Куликов А.А., Горелкин А.С., Нефедов А.А. Разработка мобильных приложений на базе ОС IOS с внедрением COREML технологий // Международный журнал информационных технологий и энергоэффективности. – 2024. – Т. 9 № 5(43) с. 60–70

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.9

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА БАЗЕ ОС IOS С ВНЕДРЕНИЕМ COREML ТЕХНОЛОГИЙ

Куликов А.А., [1]Горелкин А.С., Нефедов А.А.
ФГБОУ ВО «МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ», Москва, Россия, (119454, г. Москва, просп. Вернадского, 78, стр. 4.), e-mail: [1]gorelk12222@bk.ru

В статье рассматривается ситуация на рынке внедрения и применения искусственного интеллекта при разработке мобильных приложений. Дается обзор современных инструментов и фреймворков, предназначенных для упрощения процесса интеграции и работы с моделями машинного обучения в iOS-приложениях. Целью данной работы является изучение основных принципов разработки приложений для iOS с интеграцией CoreML в мобильные приложения, включая описание основных возможностей, подготовки и обработки данных для моделей машинного обучения, а также создание и обучение таких моделей. Также рассматривается метод тестирования моделей машинного обучения, а также приводятся заключительные выводы исследования.

Ключевые слова: IOS, CoreML, мобильное приложение, машинное обучение, методы тестирования моделей, SwiftUI.

DEVELOPMENT OF MOBILE APPLICATIONS BASED ON IOS WITH THE INTRODUCTION OF COREML TECHNOLOGIES

Kulikov A.A., [1] Gorelkin A.S., Nefedov A.A.
MIREA - RUSSIAN TECHNOLOGICAL UNIVERSITY, Moscow, Russia (119454, Moscow, avenue. Vernadsky, 78, b. 4), e-mail: [1]gorelk12222@bk.ru

The article examines the situation on the market of the introduction and application of artificial intelligence in the development of mobile applications. An overview of modern tools and frameworks designed to simplify the process of integration and working with machine learning models in iOS applications is given. The purpose of this work is to study the basic principles of developing applications for iOS with CoreML integration into mobile applications, including a description of the main features, preparation and processing of data for machine learning models, as well as the creation and training of such models. The method of testing machine learning models is also considered, and the final conclusions of the study are presented.

Keywords: IOS, CoreML, mobile application, machine learning, model testing methods, SwiftUI.

1. Основы разработки мобильных приложений под iOS

Разработка мобильных приложений под iOS [1] требует знания основных принципов и инструментов. Для начала разработки приложений под iOS необходимо изучить язык программирования Swift, который широко используется для создания приложений для этой ОС. Также важно ознакомиться с основным инструментом разработки - Xcode, интегрированной средой разработки, которая предоставляет набор инструментов для создания, отладки и тестирования приложений. При разработке пользовательского

интерфейса необходимо использовать подходы, основанные на готовых компонентах и элементах управления, чтобы обеспечить удобство использования приложения. Анализ подходов для реализации пользовательского интерфейса позволит выбрать наиболее подходящий способ создания интерактивных и интуитивно понятных экранов приложения.

1.1. Анализ подходов для реализации пользовательского интерфейса

Анализ подходов для реализации пользовательского интерфейса является важным этапом разработки мобильных приложений под iOS. Имеется несколько основных подходов, из которых разработчики могут выбрать в соответствии с требованиями проекта: UIKit и SwiftUI [3].

Сравнительный анализ UIKit и SwiftUI

UIKit и SwiftUI являются двумя разными фреймворками для разработки приложений пользовательских интерфейсов в iOS. Главное отличие между ними заключается в подходе к построению пользовательского интерфейса. UIKit использует императивный подход, где разработчику нужно явно указывать каждый шаг при создании интерфейса. В то время как SwiftUI предлагает декларативный подход, где разработчик описывает желаемый интерфейс, и система сама заботится о его построении. Это позволяет сократить количество кода и упростить процесс создания интерфейса в SwiftUI. Сравнительный анализ UIKit и SwiftUI представлен в Таблице 1.

Таблица 1 – Анализ UIKit и SwiftUI

	SwiftUI	UIKit
Скорость вёрстки	+	-
Совместимость с легаси-кодом	-	+
Дебаггинг	-	+
Мультиплатформенность	+	-
Поддержка сообщества	-	+
Количество кода	+	-
Совместимость с Modern Concurrency	+	-
Поддерживаемость	-	+
Быстродействие	+	-

В итоге, выбор между UIKit и SwiftUI зависит от конкретного проекта и предпочтений команды разработчиков.

1.2. Сравнительный анализ архитектурных решений

Архитектура мобильных приложений на iOS представляет собой структурную основу, определяющую организацию кода, разделение обязанностей и взаимодействие компонентов приложения. Архитектурные шаблоны, такие как MVC, MVP, MVVM и VIPER [4], предоставляют разработчикам инструменты для организации кода, управления зависимостями и обеспечения легкости поддержки и масштабируемости приложения.

Сравнительный анализ MVC, MVVM, MVP, VIPER представлен в Таблице 2.

Таблица 2 – Анализ MVC, MVVM, MVP и VIPER

	MVC	MVP	MVVM	VIPER
Разделение ответственности	Неявное разделение ответственности, часто приводит к "толстым" контроллерам и увеличению связности	Presenter отвечает за бизнес-логику и обработку событий, что упрощает тестирование и поддержку	ViewModel управляет предоставлением данных для отображения, что позволяет легче тестировать и поддерживать приложение	Каждый компонент имеет четко определенные обязанности, что упрощает понимание и поддержку кода
Тестирование	Тестирование может быть затруднено из-за сильной связности между компонентами и отсутствия четкого разделения ответственности	Улучшенное тестирование благодаря вынесению логики из View в Presenter	Улучшенное тестирование за счет отделения логики отображения от бизнес-логики в ViewModel	Улучшенная тестируемость за счет отдельных компонентов и четкого разделения ответственности
Расширяемость	Масштабирование и расширение может быть затруднено из-за высокой связности и "толстых" контроллеров	Высокая расширяемость благодаря отделению представления от логики в Presenter	Улучшенная расширяемость и масштабируемость благодаря четкому разделению ответственности и декларативному связыванию данных	Высокая расширяемость за счет отдельных компонентов и четкого разделения ответственности
Сложность	Простота и легкость понимания, но возможно возникновение проблем при масштабировании	Сложность управления большим количеством Presenter и взаимосвязей между ними	Увеличение сложности приложения из-за дополнительных слоев ViewModel и дополнительных механизмов связи	Увеличение сложности проекта из-за большого количества слоев и взаимодействий между ними
Популярность	Широко используется и понятен многим разработчикам, особенно в начале карьеры	Популярен в некоторых областях, но не так широко распространен, как MVC	Пользуется популярностью, особенно в средних и больших проектах	Используется в некоторых проектах, но не так широко распространен, как MVC или MVVM

Каждый из архитектурных подходов имеет свои достоинства и недостатки, которые могут варьироваться в зависимости от конкретного проекта и команды разработчиков.

2. Внедрение технологии CoreML в мобильное приложение для iOS

CoreML [5] — это новая технология, представленная в iOS 11. Она является важным и преобладающим достижением в разработке приложений. CoreML позволяет разработчикам интегрировать модель машинного обучения в приложение для iPhone или iPad.

Основная идея CoreML заключается в том, чтобы облегчить разработчикам интеграцию моделей машинного обучения в приложения. Хотя iOS предлагает и другие способы интеграции машинного обучения, например, использование Python и последующая конвертация модели в формат, совместимый с iOS, CoreML дает множество преимуществ. Например, он оптимизирован для повышения энергоэффективности. Это очень важно для мобильных устройств, поскольку мы хотим максимально эффективно использовать ограниченный заряд батареи.

Причина, по которой CoreML может достичь этого, заключается в том, что для выполнения вычислений машинного обучения в нем используются шейдеры Metal Performance Shaders. Эти шейдеры высоко оптимизированы и работают на GPU. Помимо энергоэффективности, CoreML также обеспечивает очень простой и удобный процесс разработки.

2.1 Инициализация проекта

Чтобы можно было воспользоваться CoreML необходимо обучить модель нейронной сети. Компания Apple представила новый инструмент для работы с моделями в 2019 году под название CreateML [6]. Стартовая страница CreateML представлена на Рисунке 1.

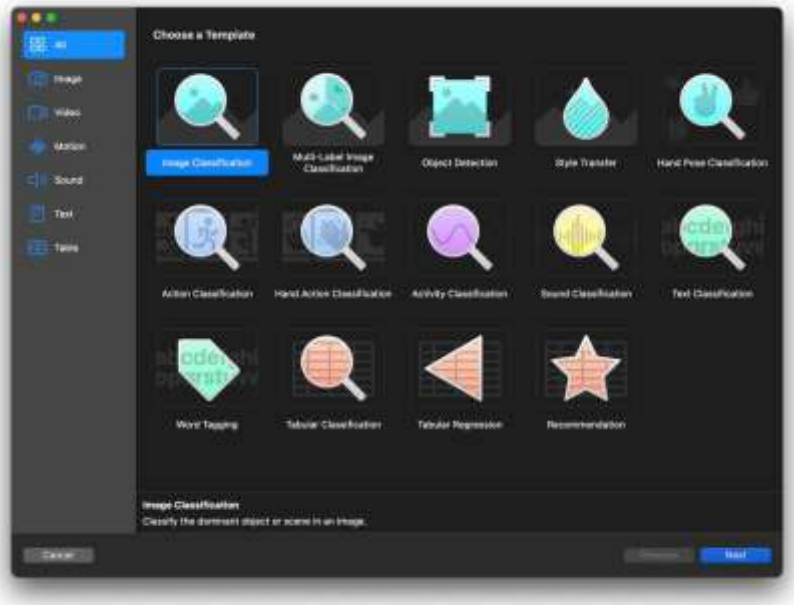

Рисунок 1 – Стартовый экран CreateML

На данный момент CreateML предоставляет возможность обучения на 6 типах данных и 14 шаблонах модели.

При обучении с использованием изображение доступны 5 шаблонов:

- Image Classification. Этот шаблон необходим для задач классификации изображений, где модель должна определить, к какому классу или категории принадлежит данное изображение. Например, классификация фотографий по типу животных, распознавание рукописных цифр и т. д.
- Multi-Label Image Classsification. В отличие от обычной классификации изображений, этот шаблон позволяет присваивать несколько меток или классов одному изображению. Это полезно, когда на изображении присутствуют различные объекты, которые могут быть классифицированы по разным категориям.
- Object Detection. Данный шаблон используется для обнаружения и локализации объектов на изображении. Он определяет прямоугольные области, где находятся объекты, и классифицирует их, позволяя модели точно определять и различать объекты на изображениях.
- Style Transfer. Этот шаблон позволяет переносить стиль одного изображения на другое, создавая новое изображение с сохранением содержания и характеристик первого изображения, но со стилем второго. Это позволяет создавать уникальные художественные эффекты и стилизованные изображения.
- Hand Pose Classification. Данный шаблон необходим для определения позы рук на изображениях. Это полезно для решения задач, связанных с анализом жестов, управлением интерфейсами с помощью жестов или виртуальной реальности.

При обучении с использованием видео доступно 3 шаблона:

- Style Transfer.
- Action Classification. Этот шаблон используется для классификации действий в видео. Он позволяет модели определять различные виды действий, происходящих в видеопотоке, что полезно для мониторинга видеонаблюдения, анализа деятельности и т. д.
- Hand Action Classification. Аналогично шаблону "Action Classification", но специализирован для классификации действий, связанных с движениями рук. Это может быть полезно для систем управления жестами или для анализа движений рук в реальном времени.

Для работы с активностью представлен лишь 1 шаблон:

- Activity Classification. Этот шаблон применяется для классификации общих активностей или событий в видеопотоке. Например, определение, происходит ли на видео ходьба, бег, игра в футбол и т. д. Это полезно для видеоаналитики и мониторинга поведения людей.

Также, как и с активностью, при работе со звуками доступен лишь один шаблон, а именно:

- Sound Classification. Данный шаблон используется для классификации звуковых сигналов. Например, определение типа звука (речь, музыка, шум), распознавание ключевых слов в аудиозаписи и т. д. Это полезно для систем анализа аудиоданных и обработки речи.

Для работы с текстом представлены 2 шаблона:

- Text Classification. Этот шаблон необходим для классификации текстов по определенным категориям или меткам. Например, определение темы новостной

статьи, классификация отзывов на продукты по тональности и т. д. Это полезно для автоматической обработки и анализа текстовой информации.

- Word Tagging. Данный шаблон позволяет размечать отдельные слова или токены в тексте с присвоением им соответствующих меток или категорий. Это помогает в семантическом анализе текста и его понимании компьютером.

Также для работы с таблицами предсталены 3 шаблона:

- Tabular Classification. Этот шаблон используется для классификации данных, представленных в табличной форме, по определенным категориям или меткам. Например, классификация клиентов по их покупательским предпочтениям или определение категории риска в финансовых операциях.
- Tabular Regression. Схож с классификацией, но вместо присваивания категорий предсказывает непрерывные значения для каждого объекта. Например, предсказание цены недвижимости на основе ее характеристик.
- Recommendation. Этот шаблон используется для предсказания предпочтений пользователей или рекомендации контента на основе их предыдущих действий или интересов. Это полезно для создания персонализированных рекомендательных систем в различных приложениях и сервисах.

Инициализируем проект и переходим к обучению. Выберем тип данных изображение и шаблон Image Classification. Экран инициализации проекта CreateML и экран обучения модели нейронной сети представлены на Рисунках 2-3.

Рисунок 2 – Экран инициализации проекта CreateML.

Рисунок 3 – Экран обучения модели нейронной сети

2.2 Подготовка и разметка данных

Следующий важный шаг после создания проекта - подготовка данных, необходимых для обучения модели машинного обучения. Обучающие данные обычно состоят из набора пар вход-выход. Обучающие данные используются для обучения модели машинного обучения, а тестовые - для проверки точности модели. Обычно данные разбиваются случайным образом, при этом значительная часть данных, например 70-80 %, используется для обучения, а оставшаяся часть - для тестирования.

Обучающие данные должны быть в формате, который распознается моделью, используемой для обучения. Структура и формат файлов для Image Classification представлены на Рисунке 4.

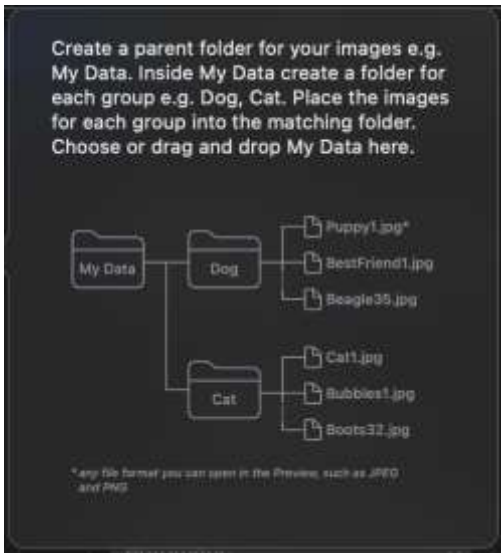

Рисунок 4 – Структура и формат данных для Image Classification

2.3 Обучение нейронной сети

После разметки данных и их импорта в проект необходимо выставить Parameters. Результаты загрузки данных в проект представлены на Рисунке 5.

Рисунок 5 – Результаты загрузки данных в проект

1. Feature extraction или превращение данных, специфических для предметной области, в понятные для модели векторы. Их доступно 2 на выбор:

- Image Feature Print V1 – модель извлечения объектов масштабирует входное изображение до 299 x 299 и выдает размер встраиваемого объекта 2048. Доступно на более старых версиях iOS
- Image Feature Print V2 - Модель извлечения объектов масштабирует входное изображение до 360 x 360 и выдает размер встраиваемого объекта 768. Доступно только с iOS 17

2. Iteration (Итерация):

Это количество повторений обучения, через которые проходят данные. Чем больше итераций, тем больше возможности для модели улучшить свои предсказания, но это также может привести к переобучению модели на обучающих данных.

3. Augmentations (доп. Настройки)

- Add noise (Добавить шум)
- Blur (Размытие)
- Crop (Обрезка)
- Expose (Экспозиция)
- Flip (Отражение)
- Rotate (Поворот)

Так как в iOS разработке принято поддерживать на 2 версии ОС меньше, то оставим Image Feature Print V1. Для более точного предсказания выставим 500 итераций и не будем добавлять другие эффекта.

Запустим обучения с помощью кнопки Train. Результаты обучения модели нейронной сети представлены на Рисунке 6.

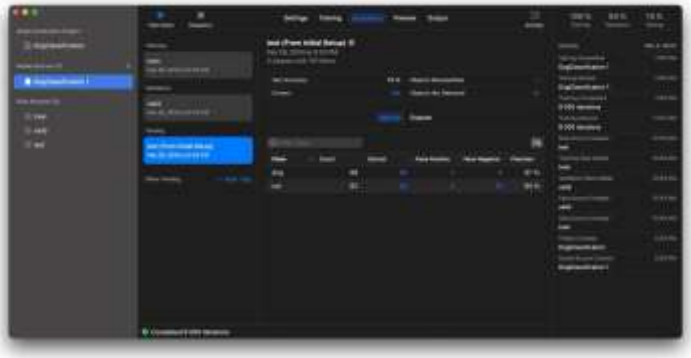

Рисунок 6 – Результаты обучения модели нейронной сети

2.4 Интеграция модели нейронной сети в iOS приложение.

Для начала создадим тестовый проект для проверки нейронной сети в IDE XCode. В качестве архитектурного подхода будем использовать MVC, а для реализации функционала и интерфейса UIKit.

Перенесем нашу созданную модель в проект и проверим ее настройки. Структура проекта и настройки модели представлены на Рисунке 7.

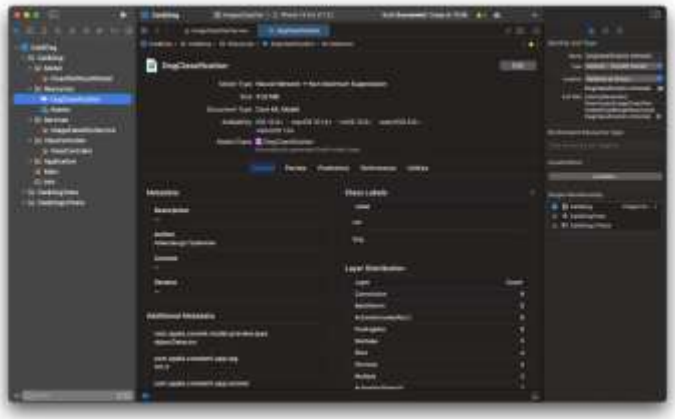

Рисунок 7 – Структура проекта и настройки модели

Следующим шагом реализуем контроллер, который будет брать изображение и прогонять его через классификатор, чтобы на выходе получить распознанный объект. Затем проверим его работоспособность на реальных тест-кейсах.

2.5 Тестирование нейронной сети

Тестирование можно производить двумя способами [6]. А именно:

- В самом приложение CreateML
- На мобильном устройстве или эмуляторе.

Мы будет тестировать вручную на реальном устройстве. Результаты тестирования представлены на Рисунке 8.

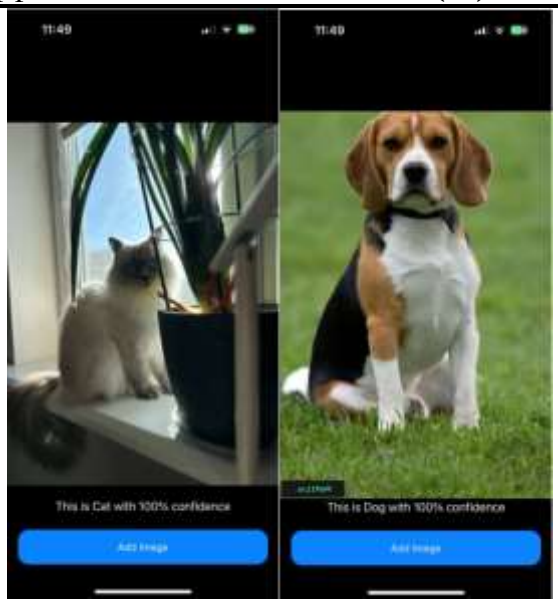

Рисунок 8 – Результаты тестирования

Как можно увидеть, в результате тестирования модель нейронной сети успешно классифицирует объекты на изображении с вероятностью 100%.

Результаты исследования

Исследование позволило выявить основные моменты, связанные с разработкой мобильных приложений на базе ОС iOS с внедрением CoreML технологий. Были освещены основы разработки мобильных приложений на iOS. Также были проанализированы возможности интеграции CoreML в мобильные приложения, включая подготовку и обработку данных для моделей машинного обучения, а также создание и тренировку самих моделей. Также были представлены методы тестирования моделей машинного обучения при интеграции CoreML.

Список литературы

1. Ларионов Д.А. Мобильное приложение на основе iOS. Москва: "ГелиосАРТ", 2017.
2. Императивный UIKit vs Декларативный SwiftUI - 2023, https://habr.com/ru/companies/ozontech/articles/742750/
3. Raúl Ferrer García. iOS Architecture Patterns: MVP, MVVM, VIPER, and VIP in Swift. - 2023, C.397.
4. Дьяков А.В., Наумова Е.А. Машинное обучение на языке Swift с использованием CoreML. Москва: "LRF Media", 2021.
5. Create ML. Create machine learning models for use in your app. – 2024, https://developer.apple.com/documentation/createml
6. Avi Tsadok. Pro iOS Testing – 2020, C.320.

References

1. Larionov D.A. Mobile application based on iOS. Moscow: HeliosART, 2017.
2. Imperative UIKit vs Declarative Swiftai - 2023, https://habr.com/ru/companies/ozontech/articles/742750/

3. Paul Ferrer Garcia. iOS Architecture Patterns: MVC, MVVM, VIPER, and VIP in Swift. - 2023, pp.397.
4. Dyakov A.V., Naumova E.A. Machine learning in Swift using Corel. Moscow: LRF Media, 2021.
5. Create ML. Create machine learning models for use in your app. – 2024, https://developer.apple.com/documentation/createml
6. Avi Tsadok. Pro iOS Testing – 2020, pp.320.