Сравнение различных библиотек для машинного обучения: scikit-learn, TensorFlow и PyTorch

Содержание

[Обзор проекта 1](#__RefHeading___1)

[Структура проекта 2](#__RefHeading___2)

[Классификация и регрессия с использованием библиотеки Scikit-learn 2](#__RefHeading___12)

[Классификация с использованием библиотеки для машинного обучения TensorFlow 4](#__RefHeading___4)

[Классификация с использованием библиотеки для машинного обучения PyTorch 7](#__RefHeading___10)

[Сравнительный анализ библиотек 10](#__RefHeading___6)

[Заключение 12](#__RefHeading___7)

[Приложение 1. Список сайтов используемых библиотек и ссылки на ресурсы 13](#__RefHeading___8)

[Приложение 2. Список необходимых библиотек 14](#__RefHeading___9)

# Обзор проекта

Реализовать задачи классификации и регрессии с использованием scikit-learn, TensorFlow и PyTorch, сравнить их производительность и удобство использования.

Цель исследования: провести сравнительный анализ библиотек scikit-learn, TensorFlow и PyTorch в контексте решения задач классификации и регрессии, оценить их производительность и удобство использования.

Для достижения этой цели поставлены следующие задачи:

1. Изучить теоретические основы машинного обучения, уделяя особое внимание задачам классификации и регрессии.

2. Провести обзор библиотек scikit-learn, TensorFlow и PyTorch, рассмотрев их историю, архитектуру и основные компоненты.

3. Реализовать типовые задачи классификации и регрессии с использованием каждой из рассматриваемых библиотек.

4. Провести сравнительный анализ производительности библиотек на основе скорости обучения, точности предсказаний и использования ресурсов.

5. Оценить удобство использования каждой библиотеки, включая простоту написания кода, качество документации и поддержку сообщества.

6. Сформулировать рекомендации по выбору библиотеки в зависимости от специфики решаемой задачи.

# Структура проекта

Проект состоит из блокнотов google colab в которых реализована задача классификации и регрессии с использованием scikit-learn, TensorFlow и PyTorch :

## Классификация и регрессия с использованием библиотеки Scikit-learn

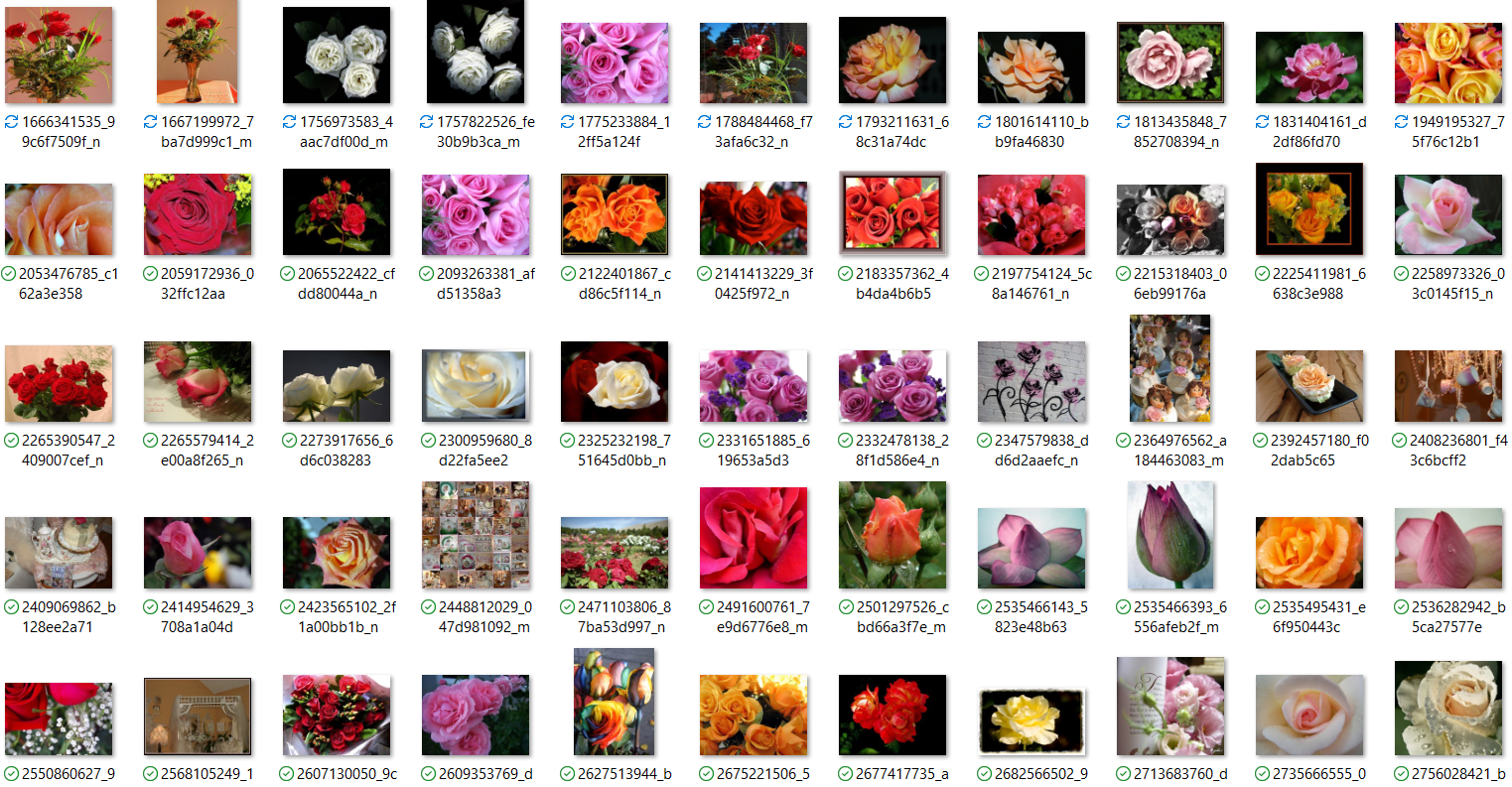
Архитектура Scikit-learn построена вокруг концепции оценщиков (estimators), которые реализуют методы fit() и predict().

Ссылка на проект в google colab:

<https://colab.research.google.com/drive/1XKryA6HhoXraLTwFMiJRDeQdFL6jM3RS?usp=sharing>

Использовался dataset состоящий из фотографий цветов «flower\_photos»: <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

На фото ниже представлен фрагмент датасета фотографий цветов класса «daisy», Рисунок 1:



*Рисунок 1 Фрагмент датасета фотографий цветов класса «roses»*

Всего изображений в датасете: 3670. Разделённых на 5 классов цветов: 'daisy', 'dandelion', 'roses', 'sunflowers', 'tulips'

В качестве примера для предсказания класса нового изображения использовалось фото "Роза крупным планом", которой не было в датасете. Представленная на рисунке 2



*Рисунок 2 Фотографий цветка «Роза крупным планом»*

Ссылка на изображение:

<https://vsegda-pomnim.com/uploads/posts/2022-04/1650490269_13-vsegda-pomnim-com-p-roza-krupnim-planom-foto-17.jpg>

В результате работы программы, которая включает:

1. Загрузку и предобработку данных:

- Изменение размера изображений

- Преобразование в плоские массивы

- Нормализацию данных

2. Обучение модели алгоритмом классификации Random Forest Classifier

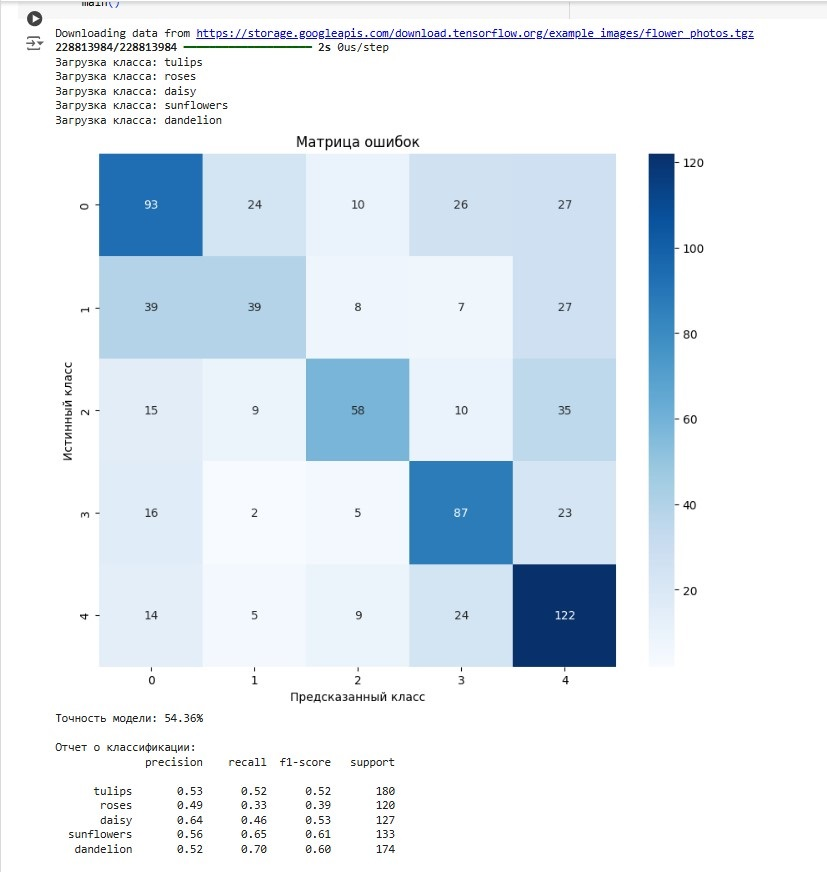
3. Оценку производительности:

4. Визуализацию результатов сравнения моделей

Показала точность модели: 54.36%

Предсказанный класс: roses с уверенностью 36.00%

Отчет о классификации с матрицей ошибок представлен на русинке 3.



*Рисунок 3 Отчёт о классификации с матрицей ошибок*

## Классификация с использованием библиотеки для машинного обучения TensorFlow

TensorFlow – открытая программная библиотека для машинного обучения, разработанная компанией Google.

Особенности TensorFlow:

- Поддержка как классического машинного обучения, так и глубоких нейронных сетей

- Возможность распределенного обучения на нескольких GPU или TPU

- Поддержка мобильных и встраиваемых платформ (TensorFlow Lite)

Ссылка на проект в google colab:

<https://colab.research.google.com/drive/1mYTmwkUVuCvZq8bJT_t3KUGUh1ShgpbV?usp=sharing>

Использовался тот же dataset, для чистоты эксперимента, состоящий из фотографий цветов «flower\_photos»: <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

Основные компоненты программы:

1. Импортированы все необходимые библиотеки

2. Реализована загрузка и распаковка датасета

3. Добавлен подсчет количества изображений

4. Реализована аугментация данных с помощью (для увеличения датасета):

- RandomFlip для горизонтального отражения

- RandomRotation для поворота на 10%

- RandomZoom для случайного приближения

- RandomContrast для изменения контраста

5. Данные разделены на обучающую и тестовую выборки (80/20)

6. Создана модель с тремя блоками свертки, включая:

- Слои Conv2D

- MaxPooling

- Dropout для регуляризации (предотвращения переобучения)

7. Добавлена визуализация результатов обучения

8. Реализована функция для инференса на новых изображениях, для предсказания его класса

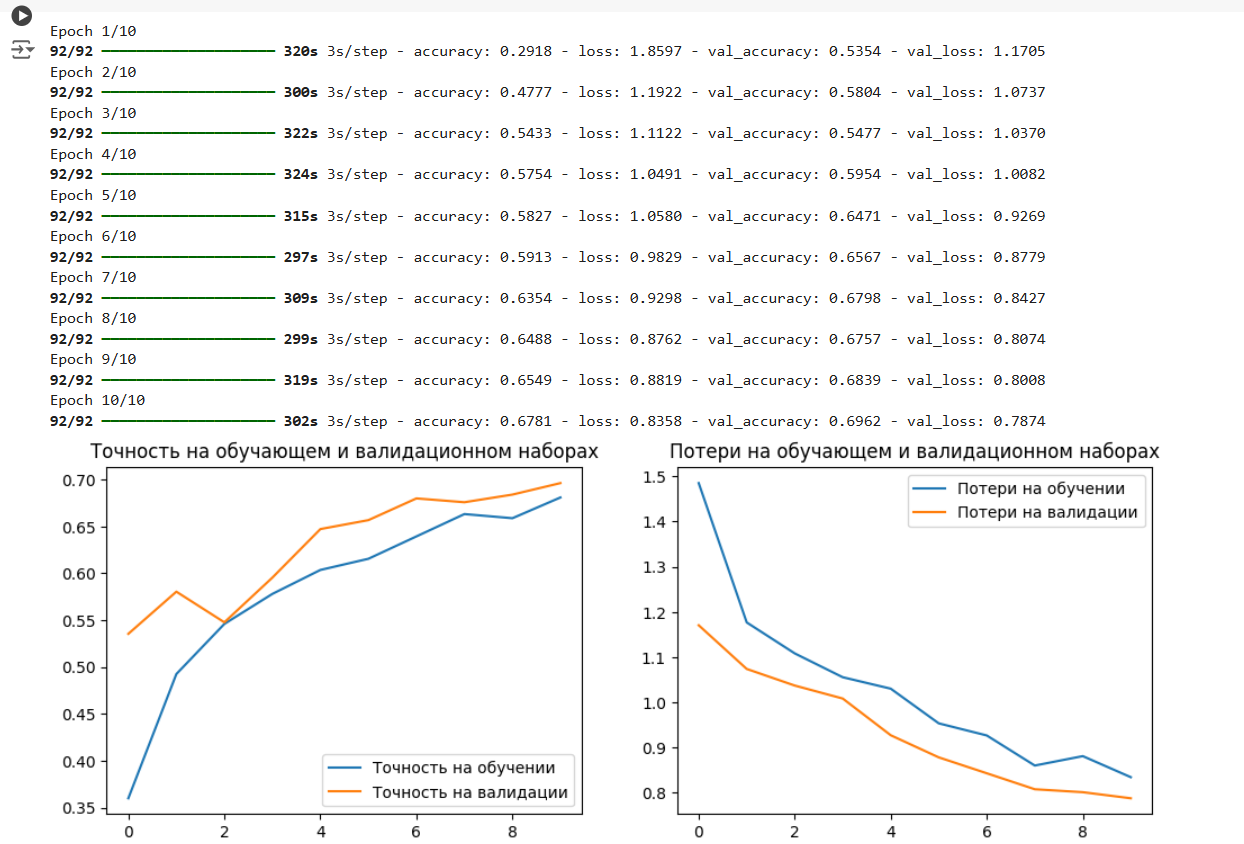
Функция также выводит процентные значения уверенности модели для каждого класса, что помогает лучше понять, как модель "думает".

Реализована визуализация процесса обучения с графиками точности и потерь

Обучающая точность, максимальная: 64.71%

Валидационная точность, максимальная: 92.69%

На рисунке 4 представлен результат работы программы с использованием библиотеки TensorFlow. С результатами обучения на каждой из 10 эпох и графиками точности и потерь на обучении и валидации (тестировании) . Получены хорошие результаты.



*Рисунок 4 Результат работы программы с использованием библиотеки TensorFlow .*

Создана удобная функция для инференса, предсказания класса нового изображения , представленная на рисунке. Которая показывает предсказания для всех классов, визуализирует изображение, выводит уровень уверенности модели.



*Рисунок 5 Результат работы инференса, фото цветка подсолнуха с результатами предсказания*

## Классификация с использованием библиотеки для машинного обучения PyTorch

PyTorch\*\* – открытая библиотека машинного обучения, разработанная Facebook AI Research (FAIR).

Архитектура PyTorch основана на концепции динамических вычислительных графов, которые строятся "на лету" во время выполнения.

Ссылка на данный блокнот:

<https://colab.research.google.com/drive/1rq0rMDt01y8BZ181DBvsuPQxA4DGJZRX?usp=sharing>

Использовался тот же dataset, для чистоты эксперимента, состоящий из фотографий цветов «flower\_photos»: <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

Описание основных компонентов программы с использованием библиотеки для машинного обучения PyTorch:

1. Загрузка и подготовка данных:

* Автоматическая загрузка и распаковка датасета
* Подсчет количества изображений
* Разделение на обучающую и валидационную выборки (80/20)

2. Архитектура модели:

* Использование предобученной ResNet18
* Замораживание весов базовой модели
* Добавление dropout для борьбы с переобучением

3. Аугментация данных:

* Случайное изменение размера и обрезка
* Горизонтальное отражение
* Поворот изображений
* Изменение яркости, контраста и насыщенности

4. Процесс обучения:

* L2 регуляризация
* Планировщик скорости обучения (ReduceLROnPlateau)
* Ранняя остановка
* Сохранение лучшей модели

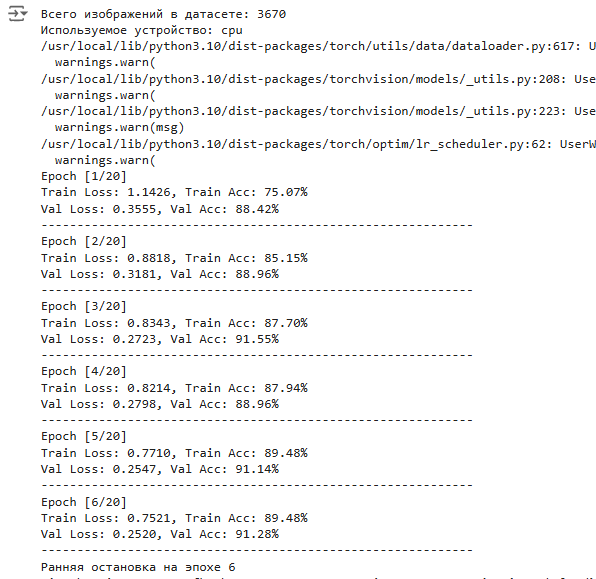
5. Оптимизация:

* Использование оптимизатора AdamW
* Многопоточная загрузка данных

6. Визуализация:

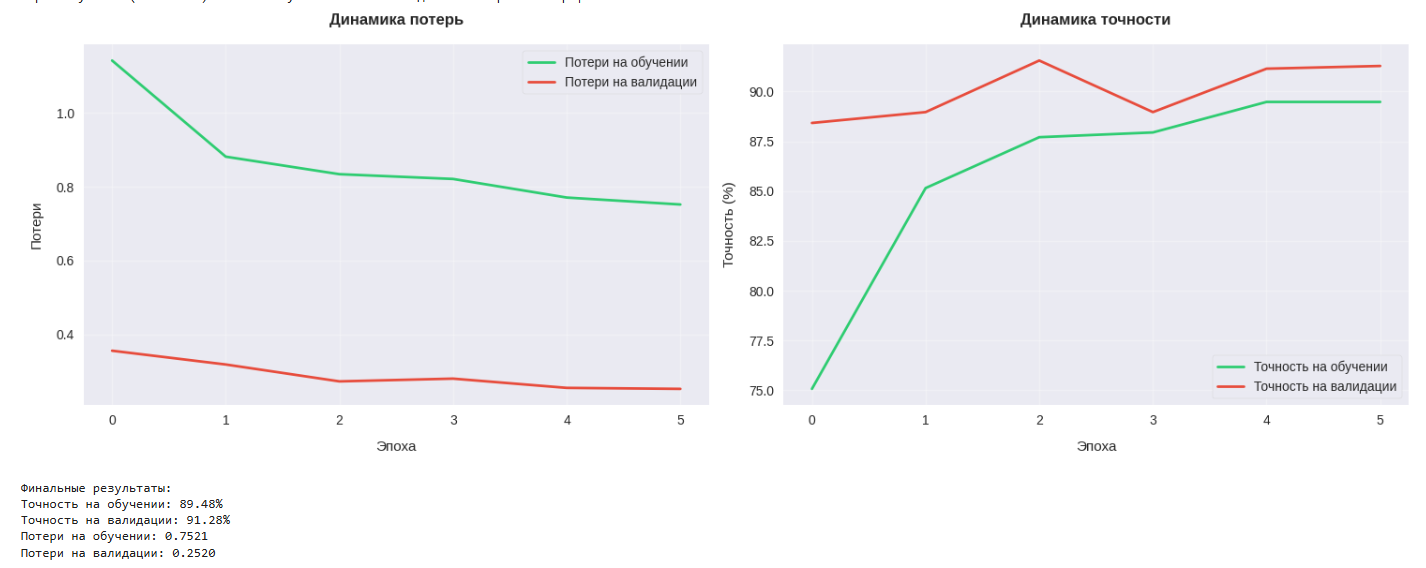
* Графики потерь и точности
* Отдельные кривые для обучающей и валидационной выборок

На рисунке 6 представлен результат работы программы с использованием библиотеки машинного обучения PyTorch, процентное содержание точности предсказания и потерь, по эпохам:



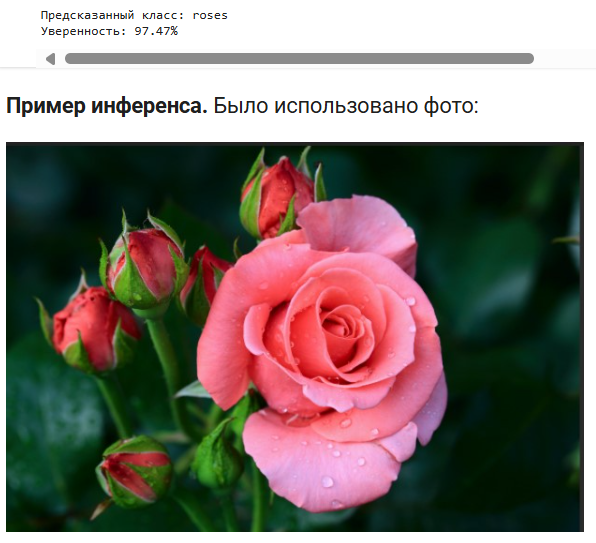
*Рисунок 6 Результат точности предсказания и потерь, по эпохам*

На рисунке 7 представлен результат визуализации, графики потерь и точности с отдельными кривыми для обучающей и валидационной выборок с финальными результатами точности и потерь:



*Рисунок 7 Графики динамики точности и потерь, финальные результаты предсказания*

Для предсказания класса нового изображения было выбрано то же фото, что использовалось в библиотеке Scikit-learn, для сравнения. Результат работы функции представлен на рисунке 7:



*Рисунок 8 Результат работы функции*

**Анализ полученных результатов:**

1. Точность на валидационной выборке достигла ~91% (лучший результат 91.55% на 3-й эпохе)

2. Разрыв между обучающей и валидационной точностью всего ~2%, что говорит о хорошем балансе между обучением и обобщение:

Обучающая точность: 89.48% (последняя эпоха)

Валидационная точность: 91.28% (последняя эпоха)

3. Ранняя остановка сработала на 6-й эпохе, так как:

Лучшая валидационная точность была достигнута на 3-й эпохе (91.55%)

В течение следующих 3-х эпох улучшения не наблюдалось

Это помогло избежать переобучения и сэкономить время

4. Величина потерь остается относительно высокой, но это может быть связано с:

* Использованием аугментации данных, которая намеренно усложняет задачу
* L2-регуляризацией, которая добавляет штраф к функции потерь
* Dropout, который вносит случайный шум в процесс обучения

5. Предсказанный класс: roses. Уверенность: 97.47%. Это прекрасный результат.

Однако, были достигнуты хорошие результаты благодаря:

Использованию transfer learning с предобученной ResNet18

Грамотной аугментации данных

Механизмам регуляризации

Ранней остановке обучения

## Сравнительный анализ библиотек

**1.1. Производительность**

**Скорость обучения моделей**

**- Scikit-learn:** Показал самую высокую скорость обучения благодаря оптимизированным алгоритмам для классического машинного обучения. Время обучения модели Random Forest составило несколько минут.

**- TensorFlow:** Потребовал больше времени на обучение (10 эпох), но продемонстрировал стабильный прогресс с каждой эпохой.

**- PyTorch:** Благодаря использованию предобученной модели ResNet18 и ранней остановке, обучение заняло 6 эпох, что оптимизировало временные затраты.

**Точность предсказаний**

**- Scikit-learn:** Достиг точности 54.36% на тестовой выборке, что является наименьшим показателем среди трёх библиотек.

**- ТensorFlow:** Продемонстрировал высокую точность с максимальной валидационной точностью 92.69%.

**- PyTorch:** Показал отличные результаты с валидационной точностью 91.55% (лучший результат на 3-й эпохе).

**Использование ресурсов**

**- Scikit-learn:** Наиболее эффективен в плане использования памяти и CPU, не требует GPU.

**- ТensorFlow:** Требует значительных вычислительных ресурсов, но эффективно использует GPU-ускорение.

**- PyTorch:** Оптимально использует доступные ресурсы благодаря динамическим вычислительным графам и эффективной работе с GPU.

**1.2. Удобство использования**

**Простота написания кода**

**- Scikit-learn:** Отличается наиболее простым и понятным API, требует минимального количества кода для базовых задач.

**- TensorFlow:** Имеет более сложный синтаксис, но предоставляет высокоуровневый API Keras, упрощающий разработку.

**- PyTorch:** Предлагает интуитивно понятный Python-подобный синтаксис и более гибкий подход к построению моделей.

Документация и поддержка сообщества

**- Scikit-learn:** Имеет обширную документацию с подробными примерами и активное сообщество.

**- TensorFlow:** Предоставляет comprehensive документацию, множество обучающих материалов и крупнейшее сообщество.

**- PyTorch:** Отличается качественной документацией и растущим сообществом, особенно в академической среде.

Интеграция с другими инструментами

**- Scikit-learn:** Хорошо интегрируется с pandas, numpy и другими инструментами анализа данных.

**- TensorFlow:** Предлагает широкую экосистему инструментов, включая TensorBoard для визуализации.

**- PyTorch:** Обеспечивает простую интеграцию с Python-экосистемой и научными библиотеками.

**1.3. Гибкость и масштабируемость**

**- Scikit-learn:** Оптимален для небольших и средних датасетов, ограничен в масштабируемости.

**- TensorFlow:** Отлично подходит для промышленного применения и масштабных проектов.

**- PyTorch:** Предлагает гибкую архитектуру, удобную для исследований и прототипирования.

# Заключение

**2.1. Выводы по результатам исследования**

На основе проведенного исследования можно сделать следующие выводы:

1. Каждая библиотека имеет свои сильные стороны и оптимальные области применения

2. В задаче классификации изображений:

- PyTorch и TensorFlow показали сопоставимо высокую точность (>90%)

- Scikit-learn продемонстрировал более скромные результаты, что ожидаемо для задач компьютерного зрения

**2.2. Рекомендации по выбору библиотеки в зависимости от задачи**

**Scikit-learn** рекомендуется для:

- Классических задач машинного обучения

- Быстрого прототипирования

- Проектов с ограниченными вычислительными ресурсами

**ТensorFlow** оптимален для:

- Промышленных проектов

- Развертывания моделей в production

- Мобильных и встраиваемых систем

P**yTorch** подходит для:

- Исследовательских проектов

- Быстрого экспериментирования с архитектурами

- Академических задач

**2.3. Перспективы дальнейших исследований**

1. Расширение сравнительного анализа на другие типы задач машинного обучения

2. Исследование производительности библиотек на различных аппаратных платформах

3. Анализ новых возможностей библиотек в контексте последних достижений в области ИИ

4. Изучение возможностей интеграции различных библиотек в единый пайплайн обработки данных

# Приложение 1. Список сайтов используемых библиотек и ссылки на ресурсы

1. Сайт TensorFlow: https://www.tensorflow.org/?hl=ru

2. Сайт PyTorch: https://pytorch.org/

3. Сайт scikit-learn: https://scikit-learn.org/

4. Цветочный датасет: <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

5.Подборка датасетов для машинного обучения:

<https://habr.com/ru/articles/452392/>

6. Ссылка на изображение «Роза крупным планом»:

<https://vsegda-pomnim.com/uploads/posts/2022-04/1650490269_13-vsegda-pomnim-com-p-roza-krupnim-planom-foto-17.jpg>

# Приложение 2. Список необходимых библиотек

matplotlib 3.8.0

matplotlib-inline 0.1.7

matplotlib-venn 1.1.1

numpy 1.26.4

torch 2.5.0+cu121

torchaudio 2.5.0+cu121

torchsummary 1.5.1

torchvision 0.20.0+cu121