

# 1 ЗАДАЧИ МАШИННОГО ОБУЧЕНИЯ НА МОБИЛЬНЫХ УСТРОЙСТВАХ

Машинное обучение, как известно, очень тесно связано с нашей жизнью, и во многих областях используются разные модели машинного обучения. Рассмотрим задачи машинного обучения на мобильных устройствах:

- кастомизация;
- распознавание фото, текста и видео;
- распознавание звука;
- анализ данных с сенсоров;
- навигация.

## 1.1 КАСТОМИЗАЦИЯ

Подбор музыки, новостей, рекламы и т.д. – все это достигается с помощью машинного обучения. Приложение получает ваши персональные данные и показывает вам то, что понравилось людям с наиболее подходящими данными. Основная задача: проинформировать пользователя о товарах или услугах, которые будут для него наиболее интересными и актуальными.

Разнообразие таких систем можно проиллюстрировать основными характеристиками:

- предмет рекомендации;
- цель рекомендации;
- контекст рекомендации;
- источник рекомендации;
- степень персонализации;
- формат рекомендации;
- прозрачность рекомендации.

В центре таких систем лежит матрица предпочтений. В данной матрице одна из осей отвечает за пользователей, вторая за объекты рекомендации. Заполнена же эта матрица значениями по заданной шкале. Каждый пользователь обычно может оценить только небольшую часть объектов. Поэтому матрица очень разрежена. Задача системы – обобщение информации и предсказание отношения пользователя к объекту (заполнение пропущенных значений матрицы).

Пример приложений: Spotify, TikTok

## 1.2 РАСПОЗНАВАНИЕ ФОТО, ТЕКСТА И ВИДЕО

Распознавание фото и видео на мобильных телефонах мало чем отличается от обычных компьютерных методов, только цели немного другие. Например, некоторые мобильные телефоны распознают владельца через Face ID с помощью фронтальной камеры. Есть приложения для определения возраста, пола.

Распознавание текста на изображениях (оптическое распознавание символов, OCR) – одно из направлений распознавания образов, задача которого заключается в переводе изображений рукописного, машинного или печатного текста в текстовые данные, используемые для представления символов в компьютере.

Системы OCR применяются во многих областях. Вот некоторые из задач, которые решают системы распознавания текстов:

- считывание данных с бланков и анкет;
- автоматическое распознавание номерного знака;
- извлечение информации из визитных карточек в список контактов;
- создание цифровых версий печатных и рукописных документов;
- технология для помощи слепым и слабовидящим.

Существует несколько известных библиотек для работы с изображениями в мобильных приложениях: Tesseract, OpenCV, Mobile Vision Google, ML Kit.

Пример приложений: Google Lens, Pinterest

## 1.3 РАСПОЗНАВАНИЕ ЗВУКА

Распознавание звука и его парсинг тоже очень важная задача машинного обучения. Голосовые помощники, голосовой ввод, умные дома – все это нужно для нашей жизни.

Распознавание речи (англ. Speech Recognition) — процесс преобразования речевого сигнала в цифровую информацию. Задачей распознавания является сопоставление набору акустических признаков речевого сигнала или наблюдений последовательности слов, имеющих наибольшую вероятность правдоподобия среди всех кандидатов.

Для распознавания речи существует библиотека Pocketsphinx

Пример приложения: Google Assistant

## 1.4 АНАЛИЗ ДАННЫХ С СЕНСОРОВ

Телефон получает данные об окружающем мире с помощью специальных датчиков и сенсоров. В мире их существует огромное количество, и, к сожалению, большинство из них работает недостаточно точно. Но это исправляет машинное обучение, уточняя данные с сенсоров. Это важно для людей с опасными для жизни болезнями, например, может произойти сердечный приступ, и процессор лучше любого человека скажет, что это он, и, считав местоположение с навигатора, вызовет скорую помощь в тот же момент.

Пример приложений:

Приложение	Датчик
<b>Распознавание человеческой активности</b>	акселерометр; гироскоп; магнитометр; монитор сердечного ритма
<b>Распознавание жестов</b>	акселерометр; гироскоп
<b>Распознавание падений</b>	акселерометр; гироскоп; датчики близости
<b>Распознавание аварий</b>	акселерометр; приемник gps
<b>Распознавание условий окружающей среды</b>	датчик окружающего света; датчик влажности; термометр; датчик давления; приемник gps
<b>Распознавание стресса</b>	микрофон; электрокардиограф; монитор сердечного ритма; датчик проводимости кожи
<b>Распознавание эмоций</b>	микрофон; камера; электрокардиограф; монитор сердечного ритма; датчик проводимости кожи

## 1.5 НАВИГАЦИЯ.

Навигационные приложения можно значительно улучшить, если интегрировать в них алгоритмы по распознаванию фото и видео. К примеру, если приложение подключается к камере в автомобиле, оно может анализировать ситуацию на дороге и предупреждать водителя в случае возможной опасности. Так можно распознавать пробки, дорожные знаки по ограничению скорости, агрессивное поведение окружающих водителей и другие характеристики дорожного движения.

Пример приложения: карты Google

## **2 НЕЙРОННЫЕ СЕТИ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ**

Существуют различные типы нейронных сетей, которые обычно используются при разработке мобильных приложений. Рассмотрим некоторые из наиболее часто используемых.

### **2.1 КОНВОЛЮЦИОННЫЕ НЕЙРОННЫЕ СЕТИ (CNN)**

Конволюционные нейронные сети (CNN) обычно нужны для распознавания изображений, что позволяет им хорошо интегрироваться в приложениях, требующих анализа картинок. Архитектура CNN разработана для автоматического определения и извлечения соответствующих характеристик из изображений. Это дает возможность мобильным приложениям выполнять такие задачи, как распознавание лиц, обнаружение объектов и классификация изображений.

Например, приложение Snapchat использует CNN для применения фильтров к лицам пользователей в режиме реального времени. Нейронная сеть способна распознавать черты лица пользователя и применять фильтры, повторяющие движения и мимику. Аналогичным образом приложение Google Photos использует CNN для автоматической классификации фотографий пользователя по тематике, что облегчает поиск и упорядочивание изображений.

### **2.2 РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ (RNN)**

Рекуррентные нейронные сети (RNN) широко используются в обработке естественного языка, что делает их хорошо подходящими для мобильных приложений, требующих анализа текста. RNN предназначены для анализа последовательностей данных и выявления закономерностей во времени. Это позволяет мобильным приложениям выполнять такие задачи, как распознавание речи, генерация текста и перевод языка.

Например, приложение Google Translate использует RNN для анализа структуры текста на одном языке и перевода его на другой язык. Нейронная сеть способна распознавать закономерности в тексте и генерировать точный перевод, отражающий смысл оригинального текста.

### **2.3 СЕТИ С ДЛИННОЙ КРАТКОВРЕМЕННОЙ ПАМЯТЬЮ (LSTM)**

Сети с длинной кратковременной памятью (LSTM) — это тип RNN, предназначенный для анализа последовательностей данных с долгосрочными

зависимостями. Это делает их хорошо подходящими для мобильных приложений, требующих памяти и контекстной осведомленности, таких как чат-боты и голосовые помощники. LSTM-сети способны запоминать прошлые данные и использовать эту информацию для своих прогнозов.

Например, приложение Amazon Alexa использует LSTM-сеть для анализа запросов пользователей и предоставления соответствующих ответов. Нейронная сеть способна запоминать прошлые взаимодействия и использовать эту информацию для своих ответов.

## 2.4 ДРУГИЕ ТИПЫ НЕЙРОННЫХ СЕТЕЙ

Помимо CNN, RNN и LSTM, существуют и другие типы нейронных сетей, которые можно использовать при разработке мобильных приложений. Например, глубокие сети убеждений (DBN) используются в рекомендательных системах, а автоэнкодеры применяются для сжатия изображений и обесцвечивания данных. Выбор архитектуры нейронной сети зависит от конкретных требований мобильного приложения и типа данных, которые необходимо анализировать.

## 2.5 ИНСТРУМЕНТЫ ДЛЯ ВНЕДРЕНИЯ НЕЙРОНОВ В ANDROID

Для создания и внедрения нейронных сетей в Android приложение можно применять различные инструменты. Рассмотрим наиболее популярные.

**TensorFlow Lite.** Библиотека специально разработана для работы с мобильными устройствами. Обеспечивает высокую производительность и компактный размер моделей. Поддерживает интеграцию с NNAPI: помогает ускорить выполнение на аппаратном уровне.

**Keras и PyTorch.** Обе библиотеки обладают богатым набором инструментов для обучения нейронных сетей и поддерживают экспорт моделей в формат TensorFlow Lite. Если вы хотите создать сложную нейронную сеть с нестандартной архитектурой, можно использовать эти библиотеки.

**ML Kit.** Сервис предоставляет готовые решения для ряда задач: например, распознавание изображений и звука, анализ текста и так далее. Но при использовании ML Kit вы можете ограничены в возможностях настройки моделей.

**NNAPI (Neural Networks API)** – это API, разработанное компанией Google. Позволяет использовать аппаратное ускорение для выполнения вычислений нейронных сетей на устройствах с операционной системой Android.

## 2.6 БАЗОВЫЕ ПОНЯТИЯ

Для начала рассмотрим, что такое нейрон. Нейрон принимает входные данные, выполняет некоторые вычисления и передает данные дальше.

Нейрон состоит из трёх элементов:

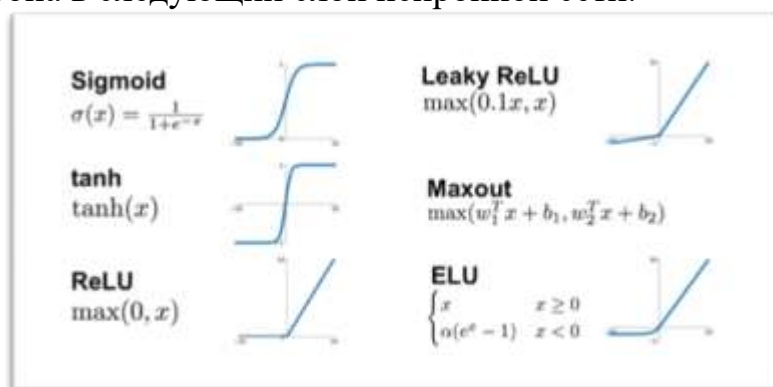
- входы,
- веса,
- функция активации.



Вес – это величина, которая показывает важность каждого входа. Оптимальные значения весов могут помочь нейронной сети достичь высокой точности в выполнении задач, для которых она была обучена. Изначально эти параметры случайны и в процессе обучения корректируются. В алгоритмах линейной регрессии обычно будет диапазон (0, 1), если функция активации – сигмоида.

Функция активации – это математическая функция, которая определяет, какой выходной сигнал будет передан из нейрона в следующий слой нейронной сети.

Функция активации применяется к выходу нейрона после того, как входные данные умножаются на соответствующие им веса. Она может добавлять нелинейность к выходу: это позволяет нейронной сети лучше обрабатывать сложные данные.



Сигнал в нейронной сети распространяется от входных нейронов к выходным нейронам через слои нейронов.

Входные данные передаются входным нейронам. Они передают сигнал следующему слою нейронов.



Каждый нейрон в слое принимает сигналы от предыдущего слоя, умножает их на соответствующие веса и передает результат следующему нейрону. Таким образом, каждый нейрон в слое обрабатывает информацию от предыдущего слоя и передаёт результат следующему.

Обучение нейросети происходит в два этапа:

- прямое распространение ошибки;
- обратное распространение ошибки.

Во время прямого распространения ошибки делается предсказание ответа. Оно сравнивается с реальным результатом, и если есть ошибка, то веса (числовые значения, которые определяют, как сильно входные сигналы влияют на выходной результат) в сети корректируются, чтобы уменьшить данную ошибку. Это происходит с использованием математического метода, который помогает найти оптимальные значения весов для минимизации ошибки.

### **Генетический алгоритм обучения**

Метод создан на принципах естественного отбора и генетической мутации. Он работает на основе того же принципа, что и эволюционные процессы в природе, которые основаны на объединении результатов. Проще говоря, происходит природный отбор, где новое поколение образуется путем комбинирования результатов с наилучшими свойствами. Если результат такого скрещивания не соответствует определенным критериям, то отбор происходит заново, пока продукт не достигнет совершенства.

## **3 ПРИМЕР ИСПОЛЬЗОВАНИЯ НЕЙРОСЕТИ В ПРИЛОЖЕНИИ**

Создадим приложение под Android, которое будет распознать, что изображено на картинке.

При выборе модели будем учитывать:

- размер модели: она должна быть достаточно маленькой, чтобы поместиться на устройстве и быстро загружаться.
- точность должна быть достаточно высокой для решения поставленной задачи.

Необходимо внимательно выбирать модель под конкретную задачу.



Выбираем модель [из TensorFlow Hub](#). Загружаем ее в assets и запрещаем в gradle сжатие. Если файл модели сжать, размер изменится и приложение может не смочь правильно загрузить модель. Это приведёт к ошибкам в работе модели и, в конечном итоге, к сбоям приложения.

```
androidResources {  
    noCompress 'tflite'  
}
```

Вставляем зависимости:

```
implementation 'org.tensorflow:tensorflow-lite-task-vision:0.4.0'  
implementation 'org.tensorflow:tensorflow-lite-gpu-delegate-plugin:0.4.0'  
implementation 'org.tensorflow:tensorflow-lite-gpu:2.9.0'
```

NNAPI уже входит в одну из них.

Создаём image classifier: он устанавливает порог оценки для результатов классификации изображения.

```
private fun setupImageClassifier() {  
    val threshold: Float = 0.5f  
    val maxResults: Int = 3  
    val numThreads: Int = 2  
    val optionsBuilder = ImageClassifier.ImageClassifierOptions.builder()  
        .setScoreThreshold(threshold)  
        .setMaxResults(maxResults)  
  
    val baseOptionsBuilder = BaseOptions.builder().setNumThreads(numThreads)  
    baseOptionsBuilder.useNnapi()  
    optionsBuilder.setBaseOptions(baseOptionsBuilder.build())  
  
    val modelName = "model.tflite"  
  
    try {  
        imageClassifier =  
            ImageClassifier.createFromFileAndOptions(context, modelName, optionsBuilder.build())  
    } catch (e: IllegalStateException) {  
        Log.e("TF", "TFLite failed to load model with error: " + e.message)  
    }  
}
```



Получился image-классификатор с уже обученной моделью. Можем выполнять код. На вход подаём bitmap — его необходимо получить из картинки. Картинку прогоняем через image-процессор и получаем выходной результат.

```
// Create preprocessor for the image.  
val imageProcessor = ImageProcessor.Builder()  
    .build()  
  
// Preprocess the image and convert it into a TensorImage for classification.  
val tensorImage = imageProcessor.process(TensorImage.fromBitmap(image))  
  
val imageProcessingOptions = ImageProcessingOptions.builder()  
    .setOrientation(getOrientationFromRotation(rotation))  
    .build()  
  
val results = imageClassifier?.classify(tensorImage, imageProcessingOptions)
```