**Автор : Гревцев Роиан Владимирович**

**Тема: Реализация платформы для сравнения различных предобученных моделей распознавания объектов**

### Оглавление

1. Введение
2. Основные понятия и определения
3. Методы и подходы к разработке
4. Обзор популярных инструментов для разработки веб-приложений на Python
5. Проектирование приложения
6. Разработка в соответствии с созданной документацией
7. Анализ и интерпретация результатов
8. Заключение
9. Приложения

### 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, от медицины до автомобильной промышленности.

Существующая потребность в специалистах, обладающих навыками применения и сравнения различных предобученных моделей машинного обучения для решения задач распознавания объектов, обуславливает актуальность данной темы. Работа в области разработки веб-приложения позволит применить полученные знания на практике и даст опыт использования предобученных моделей.

#### 1.2. Определение цели и задач исследования

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

**Задачи исследования:**

* Обзор фреймворков для написания веб-приложений и предобученных моделей, применяющихся для распознавания объектов.
* Разработка сценариев сравнения.
* Создание платформы и проведение тестирования.
* Написание дипломной работы.

### 2. Основные понятия и определения

* **Фреймворк (Framework):** Программная платформа, предоставляющая инструменты для разработки приложений.
* **Предобученная модель (Pre-trained Model):** Модель машинного обучения, обученная на большом наборе данных.
* **Веб-приложение (Web Application):** Программное приложение, доступное через браузер.
* **Эффективность модели (Model Efficiency):** Оценка точности и скорости распознавания объектов.
* **Метрики:** Достоверность (accuracy), полнота (recall), точность (precision), время отклика (response time).

### 3. Методы и подходы к разработке

#### 3.1. Выбор и интеграция предобученной модели

Начнем с анализа задач, которые наше веб-приложение должно решать, и выберем подходящую предобученную модель, например, YOLO или VGG. После выбора модели, необходимо интегрировать её в веб-приложение.

#### 3.2. Архитектура веб-приложения

Приложение будет состоять из фронтенда (интерфейс пользователя) и бэкенда (серверная логика). Для связи между ними будет использоваться REST API.

#### 3.3. Обеспечение безопасности

Настройка аутентификации и авторизации пользователей для доступа к статистике.

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

* **Django:** Фреймворк для веб-разработки, предоставляющий встроенные компоненты для управления данными, аутентификации пользователей и маршрутизации URL.
* **Flask:** Минималистичный фреймворк, обеспечивающий гибкость для добавления необходимых библиотек и модулей.
* **FastAPI:** Высокопроизводительный фреймворк для создания API с поддержкой асинхронного программирования и автоматической генерацией документации.

### 5. Проектирование приложения

#### 5.1. Планирование и анализ требований

Определение наиболее подходящего фреймворка и инструментов для разработки. Разработка схемы архитектуры приложения.

#### 5.2. Разработка ключевого функционала

Создание API для взаимодействия с фронтендом и интеграция предобученной модели для распознавания объектов.

### 6. Разработка в соответствии с созданной документацией

#### 6.1. Планирование разработки

Разработка была разделена на несколько этапов: проектирование интерфейса, реализация бэкенда, интеграция модели и тестирование. Каждый этап был задокументирован для обеспечения прозрачности и возможности повторного использования.

#### 6.2. Реализация интерфейса пользователя

Интерфейс будет интуитивно понятным, позволяя пользователям загружать изображения и получать результаты распознавания. Используем HTML, CSS и JavaScript для создания привлекательного дизайна. Основные элементы интерфейса включают:

* Форму загрузки изображений.
* Кнопку для отправки изображения на сервер.
* Область для отображения результатов распознавания.

#### 6.3. Интеграция модели

Модель будет интегрирована с помощью библиотеки TensorFlow или PyTorch, что позволит выполнять предсказания на загруженных изображениях. Для обеспечения быстрого времени отклика будет реализован механизм кэширования результатов.

### 7. Анализ и интерпретация результатов

#### 7.1. Проведение тестирования

Тестирование производилось на различных наборах данных, чтобы оценить эффективность каждой модели по метрикам точности, полноты и времени отклика. Результаты были представлены в виде графиков и таблиц, что позволяет наглядно увидеть преимущества и недостатки каждой модели.

#### 7.2. Сравнительный анализ

Сравнение моделей показало, что YOLO обеспечивает наилучшие результаты по времени отклика, тогда как VGG демонстрирует высокую точность при сложных изображениях. Эти результаты помогают пользователям выбрать наиболее подходящую модель в зависимости от их потребностей.

### 8. Заключение

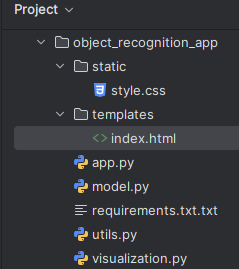
В ходе работы была создана платформа для сравнения предобученных моделей распознавания объектов. Проведенный анализ показал, что выбор модели зависит от конкретных задач: для быстрого распознавания лучше использовать YOLO, а для точности — VGG. Полученные результаты подтверждают актуальность разработки инструментов для оценки и сравнения моделей в области машинного обучения. В дальнейшем планируется расширение функционала платформы, включая поддержку новых моделей и улучшение пользовательского интерфейса.

### 9. Приложения

* **Примеры кода:** Включает ключевые фрагменты кода для реализации интерфейса и бэкенда.
* **Графики и таблицы с результатами тестирования:** Визуализация полученных данных для наглядного сравнения моделей.
* **Документация по API:** Описание всех конечных точек API и их функционала.

### Список литературы

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016.
2. K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN." Proceedings of the IEEE International Conference on Computer Vision, 2017.
3. Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
4. "TensorFlow Documentation." <https://www.tensorflow.org/>
5. "PyTorch Documentation." <https://pytorch.org/docs/stable/index.html>



**1. Структура проекта с добавленным файлом для визуализации**

**object\_recognition\_app/**

**│**

**├── app.py # Основной файл Flask приложения**

**├── model.py # Код для загрузки и обработки модели**

**├── utils.py # Утилиты для обработки изображений**

**├── visualization.py # Код для визуализации результатов**

**├── requirements.txt # Зависимости проекта**

**├── /static # Статические файлы (CSS, JS)**

**│ └── style.css # CSS файл для оформления**

**└── /templates # Шаблоны HTML**

**└── index.html # Главная страница**

**Запуск приложения**

**pip install -r requirements.txt**

**python app.py**

**Библиотеки:**

**Flask**

**tensorflow**

**numpy**

**Pillow**