-grade

GPU with 6 GB of memory, making it suitable for autonomous use in civil aviation training centers and enabling simultaneous service of multiple users.

Keywords: knowledge distillation, compact language models, fine-tuning, LoRA, personalization, civil aviation

Введение

Современные большие языковые модели, основанные на архитектуре Transformer [1], демонстрируют впечатляющие результаты в задачах генерации текста и персонализации контента [2, 3]. Модели семейства GPT [2], LLaMA [3], BERT [4] и их производные показали способность решать широкий спектр задач обработки естественного языка. Однако их развертывание в условиях реальной эксплуатации с ограниченными вычислительными ресурсами остается сложной задачей из-за высоких требований к памяти и вычислительной мощности.

Дистилляция знаний [5] представляет собой эффективный подход к переносу знаний из большой модели-учителя в компактную модель-ученика. Данный метод успешно применялся для создания эффективных моделей, таких как DistilBERT [6], и получил широкое освещение в обзорных работах [7]. Альтернативные подходы включают методы квантования [8] и эффективной адаптации параметров [9, 10].

Проблема масштабируемости

Рассмотрим практическую задачу создания системы генерации персонализированных тренировочных программ физической подготовки для пилотов и бортпроводников гражданской авиации. Пусть M_{large}— большая языковая модель (например, Qwen 3 4B [11] с 4 миллиардами параметров), а доступная видеокарта имеет ограничение по видеопамяти $V_{GPU} = 6$ ГБ (например, NVIDIA RTX 2060).

Объем памяти, необходимый для загрузки модели в режиме инференса при использовании 16-битной точности (FP16), можно оценить как:

$$V_{model} = N_{params} \times 2 \text{ байт} + V_{overhead} \quad (1)$$

где N_{params}— количество параметров модели, 2 байта на параметр для FP16, и $V_{overhead}$— накладные расходы на промежуточные активации.

Для модели Qwen 3 4B:

$$V_{Qwen} = 4 \times 10^9 \times 2 \text{ байт} + V_{overhead} \approx 8 \text{ Гб} + V_{overhead} > 6 \text{ Гб} \quad (2)$$

Таким образом, на доступном оборудовании можно запустить только один экземпляр большой модели, что создает узкое место при обслуживании множества пользователей.

Предлагаемое решение

Мы предлагаем подход, основанный на дистилляции знаний [5, 7]: использовать большую модель M_{large} для генерации обучающего датасета в духе методологии самообучения [12], а затем обучить на этих данных компактную модель M_{small} (Gemma 3 270M [13]). Это позволяет:

1. Сохранить качество генерации при значительном сокращении размера модели: 270 МБ $\ll$ 8 ГБ;
2. Запустить n $\gg$ 1 экземпляров малой модели одновременно на одной GPU;
3. Сократить время отклика и повысить пропускную способность системы.

Количество одновременно работающих экземпляров оценивается как:

$$n_{instances} = \left\lfloor \frac{V_{GPU}}{V_{small} + V_{batch}} \right\rfloor \quad (3)$$

где V_{small} — память для одного экземпляра малой модели, V_{batch} — память для обработки батча запросов.

Методы

Предлагаемый подход включает несколько последовательных этапов: генерацию синтетического обучающего набора данных с помощью модели-учителя, дообучение компактной модели методом низкоранговой адаптации и оптимизацию процесса обучения с учётом ограничений доступного оборудования.

Генерация синтетического датасета

Обучающий датасет был сгенерирован с использованием модели Qwen 3 4B [11]. Подход к генерации синтетических данных основан на методологии самообучения [12], позволяющей использовать большую модель для создания

обучающих примеров. Для каждого примера датасета формировался входной запрос, содержащий:

– Профиль члена экипажа гражданской авиации: $p = (l, f, g, c, e)$, где l — уровень подготовки, f — частота тренировок в неделю, g — цель физической подготовки, c — противопоказания, e — доступное оборудование;

– Историю последних 5 тренировок по программе физической подготовки: $H = \{w_1, w_2, \ldots, w_5\}$;

– Каталог упражнений: $\mathscr{E}$.

Модель генерировала 3 персонализированные тренировки в структурированном текстовом формате. Итоговый датасет содержит:

$$|\mathcal{D}| = 67{,}392 \quad (4)$$

Каждый пример имеет следующую структуру:

$$d_i = (x_i, y_i),\ x_i \in \mathbb{R}^{L_{prompt}},\ y_i \in \mathbb{R}^{L_{response}} \quad (5)$$

где средняя длина входного запроса $\mathbb{E}[L_{prompt}] \approx 2648$ токенов, длина ответа $\mathbb{E}[L_{prompt}] \approx 2648$ токенов.

Архитектура целевой модели

В качестве базовой модели для дообучения использовалась Gemma 3 270M Instruct [13] —модель имеет N = 298 474 112 ≈ 298,5 млн параметров, L = 24 слоёв, $d_{model} = 1024$, h = 16 голов внимания, максимальную длину контекста: $L_{max} = 4096$ токенов

Метод LoRA для эффективного обучения

Для эффективного обучения модели с минимальными вычислительными затратами применялся метод LoRA (низкоранговая адаптация, Low-Rank Adaptation) [14]. Суть метода заключается в том, что вместо обновления всех весов нейронной сети обучаются лишь небольшие дополнительные матрицы низкого ранга, которые корректируют поведение исходной модели. Это позволяет значительно сократить количество обучаемых параметров и объём требуемой памяти, сохраняя при этом качество результатов.

Математическая формулировка LoRA

Пусть $W_0 \in \mathbb{R}^{d\times k}$— предобученная весовая матрица. При стандартном дообучении обновление весов записывается как:

$$W = W_0 + \Delta W \quad (6)$$

где $\Delta W \in \mathbb{R}^{d\times k}$— полная матрица обновлений.

LoRA аппроксимирует ΔW произведением двух матриц низкого ранга:

$$\Delta W = BA,\ \ B \in \mathbb{R}^{d\times r},\ A \in \mathbb{R}^{r\times k} \quad (7)$$

где $r \ll \min(d, k)$— ранг аппроксимации (в нашем случае r = 128).

Прямой проход через слой с LoRA:

$$h = W_0 x + \frac{a}{r} BAx = W_0 x + \frac{a}{r} B(Ax) \quad (8)$$

где α — масштабирующий коэффициент (в нашем случае α = 128).

Параметры LoRA

В данной работе LoRA-адаптеры применялись к следующим компонентам трансформера [1]:

– Проекции механизма внимания: W_q, W_k, W_v, W_o;

– Проекции полносвязной сети (слои, преобразующие скрытые представления): $W_{gate}, W_{up}, W_{down}$.

Количество обучаемых параметров:

$$N_{trainable} = \sum_{l=1}^{L} \sum_{i\in\{q,k,v,o,gate,up,down\}} \left(d_{in}^{(i)} + d_{out}^{(i)}\right) \times r \quad (9)$$

В нашем случае:

$$N_{trainable} = 30,375,936 \text{ параметров } \approx 10.18\% \text{ от } N_{total} \quad (10)$$

Конфигурация обучения

Для обеспечения воспроизводимости результатов в таблице 1 приведены основные гиперпараметры, использованные при обучении модели. Значения параметров подобраны с учётом ограничений доступной видеопамяти и направлены на достижение баланса между скоростью сходимости и стабильностью процесса обучения.

Таблица 1

Гиперпараметры обучения

Параметр	**Значение**
Размер батча на устройство	4
Шаги накопления градиента	2
Эффективный размер батча	4 × 2 = 8
Скорость обучения	$\eta = 2 \times 10^{-5}$
Шагов прогрева	n_warmup = 20
Количество эпох	1
Общее количество шагов	n_steps = 8 424
Оптимизатор	AdamW 8-bit
Weight decay	$\lambda = 0{,}01$
Планировщик LR	Linear decay

Использовался оптимизатор AdamW [15] — алгоритм градиентной оптимизации с адаптивной скоростью обучения и регуляризацией весов, широко применяемый при обучении нейронных сетей. Для сокращения потребления видеопамяти состояния оптимизатора хранились в 8-битном формате вместо стандартного 32-битного, что позволяет уменьшить расход памяти приблизительно в четыре раза без существенной потери точности, аналогично подходу QLoRA (квантованная низкоранговая адаптация) [16]. Обновление параметров на шаге t:

$$\mathrm{m_t} = \beta_1 \mathrm{m_{t-1}} + (1-\beta_1)\mathrm{g_t} \tag{11}$$

$$\mathrm{v_t} = \beta_2 \mathrm{v_{t-1}} + (1-\beta_2)\mathrm{g_t^2} \tag{12}$$

$$\hat{\mathrm{m}}_\mathrm{t} = \frac{\mathrm{m_t}}{1-\beta_1^\mathrm{t}},\ \hat{\mathrm{v}}_\mathrm{t} = \frac{\mathrm{v_t}}{1-\beta_2^\mathrm{t}} \tag{13}$$

$$\theta_t = \theta_{t-1} - \eta_t \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} + \lambda \theta_{t-1} \right) \tag{14}$$

где g_t − градиент на шаге $t, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

Скорость обучения изменялась по линейному расписанию с прогревом:

$$\eta_t = \begin{cases} \frac{t}{n_{warmup}} \eta_{max}, & t \le n_{warmup} \\ \eta_{max}\left(1 - \frac{t-n_{warmup}}{n_{steps}-n_{warmup}}\right), & t > n_{warmup} \end{cases} \tag{15}$$

Функция потерь

Использовалась стандартная функция потерь — перекрёстная энтропия, вычисляемая только на токенах ответа модели, без учёта входного запроса, что соответствует практике дообучения с инструкциями [17]:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{t\in\tau_{response}^{(i)}} logP_\theta\left(y_t^{(i)}|x^{(i)}, y^{(i)} < t\right) \quad (16)$$

где $\tau_{response}^{(i)}$ — множество индексов токенов в ответе для примера i, P_θ — распределение вероятностей, предсказываемое моделью.

Вероятность предсказания токена вычисляется через softmax:

$$P_\theta(y_t = j|h_t) = \frac{exp(w_j^T h_t/\tau)}{\sum_{j'=1}^{|V|} exp\left(w_{j'}^T h_t/\tau\right)} \quad (17)$$

где h_t — скрытое состояние на позиции t, w_j — вектор эмбеддинга токена j, |V| — размер словаря, τ = 1 — температура.

Маскирование входных данных

Для корректного обучения модели необходимо, чтобы функция потерь вычислялась только по ответам модели, а не по тексту входного запроса. С этой целью применяется маскирование входных данных — приём, при котором токены входного запроса исключаются из расчёта ошибки, и модель обучается генерировать только содержательную часть ответа.

Для предотвращения переобучения на входных запросах применялось маскирование меток:

$$\tilde{y}_t = \begin{cases} -100, & t \in \tau_{prompt} \\ y_t, & t \in \tau_{response} \end{cases} \quad (18)$$

где −100 — специальное значение, игнорируемое функцией потерь.

Техническая инфраструктура

Обучение проводилось на следующем оборудовании с использованием библиотек Transformers [18] и Unsloth [19]:

- GPU: NVIDIA GeForce RTX 5090 (32 ГБ VRAM);
- Программная платформа: Unsloth [19] + Transformers 4.56.2 [18];
- Точность вычислений: FP16 (bfloat16 для активаций);

– Сохранение промежуточных активаций: включено (экономия 30% памяти).

Пиковое использование памяти GPU:

$$V_{peak} = 30.209 \text{ Гб} \approx 96.3\% \text{ от } V_{GPU} \quad (19)$$

Результаты

Для оценки качества дистиллированной модели проведена серия экспериментов, включающая анализ метрик обучения, динамики функции потерь, сравнение размеров моделей и качественную оценку генерируемых программ тренировок.

Метрики обучения

Обучение модели завершилось успешно за одну эпоху. Основные результаты представлены в таблице 2.

Таблица 2

Результаты обучения модели

Метрика	Значение
Количество обучающих примеров	67 392
Общее количество шагов	8 424
Время обучения	3 ч 9 мин 45 сек (11 385 сек)
Время на шаг	≈ 1,35 сек
Примеров в секунду	≈ 5,92
Начальное значение функции потерь	0,0299 (шаг 10)
Финальное значение функции потерь	0,0290 (шаг 8420)
Пиковая память GPU	30,209 ГБ
Использование памяти	96,31%

Динамика функции потерь

График изменения функции потерь на этапе обучения показывает стабильную сходимость модели. Важные наблюдения:

1. Быстрое снижение значения функции потерь в первые 100 шагов после прогрева (шаги 20-120);
2. Стабилизация функции потерь на уровне ~0,03 после шага 500;
3. Отсутствие резких скачков, свидетельствующее о стабильности обучения;
4. Небольшое дальнейшее улучшение в конце обучения: 0,029 на шаге 8420.

Среднее значение функции потерь по последним 1000 шагам:

$$\bar{\mathcal{L}}_{7420:8420} = \frac{1}{100}\sum_{t=7420}^{8420}\mathcal{L}_t \approx 0.0299 \quad (20)$$

Сравнение размеров моделей

Таблица 3 демонстрирует значительное уменьшение размера модели при сохранении функциональности. Квантование [8] позволяет дополнительно сократить размер модели.

Таблица 3

Сравнение размеров моделей

Модель	Параметры	Размер (FP16)	Память инференса
Qwen 3 4B	$4{,}0 \times 10^9$	~8 ГБ	> 8 ГБ
Gemma 3 270M (базовая)	$2{,}98 \times 10^8$	596 МБ	~1 ГБ
Gemma 3 270M (Q8)	$2{,}98 \times 10^8$	300 МБ	~600 МБ
LoRA адаптеры	$3{,}04 \times 10^7$	61 МБ	—

Качественная оценка выходов модели

Ниже представлен пример тренировочной программы, сгенерированной моделью для члена лётного экипажа с начальным уровнем физической подготовки (1 тренировка в неделю, цель: снижение веса):

Тренировка 1 (всё тело): жим ногами в тренажёре, тяга в тренажёре, сгибание ног лёжа, жим от груди в тренажёре.

Модель корректно определила частоту тренировок (1 раз в неделю), выбрала подходящую схему занятий (тренировка на всё тело при низкой частоте), подобрала упражнения для начинающих (безопасное оборудование), что соответствует требованиям программ физической подготовки экипажа гражданской авиации.

Для члена кабинного экипажа со средним уровнем подготовки (3 тренировки в неделю, цель: поддержание формы) модель сгенерировала трёхдневную программу с разделением по типу движения:

Тренировка 1 (жимовые): жим штанги лёжа, жим над головой, разведение гантелей в стороны, разгибание на трицепс.

Тренировка 2 (тяговые): тяга штанги в наклоне, тяга верхнего блока, тяга к лицу, сгибание на бицепс.

Тренировка 3 (ноги): приседания со штангой, румынская тяга, разгибание ног, подъём на носки.

Для пилота с продвинутым уровнем подготовки (3 тренировки в неделю, цель: набор мышечной массы) модель предложила трёхдневную программу с проработкой отдельных групп мышц:

Тренировка 1 (грудь): жим штанги лёжа, жим гантелей на наклонной скамье, сведение рук в кроссовере, отжимания на брусьях.

Тренировка 2 (спина): становая тяга, подтягивания, тяга сидя, тяга верхнего блока.

Тренировка 3 (плечи и руки): жим над головой, разведение гантелей в стороны, сгибание штанги на бицепс, разгибание на трицепс.

Приведённые примеры демонстрируют, что модель адаптирует структуру программы к уровню подготовки и целям члена экипажа: для начинающих предлагается одна тренировка на всё тело с использованием тренажёров, для среднего уровня — трёхдневное разделение по типу движения, для продвинутого — трёхдневная программа с проработкой каждой группы мышц.

Результаты

Полученные экспериментальные результаты позволяют оценить эффективность предложенного подхода, определить границы его применимости в системе подготовки экипажа гражданской авиации и наметить направления дальнейших исследований.

Полученные результаты свидетельствуют об успешности подхода к дистилляции знаний [5, 7] через дообучение на синтетических данных. Низкое финальное значение функции потерь (0,029) указывает на то, что компактная модель с высокой точностью воспроизводит поведение модели-учителя в задаче генерации тренировочных программ. Достигнутое сжатие в 27 раз означает переход от модели, требующей выделенного сервера, к модели, способной работать на потребительском оборудовании. Метод LoRA [14] оказался особенно эффективным для данной задачи: обучение лишь 10,18% параметров позволило достичь сходимости за одну эпоху (~3,2 часа), а модульность адаптеров открывает возможность обучения различных конфигураций для разных категорий пользователей без повторного обучения базовой модели.

Обученная модель может быть развёрнута в условиях реальной эксплуатации системы подготовки лётного и кабинного экипажа гражданской авиации со следующими преимуществами:

– Низкие требования к оборудованию: модель работает на потребительских графических процессорах (например, RTX 2060 с 6 ГБ памяти), что позволяет использовать её в учебных центрах без специализированной серверной инфраструктуры.

– Высокая пропускная способность: возможность одновременного запуска до 12 экземпляров модели на одном графическом процессоре обеспечивает параллельное обслуживание множества обучающихся.

– Малое время отклика: генерация персонализированной тренировочной программы занимает менее одной секунды.

– Возможность развертывания на мобильных устройствах с ограниченной памятью благодаря квантованию [8], что расширяет сценарии автономного использования.

Несмотря на достигнутые результаты, существуют направления для дальнейшего развития:

1. Количественная оценка качества: необходима разработка метрики для автоматической оценки сгенерированных тренировочных программ (например, экспертная разметка или сравнительное тестирование)
2. Оценка обобщающей способности: проверка качества модели на отложенной выборке с профилями членов экипажа, не представленными в обучающем наборе данных.
3. Расширение датасета: увеличение покрытия различных профилей членов экипажа, уровней подготовки и типов тренировочных программ.
4. Оптимизация вывода: применение FlashAttention [20] для ускорения генерации.
5. Исследование методов дистилляции с явным моделированием выходного распределения модели-учителя [5].

Заключение

В данной работе предложен и реализован подход к созданию компактной специализированной языковой модели для генерации персонализированных тренировочных программ физической подготовки лётного и кабинного экипажа гражданской авиации. Путём дистилляции знаний [5, 7] из модели Qwen 3 4B [11] в модель Gemma 3 270M [13] с использованием метода LoRA [14] достигнуто сжатие в 27 раз при сохранении качества генерации. Результирующая модель объёмом 300 МБ может быть развёрнута на потребительском графическом процессоре с 6 ГБ памяти, что делает её пригодной для автономного использования в учебных центрах гражданской авиации. Дальнейшие исследования целесообразно направить на разработку количественных метрик оценки качества и расширение области применения предложенного подхода.

Конфликт интересов

Авторы заявляют об отсутствии конфликта интересов.

Conflict of interest

The authors declare no conflict of interest.

Список источников/ References

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. Attention is all you need. *Advances in Neural Information Processing Systems,* 2017, vol. 30, pp. 5998–6008.

2. Brown T.B., Mann B., Ryder N., Subbiah M., Kaplan J.D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., Amodei D. Language models are few-shot learners. *Advances in Neural Information Processing Systems,* 2020, vol. 33, pp. 1877–1901.

3. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample G. *LLaMA:* Open

and efficient foundation language models. *ArXiv :* website*.* Available at: https://arxiv.org/abs/2302.13971 (accessed: 12 December 2025).

4. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Human Language Technologies, Minneapolis, 2019, pp. 4171–4186.

5. Hinton G., Vinyals O., Dean J. Distilling the knowledge in a neural network. *ArXiv* : website. Available at: https://arxiv.org/abs/1503.02531 (accessed: 12 December 2025).

6. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing at NeurIPS 2019*, Vancouver, 2019, 5 p.

7. Gou J., Yu B., Maybank S.J., Tao D. Knowledge Distillation: a survey. *International Journal of Computer Vision*, 2021, vol. 129, pp. 1789–1819.

8. Gholami A., Kim S., Dong Z., Yao Z., Mahoney M.W., Keutzer K. A Survey of Quantization methods for efficient neural network inference. *ArXiv* : website. Available at: https://arxiv.org/abs/2103.13630 (accessed: 12 December 2025).

9. Li X.L., Liang P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021, pp. 4582–4597.

10. Houlsby N., Giurgiu A., Jastrzebski S., Morrone B., De Laroussilhe Q., Gesmundo A., Attariyan M., Gelly S. Parameter-Efficient Transfer Learning for NLP. *Proceedings of the 36th International Conference on Machine Learning (ICML 2019), Long Beach,* 2019, pp. 2790–2799.

11. Bai J., Bai S., Chu Y., Cui Z., Dang K., Deng X., Fan Y., Ge W., Han Y., Huang F. et al. Qwen Technical Report. *ArXiv* : website. Available at: https://arxiv.org/abs/2309.16609 (accessed 12 December 2025).

12. Wang Y., Kordi Y., Mishra S., Liu A., Smith N.A., Khashabi D., Hajishirzi H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto*, 2023, pp. 13484–13508.

13. Gemma Team, Mesnard T., Hardin C., Dadashi R., Bhupatiraju S., Pathak S., Sifre L., Rivière M., Kale M.S., Love J. et al. Gemma: Open Models Based on Gemini Research and Technology. *ArXiv :* website. Available at: https://arxiv.org/abs/2403.08295 (accessed: 12 December 2025).

14. Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L., Chen W. LoRA: Low-Rank Adaptation of Large Language Models. *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022),* 2022, 13 p.

15. Loshchilov I., Hutter F. Decoupled Weight Decay Regularization. *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019),* New Orleans, 2019, 19 p.

16. Dettmers T., Pagnoni A., Holtzman A., Zettlemoyer L. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, 2023, vol. 36, pp. 10088–10115.

17. Wei J., Bosma M., Zhao V.Y., Guu K., Yu A.W., Lester B., Du N., Dai A.M., Le Q.V. Finetuned Language Models Are Zero-Shot Learners. *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*, 2022, 46 p.

18. Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q., Rush A.M. Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

19. Unsloth: Fine-tuning & Reinforcement Learning for LLMs. *Github.com* : website. Available at: https://github.com/unslothai/unsloth (accessed 12 December 2025).

20. Dao T., Fu D.Y., Ermon S., Rudra A., Ré C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. Advances in Neural Information Processing Systems,* 2022, vol. 35, pp. 16344–16359.

Получено 18 декабря 2025 ● Принято к публикации 20 февраля 2026 ● Опубликовано 27 февраля 2026
Received 18 December 2025 ● Accepted 20 February 2026 ● Published 27 February 2026