**Автор : Гревцев Роиан Владимирович**

Дипломная работа по курсу "Python-разработчик"

Тема: Реализация платформы для сравнения различных предобученных моделей распознавания объектов

Оглавление

1. Введение

1.1. Обоснование выбора темы

1.2. Определение цели и задач исследования

2. Основные понятия и определения

3. Методы и подходы к разработке

3.1. Выбор и интеграция предобученной модели

3.2. Архитектура веб-приложения

3.3. Обеспечение безопасности

4. Обзор популярных инструментов для разработки веб-приложений на Python

4.1. Django

4.2. Flask

4.3. FastAPI

5. Проектирование приложения

5.1. Планирование и анализ требований

5.2. Основные требования

5.3. Технические требования

6. Разработка в соответствии с созданной документацией

6.1. Планирование разработки

6.2. Разработка

7. Анализ и интерпретация результатов

7.1. Сравнение моделей

7.2. Интерпретация результатов

7.3. Рекомендации по выбору модели в зависимости от проекта

8. Заключение

8.1. Обзор выполненной работы

8.2. Дальнейшие планы

9. Приложения

9.1. Пример файловой структуры проекта

9.2. Список необходимых библиотек

1. Введение

1.1. Обоснование выбора темы

В современном мире искусственный интеллект и глубокое обучение становятся неотъемлемой частью различных областей, включая обработку изображений. Классификация объектов на изображениях представляет собой одну из задач, решаемых с помощью машинного обучения. Веб-приложение, использующее предобученные модели для классификации объектов на изображениях, имеет широкий спектр применений в различных отраслях.

Существует растущая потребность в специалистах, обладающих навыками применения и сравнения различных предобученных моделей машинного обучения для решения задач распознавания объектов на изображениях. Компании стремятся использовать наиболее эффективные модели для улучшения качества своих продуктов и услуг.

Работа в области разработки веб-приложения позволит применить полученные знания на практике и даст опыт использования предобученных моделей. Это позволит создавать интересные и полезные для пользователей возможности.

1.2. Определение цели и задач исследования

Цель исследования: Создать платформу для сравнения различных моделей распознавания объектов, чтобы оценить их эффективность.

Задачи исследования:

1. Анализ современных подходов и инструментов в области разработки.

2. Выбор конкретной темы исследования для дальнейшего изучения.

3. Проведение теоретического обзора и анализа литературы по выбранной теме.

4. Оценка результатов и формулирование выводов по проведенному исследованию.

5. Написание и оформление дипломной работы согласно требованиям учебного заведения.

2. Основные понятия и определения

1. Фреймворк (Framework): Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений. В контексте веб-приложений часто используются Django, FastAPI или Flask.

2. Предобученная модель (Pre-trained Model): Модель машинного обучения, которая была заранее обучена на большом наборе данных и может быть использована для решения задач распознавания объектов. Примеры таких моделей включают Inception, ResNet, VGG и YOLO.

3. Веб-приложение (Web Application): Программное приложение, которое работает на веб-сервере и доступно через браузер. В контексте данного проекта веб-приложение будет использоваться для сравнения и демонстрации эффективности различных предобученных моделей распознавания объектов.

4. Эффективность модели (Model Efficiency): Оценка того, насколько точно и быстро модель распознает объекты. Основные метрики для оценки эффективности включают достоверность (accuracy), полноту (recall), точность (precision) и время отклика (response time).

5. Достоверность (Accuracy): Процент правильно классифицированных объектов от общего числа объектов. Высокая точность указывает на то, что модель делает меньше ошибок.

6. Полнота (Recall): Способность модели обнаруживать все истинные положительные примеры среди всех положительных примеров в данных. Высокая полнота означает, что модель пропускает меньше истинных положительных объектов.

7. Точность (Precision): Способность модели правильно определять положительные примеры среди всех примеров, которые она классифицировала как положительные. Высокая точность означает, что модель делает меньше ложных срабатываний.

8. Время отклика (Response Time): Время, необходимое модели для обработки входных данных и выдачи результата. Время отклика является важной метрикой, особенно в приложениях, где скорость критична.

9. Матрица ошибок (Confusion Matrix): Таблица, используемая для оценки производительности модели классификации. Она показывает количество правильных и неправильных предсказаний, разбитых по каждой категории.

10. ROC-кривая (Receiver Operating Characteristic Curve): График, который показывает соотношение между полнотой и ложноположительными срабатываниями для модели при различных порогах классификации. ROC-кривая помогает визуально оценить способность модели различать положительные и отрицательные классы.

11. F1-score: Среднее гармоническое значение точности и полноты, которое используется как единая метрика для оценки эффективности модели, особенно когда имеется несбалансированный набор данных.

12. IoU (Intersection over Union): Метрика, используемая для оценки качества сегментации объектов, которая измеряет степень совпадения между предсказанными и истинными границами объектов.

3. Методы и подходы к разработке

3.1. Выбор и интеграция предобученной модели

Для задач распознавания объектов в данном проекте была выбрана предобученная модель MobileNet SSD, которая хорошо зарекомендовала себя в задачах компьютерного зрения благодаря своей высокой скорости обработки и приемлемой точности.

Интеграция модели включает следующие этапы:

• Загрузка модели и её конфигурации.

• Подготовка изображений для обработки (изменение размера, нормализация).

• Использование модели для предсказания классов объектов на загруженных изображениях.

3.2. Архитектура веб-приложения

Приложение состоит из двух основных компонентов: фронтенда и бэкенда.

• Фронтенд: Интерфейс пользователя реализован с использованием HTML, CSS и JavaScript, с использованием фреймворка Bootstrap для стилизации и адаптивности.

• Бэкенд: Серверная логика реализована на Python с использованием фреймворка Django. Бэкенд отвечает за обработку запросов, взаимодействие с базой данных и выполнение предсказаний с использованием модели машинного обучения.

REST API: Для связи между фронтендом и бэкендом используется REST API, что позволяет фронтенду отправлять изображения на сервер для обработки и получать результаты.

3.3. Обеспечение безопасности

Для обеспечения безопасности приложения реализована аутентификация и авторизация пользователей. Каждый пользователь может создать свою учетную запись, а затем входить в систему, чтобы загружать и обрабатывать изображения. Это гарантирует, что пользователи видят только свои загруженные данные

4. Обзор популярных инструментов для разработки веб-приложений на Python

4.1. Django

Django — это мощный фреймворк для веб-разработки на Python, который предоставляет множество встроенных функций, таких как:

• Встроенная страница администрирования для управления данными.

• ORM (Object-Relational Mapping) для работы с базами данных.

• Поддержка аутентификации пользователей и управления доступом.

Особенности: Идеально подходит для больших и сложных проектов благодаря своей полной экосистеме.

4.2. Flask

Flask — это минималистичный фреймворк, который обеспечивает простоту и гибкость. Он подходит для небольших и средних проектов, где требуется большая гибкость.

Основные возможности:

• Минималистичный подход: включает только основные компоненты без лишних зависимостей.

• Высокая гибкость для добавления нужных библиотек и модулей.

• Поддержка шаблонов с использованием Jinja2 и маршрутизации с помощью Werkzeug.

Особенности: Flask позволяет быстро разрабатывать прототипы и предоставляет разработчику полный контроль над архитектурой приложения.

4.3. FastAPI

FastAPI — это современный, высокопроизводительный фреймворк для создания API с использованием Python 3.7+ на основе стандартов OpenAPI и JSON Schema.

Основные возможности:

• Высокая производительность благодаря использованию асинхронного программирования (async/await).

• Автоматическая генерация интерактивной документации API, что облегчает тестирование и использование API.

• Поддержка валидирования и аннотаций типов данных, что позволяет разработчикам легко определять ожидаемые форматы данных.

Особенности: FastAPI идеально подходит для создания микросервисов и высоконагруженных приложений, где важна производительность и скорость обработки запросов.

5. Проектирование приложения

5.1. Планирование и анализ требований

Выбор фреймворка и инструментов: На основании анализа требований к проекту был выбран фреймворк Django, так как он предоставляет все необходимые инструменты для разработки сложных веб-приложений с поддержкой аутентификации и работы с базами данных.

Определение структуры приложения: Разработка схемы архитектуры, включающей фронтенд, бэкенд, базу данных и интеграцию с предобученной моделью.

5.2. Основные требования

• Загрузка изображений: Пользователь должен иметь возможность загружать изображения для анализа.

• Распознавание объектов: Приложение должно использовать предобученную модель для распознавания объектов на загруженных изображениях.

• Отображение результатов: Приложение должно отображать результаты распознавания объектов, включая метки и вероятность для каждого объекта.

5.3. Технические требования

• Фронтенд: Использование HTML, CSS, JavaScript и Bootstrap для создания интерфейса.

• Бэкенд: Python с использованием фреймворка Django.

• База данных: Использование SQLite для хранения данных о пользователях и истории загрузок.

6. Разработка в соответствии с созданной документацией

6.1. Планирование разработки

Разработка была разделена на несколько основных этапов:

1. Проектирование интерфейса.

2. Реализация серверной логики.

3. Интеграция модели машинного обучения.

6.2. Разработка

Фронтенд-разработка:

• Создан пользовательский интерфейс с использованием HTML, CSS и JavaScript.

• Использован фреймворк Bootstrap для стилизации и адаптивности.

• Разработаны формы загрузки изображений и страницы отображения результатов.

Бэкенд-разработка:

• Реализована серверная логика с использованием фреймворка Django.

• Настроены маршруты для обработки запросов от фронтенда.

• Использована база данных SQLite для хранения данных о пользователях и истории загрузок.

Интеграция модели машинного обучения:

• Подключена и настроена предобученная модель MobileNet SSD.

• Реализована обработка изображений с использованием модели.

7. Анализ и интерпретация результатов

7.1. Сравнение моделей

Были протестированы различные предобученные модели, включая VGG, YOLO и MobileNet, на различных наборах данных.

Основные метрики:

• Достоверность (accuracy)

• Полнота (recall)

• Точность (precision)

• Время отклика (response time)

7.2. Интерпретация результатов

Полученные данные анализируются для выявления сильных и слабых сторон каждой модели. Например:

• Модель VGG показала высокую точность, но время отклика было выше по сравнению с легковесными моделями.

• YOLO обеспечила быстрые результаты, но с некоторыми потерями в точности.

Результаты тестирования моделей показывают, что выбор модели зависит от конкретных требований задачи. Важно учитывать как точность распознавания, так и время отклика, особенно в реальных приложениях, где пользователи ожидают быстрой обработки.

7.3. Рекомендации по выбору модели в зависимости от проекта

• Для задач, требующих высокой точности и допускающих большее время отклика: рекомендуется использовать модель VGG. Она подходит для приложений, где критична точность распознавания, например, в медицинской диагностике или системах безопасности.

• Для задач, где важна скорость обработки: можно рассмотреть использование моделей YOLO или MobileNet. Эти модели обеспечивают быстрое распознавание объектов, что делает их подходящими для приложений в реальном времени, таких как видеонаблюдение или системы автономного вождения.

8. Заключение

8.1. Обзор выполненной работы

В данной дипломной работе была успешно разработана платформа для сравнения различных предобученных моделей распознавания объектов. Приложение позволяет пользователям загружать изображения и получать результаты распознавания, используя модель MobileNet SSD.

Реализованное приложение:

• Позволяет пользователям загружать изображения и получать результаты распознавания объектов.

• Использует предобученные модели для обработки изображений.

• Обеспечивает безопасность с помощью аутентификации и авторизации пользователей.

8.2. Дальнейшие планы

В будущем планируется:

• Расширение функционала приложения, включая возможность выбора различных моделей для распознавания объектов.

• Улучшение пользовательского интерфейса для повышения удобства работы с приложением.

• Добавление функционала для сохранения истории загрузок и результатов.

• Проведение дополнительных тестов для более глубокого анализа производительности различных моделей.

9. Приложения

9.1. Пример файловой структуры проекта

project/

│

├── app/ # Корневая папка проекта

│ ├── \_\_init\_\_.py # Инициализация пакета

│ ├── settings.py # Настройки проекта

│ ├── urls.py # URL-адреса проекта

│ ├── wsgi.py # WSGI-совместимый интерфейс

│ └── asgi.py # ASGI-совместимый интерфейс (если используется)

│

├── object\_detection/ # Приложение для распознавания объектов

│ ├── \_\_init\_\_.py # Инициализация пакета

│ ├── admin.py # Настройки административной панели

│ ├── apps.py # Конфигурация приложения

│ ├── forms.py # Формы для обработки данных

│ ├── models.py # Модели базы данных

│ ├── tests.py # Тесты приложения

│ ├── urls.py # URL-адреса приложения

│ ├── views.py # Представления (логика обработки запросов)

│ ├── mobilenet\_iter\_73000.caffemodel # Предобученная модель (если необходимо)

│ └── mobilenet\_ssd\_deploy.prototxt # Конфигурация модели (если необходимо)

│

├── media/ # Папка для загруженных пользователем файлов

│ ├── images/ # Загружаемые изображения

│ └── processed\_images/ # Обработанные изображения

│

├── templates/ # Шаблоны HTML

│ └── object\_detection/ # Шаблоны для приложения распознавания объектов

│ ├── add\_image\_feed.html # Шаблон для загрузки изображения

│ ├── base\_generic.html # Базовый шаблон

│ ├── dashboard.html # Шаблон для дашборда

│ ├── home.html # Шаблон для домашней страницы

│ ├── login.html # Шаблон для страницы входа

│ └── register.html # Шаблон для страницы регистрации

│

├── manage.py # Скрипт для управления проектом

├── requirements.txt # Список необходимых библиотек

└── db.sqlite3 # База данных SQLite помоги написать код для каждой папки

9.2. Список необходимых библиотек

Django==3.2.9

numpy==1.21.2

opencv-python==4.5.4

Pillow==9.0.0

10. Заключительные замечания

В ходе выполнения данной дипломной работы была разработана платформа для сравнения различных предобученных моделей распознавания объектов. Приложение предоставляет пользователям возможность загружать изображения, обрабатывать их с помощью предобученной модели MobileNet SSD и получать результаты распознавания.

Основные выводы:

1. Анализ моделей: В процессе работы были проанализированы различные предобученные модели, такие как VGG, YOLO и MobileNet. Каждая из них имеет свои сильные и слабые стороны, что позволяет пользователям выбирать наиболее подходящую модель в зависимости от специфики задачи.

2. Интерфейс пользователя: Разработанный интерфейс обеспечивает удобство взаимодействия с приложением, позволяя пользователям легко загружать изображения и получать результаты.

3. Безопасность: Реализованная система аутентификации и авторизации пользователей гарантирует безопасность данных и конфиденциальность информации.

11. Список использованной литературы

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville. "Deep Learning". MIT Press, 2016.

2. Francois Chollet. "Deep Learning with Python". Manning Publications, 2017.

3. Django Documentation. https://docs.djangoproject.com/en/stable/

4. Flask Documentation. https://flask.palletsprojects.com/en/2.0.x/

5. FastAPI Documentation. https://fastapi.tiangolo.com/

6. TensorFlow Documentation. https://www.tensorflow.org/api\_docs