Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение

«Академия управления городской средой, градостроительства и печати»

УТВЕРЖДАЮ
Заместитель директора
по учебно-методической работе
_____О.В. Фомичёва
«____» _____ 20_____г.

Методические рекомендации по организации и проведению практических занятий

ПМ.01 «РАЗРАБОТКА КОДА ДЛЯ ОБУЧЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»
МДК.01.02. Разработка мобильных приложений с поддержкой искусственного интеллекта

для специальности

специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта

Форма обучение –очная

Санкт-Петербург
2025

Разработчик: Ипатова С.В./Оболенская Е.Г., методисты СПб ГБПОУ АУГСГиП

Одобрены на заседании цикловой комиссии

Общетехнических дисциплин и компьютерных технологий

Протокол № 4

09.12.2025 г.

Председатель цикловой комиссии:

Шурухина И.Е.

В результате изучения профессионального модуля обучающихся должен освоить основной вид деятельности ВД1. «Разработка кода для обучения искусственного интеллекта» и соответствующие ему общие компетенции и профессиональные компетенции:

1.1.1. Перечень общих компетенций

Код	Наименование общих компетенций
ОК 01	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях
ОК 04	Эффективно взаимодействовать и работать в коллективе и команде
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 06	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения
ОК 07	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях
ОК 08	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
ОК 09	Пользоваться профессиональной документацией на государственном и иностранном языках

1.1.2. Перечень профессиональных компетенций

Код	Наименование видов деятельности и профессиональных компетенций
ВД 1	Разработка кода для искусственного интеллекта
ПК 1.1	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
ПК 1.2	Разрабатывать программные модули в соответствии с техническим заданием.
ПК 1.3	Оформлять программный код в соответствии с техническим заданием.
ПК 1.4	Использовать систему контроля версий программного кода с учетом обеспечения возможности организации групповой разработки.
ПК 1.5	Выполнять отладку программных модулей с использованием специализированных программных средств.
ПК 1.6	Выполнять тестирование программного кода.
ПК1.7.	Составлять тестовые сценарии.

1.1.3. В результате освоения профессионального модуля обучающийся должен:

Иметь практический опыт	– Разработки, оптимизации и оценки сложности алгоритмов для ИИ-программ. – Использования библиотек и инструментов для работы с алгоритмами и данными (например: Pandas, NumPy, Scikit-learn).

	– Применения структур данных (деревья, графы, списки) для реализации алгоритмов. – Разработки модульных ИИ-систем, соответствующих требованиям производительности и безопасности. – Внедрения разработанных ИИ-модулей в комплексные программные системы. – Оптимизации кода и работы с интерфейсами для взаимодействия между модулями. – Оформления, документирования и структурирования кода для последующей поддержки. – Использования инструментов статического анализа кода для выявления ошибок и улучшения качества. – Работы с системами документирования кода (например, Doxygen, Sphinx). – Управления проектами с использованием систем контроля версий для организации командной работы. – Разрешения конфликтов при слиянии веток и использования pull request для рецензирования кода. – Настройки процессов CI/CD для автоматического тестирования и развертывания кода. – Отладки программных модулей с использованием пошаговой проверки. – Применения методов логирования и профилирования производительности. – Использования специальных средств для отладки многопоточных программ. – Выполнения статического тестирования программного кода на предмет выявления ошибок/дефектов алгоритмов, в том числе – на наличие обработки исключений – Выполнения тестирования программных модулей в соответствии в тест-планом – Генерирования тестовых данных – Выполнения интеграционного тестирования в соответствии с заданием – Выполнения регрессионного тестирования в соответствии с заданием. – Работы с CI/CD пайплайнами для автоматизации тестирования. – Разработки тестовых сценариев в соответствии с тестовым планом (тестирование производительности, надежности, UI-тестирование), в том числе с применением средств автоматизации проектирования. – Разработки тестовых пакетов и заданий на выполнение тестирования. – Оценки тестовых данных на предмет покрытия строк и покрытия ветвей, выполнения валидации данных. – Автоматизации создания и выполнения тестовых сценариев.
Уметь	– Анализировать технические задания и выявлять требования к алгоритмам. – Применять методы алгоритмизации для решения задач программирования.

	– Разрабатывать оптимальные алгоритмы для решения задач в области ИИ. – Реализовывать программные модули на основе требований технического задания. – Соблюдать при разработке принципы «чистого кода». – Использовать стандартные библиотеки и фреймворки для ускорения разработки. – Оформлять код в соответствии с принятыми стандартами и требованиями. – Документировать разработанный программный код. – Соблюдать соглашения о наименованиях переменных, функций и классов (например, PEP8 для Python). – Работать с системами контроля версий для управления проектами. – Организовывать совместную работу над проектом через ветки разработки и слияние изменений. – Разрешать конфликты при слиянии кода. – Использовать инструменты для отладки программного кода. – Идентифицировать и исправлять ошибки в программе. – Применять методы логирования для анализа выполнения программ. – Проводить различные виды тестирования (юнит-тестирование, интеграционное тестирование). – Выполнять настройки окружения т подготовку тестовых данных – Фиксировать результаты выполнения тестов и подготавливать отчеты о результатах тестов. – Определять уровень критичности дефектов. – Разрабатывать автоматизированные тесты для тестирования модулей и/или отдельных функций – Восстанавливать окружение и тесты после сбоя – Проектировать тестовые сценарии на основе тестовых планов. – Разрабатывать тестовые пакеты и задания на выполнение тестирования. – Использовать шаблоны для написания тест-кейсов. – Оценивать риски при отборе тестов для регрессионного тестирования. – Оценивать тесты на соответствие целям тестирования.
Знать	– Основные методы и подходы к построению алгоритмов (типовые поисковые алгоритмы, жадные алгоритмы, динамическое программирование, рекурсивные подходы). – Принципы эффективной обработки данных. – Языки программирования, применяемые для разработки алгоритмов. – Принципы модульного программирования. – Языки программирования для разработки модулей. – Стандартные фреймворки и библиотеки для работы с ИИ. – Основные принципы чистого кода (Clean Code). – Стандарты и практики документирования программного обеспечения. – Инструменты для автоматической проверки качества кода (например, PyLint, ESLint).

	– Принципы работы распределенных систем контроля версий. – Основные команды и операции в системах контроля версий (например: commit, pull, push, merge). – Методы разрешения конфликтов в ходе групповой разработки. – Принципы работы отладчиков и логирования. – Способы выявления ошибок в программе (отладка по шагам, точки останова). – Инструменты для отладки кода (например, PyCharm, Visual Studio Debugger). – Техники выполнения тестовых прогонов. – Инструменты и среды выполнения тестирования – Языки разработки автоматизированных тестов – Инструменты для тестирования программного кода. – Правила выполнения отчетов о тестировании – Цели, задачи и виды тестирования. Понятие стратегии тестирования. – Жизненный цикл дефекта. – Основы тест-дизайна: тестовый сценарий, тестовый пакет, чек-лист, основные шаблоны. – Основные инструменты проектирования тестов. – Методы и подходы к написанию тестов (Test-Driven Development, Behavior-Driven Development).

Практические работы

тема	название ПР	часы
МДК.01.02. Разработка мобильных приложений с поддержкой искусственного интеллекта		
Тема 2.1. Платформы и инструменты мобильной разработки	**Практическая работа №1.** Создание первого Android-приложения с базовыми интерфейсами.	14
	Практическая работа №2. Разработка пользовательского интерфейса для мобильного приложения.	14
Тема 2.2. Интеграция ИИ в мобильные приложения	**Практическая работа №3.** Внедрение TensorFlow Lite модели в Android-приложение.	14
	Практическая работа №4. Оптимизация ИИ-модели для мобильного устройства.	14
Тема 2.3. Разработка интерактивных мобильных ИИ-приложений	**Практическая работа №5.** Разработка мобильного приложения для распознавания изображений.	14
	Практическая работа №6. Внедрение голосового помощника на основе ИИ в мобильное приложение.	16
Тема 2.4. Развертывание мобильных приложений с ИИ	**Практическая работа №7**. Автоматизация тестирования мобильного ИИ-приложения.	14
	Практическая работа №8. Развертывание мобильного приложения в магазинах мобильных приложений.	16
		116

МДК.01.02. Разработка мобильных приложений с поддержкой искусственного интеллекта

Практическая работа №1. Создание первого Android-приложения с базовыми интерфейсами.

Цель работы: освоить базовые этапы создания Android-приложения в среде **Android Studio**: от инициализации проекта до запуска на устройстве/эмуляторе, а также познакомиться с основными компонентами интерфейса (Activity, XML-разметка, ресурсы).

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Оборудование и ПО

- Компьютер с ОС Windows/macOS/Linux.
- Установленная **Android Studio** (последняя стабильная версия).
- USB-кабель (для подключения реального устройства) *или* настроенное виртуальное устройство (AVD).

Шаг 1. Создание нового проекта

1. Откройте **Android Studio**.
2. На стартовом экране выберите **Start a new Android Studio project**.
3. В окне **New Project**:
   - **Name**: введите название приложения (например, `MyFirstApp`).

- **Package name**: укажите уникальное имя пакета (например, `com.example.myfirstapp`).
- **Save location**: выберите папку для сохранения проекта.
- **Language**: выберите **Java** или **Kotlin** (по умолчанию — Kotlin).
- **Minimum API level**: оставьте значение по умолчанию (обеспечивает совместимость с большинством устройств).

4. Нажмите **Finish**.

Примечание: Android Studio создаст проект с базовой структурой.

Шаг 2. Изучение структуры проекта

После создания проекта вы увидите следующие ключевые папки:

- **app/manifests**

Файл `AndroidManifest.xml` — описывает конфигурацию приложения (разрешения, компоненты).

- **app/java**

Код на Java/Kotlin. По умолчанию содержит класс `MainActivity.kt` (или `MainActivity.java`), который запускается при старте приложения.

- **app/res**

Ресурсы приложения:

- `layout/` — XML-файлы интерфейса (например, `activity_main.xml`).
- `values/` — строковые ресурсы (`strings.xml`), цвета (`colors.xml`), стили.
- `drawable/` — изображения.
- `mipmap/` — иконки приложения для разных разрешений.

Шаг 3. Настройка интерфейса (XML-разметка)

1. Откройте файл `res/layout/activity_main.xml`.
2. В режиме **Design** добавьте элементы интерфейса:
   - **TextView**: перетащите на экран, измените текст (например, «Добро пожаловать!») через атрибут `text`.
   - **Button**: добавьте кнопку, задайте текст (например, «Нажми меня»).
   - **EditText**: поле для ввода текста.
3. В режиме **Code** проверьте XML-код. Пример:

```xml
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
```

```
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Добро пожаловать!"
        android:textSize="24sp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Нажми меня" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Введите текст" />
</LinearLayout>
```

Пояснения:

- `android:id` — уникальный идентификатор элемента.
- `android:layout_width` и `android:layout_height` — размеры.
- `android:text` — текст элемента.

Шаг 4. Программирование логики (Java/Kotlin)

1. Откройте файл `java/com.example.myfirstapp/MainActivity.kt` (или `.java`).
2. Добавьте обработчик нажатия кнопки. Пример для **Kotlin**:

```
kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val button = findViewById<Button>(R.id.button)
        val textView = findViewById<TextView>(R.id.textView)
        val editText = findViewById<EditText>(R.id.editText)

        button.setOnClickListener {
            val inputText = editText.text.toString()
            textView.text = "Вы ввели: $inputText"
        }
    }
}
```

Для **Java** код будет аналогичным (с учётом синтаксиса Java).

Объяснение:

- `findViewById()` — находит элемент по ID.
- `setOnClickListener()` — задаёт действие при нажатии кнопки.
- `editText.text.toString()` — получает текст из поля ввода.

Шаг 5. Запуск приложения

Вариант 1. На реальном устройстве

1. Подключите устройство к компьютеру через USB.
2. Активируйте **режим разработчика** на устройстве:
   - Перейдите в **Настройки → О телефоне → Номер сборки** и нажмите 7 раз.
3. Включите **отладку по USB** в настройках разработчика.
4. В Android Studio нажмите **Run** (зелёная стрелка).
5. Выберите своё устройство в списке и нажмите **OK**.

Вариант 2. На эмуляторе (AVD)

1. В Android Studio откройте **AVD Manager** (кнопка на панели инструментов).
2. Нажмите **Create Virtual Device**, выберите модель устройства (например, **Pixel 3**) и образ системы.
3. Запустите эмулятор, затем нажмите **Run** в Android Studio и выберите эмулятор.

Шаг 6. Проверка работы

После запуска:

1. Введите текст в поле **EditText**.
2. Нажмите кнопку **Нажми меня**.
3. Убедитесь, что `TextView` отображает введённый текст.

Контрольные вопросы

1. Какие основные компоненты входят в структуру Android-проекта?
2. Для чего используется файл `AndroidManifest.xml`?
3. Как добавить обработчик нажатия кнопки в Kotlin/Java?
4. Какие способы запуска приложения существуют?

Отчёт по работе

Подготовьте отчёт, включающий:

1. Скриншоты этапов создания проекта (структура, XML-разметка, код Activity).
2. Описание выполненных действий.
3. Ответы на контрольные вопросы.
4. Вывод о проделанной работе.

Возможные ошибки и их решения

- **«Failed to launch emulator»**: проверьте, включён ли **Intel HAXM** (в SDK Manager → SDK Tools).

- **«App keeps stopping»**: проверьте логи в **Logcat** (внизу Android Studio) для поиска ошибок.
- **«Device not detected»**: убедитесь, что USB-драйверы установлены и отладка по USB активирована.

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №2. Разработка пользовательского интерфейса для мобильного приложения.

Цель работы: спроектировать пользовательский интерфейс мобильного приложения, соблюдая принципы UX/UI-дизайна и современные гайдлайны (Material Design, Human Interface Guidelines).

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической

работы

Задачи

1. Определить целевую аудиторию и сценарии использования приложения.
2. Разработать структуру экранов и навигацию.
3. Создать вайрфреймы (черновые макеты) интерфейсов.
4. Пропроектировать визуальный стиль (цвета, шрифты, иконки).
5. Собрать интерактивный прототип.
6. Провести базовое юзабилити-тестирование.

Ход работы

1. Анализ требований и пользовательских сценариев

- Определите **назначение приложения** (например: трекер привычек, онлайн-магазин, мессенджер).
- Выделите **ключевые пользовательские сценарии** (например: «регистрация → выбор товара → оформление заказа»).
- Составьте список **обязательных экранов** (главный, настройки, детализация и т. п.).

2. Структура и навигация

- Постройте **схему экранов** (sitemap) с указанием переходов.
- Выберите тип навигации:
  - Bottom Tab Bar (вкладки внизу);
  - Hamburger Menu (боковое меню);
  - Stack Navigation (последовательные экраны).
- Продумайте **обратные переходы** (кнопка «Назад», жесты).

3. Вайрфреймы

Создайте чёрно-белые макеты каждого экрана с размещением:

- заголовков;
- основных блоков контента;
- кнопок и полей ввода;
- навигационных элементов.

Инструменты: Figma, Adobe XD, Sketch, Balsamiq.

4. Визуальный дизайн

Разработайте **дизайн-систему**:

- **Цветовая палитра**: 2–3 основных цвета + акценты (используйте инструменты вроде Coolors или Adobe Color).
- **Шрифты**: 1–2 гарнитуры (например, Roboto + Montserrat).
- **Иконки**: набор унифицированных символов (библиотеки: Material Icons, Font Awesome).
- **Отступы и сетки**: модульная сетка (например, 8-pt grid).
- **Состояние элементов**: активные/неактивные кнопки, загрузки, ошибки.

5. Прототипирование

- Соедините экраны **интерактивными связями** (onTap, onScroll).
- Добавьте **микроанимации** (например, плавное появление элементов).
- Проверьте **адаптивность** под разные размеры экрана (smartphone, tablet).

Инструменты: Figma (Prototype mode), Adobe XD, InVision.

6. Юзабилити-тестирование

Проведите тестирование с 3–5 участниками:

- Дайте **задачи** (например: «Найдите товар и добавьте в корзину»).
- Фиксируйте:
  - время выполнения действий;
  - затруднения;
  - субъективные комментарии.
- Внесите **итерационные правки** в прототип.

Требования к отчёту

Предоставьте:

1. **Описание приложения**: цель, аудитория, ключевые функции.
2. **Схему навигации** (изображение или ссылка на инструмент).
3. **Вайрфреймы** всех экранов.
4. **Дизайн-систему** (цвета, шрифты, примеры компонентов).
5. **Интерактивный прототип** (ссылка на Figma/Adobe XD).
6. **Результаты тестирования**: 2–3 выявленные проблемы и способы их решения.

Критерии оценки

- Логичность навигации и пользовательских сценариев.
- Соблюдение принципов UX/UI (контрастность, читаемость, консистентность).
- Качество интерактивного прототипа (плавные переходы, обратная связь).
- Обоснованность решений на основе тестирования.

Пример структуры приложения (трекер привычек)

1. **Экран входа/регистрации** — кнопка «Войти», «Зарегистрироваться».
2. **Главный экран** — список привычек, кнопка «+».
3. **Детализация привычки** — прогресс, история, настройки.
4. **Настройки** — профиль, уведомления, тема.
5. **Статистика** — графики выполнения за неделю/месяц.

Полезные ресурсы

- **Гайдлайны**:
  - Material Design (material.io);
  - Apple HIG (developer.apple.com/design).
- **Библиотеки компонентов**:
  - Figma Community (готовые UI-киты);
  - Google Material Symbols.
- **Тестирование**:
  - UserTesting.com (для удалённого тестирования);
  - Lookback.io (запись экрана и эмоций).

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №3. Внедрение TensorFlow Lite модели в Android-приложение.

Цель работы: освоить процесс интеграции обученной модели машинного обучения (в формате TensorFlow Lite) в мобильное приложение на платформе Android.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Время на выполнение работы*:** 90 мин

Оборудование, технические средства и инструменты:

3. Методические рекомендации для практических занятий
4. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Необходимые инструменты и ПО

- Android Studio (актуальная версия);
- Python + TensorFlow (для обучения и конвертации модели);
- JDK 11+;
- эмулятор Android или физическое устройство для тестирования.

Шаг 1. Подготовка модели TensorFlow Lite

1. **Обучите модель** в Python (например, с использованием Keras):

```
python
```

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    # слои вашей модели
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10)
```

2. **Конвертируйте модель в формат TFLite**:

```
python
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Сохраняем модель
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

Шаг 2. Настройка Android-проекта

1. Создайте новый проект в Android Studio (Empty Activity, язык — Kotlin).
2. Добавьте зависимости в `app/build.gradle` (в блок `dependencies`):

```
gradle
implementation 'org.tensorflow:tensorflow-lite:2.14.0'
implementation 'org.tensorflow:tensorflow-lite-support:0.5.0'
```

3. Синхронизируйте проект (Sync Now).

Шаг 3. Размещение модели в проекте

1. Создайте папку `assets` в `app/src/main/`.
2. Поместите файл `model.tflite` в эту папку.

Шаг 4. Загрузка модели в приложении

Создайте класс-обработчик модели (например, `TFLiteModelHandler.kt`):

```
kotlin
import org.tensorflow.lite.Interpreter
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel
import android.content.Context
import java.io.FileInputStream

class TFLiteModelHandler(private val context: Context) {

    private lateinit var interpreter: Interpreter

    @Throws(IOException::class)
    fun loadModel() {
        val assetFileDescriptor = context.assets.openFd("model.tflite")
        val inputStream = FileInputStream(assetFileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = assetFileDescriptor.startOffset
```

```
            val declaredLength = assetFileDescriptor.declaredLength
            val mappedByteBuffer = fileChannel.map(
                FileChannel.MapMode.READ_ONLY, startOffset, declaredLength
            )
            interpreter = Interpreter(mappedByteBuffer)
            assetFileDescriptor.close()
        }

        fun closeModel() {
            interpreter.close()
        }
    }
```

Шаг 5. Предобработка входных данных

Пример для изображения (размер 224×224, RGB):

```
kotlin
private fun preprocessImage(bitmap: Bitmap): FloatArray {
    val scaledBitmap = Bitmap.createScaledBitmap(bitmap, 224, 224, true)
    val input = FloatArray(224 * 224 * 3)

    for (y in 0 until 224) {
        for (x in 0 until 224) {
            val pixel = scaledBitmap.getPixel(x, y)
            input[y * 224 * 3 + x * 3] = ((pixel shr 16) and 0xFF) / 255.0f
            input[y * 224 * 3 + x * 3 + 1] = ((pixel shr 8) and 0xFF) / 255.0f
            input[y * 224 * 3 + x * 3 + 2] = (pixel and 0xFF) / 255.0f
        }
    }
    return input
}
```

Шаг 6. Запуск инференса

Вызов модели в фоновом потоке (например, через Coroutine):

```
kotlin
fun runInference(inputData: FloatArray): FloatArray {
    val output = FloatArray(NUM_CLASSES) // замените на число классов вашей модели
    interpreter.run(inputData, output)
    return output
}
```

Шаг 7. Постобработка результатов

Пример интерпретации вывода (классификация):

```
kotlin
private fun interpretOutput(output: FloatArray): String {
    val maxIndex = output.indices.maxByOrNull { output[it] } ?: 0
    return classLabels[maxIndex] // массив меток классов
}
```

Шаг 8. Интеграция с UI

1. В `activity_main.xml` добавьте:
   - `ImageView` для загрузки изображения;
   - `Button` для запуска инференса;
   - `TextView` для вывода результата.
2. В `MainActivity.kt` обработайте клик по кнопке:

```
kotlin
button.setOnClickListener {
    lifecycleScope.launch(Dispatchers.Default) {
        val input = preprocessImage(imageView.drawable.toBitmap())
        val output = runInference(input)
        val result = interpretOutput(output)

        withContext(Dispatchers.Main) {
            resultTextView.text = result
        }
    }
}
```

Рекомендации по оптимизации

1. **Квантование модели** (уменьшение размера и ускорение):

```
python
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
```

2. **Использование GPU/DSP** (добавьте зависимость):

```
gradle
implementation 'org.tensorflow:tensorflow-lite-gpu:2.14.0'
```

Инициализация с ускорением:

```
kotlin
val options = Interpreter.Options()
options.setUseGPU(true)
interpreter = Interpreter(mappedByteBuffer, options)
```

3. **Фоновые потоки** — всегда запускайте инференс в `Dispatchers.Default` или `Executors`.

Контроль качества

1. Проверьте:
   - корректность загрузки модели (`Interpreter` не `null`);
   - совпадение размеров входных/выходных тензоров;
   - точность предсказаний на тестовых данных.
2. Протестируйте на устройствах с разными версиями Android (minSdkVersion ≥ 21).

Отчёт по работе

Включите в отчёт:

1. Скриншот работающего приложения.

2. Листинги ключевых фрагментов кода (загрузка модели, предобработка, инференс).

3. Описание возникших проблем и их решений.

4. Анализ производительности (время инференса, размер модели).

Критерии оценки:

- корректная загрузка модели;
- работоспособность инференса;
- соблюдение принципов асинхронности;
- качество постобработки результатов.

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1.Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №4. Оптимизация ИИ-модели для мобильного устройства.

Цель работы: разработать методы, которые позволят эффективно использовать ограниченные вычислительные ресурсы мобильных устройств. Например, улучшить производительность алгоритмов распознавания изображений на основе машинного обучения.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.

2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.

4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).

5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Задачи

- **Изучить существующие методы оптимизации**. Например, квантование (уменьшение точности чисел с плавающей запятой) или прунинг (обрезка) — удаление наименее значимых нейронов и связей в нейросети.
- **Выбрать специализированные фреймворки** для мобильного ИИ, которые поддерживают современные модели машинного обучения и оптимизированы под ограниченные ресурсы смартфонов. Например, TensorFlow Lite, Core ML.
- **Прототипировать модель** на ПК, проверить её точность, а затем протестировать на целевых устройствах.
- **Сравнить альтернативы** — иногда простая эвристика работает лучше, чем «тяжёлый» ИИ.

cyberleninka.rudeepme.ru

Методика

Пошаговая стратегия оптимизации: deepme.ru

1. Замерить базовые показатели — размер модели, время вывода, использование памяти.
2. Применить квантование — самый простой метод с предсказуемым результатом.
3. Экспериментировать с прунингом — начать с 20–30% обрезки.
4. Тестировать на реальных устройствах — разные процессоры могут давать разную производительность.
5. Повторять цикл оптимизации до достижения целевых показателей.

deepme.ru

Важно: идеальной оптимизации не существует — важно найти баланс между размером модели, скоростью работы и точностью предсказаний конкретно для конкретного use-case. deepme.ru

Ресурсы

Для выполнения работы можно использовать:

- **Готовые инструменты** для мобильного ИИ, например, ML Kit от Google (включает готовые решения для распознавания текста, лиц).
- **Специализированные архитектуры** — MobileNet, EfficientNet и другие сети, разработанные специально для мобильных устройств.
- **Разделение модели** — загрузка части вычислений на сервер при наличии интернет-соединения.

deepme.ru

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №5. Разработка мобильного приложения для распознавания изображений.

Цель работы: разработать функциональное мобильное приложение, способное распознавать заданные объекты на изображениях с использованием технологий машинного обучения.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

***Время на выполнение работы*:** 90 мин

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Задачи

1. Изучить существующие подходы к распознаванию изображений.
2. Выбрать подходящую архитектуру нейронной сети.

3. Подготовить датасет для обучения модели.
4. Обучить модель распознавания.
5. Интегрировать модель в мобильное приложение.
6. Реализовать интерфейс пользователя.
7. Протестировать работоспособность приложения.

Теоретическая часть

Основные подходы к распознаванию изображений

- **Классические методы** (алгоритм Виолы-Джонса, SIFT, HOG) — основаны на выделении ключевых признаков вручную.
- **Глубокое обучение** — использование свёрточных нейронных сетей (CNN), автоматически извлекающих признаки.

Рекомендуемые архитектуры CNN

- **MobileNetV2** — оптимизирована для мобильных устройств, малый размер и высокая скорость.
- **EfficientNet** — баланс точности и производительности.
- **YOLO (You Only Look Once)** — для задач реального времени.

Фреймворки для мобильной интеграции

- **TensorFlow Lite** — официальная оптимизация моделей TensorFlow для мобильных платформ.
- **PyTorch Mobile** — поддержка моделей PyTorch на Android/iOS.
- **Core ML** (iOS-специфичный).

Практическая реализация

Шаг 1. Подготовка датасета

1. Собрать изображения объекта (минимум 200-300 экземпляров).
2. Разметить данные (bounding boxes или классы).
3. Разделить на train/validation/test (70%/15%/15%).
4. Применить аугментацию (поворот, яркость, обрезка).

Шаг 2. Обучение модели

1. Выбрать предобученную модель (например, MobileNetV2).
2. Дообучить на своём датасете (transfer learning).
3. Оценить метрики: точность (accuracy), полнота (recall), F1-score.
4. Конвертировать в мобильный формат (TFLite, Core ML).

Пример кода обучения (TensorFlow/Keras):

```
python
model = tf.keras.applications.MobileNetV2(input_shape=(224, 224, 3),
                                           include_top=False,
                                           weights='imagenet')
model.trainable = False

global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction_layer = tf.keras.layers.Dense(num_classes, activation='softmax')

model = tf.keras.Sequential([
  model,
  global_average_layer,
```

```
    prediction_layer
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_data, epochs=10, validation_data=val_data)
```

Шаг 3. Разработка мобильного приложения

Для Android (Kotlin/Java):

1. Добавить зависимость TensorFlow Lite в `build.gradle`.
2. Разместить модель в `assets/`.
3. Создать класс для инференса:

```
kotlin
val interpreter = Interpreter(fileDescriptor)
val input = Array(1) { Array(224) { Array(224) { FloatArray(3) } } }
val output = Array(1) { FloatArray(numClasses) }
interpreter.run(input, output)
```

Для iOS (Swift):

1. Импортировать Core ML модель.
2. Использовать `VNCoreMLRequest` для обработки изображений.

текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1.Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №6. Внедрение голосового помощника на основе ИИ в мобильное приложение.

Цель работы: освоить процесс интеграции голосового ИИ-помощника в мобильное приложение: от постановки задачи до тестирования и развёртывания.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Теоретические основы

Голосовой помощник — программный агент, использующий технологии:

- **ASR (Automatic Speech Recognition)** — распознавание речи;
- **NLP (Natural Language Processing)** — обработка естественного языка;
- **TTS (Text-to-Speech)** — синтез речи.

Основные платформы и инструменты:

- **Облачные API:** Google Cloud Speech-to-Text + Dialogflow, Amazon Lex, Microsoft Azure Cognitive Services.
- **Локальные фреймворки:** TensorFlow Lite, Core ML (iOS), ML Kit (Android).
- **Готовые решения:** Speechify TTS API, IBM Watson Assistant.

Пошаговый алгоритм внедрения

Шаг 1. Определение целей и сценариев

Задайте:

- Какие задачи решит помощник? (навигация, ответы на вопросы, выполнение действий).
- Какие команды будут поддерживаться? (например: *«Открой настройки»*, *«Найди ближайший магазин»*).
- Каковы KPI успеха? (снижение нагрузки на поддержку на 30%, увеличение вовлечённости на 20%).

Шаг 2. Выбор технологической платформы

Варианты:

- **Облачное решение** (быстро, но требует интернета):
  - Google Dialogflow + Speech-to-Text API.
  - Amazon Lex.
- **Локальная обработка** (офлайн, выше приватность):
  - TensorFlow Lite с моделью распознавания речи.
  - Core ML для iOS.

Критерии выбора:

- Необходимость работы без интернета.
- Требования к конфиденциальности данных.
- Бюджет и сроки разработки.

Шаг 3. Проектирование диалогового сценария

Создайте **блок-схему диалога** с вариантами ответов пользователя. Пример:

```
Пользователь: «Покажи баланс»
→ Система: «Ваш баланс: 5 000 ₽. Что ещё хотите узнать?»
→ Пользователь: «История операций»
→ Система: «Последние 5 операций: ...»
```

Шаг 4. Интеграция API/фреймворка

Для облачного решения (на примере Dialogflow):

1. Создайте проект в Google Cloud Console.
2. Настройте агента в Dialogflow:
   - Определите **intents** (намерения) — например, «Узнать баланс».
   - Добавьте **training phrases** (примеры фраз пользователя).
   - Задайте **responses** (ответы системы).
3. Получите API-ключ для мобильного приложения.

Для локальной модели (TensorFlow Lite):

1. Обучите модель на наборе данных (например, Common Voice).
2. Конвертируйте модель в формат TFLite.
3. Добавьте файл модели в ресурсы приложения.

Шаг 5. Разработка мобильного клиента

Ключевые компоненты:

1. **Микрофонный ввод:**
   - Android: `MediaRecorder` или `SpeechRecognizer`.
   - iOS: `AVAudioRecorder` + `SFSpeechRecognizer`.
2. **Отправка аудио на обработку:**
   - Для облачных API — HTTP-запрос с аудиоданными.
   - Для локальных моделей — вызов TFLite-интерпретатора.
3. **Синтез речи:**
   - Используйте TTS-API (например, Google Cloud Text-to-Speech).
   - На iOS: `AVSpeechSynthesizer`.
   - На Android: `TextToSpeech` API.

Пример кода (Android, распознавание речи через Google API):

```
java
private void startRecognition() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    startActivityForResult(intent, REQUEST_CODE_SPEECH_INPUT);
}
```

Шаг 6. Тестирование

Проверьте:

- Точность распознавания команд (тест на шуме, разных акцентах).
- Задержку ответа (оптимально < 1 с).
- Работу в офлайн-режиме (если предусмотрено).
- Корректность синтеза речи.

Методы:

- Юнит-тесты для логики диалога.
- Ручное тестирование на устройствах.
- A/B-тестирование с реальными пользователями.

Шаг 7. Развёртывание и мониторинг

1. Выпустите бета-версию для ограниченной группы пользователей.
2. Соберите обратную связь (например, через Firebase Analytics).
3. Отслеживайте метрики:
   - % успешных распознаваний;
   - среднее время ответа;
   - количество отказов.
4. Дообучайте модель на новых данных.

Типичные проблемы и решения

- **Низкое качество распознавания:**
  - Добавьте больше тренировочных фраз в intents.
  - Используйте шумоподавление перед отправкой аудио.
- **Высокие задержки:**
  - Переходите на локальную обработку (TFLite).
  - Кэшируйте часто используемые ответы.
- **Проблемы с приватность:**
  - Обрабатывайте данные на устройстве.
  - Получите согласие пользователя на сбор аудио.

5. Контрольные вопросы

1. Какие технологии лежат в основе голосовых помощников?
2. В чём разница между облачным и локальным подходом к обработке речи?
3. Как протестировать точность распознавания команд?
4. Какие метрики важны для оценки работы помощника?

6. Задание для самостоятельной работы

Разработайте прототип мобильного приложения (Android/iOS) с голосовым помощником, который:

- распознаёт 5 команд (например: «Открыть карту», «Проверить баланс», «Позвонить в поддержку»);
- отвечает синтезированным голосом;
- работает в офлайн-режиме (используйте TFLite или локальный TTS).

Требования к отчёту:

1. Описание выбранных технологий и аргументов в их пользу.
2. Схема диалогового сценария.
3. Скриншоты интерфейса приложения.
4. Результаты тестирования (точность распознавания, задержки).

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №7. Автоматизация тестирования мобильного ИИ-приложения.

Цель работы: освоить методы и инструменты автоматизации тестирования мобильных приложений с элементами искусственного интеллекта, сформировать набор автотестов и проанализировать их эффективность.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ

- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.
3. Предъявить преподавателю результаты работы.
4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Задачи

1. Выбрать инструменты автоматизации для мобильного ИИ-приложения.
2. Разработать тестовые сценарии, учитывающие специфику ИИ-функционала.
3. Реализовать автотесты для ключевых пользовательских сценариев.
4. Провести тестирование, собрать и проанализировать результаты.
5. Оценить преимущества и ограничения автоматизации с ИИ.

Теоретическая часть

Особенности тестирования ИИ-приложений

- **Недетерминированность**: результаты работы ИИ могут варьироваться при одинаковых входных данных.
- **Обучаемость**: модель меняется после дообучения, требуется регрессионное тестирование.
- **Чёрно-ящичность**: сложно проверить внутреннюю логику модели.
- **Данные**: качество тестов зависит от репрезентативности тестовых датасетов.

Ключевые виды тестирования

1. **Функциональное** — проверка соответствия требованиям (например, корректность распознавания объектов).
2. **Производительность** — время ответа модели, нагрузка на устройство.
3. **Стабильность** — поведение при длительных сессиях.
4. **Безопасность** — защита персональных данных, устойчивость к атакам.
5. **Юзабилити** — удобство интерфейса для взаимодействия с ИИ.

Практическая часть

Шаг 1. Выбор инструментов

Рекомендуемые стеки:

- **Appium** + **JUnit/TestNG** (кросс-платформенная автоматизация).
- **Espresso** (Android) / **XCUITest** (iOS) для нативных тестов.
- **Selenium** + **Appium** для веб-компонентов.
- **Applitools** или **Percy** для визуального тестирования (сравнение скриншотов).
- **TensorFlow Lite Inspector** для валидации моделей на устройстве.

Шаг 2. Разработка тестовых сценариев

Примеры сценариев для ИИ-приложения (например, фоторедактор с нейрофильтрами):

1. **Загрузка изображения**:
   - Проверить поддержку форматов (JPEG, PNG).
   - Валидировать ограничение по размеру файла.
2. **Применение фильтра**:
   - Убедиться, что фильтр применяется без ошибок.
   - Сравнить выходное изображение с эталонным (допустимое отклонение: ≤5% по метрике SSIM).
3. **Отмена действия**:
   - Проверить возврат к исходному изображению.
4. **Обработка ошибок**:
   - Имитировать отсутствие интернета — должно появиться сообщение.
   - Загрузить некорректный файл — приложение не должно крашиться.

Шаг 3. Реализация автотестов (пример на Appium + Java)

```java
@Test
public void applyFilterTest() {
    // 1. Загрузка изображения
    driver.findElement(By.id("upload_btn")).click();
    driver.findElement(By.id("select_image")).sendKeys("test.jpg");

    // 2. Применение фильтра
    driver.findElement(By.id("filter_btn")).click();
    driver.findElement(By.xpath("//android.widget.TextView[@text='Neural Style']")).click();

    // 3. Проверка результата
    WebElement resultImage = driver.findElement(By.id("result_image"));
    Assert.assertTrue(resultImage.isDisplayed());
}
```

Шаг 4. Запуск и анализ результатов

1. Выполнить тесты на реальных устройствах и эмуляторах.
2. Собрать отчёты (например, через **Allure**).
3. Проанализировать:
   - Процент пройденных тестов ($P \geq 90\%$ — успешно).
   - Время выполнения (целевое: ≤5 мин для полного набора).
   - Ложные срабатывания (допустимо: ≤2%).

Шаг 5. Оценка эффективности

Плюсы автоматизации:

- Сокращение времени тестирования на 40−60%.
- Повторное использование тестов при дообучении модели.
- Раннее выявление деградации качества ИИ.

Ограничения:

- Сложность тестирования «чёрных ящиков».
- Необходимость актуализации тестовых данных.
- Высокие требования к инфраструктуре (GPU для инференса).

Требования к отчёту

1. Перечень использованных инструментов.
2. Таблица тестовых сценариев (столбцы: ID, описание, шаги, ожидаемый результат).
3. Скриншоты результатов выполнения тестов.

4. Анализ метрик (процент пройденных тестов, время, ошибки).
5. Выводы о применимости автоматизации для ИИ-приложений.

Контрольные вопросы

1. Какие риски возникают при тестировании ИИ-приложений?
2. Как проверить качество работы нейросетевой модели в мобильном приложении?
3. В чём отличие автоматизации тестирования ИИ-приложений от классических?
4. Какие метрики использовать для оценки визуального тестирования?
5. Как часто нужно обновлять автотесты для ИИ-системы?

Литература

1. «Automation in Testing» — Richard Bradshaw.
2. Документация Appium (appium.io).
3. «AI in Software Testing» — Jason Arbon.
4. Официальные гайды Espresso и XCUITest.

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Практическая работа №8. Развертывание мобильного приложения в магазинах мобильных приложений.

Цель работы: освоить процесс публикации мобильного приложения в основных магазинах приложений (App Store и Google Play), включая подготовку материалов, заполнение метаданных и прохождение модерации.

Задачи:

- Систематизировать подходы к изучению предмета.
- Научить правильному плану работы ИИ
- Показать основные приемы эффективного использования возможностей ИИ

Оборудование, технические средства и инструменты:

1. Методические рекомендации для практических занятий
2. Компьютер с подключением к сети Интернет

Ход практического занятия:

1. Для выполнения данного задания необходимо изучить теоретический материал по данной теме.
2. Руководствуясь методическими рекомендациями (инструкционная карта работы), студенты выполняют практическую работу.

3. Предъявить преподавателю результаты работы.

4. Оформить отчет по практической работе в соответствии с требованиями (содержание отчета см. ниже).
5. Подготовить ответы на контрольные вопросы.

Теоретические и учебно-методические материалы по теме практической работы

Теоретическая часть

Основные этапы развёртывания

1. **Подготовка приложения к публикации** (финальное тестирование, оптимизация, сборка релиза).
2. **Создание аккаунта разработчика** в целевых магазинах.
3. **Подготовка маркетинговых материалов** (иконка, скриншоты, видео, описание).
4. **Заполнение метаданных** (название, категории, ключевые слова).
5. **Настройка монетизации** (если применимо).
6. **Загрузка сборки** в консоль разработчика.
7. **Прохождение модерации**.
8. **Публикация и мониторинг**.

Требования магазинов

App Store (Apple):

- аккаунт разработчика ($99/год);
- сборка в формате .ipa;
- сертификат подписи;
- соответствие ;
- обязательное наличие Privacy Policy.

Google Play:

- аккаунт разработчика ($25 единоразово);
- сборка в формате .apk или .aab;
- цифровой сертификат;
- соответствие ;
- обязательная Privacy Policy.

Практическая часть

Шаг 1. Подготовка приложения

1. Проведите финальное **тестирование** на:
   - совместимость с целевыми ОС и устройствами;
   - производительность (время запуска, потребление ресурсов);
   - безопасность (защита данных, валидация ввода).
2. Оптимизируйте:
   - размер APK/IPA;
   - скорость загрузки экранов;
   - энергопотребление.
3. Соберите **релизную версию**:
   - для iOS: через Xcode (архив → Export as .ipa);
   - для Android: через Android Studio (Build → Generate Signed Bundle/APK).

Шаг 2. Создание аккаунтов разработчика

1. **App Store Connect**:
   - зарегистрируйтесь на ;
   - оплатите годовую подписку;
   - настройте Team Agent.
2. **Google Play Console**:

- зарегистрируйтесь на ;
- заполните налоговую информацию;
- примите соглашения.

Шаг 3. Подготовка маркетинговых материалов

1. **Иконка**:
   - iOS: 1024×1024 px, PNG, без прозрачности;
   - Android: 512×512 px, 32-bit PNG, с прозрачностью.
2. **Скриншоты**:
   - минимум 3–5 штук для каждой платформы;
   - разрешение: от 640×960 до 1242×2688 px;
   - покажите ключевые функции.
3. **Видео** (опционально):
   - длительность: до 30 сек;
   - формат: MP4, H.264;
   - демонстрация интерфейса без звука.
4. **Описание**:
   - заголовок: до 50 символов;
   - краткое описание: до 170 символов;
   - полное описание: до 4000 символов (iOS), до 8000 (Google Play).

Шаг 4. Заполнение метаданных

1. **Название приложения** – уникальное, с ключевыми словами.
2. **Категории** – выберите 1–2 релевантные.
3. **Ключевые слова** – список через запятую (iOS), в тексте описания (Google Play).
4. **Возрастной рейтинг** – заполните опросник в консоли.
5. **Контакты** – укажите email для поддержки.

Шаг 5. Настройка монетизации

1. **Бесплатная модель**:
   - без встроенных покупок;
   - возможна реклама.
2. **Платная модель**: всвязи
   - единоразовая покупка;
   - подписки (еженедельные/месячные/годовые).
3. **Гибридная модель**:
   - бесплатный базовый функционал + премиум-опции.

Шаг 6. Загрузка сборки

1. **App Store**:
   - в App Store Connect создайте новое приложение;
   - загрузите `.ipa`-файл через Xcode или Transporter;
   - укажите версию и номер сборки.
2. **Google Play**:
   - в Play Console создайте новый релиз;
   - загрузите `.aab` или `.apk`;
   - выберите трек (Production, Beta, Alpha).

Шаг 7. Модерация

1. **Сроки**:
   - App Store: 1–7 дней;
   - Google Play: 1–3 дня.
2. **Возможные причины отказа**:
   - нарушение правил магазина;
   - баги в приложении;

- некорректные метаданные.

3. **Действия при отказе**:
   - изучите комментарии модераторов;
   - исправьте ошибки;
   - отправьте новую версию.

Шаг 8. Публикация

1. После одобрения:
   - установите дату релиза (можно «сразу»);
   - проверьте страницу приложения;
   - протестируйте установку.
2. **Пост-релизные действия**:
   - отслеживайте отзывы;
   - анализируйте метрики (загрузки, удержание);
   - выпускайте обновления для исправления багов.

Контрольные вопросы

1. Какие документы обязательны для публикации в App Store и Google Play?
2. В чём разница между форматами .apk и .aab?
3. Какие ключевые элементы ASO (App Store Optimization) влияют на видимость приложения?
4. Каковы типичные сроки модерации в каждом магазине?
5. Какие метрики нужно отслеживать после публикации?

Отчёт по работе

Подготовьте документ, включающий:

1. Скриншоты консоли разработчика (создание приложения, загрузка сборки).
2. Примеры маркетинговых материалов (иконка, скриншоты).
3. Описание процесса модерации (сроки, комментарии).
4. Анализ первых 7 дней после публикации (загрузки, рейтинги).

Оформление результатов работы

Оформить отчёт о проделанной работе, который должен содержать исчерпывающие текстовые ответы на поставленные вопросы с решениями, пояснениями, результатами решения:

Содержание отчёта

1. Цель.

2. Раскрытие темы работы с сопровождением необходимыми материалами по построению и кодированию.

Ответить на контрольные вопросы. (устно)

Сформулировать выводы по результатам работы.

Сдать и защитить работу.

Критерии оценки

Оценка «отлично» ставится в том случае, если студент:

- свободно применяет полученные знания при выполнении практических заданий;

- выполнил работу в полном объеме с соблюдением необходимой последовательности действий;

- в письменном отчете по работе правильно и аккуратно выполнены все записи;

- при ответах на контрольные вопросы правильно понимает их сущность, дает точное определение и истолкование основных понятий, использует специальную терминологию дисциплины, не затрудняется при ответах на видоизмененные вопросы, сопровождает ответ примерами.

Оценка «хорошо» ставится, если:

- выполнены требования к оценке «отлично», но допущены 2 – 3 недочета при выполнении практических заданий и студент может их исправить самостоятельно или при небольшой помощи преподавателя;

- в письменном отчете по работе делает незначительные ошибки;

- при ответах на контрольные вопросы не допускает серьезных ошибок, легко устраняет отдельные неточности, но затрудняется в применении знаний в новой ситуации, приведении примеров.

Оценка «удовлетворительно» ставится, если:

- практическая работа выполнена не полностью, но объем выполненной части позволяет получить правильные результаты и выводы;

- в ходе выполнения работы студент продемонстрировал слабые практические навыки, были допущены ошибки;

- студент умеет применять полученные знания при решении простых задач по готовому алгоритму;

- в письменном отчете по работе допущены ошибки;

- при ответах на контрольные вопросы правильно понимает их сущность, но в ответе имеются отдельные пробелы и при самостоятельном воспроизведении материала требует дополнительных и уточняющих вопросов преподавателя.

Оценка «неудовлетворительно» ставится, если:

- практическая работа выполнена не полностью и объем выполненной работы не позволяет сделать правильных выводов, у студента имеются лишь отдельные представления об изученном материале, большая часть материала не усвоена;

- в письменном отчете по работе допущены грубые ошибки, либо он вообще отсутствует;

- на контрольные вопросы студент не может дать ответов, так как не овладел основными знаниями и умениями в соответствии с требованиями программы.