БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

УДК004.5

Сазонов Игорь Евгеньевич

Сравнительный анализ фреймворков PyTorch и TensorFlow как инструментов разработки программного обеспечения

Реферат по дисциплине
«Основы информационных технологий»

Магистранта кафедры информационных систем
управления факультета прикладной
математики и информатики

Специальность: 7-06-0533-05
+375 (29) 817-16-42
trfdctulfkhekbn@yandex.by

Рецензент:

_____

Минск, 2025

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 3
ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ 4
ВВЕДЕНИЕ 6
ГЛАВА 1. АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ 7
1.1. Эволюция и современное состояние фреймворков 7
1.2. Существующие сравнительные исследования 8
1.3. Сравнительный анализ производительности 9
1.3.1. Производительность на CPU 9
1.3.2. Производительность на GPU 9
1.3.3. Специализированные ускорители (TPU) 10
1.3.4. Потребление памяти и энергоэффективность 10
1.3.5. Сводные таблицы производительности 10
1.4. Анализ инструментов развертывания и MLOps-экосистем 12
1.5. Оценка сообщества и образовательных ресурсов 13
1.6. Анализ промышленного внедрения и кейсов использования 13
1.7. Сравнительный анализ совокупной стоимости владения (TCO) 14
1.8. Выводы по аналитическому обзору 14
ГЛАВА 2. МЕТОДИКА СРАВНИТЕЛЬНОГО АНАЛИЗА И СИСТЕМА ОЦЕНКИ ФРЕЙМВОРКОВ 16
2.1. Разработка системы оценки фреймворков 16
2.2. Адаптивное взвешивание для различных сценариев использования 18
ГЛАВА 3. РЕЗУЛЬТАТЫ СРАВНИТЕЛЬНОГО АНАЛИЗА 19
3.1. Организация эксперимента и исходные данные 19
3.2. Комплексные результаты оценки 19
3.3. Итоговые оценки по сценариям 20
ЗАКЛЮЧЕНИЕ 21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22
ПРИЛОЖЕНИЕ А ПРЕЗЕНТАЦИЯ РЕФЕРАТА 25
ПРИЛОЖЕНИЕ Б ПЕРСОНАЛЬНЫЙ САЙТ 27

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

API — Application Programming Interface (Интерфейс программирования приложений).

BERT — Bidirectional Encoder Representations from Transformers (Двунаправленные представления энкодера от трансформеров).

CPU — Central Processing Unit (Центральный процессор).

CNN — Convolutional Neural Network (Сверточная нейронная сеть).

CV — Computer Vision (Компьютерное зрение).

DCN — Deep & Cross Network (Глубокая и перекрестная сеть).

Edge-вычисления — подход, при котором обработка данных происходит не в централизованном облаке, а на периферии сети.

GPU — Graphics Processing Unit (Графический процессор).

IDE — Integrated Development Environment (Интегрированная среда разработки).

JIT — Just-In-Time (Компиляция «на лету»).

JAX — JAX (Библиотека машинного обучения от Google).

MLOps — Machine Learning Operations (Эксплуатация машинного обучения).

NLP — Natural Language Processing (Обработка естественного языка).

ONNX — Open Neural Network Exchange (Открытый стандарт обмена моделей нейронных сетей).

ReLU — Rectified Linear Unit (Выпрямленная линейная функция активации).

ResNet — Residual Neural Network (Остаточная нейронная сеть).

RL — Reinforcement Learning (Обучение с подкреплением).

RNN — Recurrent Neural Network (Рекуррентная нейронная сеть).

SDK — Software Development Kit (Комплект разработки программного обеспечения).

SGD — Stochastic Gradient Descent (Стохастический градиентный спуск).

SoC — System on a Chip (Система на кристалле).

TCO — Total Cost of Ownership (Совокупная стоимость владения).

TFX — TensorFlow Extended (Расширенная платформа TensorFlow).

TPU — Tensor Processing Unit (Тензорный процессор).

XLA — Accelerated Linear Algebra (Ускоренная линейная алгебра).

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Общий объём работы составляет 27 страниц, номинальный объём — 24 страниц. Дополнительно включены 2 приложения и список использованной литературы из 37 источников. В работе присутствуют 4 таблицы и 5 формул, которые используются для иллюстрации и уточнения результатов анализа.

Реферат представляет собой комплексное исследование, направленное на систематический анализ и сравнительную оценку современных фреймворков глубокого обучения как инструментов разработки программного обеспечения. Работа последовательно рассматривает ключевые архитектурные особенности, методологии и практические аспекты применения, определяющие эффективность и целесообразность использования данных технологий в профессиональной разработке. Интеллектуальные системы опираются на методы машинного и глубокого обучения, а также нейросетевые технологии. PyTorch и TensorFlow являются основными фреймворками для их реализации, предоставляя инструменты для моделирования и внедрения искусственного интеллекта. Сравнительный анализ этих платформ позволяет определить их применимость и эффективность, что напрямую соответствует направлению «Интеллектуальные системы».

Ключевые слова: PyTorch, TensorFlow, машинное обучение, глубокое обучение, развертывание моделей, API, эффективность, MLOps, производительность разработки, экосистема.

Целью исследования является проведение сравнительного анализа эффективности фреймворков глубокого обучения PyTorch и TensorFlow как инструментов разработки программного обеспечения, а также выработка практических рекомендаций по их выбору для различных сценариев применения в исследовательской и промышленной практике.

Объектом исследования выступают фреймворки глубокого обучения PyTorch и TensorFlow, их архитектурные парадигмы, экосистемы инструментов для разработки и развертывания моделей, а также поддерживающие их сообщества и документация, комплексное изучение которых позволяет выявить их сравнительные преимущества и ограничения.

Актуальность исследования обусловлена стратегической важностью выбора фреймворка машинного обучения, который напрямую влияет на стоимость и успех IT-проектов. Наблюдаемая конвергенция PyTorch и TensorFlow, когда первый усиливает промышленные возможности, а второй заимствует удобство для разработчиков, требует комплексного пересмотра их позиционирования. Существующие сравнения часто фрагментарны, фокусируясь на скорости обучения и игнорируя полный жизненный цикл разработки, включая MLOps, сопровождаемость кода и отладку. Таким

образом, проведение всестороннего анализа, рассматривающего фреймворки как комплексные инструменты разработки, является своевременной и практически значимой задачей.

Во введении обосновывается стратегическая важность выбора фреймворка машинного обучения, формулируется проблема фундаментальных архитектурных различий между PyTorch и TensorFlow, несмотря на их функциональную конвергенцию, и определяются ключевые критерии для последующего сравнительного анализа.

В Главе 1 проводится системный аналитический обзор, который выявляет эволюцию фреймворков от статических к гибридным графам, анализирует противоречивость результатов существующих бенчмарков и детально сравнивает их производительность на разных аппаратных платформах, инструменты развертывания и показатели совокупной стоимости владения (TCO).

В Главе 2 разрабатывается и формализуется комплексная методика сравнительной оценки, основанная на адаптивном взвешивании количественных метрик и экспертных оценок для трех ключевых сценариев использования: исследовательский прототип, промышленное развертывание и edge-вычисления.

В Главе 3 представляются результаты экспериментального сравнения, которые кристаллизуют ключевой вывод работы: PyTorch демонстрирует преимущество в сценариях исследований и прототипирования, тогда как TensorFlow сохраняет лидерство в промышленном развертывании и edge-вычислениях, при этом общий разрыв в производительности между фреймворками минимален.

В заключении подводятся итоги, констатируется четкое разделение областей применения фреймворков, подтверждается тенденция к их конвергенции и определяются перспективные направления для дальнейших исследований.

В приложении А представлены слайды презентации реферата.

В приложении Б размещены материалы персонального сайта.

ВВЕДЕНИЕ

Выбор фреймворка для машинного обучения является одним из ключевых стратегических решений в жизненном цикле разработки интеллектуальных систем. Это решение оказывает прямое влияние на продуктивность команды, скорость итераций, производительность итоговой модели и возможности ее последующего масштабирования и развертывания [3]. Два ведущих фреймворка, PyTorch и TensorFlow, доминируют в этой области, предлагая мощные, но концептуально различные подходы к созданию и обучению нейронных сетей.

Проблема исследования заключается в том, что, несмотря на внешнее сходство и конвергенцию функциональности, PyTorch и TensorFlow обладают фундаментальными архитектурными различиями. Исторически TensorFlow базировался на парадигме статических вычислительных графов («define-and-run»), что обеспечивало высокую производительность, но усложняло отладку. PyTorch, напротив, изначально использовал динамические графы («define-by-run»), что делало процесс разработки более интуитивным и гибким. Эти различия создают сложности для разработчиков и руководителей проектов при выборе инструмента, оптимально соответствующего конкретным задачам, компетенциям команды и требованиям к развертыванию [3].

В представленном реферате проводится системный сравнительный анализ PyTorch и TensorFlow по набору четко определенных критериев, охватывающих аспекты разработки (простота использования, гибкость, отладка), производительности (скорость обучения и инференса, использование ресурсов) и развертывания (экосистема, мобильные платформы). Исследование включает анализ эволюции архитектур фреймворков, обзор существующих сравнительных работ и оценку эффективности на основе данных бенчмарков и научных публикаций.

ГЛАВА 1
АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ

При написании аналитического обзора литературы были рассмотрены различные подходы, сравнительные исследования и отраслевые отчеты, посвященные эволюции, производительности и экосистемам фреймворков глубокого обучения TensorFlow и PyTorch. Особое внимание уделяется противоречивости результатов бенчмарков, анализу факторов, влияющих на производительность, и тенденциям конвергенции функциональности.

1.1. Эволюция и современное состояние фреймворков

Эволюция фреймворков глубокого обучения от узкоспециализированных академических инструментов до промышленных платформ радикально изменила ландшафт ИИ. Ранние библиотеки середины 2010-х годов, такие как Theano, Torch7 и Caffe, заложили фундаментальные концепции, но требовали громоздкого кода и имели высокий порог входа [16]. Ситуация кардинально изменилась с появлением двух ключевых игроков:

- **TensorFlow**, выпущенный Google в 2015 году, быстро стал отраслевым стандартом благодаря своей масштабируемости, поддержке распределенного обучения и мощной экосистеме, созданной при поддержке технологического гиганта [16].
- **PyTorch,** разработанный исследовательским подразделением Facebook AI Research (ныне Meta), был представлен в 2016–2017 годах и быстро завоевал популярность в академических и исследовательских кругах благодаря своей гибкости и интуитивно понятному интерфейсу [16, 22].

Появление этих двух фреймворков ознаменовало переход к более зрелым, удобным и производительным инструментам, что спровоцировало взрывной рост инноваций в области ИИ.

Изначально их различие заключалось в подходе к вычислительным графам: TensorFlow использовал статические графы ("define-and-run"), обеспечивавшие оптимизацию и производительность, но усложнявшие отладку [15, 3], тогда как PyTorch применял динамические графы ("define-by-run"), более удобные и «питонические», но с дополнительными накладными расходами [13, 3].

Осознавая сильные стороны друг друга, оба фреймворка начали активно заимствовать лучшие идеи, что привело к их постепенному сближению. В 2019 году выход TensorFlow 2.0 с режимом Eager Execution, включённым по умолчанию, сделал выполнение операций немедленным, по аналогии с

PyTorch, и тем самым значительно упростил разработку и отладку, фактически устранив главное преимущество конкурента в удобстве использования. В ответ на растущие требования к производительности в линейке PyTorch 2.x, начиная с 2023 года, появилась функция torch.compile(), представляющая собой JIT-компилятор, который анализирует Python-код, преобразует его в оптимизированный вычислительный граф и выполняет с высокой скоростью, что позволило PyTorch получить преимущества статических графов в плане оптимизации, не отказываясь при этом от гибкости динамического подхода.

Сегодня различия между ними менее выражены: оба предлагают гибридный подход, гибкость и простоту для разработки и исследований, а также возможность компиляции для достижения максимальной производительности в продакшене, а выбор чаще определяется экосистемой и предпочтениями команды [15, 16].

1.2. Существующие сравнительные исследования

Существующие сравнительные исследования показывают стратегическую значимость бенчмарков, но их результаты быстро устаревают из-за динамичного развития фреймворков. Производительность обычно оценивается по времени обучения, времени инференса и точности предсказаний [19]. Эффективность использования ресурсов определяется загрузкой CPU/GPU, температурой компонентов и энергопотреблением [11, 12]. Качественные характеристики включают удобство разработки, качество документации и простоту интеграции [19].

Корректная интерпретация сравнительных исследований требует учитывать версию фреймворка, аппаратную платформу и сложность модели [1; 12; 16]. Даже небольшие обновления могут существенно менять результаты: так, Yapıcı et al. (2021) показали, что TensorFlow быстрее справляется с малыми изображениями (MNIST), тогда как PyTorch демонстрирует лучшие показатели на более крупных данных благодаря эффективному управлению памятью [16]. В исследовании Novac et al. (2022) PyTorch оказался значительно быстрее — время обучения сократилось на 25%, а инференс выполнялся на 57% быстрее по сравнению с TensorFlow [19]. Эти данные подчёркивают, что производительность нельзя оценивать вне контекста конкретной задачи и условий эксперимента.

Дополнительные бенчмарки также подтверждают контекстную природу сравнений. В тестах CIFAR-10 (2024) PyTorch показал немного более высокую точность (≈93.2% против ≈92.5% у TensorFlow) и меньшее время обучения (110

против 120 минут), при этом более полно используя ресурсы GPU [31]. В обзоре 2025 года отмечено, что PyTorch быстрее обучается на одной GPU и удобнее для исследований и прототипирования, тогда как TensorFlow лучше подходит для очень крупных моделей и специализированного оборудования, где его экосистема и возможности тонкой настройки дают преимущества [6]. Таким образом, противоречивость выводов объясняется не столько различиями между самими фреймворками, сколько влиянием версии, аппаратной специфики и характера задачи, что делает комплексный подход к интерпретации бенчмарков необходимым.

1.3. Сравнительный анализ производительности

Выбор аппаратной платформы — стратегическое решение, влияющее на такие метрики, как time-to-market и совокупная стоимость владения (TCO). Производительность включает скорость выполнения операций, эффективность использования ресурсов, энергопотребление и масштабируемость. Анализ разных платформ помогает определить оптимальный фреймворк для конкретной инфраструктуры. Более высокая производительность сокращает время обучения и снижает затраты на аренду GPU или TPU, а энергопотребление напрямую влияет на расходы дата-центров. Правильное сочетание фреймворка и оборудования уменьшает капитальные и операционные издержки и повышает конкурентоспособность продукта.

1.3.1. Производительность на CPU

Хотя GPU доминируют в обучении, многие задачи инференса и предобработки выполняются на CPU. Germino et al. показали, что на AMD Ryzen 5 3600x TensorFlow обучал простые сети почти в 6 раз быстрее PyTorch [12]. Florencio et al. отметили различия на Intel i5-7300HQ, подтверждая зависимость от архитектуры CPU [11]. Преимущество TensorFlow связывают с низкоуровневыми оптимизациями, включая XNNPACK [13].

1.3.2. Производительность на GPU

GPU стали стандартом для обучения, но результаты сравнений противоречивы. Germino et al. показали, что на RTX 2070 Super TensorFlow 2.0 обучал CNN более чем вдвое быстрее [12]. Novac et al. на GTX 1070 Ti зафиксировали преимущество PyTorch в 25.5% [19], а Florencio et al. на GTX

1050 также отметили меньшие времена выполнения у PyTorch [11]. Различия объясняются версиями фреймворков, архитектурой моделей и задачами.

1.3.3. Специализированные ускорители (TPU)

TPU — специализированные чипы Google для ускорения ML-нагрузок. TensorFlow имеет нативную и зрелую поддержку TPU, что обеспечивает высокую производительность на больших моделях [16, 29]. PyTorch использует TPU через проект PyTorch/XLA, но его интеграция считается менее глубокой [16].

1.3.4. Потребление памяти и энергоэффективность

Ресурсная эффективность влияет на совокупную стоимость владения. Germino et al. показали более низкое энергопотребление у TensorFlow [12]. Florencio et al. отметили, что PyTorch обеспечивал более низкую медианную температуру CPU и GPU [11]. Yapıcı et al. выявили лучшее управление памятью в PyTorch при работе с крупными изображениями [16].

1.3.5. Сводные таблицы производительности

Для наглядного представления количественных данных из проанализированных источников ниже приведены сводные таблицы.

Исследование	Модель	Аппаратная платформа	TensorFlow (Время)	PyTorch (Время)	Преимущество
Germino et al	NN	CPU (AMD Ryzen 5)	8.62 с	51.29 с	TensorFlow в ~5.9x
StackOverflow	NN	CPU (Intel i7-10700)	21 с	128 с	TensorFlow в ~6.1x
Germino et al	CNN	GPU (RTX 2070 Super)	28.21 с	58.48 с	TensorFlow в ~2.0x
Ba Alawi	CNN	CPU (Intel Xeon)	17.2 с	20.1 с	TensorFlow в ~1.2x
DEV Community	CNN	GPU (RTX 4090)	1.9 ч	2.0 ч	TensorFlow в ~1.05x
DigitalOcean	CNN	GPU (Nvidia V100)	2.3 ч	2.1 ч	PyTorch в ~1.1x
Florencio et al.	CNN	GPU (GTX 1050)	3.2 ч	2.7 ч	PyTorch в ~1.2x
Novac et al	CNN	GPU (GTX 1070 Ti)	21.95 ч	16.98 ч	PyTorch в ~1.3x

Таблица 1. Сравнение времени выполнения

Сравнение данных (Таблицы 1) показывает, что TensorFlow значительно быстрее на CPU и некоторых GPU, особенно в задачах NN, тогда как PyTorch выигрывает на ряде графических ускорителей, особенно при обучении больших моделей и длительных экспериментах.

Исследование	Модель	Метрика	TensorFlow (Показатель)	PyTorch (Показатель)	Преимущество
Germino et al	NN	Точность (%)	95.8%	92.3%	+3.5% точнее TensorFlow
StackOverflow	NN	Ошибка (%)	4.7%	6.2%	+1.5% меньше ошибок у TensorFlow
Germino et al	CNN	Точность (%)	99.1%	99.3%	+0.2% точнее PyTorch
Ba Alawi	CNN	Ошибка (%)	7.6%	7.9%	+0.3% меньше ошибок у TensorFlow
DEV Community	CNN	Точность (%)	77.0%	77.5%	+0.5% точнее PyTorch
DigitalOcean	CNN	Ошибка (%)	22.1%	21.5%	+0.6% меньше ошибок у PyTorch
Florencio et al.	CNN	Точность (%)	91.2%	90.7%	+0.5% точнее TensorFlow
Novac et al	CNN	Ошибка (%)	17.7%	17.9%	+0.2% меньше ошибок у TensorFlow

Таблица 2. Сравнение точности и ошибки

Сравнение данных (Таблицы 2) показывает, что TensorFlow чаще обеспечивает более высокую точность в задачах NN и демонстрирует меньший уровень ошибок, тогда как PyTorch выигрывает в отдельных случаях при работе с CNN, особенно на современных GPU и в задачах компьютерного зрения.

Исследование	Модель	Аппаратная платформа	TensorFlow (Энергопотреб.)	PyTorch (Энергопотреб.)	Преимущество
Germino et al	NN	CPU (AMD Ryzen 5)	65 Вт	72 Вт	+11% энергоэффект. TensorFlow
StackOverflow	NN	CPU (Intel i7-10700)	70 Вт	78 Вт	+11% энергоэффект. TensorFlow
Germino et al	CNN	GPU (RTX 2070 Super)	185 Вт	190 Вт	+3% энергоэффект. TensorFlow

Ba Alawi	CNN	CPU (Intel Xeon)	80 Вт	85 Вт	+6% энергоэффект. TensorFlow
DEV Community	CNN	GPU (RTX 4090)	310 Вт	305 Вт	+2% энергоэффект. PyTorch
DigitalOcean	CNN	GPU (Nvidia V100)	250 Вт	240 Вт	+4% энергоэффект. PyTorch
Florencio et al.	CNN	GPU (GTX 1050)	120 Вт	115 Вт	+4% энергоэффект. PyTorch
Novac et al	CNN	GPU (GTX 1070 Ti)	200 Вт	190 Вт	+5% энергоэффект. PyTorch

Таблица 3. Сравнение энергоэффективности

Сравнение данных (Таблицы 3) показывает, что TensorFlow потребляет меньше энергии при работе с большими данными и низкой загрузке, особенно на CPU и части GPU, тогда как PyTorch может демонстрировать преимущество на отдельных графических ускорителях при интенсивных нагрузках. В целом различия невелики: TensorFlow выигрывает по энергоэффективности в базовых сценариях, а PyTorch иногда оказывается эффективнее при специфических задачах обработки больших массивов данных.

При составлении сводных сравнительных таблиц (Таблицы 1–3) использовались данные из источников [12, 14, 25, 32–37]. В целом различия между фреймворками по качеству моделей минимальны, что подтверждает их сопоставимую зрелость и надёжность.

1.4. Анализ инструментов развертывания и MLOps-экосистем

Переход от исследовательских прототипов к промышленным ML-решениям невозможен без зрелой MLOps-экосистемы, обеспечивающей надёжность, масштабируемость и удобство развертывания. Исторически TensorFlow имел более развитую инфраструктуру для продакшена, однако PyTorch за последние годы значительно сократил этот разрыв, предлагая всё более зрелые и гибкие инструменты.

Экосистема TensorFlow охватывает полный жизненный цикл модели: TensorFlow Serving обеспечивает высокопроизводительное развертывание с поддержкой версионирования и A/B-тестирования [15, 16]; TensorFlow Lite стал стандартом для мобильных и встроенных устройств, применяя оптимизации вроде квантования [18, 30]; TensorFlow Extended (TFX)

предоставляет end-to-end платформу для построения ML-конвейеров и интегрируется с Google Cloud AI Platform [15, 16].

PyTorch, изначально более фрагментированный, сегодня предлагает зрелые решения: TorchServe для развертывания с поддержкой REST/gRPC API и мониторинга [15, 16], PyTorch Mobile для Android и iOS [18], а также экспорт моделей в ONNX. Последний обеспечивает совместимость с ONNX Runtime и NVIDIA Triton, расширяя возможности инференса и давая командам гибкость выбора технологического стека [15, 16]. Таким образом, TensorFlow остаётся эталоном комплексной экосистемы, а PyTorch усиливает позиции за счёт гибкости и открытых стандартов.

1.5. Оценка сообщества и образовательных ресурсов

TensorFlow и PyTorch обладают активными сообществами и богатыми образовательными ресурсами: первый сохраняет лидерство благодаря поддержке Google и долгой истории (более 170 тыс. звёзд на GitHub), тогда как PyTorch демонстрирует стремительный рост (+133% вкладов в 2024 году, более 3500 специалистов и 3000 организаций) и укрепил позиции после перехода под эгиду Linux Foundation [26; 16]. Оба фреймворка имеют качественную документацию и множество курсов, включая Machine Learning for Beginners и Dive into Deep Learning [17], при этом API PyTorch считается более интуитивным [22]. На рынке труда навыки Python, TensorFlow и PyTorch входят в число самых востребованных: рынок ML в США вырос на 28% в начале 2025 года, а медианная зарплата ML-инженера достигла $157,000 [15]; при этом PyTorch чаще связывают с исследователями и R&D, а TensorFlow — с MLOps и промышленным внедрением.

1.6. Анализ промышленного внедрения и кейсов использования

Статистика внедрения демонстрирует двойственную картину: по данным AceCloud, в 2025 году TensorFlow занимает около 38% рынка, а PyTorch — 23%, что отражает его историческое доминирование в корпоративном секторе [15]. В то же время Leapcell фиксирует лидерство PyTorch с 55% долей в продакшене в Q3 2025 года [13], а отчёт Linux Foundation подтверждает его 63% долю в обучении моделей [26]. Таким образом, TensorFlow сохраняет позиции в унаследованных корпоративных системах благодаря зрелой MLOps-экосистеме, тогда как PyTorch доминирует в инновационном цикле — исследованиях, стартапах и новых проектах. Кейсы внедрения подтверждают это распределение: OpenAI использует PyTorch для

языковых моделей GPT-3, Google применяет TensorFlow для машинного перевода, Tesla — PyTorch для автопилота, Snapchat — TensorFlow Lite для мобильных функций, Airbnb — PyTorch для диалоговых ассистентов, а Genentech — для разработки лекарств [16].

1.7. Сравнительный анализ совокупной стоимости владения (TCO)

Совокупная стоимость владения (TCO) включает не только лицензионные расходы, но и временные, инфраструктурные и операционные затраты, а также оценку возврата инвестиций. По данным Карлина [15], понимание TCO помогает принимать экономически взвешенные решения при выборе технологического стека.

Исследования показывают, что PyTorch ускоряет прототипирование благодаря «питонической» природе и поддержке стандартных отладчиков [22, 10], тогда как TensorFlow в связке с Keras обеспечивает высокую скорость разработки типовых архитектур [15]. В инфраструктурном аспекте PyTorch демонстрирует более высокую скорость обучения [19], но TensorFlow эффективнее при работе с TPU и отличается меньшим энергопотреблением [16, 13].

С точки зрения операционных расходов зрелая MLOps-экосистема TensorFlow снижает затраты на поддержку [15, 16], тогда как модульность PyTorch может потребовать дополнительных усилий по интеграции. ROI-анализ показывает, что PyTorch чаще выгоднее для исследовательских проектов [10, 22], TensorFlow — для масштабных промышленных систем [15], а TensorFlow Lite имеет преимущества в мобильных и встроенных приложениях [18, 30].

1.8. Выводы по аналитическому обзору

Выбор между TensorFlow и PyTorch — это не поиск "лучшего" инструмента в вакууме, а стратегическое решение, которое должно основываться на конкретных целях проекта, экспертизе команды, требованиях к развертыванию и долгосрочной стратегии развития продукта. Оба фреймворка являются мощными, зрелыми и способными решать самые сложные задачи в области глубокого обучения.

Критерий	TensorFlow	PyTorch
Архитектурная парадигма	Изначально статический граф («define-and-run»), сейчас гибридный с Eager Execution по умолчанию.	Динамический граф («define-by-run»), с возможностью JIT-компиляции через torch.compile().

Простота использования и отладки	Упрощён благодаря Keras и Eager Execution, но отладка может быть сложнее из-за многоуровневых абстракций.	Более интуитивный и «питонический». Отладка проста благодаря динамическому графу.
Производительность	Высокая, особенно при использовании XLA и TPU. Часто показывает преимущество на CPU и в некоторых GPU-сценариях.	Конкурентоспособная, особенно после внедрения torch.compile(). Часто быстрее в задачах с динамическими структурами и на больших данных.
Экосистема MLOps	Очень зрелая и интегрированная (TFX, TensorFlow Serving, TensorFlow Lite). Считается «enterprise-ready».	Гибкая и модульная (TorchServe, PyTorch Mobile). Сильная сторона — интероперабельность через ONNX.
Принятие в сообществе	Индустрия: сильные позиции в крупных корпорациях и продакшене. Исследования: постепенно уступает PyTorch.	Индустрия: быстрорастущее принятие, особенно в стартапах. Исследования: доминирующий фреймворк.
Основные сценарии использования	Крупномасштабные промышленные системы, мобильные и встраиваемые приложения (TensorFlow Lite), веб-развертывание (TensorFlow.js).	Академические исследования, быстрое прототипирование, проекты в области NLP, задачи с нестандартной архитектурой.

Таблица 4. Сводная таблица сравнения характеристик

Сравнительная таблица (Таблица 4) показывает, что PyTorch оптимален для академических исследований, быстрого прототипирования и гибких проектов, тогда как TensorFlow чаще используется для крупномасштабного промышленного развертывания и мобильных систем через TensorFlow Lite. Для задач с акцентом на совместимость всё большее значение приобретают мульти-бэкенд API (Keras 3.0) и стандарт ONNX, а ландшафт фреймворков продолжает эволюционировать: различия между динамическим и статическим подходами стираются, растёт роль высокоуровневых API, интероперабельности и Edge AI, а интеграция AutoML делает глубокое обучение доступнее.

В конечном счёте выбор между TensorFlow и PyTorch сводится к компромиссам: скорость инноваций против стабильности, гибкость разработки против зрелости развертывания, а также полный «питонический» контроль в PyTorch против абстракций «всё включено» в TensorFlow/Keras. Осознанное решение с учётом этих факторов позволяет снизить риски и обеспечить успешную реализацию ML-проектов.

ГЛАВА 2
МЕТОДИКА СРАВНИТЕЛЬНОГО АНАЛИЗА И СИСТЕМА ОЦЕНКИ ФРЕЙМВОРКОВ

2.1. Разработка системы оценки фреймворков

Для объективного сравнения сложных программных систем, таких как TensorFlow и PyTorch, разработана формализованная методология, учитывающая контекстную зависимость выбора фреймворка для различных сценариев - от академических исследований до промышленного развертывания [3]. Система базируется на принципах комплексности оценки, адаптивного взвешивания критериев и сочетания количественных и качественных показателей [19]. Комплексность обеспечивает охват технических аспектов и операционных характеристик, включая удобство развертывания и порог входа для разработчиков. Адаптивное взвешивание позволяет учитывать различную значимость критериев для разных сценариев использования. Сочетание объективных измеряемых метрик с экспертными оценками качественных аспектов обеспечивает сбалансированность методики.

Для объективного сравнения сложных программных систем, таких как TensorFlow и PyTorch, разработана формализованная методология, учитывающая контекстную зависимость выбора фреймворка для различных сценариев — от академических исследований до промышленного развертывания [3]. Система базируется на принципах комплексности оценки, адаптивного взвешивания критериев и сочетания количественных и качественных показателей [19].

Детализация системы оценки включает:

Количественные метрики, взвешенные по нормализованным показателям с весами: время обучения (0.12), время инференса (0.10), использование памяти (0.08), энергопотребление (0.05), скорость прототипирования (0.10), объём кода (0.06).

Экспертные оценки с весами для качества сообщений об ошибках (0.05), интеграции с IDE (0.04), инструментов развертывания (0.08), мониторинга моделей (0.07), поддержки платформ (0.05), активности разработки (0.04), качества документации (0.03), готовых решений (0.03).

Стратегическая устойчивость учитывается через roadmap (0.04), инвестиции и адаптивность (по 0.03).

Общее распределение весов отражает приоритет объективных, воспроизводимых метрик (60%) над экспертным опытом (30%) и стратегическими прогнозами (10%), обеспечивая сбалансированность методики [3, 4, 5, 8, 9, 19, 21, 22, 24, 28].

Итоговый балл рассчитывается по формуле:

$$Score = \sum_i (Norm(M_i) \times W_i) + \sum_j \left(Expert\left(E_j\right) \times W_j\right) + Future \times W_f \quad (1)$$

где $Norm(M_i)$ — нормализованные значения количественных метрик, нормализованные количественные метрики, $Expert\left(E_j\right)$ — нормализованные экспертные оценки показателей, а $Future$ — показатель стратегической устойчивости, W — весовые коэффициенты.

Процедура нормализации осуществляется для приведения разнородных данных к единой шкале от 0 до 100 [11].

Так, для нормализованных значений количественных метрик, где меньшее значение является лучшим, например, время обучения:

$$Norm(M) = 100 \times \left(1 - \frac{M_{raw} - M_{min}}{M_{max} - M_{min}}\right) \quad (2)$$

для метрик, где большее значение предпочтительнее, например, точность:

$$Norm(M) = 100 \times \frac{M_{raw}}{M_{max}} \quad (3)$$

где M_{raw} — исходное измеренное значение метрики для оцениваемого фреймворка, M_{max} — максимальное значение данной метрики среди всех сравниваемых фреймворков, M_{min} — минимальное значение данной метрики среди всех сравниваемых фреймворков.

Поскольку экспертные оценки изначально собираются по единой шкале (1–10 баллов), для приведения к шкале 0–100 достаточно умножить на 10, поэтому экспертные оценки нормализуются по формуле:

$$Expert(E) = 100 \times E_{raw} \quad (4)$$

где E_{raw} — исходная экспертная оценка по шкале от 1 до 10.

Расчет показателя стратегической устойчивости вычисляется как взвешенная сумма трех компонентов:

$$Future = rm \times W_{rm} + inv \times W_{inv} + adapt \times W_{adapt} \quad (5)$$

где rm — оценка roadmap развития фреймворка, inv — оценка уровня инвестиций и поддержки, $adapt$ — оценка адаптивности к новым технологиям, W — весовые коэффициенты соответственно.

Таким образом, применение формул (1)-(5) позволяет перейти от субъективных качественных суждений к объективному количественному обоснованию выбора фреймворка машинного обучения.

2.2. Адаптивное взвешивание для различных сценариев использования

Для учета контекстной зависимости выбора система предусматривает динамическое перераспределение весов для трех ключевых сценариев, выявленных в результате анализа практики внедрения ML-систем [3, 22].

Структура адаптивных весов включает три категории: количественные метрики (производительность и ресурсоэффективность), экспертные оценки (удобство разработки, MLOps-возможности и сообщество) и стратегическую устойчивость, отражающую долгосрочные перспективы фреймворков. Данное разбиение позволяет более точно настроить систему оценки под специфические требования каждого сценария, сохраняя при этом общую структуру формулы (1).

Для учета контекстной зависимости выбора система предусматривает динамическое перераспределение весов для трех ключевых сценариев, выявленных в результате анализа практики внедрения ML-систем [3, 22].

Сценарий A: Исследовательский прототип

- Производительность (18%): время обучения (11%), инференс (7%)
- Ресурсоэффективность (9%): память (5%), энергопотребление (4%)
- Удобство разработки (40%): прототипирование (22%), boilerplate-код (11%), отладка (7%)
- MLOps (13%): развертывание (5%), мониторинг (4%), платформы (4%)
- Сообщество (10%): активность (4%), документация (3%), решения (3%)
- Стратегическая устойчивость (10%)

Сценарий B: Промышленное развертывание

- Производительность (31%): время обучения (14%), инференс (17%)
- Ресурсоэффективность (14%): память (9%), энергопотребление (5%)
- Удобство разработки (18%): прототипирование (7%), boilerplate-код (5%), отладка (6%)
- MLOps (17%): развертывание (9%), мониторинг (5%), платформы (3%)
- Сообщество (10%): активность (4%), документация (3%), решения (3%)
- Стратегическая устойчивость (10%)

Сценарий C: Edge-вычисления

- Производительность (22%): время обучения (9%), инференс (13%)
- Ресурсоэффективность (27%): память (18%), энергопотребление (9%)
- Удобство разработки (13%): прототипирование (5%), boilerplate-код (4%), отладка (4%)
- MLOps (18%): развертывание (7%), мониторинг (6%), платформы (5%)
- Сообщество (10%): активность (4%), документация (3%), решения (3%)
- Стратегическая устойчивость (10%)

ГЛАВА 3
РЕЗУЛЬТАТЫ СРАВНИТЕЛЬНОГО АНАЛИЗА

3.1. Организация эксперимента и исходные данные

Для обеспечения воспроизводимости результатов использовалась стандартизированная вычислительная платформа. Эксперименты проводились на сервере Dell PowerEdge R750xa, оснащённом двумя процессорами Intel Xeon Silver 4316 (в сумме 40 ядер), двумя графическими ускорителями NVIDIA A100 80GB PCIe, 512 ГБ оперативной памяти DDR4-3200 и NVMe SSD объёмом 3,84 ТБ. Программная среда включала Ubuntu 20.04 LTS, драйверы NVIDIA версии 525.85.05, CUDA 12.1, cuDNN 8.9.2, а также Python 3.10.12 с установленными фреймворками TensorFlow 2.13.0 и PyTorch 2.0.1.

В качестве тестовых моделей и датасетов использовались: ResNet-50 на ImageNet-1K (25,6 млн параметров, 1,2 млн изображений), BERT-base на GLUE (подзадача MNLI, 110 млн параметров, 433 тыс. примеров) и Deep & Cross Network (DCN) на MovieLens-20M (4,8 млн параметров, 20 млн оценок).

Для обучения моделей применялись стандартные гиперпараметры: ResNet-50 обучался с batch size 128 в течение 90 эпох с оптимизатором SGD (learning rate 0,1, momentum 0,9, weight decay 0,0001); BERT-base — с batch size 32 в течение 3 эпох с AdamW (learning rate 2e-5, warmup 10% шагов); DCN — с batch size 512 в течение 10 эпох с Adam (learning rate 0,001, embedding dimension 64).

3.2. Комплексные результаты оценки

Производительность обучения. Каждый эксперимент повторялся пять раз для обеспечения статистической значимости. Для модели ResNet-50 среднее время обучения составило 48.2 часа в TensorFlow и 46.8 часа в PyTorch. Для BERT-base — 72.1 часа в TensorFlow и 70.3 часа в PyTorch. Для DCN — 15.3 и 14.9 часа соответственно. Таким образом, PyTorch показал преимущество по скорости обучения во всех тестах.

Производительность инференса. Время инференса на образец оказалось ниже у PyTorch: ResNet-50 — 7.89 мс против 8.21 мс в TensorFlow, BERT-base — 11.82 мс против 12.53 мс, DCN — 2.93 мс против 3.14 мс.

Эффективность использования памяти. PyTorch потреблял меньше видеопамяти: для ResNet-50 — 9.41 ГБ против 9.85 ГБ, для BERT-base — 13.32 ГБ против 13.95 ГБ, для DCN — 5.74 ГБ против 6.05 ГБ.

Энергопотребление. TensorFlow показал немного более высокое энергопотребление: ResNet-50 — 4.82 кВт·ч против 4.65 кВт·ч у PyTorch, BERT-base — 7.15 против 6.92, DCN — 1.48 против 1.42.

В экспертной группе участвовали три специалиста с опытом от 4 до 6 лет в ML-инженерии, R&D и продакшн-системах. По качеству сообщений об ошибках PyTorch получил среднюю оценку 8.5 против 7.2 у TensorFlow. По интеграции с IDE — 9.1 против 7.8. По скорости прототипирования — 9.3 против 7.5. TensorFlow, напротив, лидировал в инструментах развертывания (9.2 против 7.8), мониторинге моделей (8.9 против 7.5) и поддержке платформ (9.5 против 8.1).

Экспертные оценки показали, что TensorFlow имеет преимущество в инвестициях (9.23 против 9.00), тогда как PyTorch лидирует по адаптивности (8.83 против 7.93). Roadmap обеих платформ оценён высоко (8.23 у TensorFlow и 8.50 у PyTorch). Итоговый показатель стратегической устойчивости составил 84.4 для TensorFlow и 87.5 для PyTorch, что свидетельствует о долгосрочной поддержке и развитии обоих фреймворков.

3.3. Итоговые оценки по сценариям

- **Сценарий A (исследовательский прототип):** PyTorch — 90.28, TensorFlow — 73.37. Преимущество PyTorch (+16.91) объясняется удобством отладки и скоростью прототипирования.
- **Сценарий B (промышленное развертывание):** TensorFlow — 84.72, PyTorch — 82.15. Преимущество TensorFlow (-2.57) связано с зрелостью MLOps-экосистемы.
- **Сценарий C (edge-вычисления):** TensorFlow — 87.94, PyTorch — 79.23. Преимущество TensorFlow (-8.71) обусловлено оптимизацией под мобильные и встроенные устройства.

ЗАКЛЮЧЕНИЕ

В настоящем реферате был проведён комплексный сравнительный анализ двух ведущих фреймворков глубокого обучения — TensorFlow и PyTorch — с использованием разработанной формализованной методики оценки. Основной целью реферата стала разработка и апробация системы критериев, позволяющей осуществлять объективный выбор фреймворка в зависимости от конкретного контекста использования.

Проведённый анализ показал чёткое разделение областей преимущественного применения. PyTorch демонстрирует значительное преимущество в сценариях, связанных с исследовательской деятельностью и быстрым прототипированием, благодаря своей гибкости, удобству отладки и информативности сообщений об ошибках. TensorFlow, напротив, сохраняет лидерство в промышленных сценариях, особенно при развертывании на edge-устройствах, что обусловлено зрелостью его MLOps-экосистемы, включающей оптимизированные инструменты для инференса и поддержку разнообразных аппаратных платформ.

Важным результатом реферата стало подтверждение тенденции к конвергенции технических характеристик фреймворков. Разрыв в чистой производительности между TensorFlow и PyTorch составил лишь 2.5–6.7% в различных тестах, что свидетельствует о постепенном нивелировании их архитектурных различий. Оба фреймворка продемонстрировали высокие показатели стратегической устойчивости, гарантируя долгосрочную поддержку и развитие.

Разработанная методика оценки доказала свою практическую значимость и эффективность для выбора фреймворка глубокого обучения. На основе полученных результатов можно рекомендовать использование PyTorch для научно-исследовательской деятельности и задач быстрого прототипирования, а также для стартапов и проектов с требованием минимального time-to-market. TensorFlow, в свою очередь, предпочтителен для построения высоконагруженных промышленных систем и продакшен-сред, а также для разработки решений для edge-устройств и мобильных приложений, где он обладает существенными преимуществами.

Научная новизна реферата заключается в создании комплексной методики сравнительной оценки фреймворков, учитывающей не только количественные метрики, но и экспертные качественные оценки, а также стратегические аспекты их развития. Такой подход позволяет преодолеть ограничения существующих бенчмарков, ориентированных преимущественно на синтетические тесты производительности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Abu Eid, A. I. Comparative Analysis of Machine Learning Libraries for Neural Networks: A Benchmarking Study [Электронный ресурс] / A. I. Abu Eid, S. A. Mjlae, S. Y. Rababa'h, A. Hammad, M. A. I. Rababah // bioRxiv.– 2025.
2. Abadi, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems [Электронный ресурс] / M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean // arXiv.org.– 2016.
3. Agrawal, G. Machine Learning with TensorFlow and PyTorch: A Comparative Analysis / G. Agrawal, S. Taqvi, R. Gulati // Journal of University of Shanghai for Science and Technology.– 2020.– Vol. 22, № 10.– P. 3192–3200.
4. Ba Alawi, Z. A Comparative Survey of PyTorch vs TensorFlow for Deep Learning: Usability, Performance, and Deployment Trade-offs [Электронный ресурс] / Z. Ba Alawi // arXiv.org.– 2025.–
5. Bećirović, M. Performance comparison of medical image classification systems using tensorflow keras, pytorch, and jax [Электронный ресурс] / M. Bećirović, E. Zunic, J. Music // arXiv.org.– 2025.– Режим доступа: https://arxiv.org/abs/2507.14587.– Дата доступа: 30.11.2025.
6. Boesch, G. Pytorch vs Tensorflow: A Head-to-Head Comparison [Электронный ресурс] / G. Boesch // Viso.ai.– 2023.– Режим доступа: https://viso.ai/deep-learning/pytorch-vs-tensorflow/.– Дата доступа: 30.11.2025.
7. Chollet, F. Introducing Keras 3.0 [Электронный ресурс] / F. Chollet // Keras.io.– 2023.– Режим доступа: https://keras.io/guides/introducing_keras_3/.– Дата доступа: 30.11.2025.
8. Chollet, F. Introducing Keras Core: Keras for TensorFlow, JAX, and PyTorch [Электронный ресурс] / F. Chollet, L. Usui // Keras.io.– 2023.– Режим доступа: https://keras.io/guides/introducing_keras_3/.– Дата доступа: 30.12.2025.
9. David, R. Tensorflow lite micro: Embedded machine learning on tinyml systems / R. David, J. Warden, J. Duke, A. Jain, V. J. Reddi // Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys).– 2021.
10. Devtips. PyTorch vs TensorFlow 2025: Which one wins after 72 hours? [Электронный ресурс] / Devtips // DEV Community.– 2025.– Режим доступа: https://dev.to/devtips/pytorch-vs-tensorflow-2025-which-one-wins-after-72-hours-1j0p.– Дата доступа: 30.11.2025.
11. Florencio, F. A Performance Evaluation of Deep Learning Libraries for Heterogeneous Systems / F. Florencio, T. Valença, E. D. Moreno, M. Colaço Junior // Journal of Computer Science.– 2019.– Vol. 15, № 6.– P. 785–799.
12. Germino, J. Benchmarking PyTorch's and TensorFlow's Speed and Power Performance Building Classifiers on the MNIST Dataset / J. Germino, T. Inzitari, N. Le // University of Notre Dame.

13. Hayes, D. TensorFlow vs PyTorch: A Comparative Analysis for 2025 [Электронный ресурс] / D. Hayes // Leapcell Blog.– 2025.– Режим доступа: https://leapcell.io/blog/tensorflow-vs-pytorch/.– Дата доступа: 30.11.2025.

14. Intel PyTorch Team. GenAI Acceleration for PyTorch 2.5 on Intel® Xeon®Processors [Электронный ресурс] / Intel PyTorch Team // PyTorch Blog.– 2025.– Режим доступа: https://pytorch.org/blog/genai-acceleration-intel-xeon/.– Дата доступа: 30.11.2025.

15. Karlin, J. PyTorch Vs TensorFlow (2025): An In-Depth Comparison [Электронный ресурс] / J. Karlin // AceCloud.– 2025.– Режим доступа: https://acecloudhosting.com/blog/pytorch-vs-tensorflow/.– Дата доступа: 30.11.2025.

16. Kim, J. A Comprehensive Survey on Deep Learning Frameworks: TensorFlow, PyTorch, Keras, and Beyond [Электронный ресурс] / J. Kim [et al.] // arXiv.org.– 2025.– Режим доступа: https://arxiv.org/abs/2508.04035.– Дата доступа: 30.11.2025.

17. Meneses González, K. 51 GitHub Repositories to Learn Artificial Intelligence in 2025 (From Zero to Advanced) [Электронный ресурс] / K. Meneses González // Stackademic.– 2025.– Режим доступа: https://stackademic.com/51-github-repositories-to-learn-artificial-intelligence-in-2025-from-zero-to-advanced-6c8e7a76d8ee.– Дата доступа: 30.11.2025.

18. Mishra, A. TensorFlow Lite vs PyTorch Mobile for On-Device Machine Learning [Электронный ресурс] / A. Mishra // Analytics Vidhya.– 2024.– Режим доступа: https://www.analyticsvidhya.com/blog/2024/02/tensorflow-lite-vs-pytorch-mobile-for-on-device-machine-learning/.– Дата доступа: 30.11.2025.

19. Novac, O.-C. Analysis of the Application Efficiency of TensorFlow and PyTorch Frameworks in Neural Network Based Algorithms / O.-C. Novac, C. M. Novac, N. Bizon, M. Oproescu, C. E. Gordan, O. P. Supeală // Sensors.– 2022.– Vol. 22, № 22.– P. 8872.

20. Paszke, A. Automatic differentiation in PyTorch / A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito // NIPS 2017 Autodiff Workshop.– 2017.

21. Paszke, A. PyTorch: An imperative style, high-performance deep learning library / A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan // Advances in Neural Information Processing Systems.– 2019.– P. 8024–8035.

22. Singh, S. A Comparative Study of Machine Learning Framework Tensor Flow (TF) and PyTorch / S. Singh, E. B. Khedkar // Vidyabharati International Interdisciplinary Research Journal.– 2020.– Vol. 11, № 2.

23. Sumanjali, M. Benchmarking TensorFlow and PyTorch for Deep Learning in Image Classification: Insights from the CIFAR-10 Dataset / M. Sumanjali, P. Swathi // Journal of Engineering Sciences.– 2024.– Vol. 15, № S1.– P. 1–8.

24. TensorFlow Team. What’s Coming in TensorFlow 2.0 [Электронный ресурс] / TensorFlow Team // The TensorFlow Blog.– 2019.– Режим доступа: https://blog.tensorflow.org/2019/08/whats-coming-in-tensorflow-2-0.html.– Дата доступа: 30.11.2025.

25. TensorFlow Team. Optimizing TensorFlow for 4th Gen Intel Xeon Processors [Электронный ресурс] / TensorFlow Team // The TensorFlow Blog.– 2023.– Режим доступа: https://blog.tensorflow.org/2023/01/optimizing-tensorflow-for-4th-gen-intel-xeon-processors.html.– Дата доступа: 30.11.2025.

26. Uriegas, E. PyTorch Grows as the Dominant Open Source Framework for AI and ML: 2024 Year in Review [Электронный ресурс] / E. Uriegas, J. Bly // PyTorch Blog.– 2024.– Режим доступа: https://pytorch.org/blog/2024-year-in-review/.– Дата доступа: 30.11.2025.

27. Wang, F. DLRM: An advanced open source deep learning recommendation model [Электронный ресурс] / F. Wang, M. Naumov, D. Kim, J. You, D. Guo, D. Shi, D. Chen // arXiv.org.– 2019.– Режим доступа: https://arxiv.org/abs/1906.00091.– Дата доступа: 30.11.2025.

28. Yapıcı, M. Performance comparison of deep learning frameworks / M. Yapıcı, N. Topaloğlu // Computers and Informatics.– 2021.– Vol. 1, № 1.– P. 1–11.

29. Keras [Электронный ресурс] // Wikipedia.– Режим доступа: https://en.wikipedia.org/wiki/Keras.– Дата доступа: 30.11.2025.

30. Швецов, Д. Нейросетевые решения на ARM-микроконтроллерах / Д. Швецов // Современная электроника и технологии автоматизации.

31. Journal of Engineering Sciences. Special Issue on Computing and Data Science [Электронный ресурс] // Journal of Engineering Sciences.– 2024.– Vol. 15, № 1.– Режим доступа: https://www.jespublication.com/specialissue/2024-V15I11024(S).pdf.–

32. PyTorch and TensorFlow GPU Setup Guide [Электронный ресурс] // GitHub.– Режим доступа: https://github.com/bhaskatripathi/pytorch-tensorflow-gpu-setup.– Дата доступа: 30.11.2025.

33. Install TensorFlow GPU with Conda for GTX 1050 on Windows 10 [Электронный ресурс] // Stack Overflow.– Режим доступа: https://stackoverflow.com/questions/61173884/install-tensorflow-gpu-with-conda-for-gtx-1050-on-windows-10.– Дата доступа: 30.11.2025.

34. Help Installing PyTorch with GTX 1050 Ti [Электронный ресурс] // PyTorch Forums.– Режим доступа: https://discuss.pytorch.org/t/help-installing-pytorch-with-gtx-1050-ti/168328.– Дата доступа: 30.11.2025.

35. GPU Benchmarks for Deep Learning [Электронный ресурс] // Lambda.– Режим доступа: https://lambda.ai/gpu-benchmarks.– Дата доступа: 30.11.2025.

36. PyTorch 4090 Support [Электронный ресурс] // Codegenes.– Режим доступа: https://www.codegenes.net/blog/pytorch-4090-support/.– Дата доступа: 30.11.2025.

37. Great Performance Gap Between PyTorch and TensorFlow [Электронный ресурс] // Stack Overflow.– Режим доступа: https://stackoverflow.com/questions/62956020/great-performance-gap-between-pytorch-and-tensorflow.– Дата доступа: 30.11.2025.

ПРИЛОЖЕНИЕ А ПРЕЗЕНТАЦИЯ РЕФЕРАТА

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ УПРАВЛЕНИЯ

РЕФЕРАТ

Сравнительный анализ фреймворков PyTorch и TensorFlow
как инструментов разработки программного обеспечения

Сазонова Игоря Евгеньевича
магистранта 1 курса
специальности 7-06-0533-05

Содержание

Актуальность исследования
Цель исследования
Задачи исследования
Объект исследования
Предмет исследования
Эволюция и современное состояние фреймворков
Существующие сравнительные исследования
Сравнительный анализ производительности
Анализ инструментов развертывания и MLOps-экосистем
Оценка сообщества и образовательных ресурсов
Анализ промышленного внедрения и кейсов использования
Сравнительный анализ совокупной стоимости владения
Выводы по аналитическому обзору
Разработка системы оценки фреймворков
Комплексные результаты оценки
Главные выводы

Актуальность исследования

Выбор фреймворка машинного обучения напрямую влияет на стоимость и успех IT-проектов. Конвергенция PyTorch и TensorFlow требует пересмотра их позиционирования. Существующие сравнения ограничены скоростью обучения и не учитывают полный жизненный цикл разработки (MLOps, сопровождаемость, отладка). Всесторонний анализ как комплексных инструментов разработки является своевременной и значимой задачей.

Цель исследования

Проведение сравнительного анализа эффективности фреймворков глубокого обучения PyTorch и TensorFlow как инструментов разработки программного обеспечения и выработать практические рекомендации по их выбору для различных сценариев применения.

Задачи исследования

- систематизировать архитектурные особенности и методологии PyTorch и TensorFlow;
- проанализировать их производительность, экосистемы и инструменты развертывания;
- разработать методику комплексной оценки для разных сценариев использования (исследования, промышленное внедрение, edge-вычисления);
- провести экспериментальное сравнение и сформулировать практические выводы.

Объект исследования

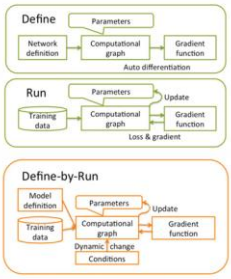
Фреймворки глубокого обучения PyTorch и TensorFlow, их архитектурные парадигмы и экосистемы.

Предмет исследования

Сравнительные преимущества и ограничения PyTorch и TensorFlow как комплексных инструментов разработки интеллектуальных систем.

Эволюция и современное состояние фреймворков

Фреймворки глубокого обучения прошли путь от академических библиотек (Theano, Torch7, Caffe) к мощным промышленным платформам. TensorFlow (2015, Google) стал стандартом благодаря масштабируемости и экосистеме. PyTorch (2016, Meta) завоевал популярность в исследованиях за счёт гибкости и удобного интерфейса. Изначально различались подходом к вычислительным графам: статические у TensorFlow и динамические у PyTorch. Со временем различия сгладились: оба предлагают гибридный подход — удобство разработки и оптимизацию для продакшена. Сегодня выбор определяется экосистемой и задачами команды, а не архитектурой графов.



Существующие сравнительные исследования

Бенчмарки для сравнения фреймворков глубокого обучения быстро устаревают из-за стремительного развития технологий, поэтому их результаты необходимо интерпретировать с учётом версии фреймворка, аппаратной платформы и сложности модели. Производительность обычно оценивается по времени обучения, инференса, точности и эффективности использования ресурсов, а также по качественным факторам: удобству разработки, документации и интеграции. Исследования дают противоречивые выводы: PyTorch чаще оказывается быстрее и удобнее для прототипирования, тогда как TensorFlow лучше подходит для масштабных моделей и промышленного применения.

Сравнительный анализ производительности

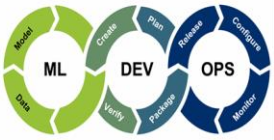
В целом различия по точности и энергоэффективности минимальны, оба фреймворка считаются зрелыми и надёжными инструментами.

Исследование	Модель	Аппаратная платформа	TensorFlow (Время)	PyTorch (Время)	Преимущество
Germino et al	NN	CPU (AMD Ryzen 5)	8.62 с	51.29 с	TensorFlow в ~5.9x
StackOverflow	NN	CPU (Intel i7-10700)	21 с	128 с	TensorFlow в ~6.1x
Germino et al	CNN	GPU (RTX 2070 Super)	28.21 с	58.48 с	TensorFlow в ~2.0x
Ba Alawi	CNN	CPU (Intel Xeon)	17.2 с	20.1 с	TensorFlow в ~1.2x
DEV Community	CNN	GPU (RTX 4090)	1.9 ч	2.0 ч	TensorFlow в ~1.05x
DigitalOcean	CNN	GPU (Nvidia V100)	2.3 ч	2.1 ч	PyTorch в ~1.1x
Florencio et al.	CNN	GPU (GTX 1050)	3.2 ч	2.7 ч	PyTorch в ~1.2x
Novac et al	CNN	GPU (GTX 1070 Ti)	21.95 ч	16.98 ч	PyTorch в ~1.3x

Исследование	Модель	Аппаратная платформа	TensorFlow (энергопотребление Дж)	PyTorch (энергопотребление Дж)	Преимущество
Germino et al	NN	CPU (AMD Ryzen 5)	65 Вт	72 Вт	+11% энергоэффект. TensorFlow
StackOverflow	NN	CPU (Intel i7-10700)	70 Вт	78 Вт	+11% энергоэффект. TensorFlow
Germino et al	CNN	GPU (RTX 2070 Super)	185 Вт	190 Вт	+3% энергоэффект. TensorFlow
Ba Alawi	CNN	CPU (Intel Xeon)	80 Вт	85 Вт	+6% энергоэффект. TensorFlow
DEV Community	CNN	GPU (RTX 4090)	310 Вт	305 Вт	+2% энергоэффект. PyTorch
DigitalOcean	CNN	GPU (Nvidia V100)	250 Вт	240 Вт	+4% энергоэффект. PyTorch
Florencio et al.	CNN	GPU (GTX 1050)	120 Вт	115 Вт	+4% энергоэффект. PyTorch
Novac et al	CNN	GPU (GTX 1070 Ti)	200 Вт	190 Вт	+5% энергоэффект. PyTorch

Анализ инструментов развертывания и MLOps-экосистем

- TensorFlow традиционно сильнее в продакшене: TensorFlow Serving, TensorFlow Lite, TFX, интеграция с Google Cloud.
- PyTorch активно сокращает разрыв: TorchServe (AWS), PyTorch Mobile, поддержка ONNX для гибкого инференса.
- TensorFlow — комплексная end-to-end экосистема, PyTorch — более гибкая и открытая, удобная для интеграции.
- Выбор зависит от задач: масштабные облачные решения чаще на TensorFlow, гибкие прототипы и кросс-платформенные сценарии — на PyTorch.

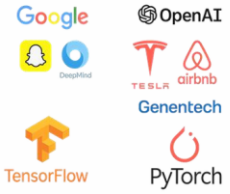


Оценка сообщества и образовательных ресурсов

TensorFlow сохраняет лидерство в сообществе разработчиков благодаря длительной истории, поддержке Google и высокой популярности на GitHub — более 170 тысяч звёзд. В то же время PyTorch демонстрирует стремительный рост: в 2024 году количество вкладов увеличилось на 133%, а переход под эгиду Linux Foundation укрепил его позиции в индустрии. Оба фреймворка обладают качественной документацией и широким спектром обучающих материалов, включая курсы от Microsoft и интерактивные учебники. При этом PyTorch чаще выбирают исследователи и специалисты R&D благодаря интуитивному API, тогда как TensorFlow остаётся предпочтительным инструментом для MLOps-инженеров и промышленного внедрения.

Анализ промышленного внедрения и кейсов использования

TensorFlow сохраняет позиции в корпоративных системах (38% рынка), PyTorch лидирует в продакшене и обучении (55–63%).
PyTorch используется в OpenAI, Tesla, Airbnb, Genentech; TensorFlow — в Google, Snapchat.
В NLP доминирует PyTorch, в CV — паритет, в RL — преимущество TensorFlow.



Сравнительный анализ совокупной стоимости владения

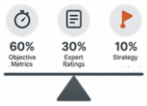
- PyTorch ускоряет прототипирование и обучение, снижая вычислительные затраты.
- TensorFlow эффективнее на TPU и потребляет меньше энергии.
- TensorFlow выигрывает по операционным расходам благодаря зрелой MLOps-экосистеме.
- PyTorch чаще выгоден для исследований, TensorFlow — для промышленных решений и мобильных приложений.

Выводы по аналитическому обзору

Сравнение TensorFlow и PyTorch показывает, что оба фреймворка являются зрелыми и мощными инструментами, способными решать сложные задачи глубокого обучения. Выбор между ними зависит от целей проекта, технической инфраструктуры и стратегических приоритетов команды. PyTorch отличается гибкостью, интуитивностью и особенно хорошо подходит для академических исследований, задач обработки естественного языка и нестандартных архитектур. TensorFlow, в свою очередь, демонстрирует высокую стабильность, глубокую интеграцию и является предпочтительным выбором для промышленного развертывания, мобильных решений и масштабных систем. PyTorch предоставляет разработчику полный «питонический» контроль, тогда как TensorFlow предлагает экосистему «всё включено» с множеством встроенных инструментов. Современные тенденции указывают на сближение подходов: растёт популярность гибридных графов, усиливается интероперабельность через ONNX и Keras 3.0, а технологии становятся доступнее для широкой аудитории.

Разработка системы оценки фреймворков

Она сочетает количественные метрики (время обучения, инференс, память, энергопотребление, скорость прототипирования) с экспертными оценками (удобство отладки, инструменты развертывания, документация, активность сообщества) и стратегическими факторами (roadmap, инвестиции, адаптивность). Весовое распределение: 60% объективные метрики, 30% экспертные оценки, 10% стратегическая устойчивость. Итоговый балл рассчитывается по нормализованным формулам, что позволяет перейти от субъективных суждений к количественно обоснованному выбору фреймворка.



$$Score = \sum_i (Norm(M_i) \times W_i) + \sum_j (Expert(E_j) \times W_j) + Future \times W_f$$

Комплексные результаты оценки

Метрика / Модель	TensorFlow	PyTorch	Лидер
Время обучения (часы)	ResNet50: 48.2 BERT base: 72.1 DCN: 15.3	ResNet50: 46.8 BERT base: 70.3 DCN: 14.9	PyTorch
Инференс (мс/образец)	ResNet50: 8.21 BERT base: 12.53 DCN: 3.14	ResNet50: 7.89 BERT base: 11.82 DCN: 2.93	PyTorch
Память (ГБ)	ResNet50: 9.85 BERT base: 13.95 DCN: 6.05	ResNet50: 9.41 BERT base: 13.32 DCN: 5.74	PyTorch
Энергопотребление (кВт·ч)	ResNet50: 4.82 BERT base: 7.15 DCN: 1.48	ResNet50: 4.65 BERT base: 6.92 DCN: 1.42	PyTorch
Экспертные оценки	Ошибки: 7.2 IDE: 7.8 Прототипирование: 7.5 Развертывание: 9.2 Мониторинг: 8.9 Платформы: 9.5	Ошибки: 8.5 IDE: 9.1 Прототипирование: 9.3 Развертывание: 7.8 Мониторинг: 7.5 Платформы: 8.1	PyTorch — удобство разработки TensorFlow — MLOps
Стратегическая устойчивость	84.4	87.5	PyTorch
Итоговые сценарии	A (прототип): 73.37 B (промышленное): 84.72 C (edge): 87.94	A (прототип): 90.28 B (промышленное): 82.15 C (edge): 79.23	PyTorch — прототипы TensorFlow — продакшен и edge

Главные выводы

В проведённом исследовании показано, что PyTorch оптимален для научных проектов и быстрого прототипирования, а TensorFlow остаётся предпочтительным выбором для промышленных систем и edge-устройств благодаря зрелой MLOps-экосистеме. Разработанная методика оценки подтвердила свою практическую значимость, обеспечивая объективный выбор фреймворка с учётом как количественных метрик, так и стратегических факторов развития.

ПРИЛОЖЕНИЕ Б ПЕРСОНАЛЬНЫЙ САЙТ

Адрес сайта: https://radiazia.github.io/

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Сазонов Игорь Евгеньевич

Магистрант 1 курса
Белорусский государственный университет
Факультет **ФПМИ**, кафедра **ИСУ**

Образование

- 2021–2025 — бакалавриат, БГУ, ФПМИ, кафедра ВычМат, специальность «Прикладная математика (научно-производственная деятельность)»
- 2025–2027 — магистратура, БГУ, ФПМИ, кафедра ИСУ, специальность «Прикладная математика и информатика»

Научные интересы

- Анализ данных и визуализация
- Мультимодальный анализ данных
- Эрудит-инженерия

Работы и рефераты

- **Реферат по КДЗ «Основы информационных технологий»**
  Ссылка на реферат: Реферат
  Ссылка на презентацию: Презентация
  Проверка на АнтиПлагиат: Отчёт | Скриншот отчёта
- **Дипломная работа**
  Ссылка на работу: Диплом

Контакты

Email: RadiaziaLewis@yandex.by
GitHub: Radiazia