Разработка механизма автоматического перевода текста на фотографии для мобильных устройств

ВЫПОЛНИЛ: СТУДЕНТ 597 ГР.
ТКАЧЕНКО Д.А.
НАУЧНЫЙ РУКОВОДИТЕЛЬ:
РОДЮКОВ А.В.

Объекты и предмет исследования

- Объекты исследования – механизмы распознавания текста на фотографии, сторонние библиотеки и механизмы проектирования и реализации мобильного приложения
- Предмет исследования – процесс создания приложения по автоматическому переводу на платформе iOS

Актуальность проблемы

Глобализация

+

Эра мобильных устройств

=

Практическая значимость автоматического перевода

+

Необходимость развития направления использования мобильных устройств

Цель работы

Основная цель работы – изучение **архитектурной** и **прикладной** разработки мобильного приложения и применение знаний в области **машинного обучения** для использования модели распознавания непосредственно на мобильном устройстве.

Поставленные задачи

1. Разработка архитектуры приложения и взаимодействия его компонент
2. Исследование и применение существующих библиотек сторонних разработчиков
3. Выбор модели для распознавания и ее обучение
4. Конвертация обученной модели в формат, пригодный для использования на мобильном устройстве
5. Тестирование полученного прототипа

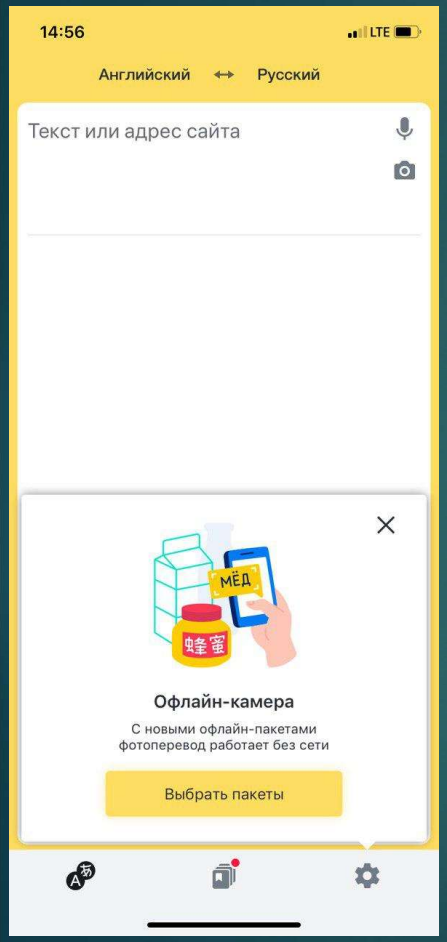
Анализ существующих решений

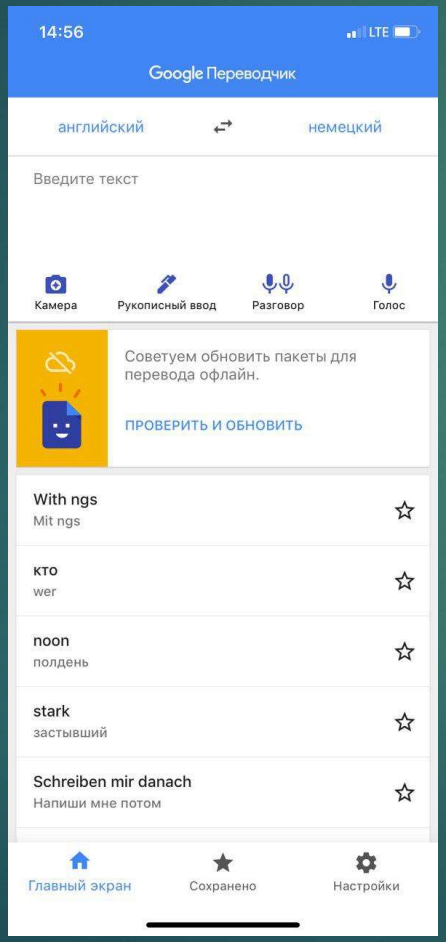
Для анализа существующих решений были выбраны 5 приложений:

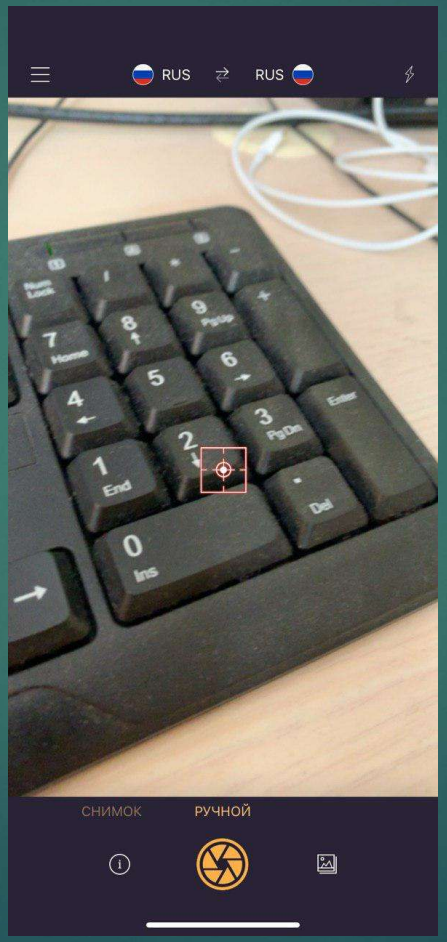
1. Google переводчик
2. Яндекс переводчик
3. Переводчик с Фото и Сканер
4. Сканер Переводчик+перевод OCR
5. Переводи фото + сканер текста

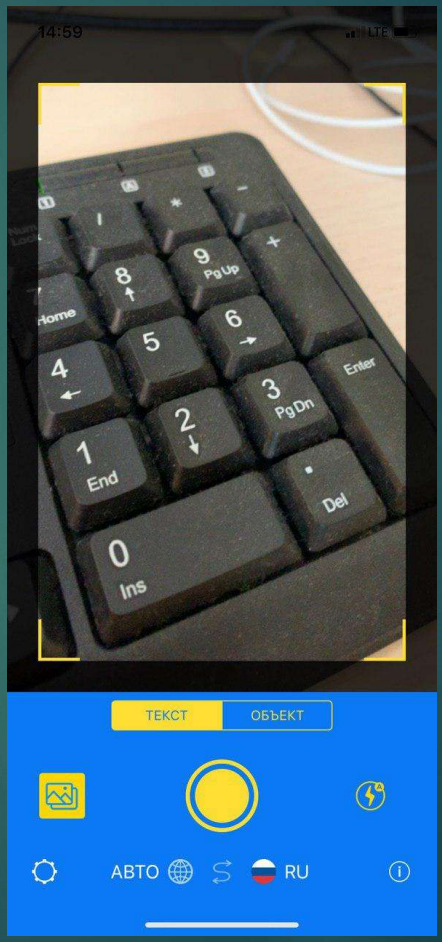
Два первых решения обладают существенным плюсом перед остальными – возможностью распознавания и перевода текста без подключения к Сети. Но их основной направленностью не является перевод текста с фотографии – основной интерфейс заточен на ввод с клавиатуры или голосовой ввод. Последние три решения работают только при подключении к Сети и не имеют возможности сохранения результатов перевода.

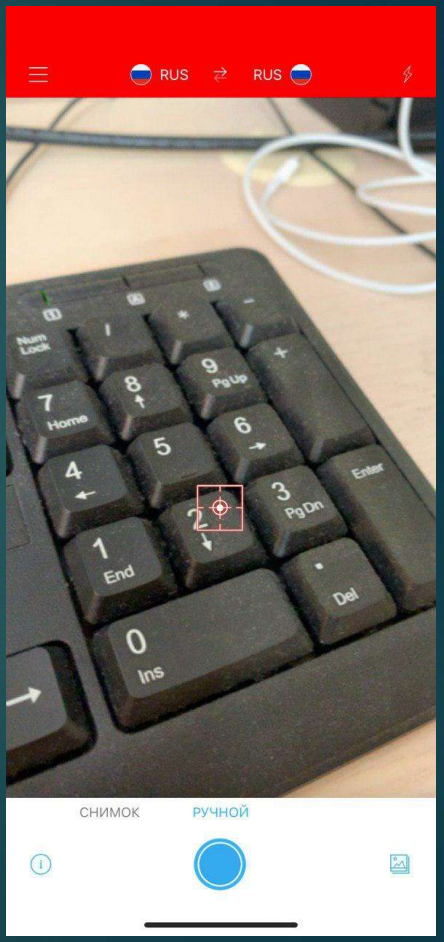
Анализ существующих решений











Архитектурная разработка

Основные структурные компоненты:

1. Сущность приложения
2. Контекст – состояние базы данных и сетевых взаимодействий
3. Объект, контролирующий переходы между экранами приложения
4. Объект, работающий с API перевода
5. Объект, работающий с видеопотоком камеры
6. Объект, распознающий текст с фотографии

Архитектурная разработка

Принципы разделения ответственности компонент и их взаимодействия

Приложения в целом:

VIPER

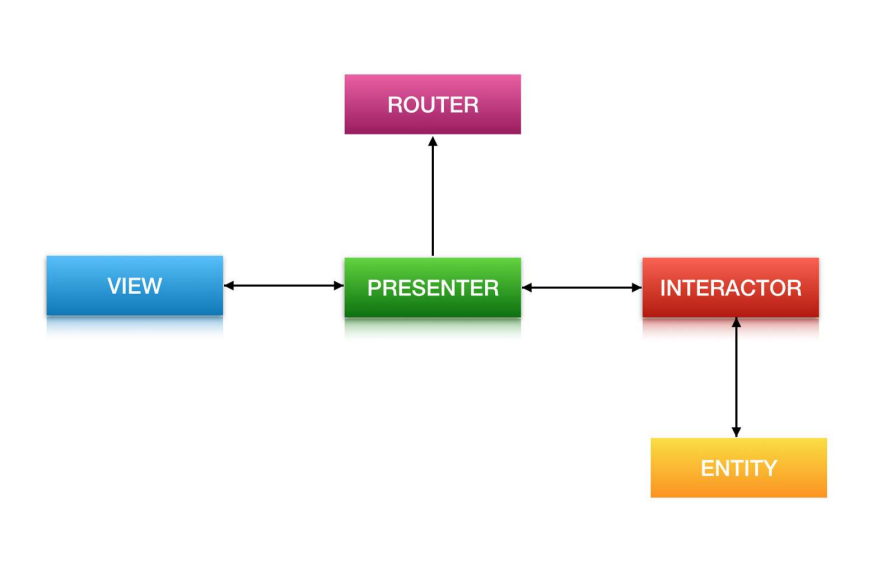
1. View
2. Interactor
3. Presenter
4. Entity
5. Router

Модулей приложения:

MVVM

1. Model
2. View
3. View-Model

Архитектурная разработка (VIPER)

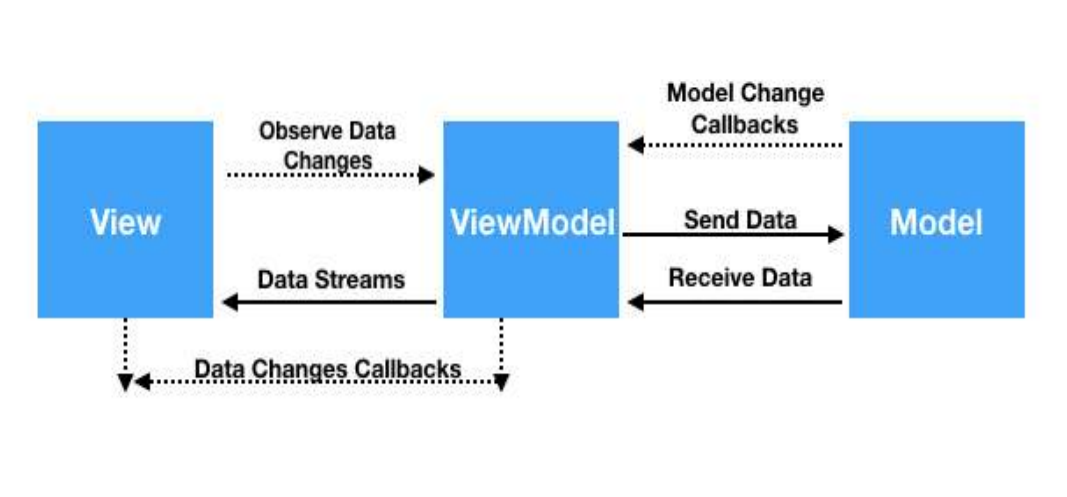


1. **View** – визуальное отображение данных и взаимодействие с пользователем
2. **Interactor** – взаимодействие с данными в базе и их изменение, взаимодействие с сетью
3. **Presenter** – взаимодействие объектов-представлений и данных
4. **Entity** – сущности базы данных, доступные только для чтения
5. **Router** – перемещение между экранами приложения

1. Тестируемость
2. Удобство поддержки
3. Четкое разделение ответственности

1. Высокий порог вхождения
2. Обилие классов

Архитектурная разработка (MVVM)



1. **Model** – создание и хранение моделей данных
2. **View** – интерфейсы представлений, логика отображения и обработка событий от пользователя
3. **View-Model** – преобразователь данных Модели для отображения во View, транслятор событий от View для обновления Модели

1. Тестируемость
2. Простота класса представления
3. Модульность

1. Нестрогая грань между Model и View-Model
2. Все взаимодействия происходят через View-Model

Разработка приложения

Основной язык разработки приложений для платформы iOS – **Swift.**

Его отличительные особенности:

1. Протоколоориентированность
2. Типизированные перечисления с присоединяемыми значениями
3. Возможность запуска C++ кода

Использование в приложении

1. Реализация базы данных с возможностью подписки на обновления
2. Переиспользуемый экран настроек
3. Обработка изображений при помощи библиотеки OpenCV

Библиотеки сторонних разработчиков

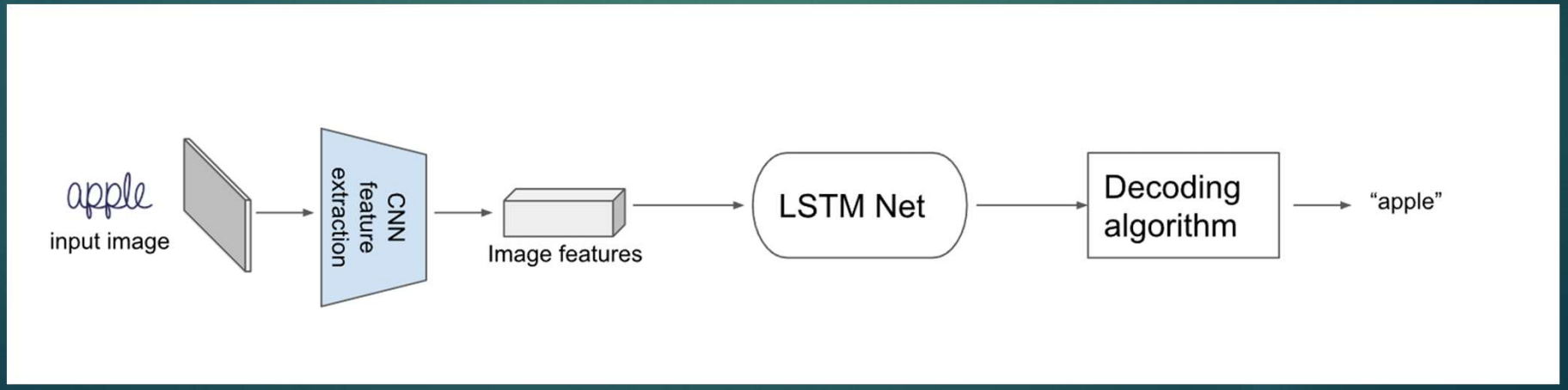
- **Google Translate API** – текстовый перевод. Отправка и получение сетевых запросов, обработка результатов.
- **Firebase** – BaaS-платформа от Google. Содержит в том числе модели для распознавания языка и текста на фотографии.
- **Tesseract** – открытая библиотека для распознавания текстов на фотографии.
- **OpenCV** – библиотека компьютерного зрения. Использовалась для обработки фотографий перед процессом распознавания.

BaaS ("Backend-as-a-Service") – модель, позволяющая разработчикам веб-приложений и мобильных приложений связать их приложения с серверным облачным хранилищем и API

Архитектура конвертируемой модели распознавания

Выбранная модель – Image OCR с функцией потерь CTC (Connectionist Temporal Classification):

1. Хорошая точность
2. Быстрое время работы
3. Использование в популярных продуктах (Dropbox document scanner)



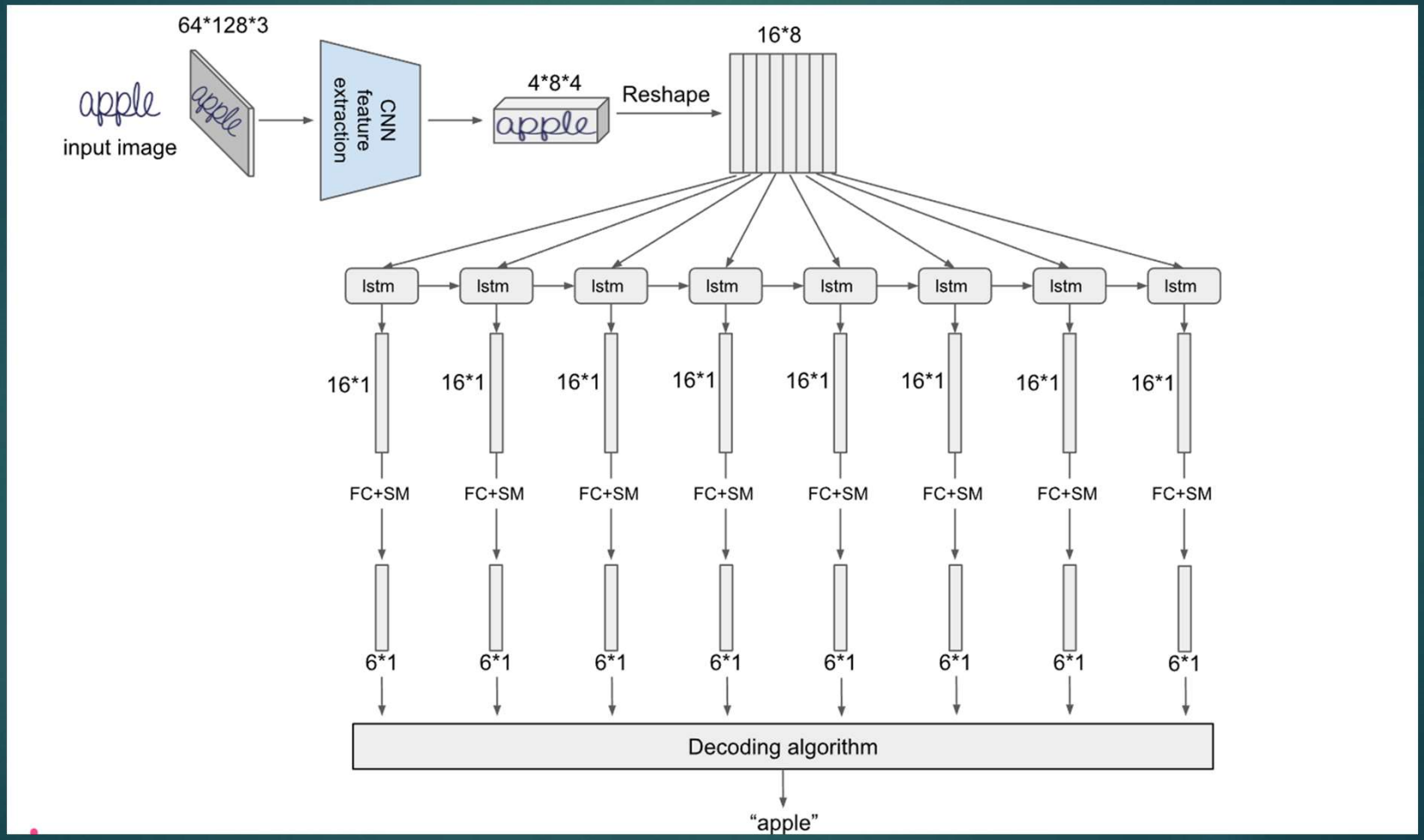
Архитектура конвертируемой модели распознавания

Верхнеуровневое описание архитектуры:

1. Конволюционные нейроны, извлекающие признаки изображения
2. Слой изменения размерности
3. Рекуррентные нейроны LSTM (long short-term memory)
4. Полносвязный слой
5. Softmax – вероятности каждого символа исходного алфавита

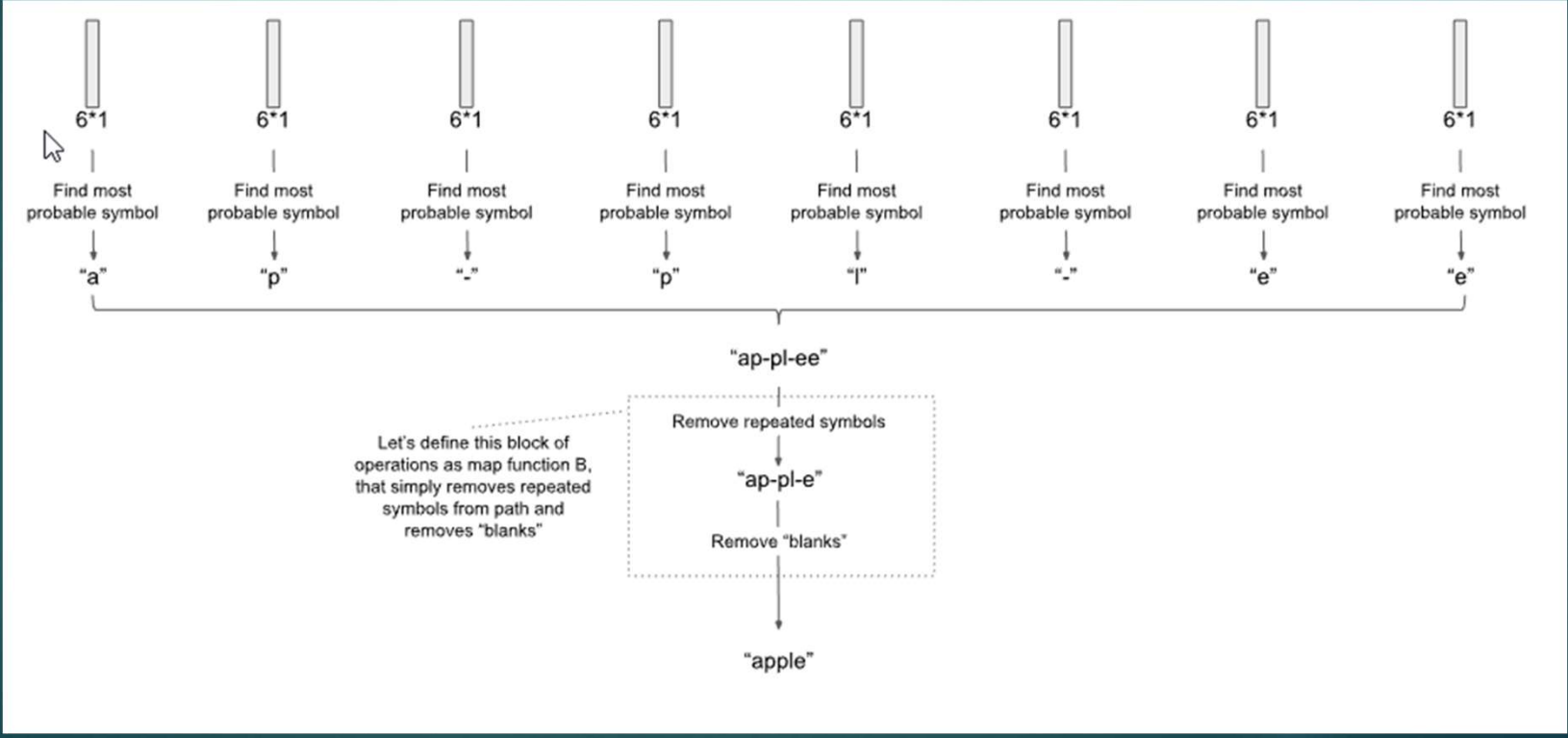
Алгоритм декодирования

Архитектура конвертируемой модели распознавания



Алгоритм декодирования

- Задача – преобразование итоговых вероятностей в строку.
- Самый распространенный подход – жадный алгоритм: выбор из каждого вектора самого вероятного элемента, затем наивная конкатенация элементов.



Конвертация и использование полученной модели

- Обученную модель необходимо конвертировать в формат, пригодный для использования на мобильном устройстве.
- Реализованная библиотека конвертации моделей Keras в требуемый формат – библиотека coremltools от Apple.
- Для обработки изображений перед процессом распознавания использована библиотека OpenCV, которая так же была использована в процессе обучения модели.

Тестирование

Тестирование используемых моделей распознавания происходило на 10 тестовых кейсах.

Номер теста	Надпись	Шрифт	Размер шрифта
1	Lemon grass jasmine	Times new Roman	40
2	Test for the testing	Arial Bold	40
3	This Is Multiline Test	Verdana	40
4	This is tricky test	Times new Roman	20, 32, 40
5	It is also tricky test	Times new Roman, Calibri, Tahoma, Impact	28

Кейсы первого этапа тестирования

Номер теста	Надпись	Шрифт
6	Courier font	Courier
7	Stix font	Stix
8	Urw chancery l font	URW Chancery L
9	Century Schoolbok font	Century Schoolbook
10	Freemono font	Freemono font

Кейсы второго этапа тестирования

Результаты тестирования

Результаты обоих этапов тестирования в долях распознанных символов:

Первый этап

	1	2	3	4	5
Firebase	100%	100%	100%	100%	100%
Tesseract	100%	100%	90%	100%	100%
Собственная	70%	80%	80%	70%	90%

Второй этап

	1	2	3	4	5
Firebase	100%	100%	100%	100%	100%
Tesseract	0%	100%	90%	100%	0%
Собственная	100%	100%	>95%	100%	100%

Заключение и результаты

- Разработана архитектура мобильного приложения
- Изучены и применены особенности языка Swift и стандартных библиотек Apple
- Реализован механизм использования C++ кода в Swift-проекте.
- Выбрана и реализована архитектура модели распознавания. Данная модель конвертирована в формат, пригодный для использования на мобильном устройстве.
- Протестированы три модели распознавания. Выявлены недостатки конвертированной модели, в связи с чем поставлены дальнейшие задачи по ее улучшению.
- Исправлены основные слабые стороны существующих решений.