Содержание

От издательства ... 13

Введение ... 14

Составители ... 16

Предисловие ... 18

Глава 1. Начало работы с TinyML ... 25

Технические требования ... 25
Представление TinyML ... 26
Что такое TinyML? ... 26
Почему ML на микроконтроллерах? ... 26
Зачем запускать ML локально? ... 27
Возможности и проблемы TinyML ... 28
Среды развертывания для TinyML ... 28
tinyML Foundation ... 30
Краткое описание глубокого обучения (DL) ... 30
Глубокие нейронные сети ... 31
Сверточные нейронные сети ... 32
Квантизация ... 35
Разница между мощностью и энергией ... 36
Различие между напряжением и током ... 36
Мощность и энергия ... 38
Программирование микроконтроллеров ... 39
Архитектура памяти ... 42
Периферийные устройства ... 43
Вход/выход общего назначения (GPIO или IO) ... 44
Аналого-цифровые преобразователи ... 45
Последовательная связь ... 45
Таймеры ... 45
Представление Arduino Nano 33 BLE Sense и Raspberry Pi Pico ... 45
Настройка Arduino Web Editor, TensorFlow и Edge Impulse ... 47
Подготовка веб-редактора Arduino Web Editor ... 47
Подготовка TensorFlow ... 47
Подготовка Edge Impulse ... 49
Как это делается ... 49
Запуск скетча на Arduino Nano 33 и Raspberry Pi Pico ... 51
Подготовка ... 51
Как это делается ... 52

Глава 2. Прототипирование на микроконтроллерах 53
Технические требования 53
Отладка кода 54
Подготовка 54
Как это делается 55
Дополнительно 58
Подключение светодиодного индикатора на макетной плате 58
Подготовка 58
Размещение прототипа на макетной плате 60
Как это делается 62
Управление внешним светодиодом с помощью GPIO 65
Подготовка 65
Представляем периферийное устройство GPIO 69
Как это делается 70
Включение и выключение светодиода с помощью кнопки 74
Подготовка 74
Как это делается 76
Использование прерываний для считывания состояния кнопки 79
Подготовка 79
Как это делается 80
Питание микроконтроллеров от батарей 82
Подготовка 82
Увеличение выходного напряжения при последовательном подключении батарей 83
Увеличение энергетической емкости за счет параллельного подключения батарей 83
Подключение батарей к плате микроконтроллера 84
Как это делается 85
Дополнительно 86

Глава 3. Создание метеостанции с помощью библиотеки TensorFlow Lite for microcontrollers 88
Импорт данных о погоде из WorldWeatherOnline 89
Подготовка 89
Как это делается 90
Подготовка набора данных 92
Подготовка 92
Подготовка сбалансированного набора данных 92
Масштабирование параметров с помощью Z-score 93
Как это делается 94
Обучение модели с помощью TF 98
Подготовка 98
Как это делается 99
Оценка эффективности модели 104
Подготовка 104
Наглядное представление эффективности с помощью матрицы ошибок 104

Оценка полноты (recall), точности (precision) и критерия F-score 106
Как это делается 106
Квантизация модели с помощью конвертера TFLite 108
Подготовка 109
Квантизация входной модели 109
Как это делается 112
Использование встроенного датчика температуры и влажности на Arduino Nano 114
Подготовка 114
Как это делается 115
Использование датчика DHT22 с Raspberry Pi Pico 116
Подготовка 116
Как это делается 117
Подготовка входных характеристик для просчета модели 119
Подготовка 119
Как это делается 120
Запуск на устройстве с помощью TFLu 122
Подготовка 122
Как это делается 123

Глава 4. Голосовое управление светодиодами с помощью Edge Impulse 127

Технические требования 128
Сбор аудиоданных с помощью смартфона 128
Подготовка 129
Сбор звуковых семплов для KWS 129
Как это делается 129
Извлечение параметров MFCC из аудиосемплов 134
Подготовка 134
Анализ звука в частотной области 134
Генерация Mel-спектрограммы 136
Извлечение MFCC 137
Как это делается 138
Дополнительно 140
Пример проектирования и обучения нейронной сети (NN) 142
Подготовка 142
Как это делается 142
Настройка эффективности модели с помощью EON Tuner 144
Подготовка 144
Как это делается 145
Классификация в реальном времени с помощью смартфона 147
Подготовка 147
Как это делается 147
Классификация в реальном времени с помощью Arduino Nano 149
Подготовка 149
Как это делается 149

Непрерывное распознавание на Arduino Nano 151
Подготовка 151
Изучение примера приложения KWS в реальном времени 151
Как это делается 153
Схема для голосового управления светодиодами на Raspberry Pi Pico 157
Подготовка 157
Представляем модуль электретного микрофона с усилителем MAX9814 158
Подключение микрофона к АЦП Raspberry Pi Pico 159
Как это делается 159
Выборка звука на Raspberry Pi Pico с помощью АЦП и прерываний по таймеру 164
Подготовка 164
Выборка звука на Raspberry PI Pico с помощью АЦП и прерываний по таймеру 164
Как это делается 165
Дополнительно 169

Глава 5. Распознавание интерьеров помещений с помощью TensorFlow Lite for Microcontrollers и Arduino Nano 171

Технические требования 172
Съемка с помощью модуля камеры OV7670 172
Подготовка 173
Как это делается 173
Захват кадров камеры через последовательный порт с помощью Python 176
Подготовка 177
Передача изображений RGB888 через последовательный порт 177
Изучаем, как преобразовать RGB565 в RGB888 179
Как это делается 179
Преобразование изображений QQVGA из YCbCr422 в RGB888 183
Подготовка 183
Преобразование YCbCr422 в RGB888 184
Как это делается 184
Создание набора данных для распознавания интерьеров помещений 186
Подготовка 186
Как это делается 187
Трансфертное обучение с помощью Keras API 189
Подготовка 189
Изучение вариантов дизайна сети MobileNet 190
Как это делается 191
Подготовка и тестирование квантизованной модели TFLite 194
Подготовка 195
Как это делается 195
Сокращение объема RAM за счет объединения функций обрезки, изменения размера, масштабирования и квантизации 197

Подготовка ... 198
Изменение размера с помощью билинейной интерполяции ... 199
Как это делается ... 200

Глава 6. Создание интерфейса на основе жестов для управления воспроизведением на YouTube ... 206

Технические требования ... 207
Подключение к MPU-6050 IMU через интерфейс I2C ... 207
Подготовка ... 208
Представляем MPU-6050 IMU ... 208
Связь с помощью I2C ... 209
Как это делается ... 211
Получение данных акселерометра ... 214
Подготовка ... 214
Как это делается ... 217
Построение набора данных с помощью инструмента пересылки данных Edge Impulse data forwarder ... 220
Подготовка ... 221
Как это делается ... 222
Разработка и обучение модели ML ... 225
Подготовка ... 225
Использование спектрального анализа для распознавания жестов ... 226
Как это делается ... 228
Классификации в реальном времени с помощью инструмента пересылки данных Edge Impulse data forwarder ... 231
Подготовка ... 231
Как это делается ... 231
Распознавание жестов на Raspberry Pi Pico в ОС Arm Mbed ... 232
Подготовка ... 232
Создание рабочих потоков с помощью RTOS API в Arm Mbed OS ... 233
Фильтрация избыточных и ложных прогнозов ... 234
Как это делается ... 235
Создание бесконтактного интерфейса с помощью PyAutoGUI ... 241
Подготовка ... 241
Как это делается ... 242

Глава 7. Запуск модели TinyML CIFAR-10 на виртуальной платформе ОС Zephyr ... 244

Технические требования ... 245
Начало работы с ОС Zephyr ... 245
Подготовка ... 245
Как это делается ... 246
Разработка и обучение малой модели CIFAR-10 ... 248
Подготовка ... 249
Замена свертки 2D на DWSC ... 249
Контроль поддержки требований модели к памяти ... 251

Как это делается.....252
Оценка достоверности модели TFLite.....255
Подготовка.....256
Как это делается.....256
Преобразование цифрового изображения в C-байтовый массив.....258
Подготовка.....258
Как это делается.....259
Подготовка основы проекта TFLu.....261
Подготовка.....261
Как это делается.....262
Создание и запуск приложения TFLu на QEMU.....263
Подготовка.....264
Как это делается.....264

Глава 8. К следующему поколению TinyML с microNPU.....269
Технические требования.....270
Настройка Arm Corstone-300 FVP.....270
Подготовка.....270
Как это делается.....272
Установка TVM с поддержкой Arm Ethos-U.....274
Подготовка.....275
Мотивация, лежащая в основе TVM.....275
Как TVM оптимизирует работу модели.....275
Как это делается.....277
Установка набора инструментов Arm и стека драйверов Ethos-U.....279
Подготовка.....280
Как это делается.....280
Генерация C-кода с помощью TVM.....282
Подготовка.....283
Запуск TVM на микроконтроллерах с помощью microTVM.....284
Как это делается.....284
Генерация C-байтовых массивов для входа, выхода и меток.....286
Подготовка.....286
Как это делается.....288
Создание и запуск модели на Arm Ethos-U55.....291
Подготовка.....291
Как это делается.....291

Предметный указатель.....295

Введение

Без сомнения, индустрия высоких технологий продолжает оказывать все большее влияние на нашу повседневную жизнь. Изменения столь же стремительны, сколь и постоянны, и происходят повсюду вокруг нас – в наших телефонах, автомобилях, интеллектуальных динамиках и микрогаджетах, которые мы используем для повышения эффективности, комфорта и возможностей коммуникации. Машинное обучение (*machine learning*, ML) – одна из самых преобразующих технологий нашего времени. Предприятия, ученые и инженерные сообщества продолжают углублять понимание, развивать и исследовать возможности этой невероятной технологии и служат все большему раскрытию ее потенциала для создания новых вариантов использования во многих отраслях.

Я менеджер по продуктам машинного обучения в компании ARM. В этой роли я нахожусь в центре революции ML, которая происходит в смартфонах, автомобильной промышленности, играх, AR, VR[1] и других областях. Мне ясно, что в ближайшем будущем функциональность ML будет в каждом отдельном электронном устройстве, – от крупнейших в мире суперкомпьютеров до самых маленьких и маломощных микроконтроллеров. Работа в области ML познакомила меня с некоторыми из самых блестящих и ярких умов в области технологий – теми, кто бросает вызов традиции, задает сложные вопросы и открывает новые ценности благодаря использованию ML.

Когда я впервые встретил Джана Марко, я едва мог произнести «ML», но в то время он уже был ветераном в этом космосе. Я был поражен широтой и глубиной его знаний и его способностью решать сложные проблемы. Вместе с командой ARM он работал над созданием Arm Compute Library (ACL) – самой эффективной библиотеки, доступной для ML на платформе ARM. Успех ACL не имеет себе равных. Он развернут на миллиардах устройств по всему миру – от серверов и флагманских смартфонов до интеллектуальных духовых шкафов.

Когда Джан Марко сказал мне, что пишет книгу о ML, моей немедленной реакцией было: «Какую часть?» Экосистема ML настолько разнообразна, что необходимо учитывать множество различных технологий, платформ и фреймворков[2]. Я знал, что благодаря обширным знаниям всех аспектов ML он был подходящим человеком для этой работы. Кроме того, Джан Марко обладает удивительной способностью объяснять вещи прямо и логично.

Книга Джана Марко раскрывает мир TinyML, проводя нас через ряд практических примеров из реального мира. Каждый пример изложен в форме рецепта, в четком и последовательном стиле, обеспечивающем простое по-

---

[1] AR (*augmented reality*) – дополненная реальность; VR (*virtual reality*) – виртуальная реальность. – *Здесь и далее прим. перев.*

[2] Фреймворк (*framework*) – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

шаговое руководство. Начиная с базовых принципов, он растолковывает основы электронных или программных технологий, которые будут использоваться в примере. Затем в книге рассказывается об используемых платформах и технологиях, за которыми следуют основы ML, – разработка моделей нейронных сетей, проведение обучения и разворачивания моделей на целевом устройстве. Это действительно стиль кулинарного руководства по приготовлению супа с орехами. Каждый пример немного сложнее предыдущего, и в них удачно сочетаются традиционные и новые технологии. Вы не просто узнаете «как», вы также получите понимание «почему». Когда дело доходит до периферийных устройств, эта книга действительно дает панорамный обзор области ML.

Машинное обучение продолжает менять все аспекты технологий, и разработчикам программного обеспечения необходимо начинать его изучение. Эта книга позволяет быстро адаптироваться благодаря использованию легкодоступных и недорогих аппаратных средств. Независимо от того, являетесь ли вы новичком в ML или имеете некоторый опыт, каждый пример будет содержать новые знания и оставлять достаточно возможностей для саморазвития и дальнейших экспериментов. Независимо от того, используете ли вы эту книгу в качестве учебника или справочника, вы создадите прочную основу в области ML для будущего развития. Это даст вашей команде возможность достичь повышения эффективности и производительности, получить новые идеи и новые возможности для ваших продуктов.

Ронан Нотон,
старший менеджер по продуктам машинного обучения в ARM

Предисловие

Эта книга о TinyML, быстрорастущей области на пересечении **машинного обучения (ML)** и встраиваемых систем, позволяющей искусственному интеллекту (ИИ) работать на устройствах на основе микроконтроллеров с чрезвычайно низким энергопотреблением.

TinyML – захватывающая область, полная возможностей. При небольшом бюджете мы можем создавать устройства, разумно взаимодействующие с окружающим миром и меняющие наш образ жизни к лучшему. Однако к этой области может быть трудно подступиться, если мы исходим из области ML и мало знакомы с микроконтроллерами. Цель этой книги – разрушить подобные барьеры и на практических примерах сделать TinyML доступным для разработчиков, не имеющих опыта программирования встраиваемых систем. Каждая глава будет представлять собой самостоятельный проект, позволяющий узнать, как использовать технологии, лежащие в основе TinyML, для взаимодействия с электронными компонентами (например, датчиками), и развертывать модели ML на устройствах с ограниченным объемом памяти.

«*Поваренная книга TinyML*» начинается с практического введения в эту междисциплинарную область, чтобы ознакомить вас с некоторыми основами развертывания интеллектуальных приложений на Arduino Nano 33 BLE Sense и Raspberry Pi Pico. По мере продвижения вы будете решать различные задачи, с которыми можете столкнуться при создании прототипов устройств на микроконтроллерах, вроде управления состоянием светодиода с помощью GPIO-вывода и кнопки или подачей питания на устройство с помощью батарей. После этого мы поговорим о примерах, касающихся измерителя температуры-влажности или датчиков **три-V** (**голос** (*voice*), **зрение** (*view*) и **вибрация** (*vibration*)), чтобы получить необходимые навыки для внедрения комплексных интеллектуальных приложений в различных сценариях. Затем вы изучите рекомендации по созданию уменьшенных моделей для микроконтроллеров с ограниченным объемом памяти. Наконец, вы познакомитесь с двумя самыми последними технологиями, microTVM и microNPU, которые помогут усовершенствовать ваши игры с TinyML.

К концу этой книги вы будете хорошо разбираться в лучших практиках и фреймворках ML, позволяющих разрабатывать ML-приложения на микроконтроллерах, и будете иметь четкое представление о ключевых аспектах, которые следует учитывать на этапе разработки.

Для кого предназначена эта книга

Эта книга предназначена для инженеров-разработчиков ML, заинтересованных в быстрой разработке приложений ML на микроконтроллерах с помощью практических примеров. Книга поможет вам расширить знания о революции TinyML, получить навыки создания комплексных интеллектуальных

проектов с использованием реальных датчиков данных на Arduino Nano 33 BLE Sense и Raspberry Pi Pico. Требуется базовое знакомство с C/C++, программированием на Python и **интерфейсом командной строки** (**CLI**). Однако никаких предварительных знаний о микроконтроллерах не требуется.

О ЧЕМ РАССКАЗЫВАЕТ ЭТА КНИГА

В главе 1 «*Начало работы с TinyML*» представлен обзор TinyML, а также возможности и проблемы, связанные с внедрением ML на микроконтроллерах с чрезвычайно низким энергопотреблением. В этой главе основное внимание уделяется фундаментальным элементам ML, энергопотреблению и микроконтроллерам, отличиям TinyML от обычного ML в облаке, на настольных компьютерах или даже смартфонах.

В главе 2 «*Прототипирование на микроконтроллерах*» представлены краткие и простые проекты, позволяющие разобраться с соответствующими основами программирования микроконтроллеров. Мы разберемся с отладкой кода и тем, как передавать данные на монитор последовательного порта Arduino. После этого мы узнаем, как программировать GPIO-периферию, управляющую выводами микроконтроллера, с помощью ARM Mbed API и использовать макетную плату для подключения внешних компонентов, таких как светодиоды и кнопки. В конце мы узнаем, как питать Arduino Nano 33 BLE Sense и Raspberry PI Pico от батареек.

Глава 3 «*Создание метеостанции с помощью библиотеки TensorFlow Lite for microcontrollers*» проведет вас через все этапы разработки приложения на основе библиотеки TensorFlow Lite for microcontrollers[4] и научит получать данные датчиков температуры и влажности. Приложение, которое будет разработано в этой главе, представляет собой метеостанцию на базе ML для прогнозирования снегопада.

Сначала мы сосредоточимся на подготовке набора данных путем получения исторических данных о погоде из WorldWeatherOnline. После этого мы представим соответствующие основы обучения и тестирования модели на основе библиотеки TensorFlow. В конце концов мы развернем модель на Arduino Nano 33 BLE Sense и Raspberry Pi Pico с библиотекой TensorFlow Lite for microcontrollers.

В главе 4 «*Голосовое управление светодиодами с помощью Edge Impulse*» показано, как разработать сквозное приложение для определения ключевых слов (**keyword spotting, KWS**) с помощью Edge Impulse и ознакомиться со сбором аудиоданных и аналого-цифровыми преобразователями (**АЦП**). Приложение, рассмотренное в этой главе, управляет цветом светодиода (красный, зеленый и синий) и количеством миганий (один, два и три).

---

[4] TensorFlow – библиотека машинного обучения, разработанная Google и ориентированная на обычные среды выполнения (серверы, облачные хранилища, ПК и смартфоны). В 2017 году была представлена специальная облегченная версия TensorFlow Lite для мобильных и малопотребляющих устройств с небольшим объемом памяти.

Сначала мы сосредоточимся на подготовке набора данных, показав, как получать аудиоданные с помощью мобильного телефона. После этого мы разработаем модель с использованием функций **MFCC**[5] и оптимизируем производительность с помощью EON Tuner. В конце концов мы доработаем приложение KWS на Arduino Nano 33 BLE Sense и Raspberry Pi Pico.

Глава 5 «*Распознавание интерьеров помещений с помощью TensorFlow Lite for microcontrollers и Arduino Nano*» призвана показать вам, как применять трансфертное обучение[6] с помощью TensorFlow и ознакомиться с лучшими практиками использования модуля камеры с микроконтроллером. Для целей этой главы мы разработаем приложение для распознавания интерьеров с помощью Arduino Nano 33 BLE Sense и модуля камеры OV7670.

В первой части мы увидим, как получать изображения с модуля камеры OV7670. После этого мы сосредоточимся на дизайне модели, применяя трансфертное обучение с помощью Keras[7] для распознавания кухонных и ванных комнат. В конце концов мы развернем квантизованную модель на Arduino Nano 33 BLE Sense с помощью TensorFlow Lite for microcontrollers.

Глава 6 «*Создание интерфейса на основе жестов для управления воспроизведением на YouTube*» направлена на разработку сквозного приложения для распознавания жестов с помощью Edge Impulse и Raspberry Pi Pico. Оно познакомит вас с инерциальными датчиками, научит использовать периферийные устройства I2C и создавать многопоточные приложения в Arm Mbed OS.

Сначала мы соберем данные акселерометра с помощью сборщика данных Edge Impulse data forwarder. После этого разработаем модель с использованием функций в частотной области для распознавания трех жестов. В конце концов мы развернем приложение на Raspberry Pi Pico и внедрим программу на Python с библиотекой PyAutoGUI для создания бесконтактного интерфейса для управления воспроизведением видео на YouTube.

В главе 7 «*Запуск модели TinyML CIFAR-10 на виртуальной платформе ОС Zephyr*» приведены рекомендации по созданию уменьшенных моделей для микроконтроллеров с ограниченным объемом памяти. В этой главе мы будем разрабатывать модель на основе набора данных для классификации изображений CIFAR-10 на виртуальном микроконтроллере на базе ARM Cortex-M3.

Сначала мы установим Zephyr, основной фреймворк, используемый в этой главе для выполнения нашей задачи. После этого разработаем малую квантизованную модель CIFAR-10 с библиотекой TensorFlow. Эта модель подойдет для микроконтроллера с объемом программной памяти всего 256 Кбайт и оперативной памятью 64 Кбайт. В конце концов мы создадим приложение

---

[5] MFCC (Mel-frequency cepstral coefficients) – сложный алгоритм анализа звуковых фрагментов с целью выделения характерных частот для распознавания речи. Подробнее об MFCC на русском языке см. https://habr.com/ru/post/140828/.

[6] Подробно о трансфертном обучении рассказывается в главе 5.

[7] Keras – высокоуровневый API, надстройка над библиотеками машинного обучения (в том числе Tensorflow), позволяющий реализовать их функциональность наиболее простым образом.

для классификации изображений с помощью TensorFlow Lite for microcontrollers и ОС Zephyr и запустим его на виртуальной платформе с помощью Quick Emulator (QEMU).

Глава 8 «*К следующему поколению TinyML с microNPU*» поможет вам ознакомиться с microNPU, новым классом процессоров для работы ML на периферийных устройствах. В этой главе мы будем запускать квантизованную модель CIFAR-10 на виртуальном контроллере ARM Ethos-U55 microNPU с помощью оптимизирующего компилятора TVM.

Сначала мы узнаем, как работает микропроцессор ARM Ethos-U55, и установим программные средства для сборки и запуска модели на фиксированной виртуальной платформе ARM Corstone-300. После этого мы будем использовать компилятор TVM для преобразования предварительно обученной модели TensorFlow Lite в код на языке Си. В конце мы покажем, как скомпилировать и развернуть код, сгенерированный TVM, в ARM Corstone-300 для выполнения вычислений с помощью ARM Ethos-U55 microNPU.

Как извлечь максимум пользы из этой книги

Вам понадобится компьютер (ноутбук или настольный компьютер) с архитектурой x86-64 и по крайней мере одним USB-портом для программирования плат Arduino Nano 33 BLE Sense и Raspberry Pi Pico. Для первых шести глав вы можете использовать Ubuntu 18.04 (или более позднюю версию) или Windows (например, Windows 10) в качестве операционной системы. Однако вам понадобится Ubuntu 18.04 (или более поздняя версия) для главы 7, посвященной запуску малой модели CIFAR-10 на виртуальной платформе с ОС Zephyr, и главы 8, посвященной следующему поколению TinyML с microNPU.

Необходимо иметь на вашем компьютере следующие программы:

- Python (рекомендуется Python 3.7),
- текстовый редактор (например, gedit в Ubuntu),
- медиаплеер (например, VLC),
- средство просмотра изображений (например, приложение по умолчанию в Ubuntu или Windows 10),
- веб-браузер (например, Google Chrome).

Для путешествия по TinyML нам понадобятся различные программные средства для разработки ML и программирования встроенных систем. Благодаря Arduino, Edge Impulse и Google эти инструменты будут находиться в облаке с доступом на основе браузера и с бесплатным планом использования.

Программы Arduino Nano 33 BLE Sense и Raspberry Pi Pico будут разрабатываться непосредственно в веб-браузере с помощью веб-редактора Arduino (https://create.arduino.cc). Однако бесплатный веб-редактор Arduino имеет ограничение в 200 с времени компиляции в день. Поэтому вы можете рассмотреть возможность перехода на любой платный тарифный план или на использование бесплатного локального редактора Arduino IDE (https://www.arduino.cc/en/software), чтобы получить неограниченное время компиляции. Если вас интересует бесплатная локальная среда разработки Arduino IDE, мы

предоставили на GitHub (https://github.com/PacktPublishing/TinyML-Cookbook/blob/main/Docs/setup_local_arduino_ide.md) инструкции по ее настройке.

В следующей таблице суммированы сведения об аппаратных устройствах и программных средствах, рассмотренных в каждой главе.

Глава	Аппаратные устройства	Программные средства
1	Arduino Nano 33 BLE Sense Raspberry Pi Pico	Arduino Web Editor
2	Arduino Nano 33 BLE Sense Raspberry Pi Pico	Arduino Web Editor, Google Colaboratory
3	Arduino Nano 33 BLE Sense Raspberry Pi Pico	Arduino Web Editor, Google Colaboratory
4	Arduino Nano 33 BLE Sense Raspberry Pi Pico	Arduino Web Editor, Edge Impulse Python 3.6 (local)
5	Arduino Nano 33 BLE Sense	Arduino Web Editor, Google Colaboratory Python 3.6 (local)
6	Raspberry Pi Pico	Arduino Web Editor, Edge Impulse Python 3.6 (local)
7	Виртуальная платформа	Google Colaboratory, Python 3.6 (local) Zephyr SDK
8	Виртуальная платформа	ARM Corstone-300, Python 3.6 (local) TVM/microTVM

Для проектов могут потребоваться датчики и дополнительные электронные компоненты для создания реалистичных прототипов TinyML и полного процесса разработки. Все компоненты перечислены в начале каждой главы и в файле *README.md* на GitHub (https://github.com/PacktPublishing/TinyML-Cookbook). Поскольку вы будете создавать настоящие электронные схемы, нам потребуется комплект электронных компонентов, который включает в себя по крайней мере беспаечную макетную плату, разноцветные светодиоды, резисторы, кнопки и соединительные провода-перемычки. Не волнуйтесь, если вы новичок в электронике, – вы познакомитесь с этими компонентами в первых двух главах этой книги. Кроме того, мы подготовили список покупок для начинающих на GitHub, чтобы вы точно знали, что купить: https://github.com/PacktPublishing/TinyML-Cookbook/blob/main/Docs/shopping_list.md[8].

Если вы используете цифровую версию этой книги, мы советуем вам вводить код самостоятельно или получить доступ к коду через репозиторий GitHub (ссылка доступна ниже). Это поможет вам избежать любых потенциальных ошибок, связанных с копированием и вставкой кода.

[8] Указанный по ссылке список ориентирован на реалии США и Европы. Рекомендации по приобретению в российских условиях оборудования или компонентов будут размещаться в разделах «*Технические требования*» соответствующих глав. Отсутствие такой рекомендации означает, что компонент широко доступен и может быть приобретен без длительного поиска в основных отечественных интернет-магазинах электроники (http://www.chipdip.ru, http://iarduino.ru, https://www.electronshik.ru, https://dip8.ru и др.), а также на https://aliexpress.ru.

Загрузка файлов с примерами кода

Вы можете загрузить файлы с примерами кода для этой книги с GitHub по адресу https://github.com/PacktPublishing/TinyML-Cookbook. В случае обновления кода он также будет обновлен в существующем репозитории GitHub.

У нас также есть другие пакеты кода из нашего богатого каталога книг и видео, доступных по адресу https://github.com/PacktPublishing. Ознакомьтесь с ними!

Загрузка цветных изображений

Мы предоставляем PDF-файл, содержащий цветные изображения скриншотов и графиков, используемых в этой книге. Вы можете скачать его здесь: https://static.packt-cdn.com/downloads/9781801814973_ColorImages.pdf.

Текстовые соглашения

В этой книге используется ряд текстовых соглашений.

Гиперссылки, домены и интернет-адреса: выделяются жирным курсивом, если они не представляют собой законченный интернет-адрес с указанием протокола (URL). В последнем случае ссылка выделена синим шрифтом: https://github.com.

Имя файла: курсивом указаны имена папок, имена файлов, расширения файлов, пути.

Пример: «Войдите в папку *~/project_npu* и создайте три папки с именами *binaries*, *src* и *sw_libs*».

Также *курсивом* указываются названия глав и разделов и указываемые в скобках эквиваленты переводных терминов (напр. «узел (*node*)»).

`Код в тексте`: моноширинный шрифт указывает кодовые обозначения в тексте, команды, имена таблиц базы данных, фиктивные URL-адреса, вводимые пользователем, и дескрипторы Twitter.

`Блок кода` задается следующим образом:

```
export PATH=~/project_npu/binaries/FVP_Corstone_SSE-300/models/Linux64_GCC-6.4:$PATH
```

Когда мы хотим привлечь внимание к определенной части блока кода, соответствующие строки или элементы выделяются **`жирным`**:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Любой ввод или вывод из командной строки записывается следующим образом:

```
$ cd ~/project_npu
$ mkdir binaries
$ mkdir src
```

Жирный шрифт: также обозначает новый термин, важное слово или названия, которые вы видите на экране.

Например, названия в меню или диалоговых окнах отображаются в тексте следующим образом: «Нажмите на **Corstone-300 Ecosystem FVPs**, а затем нажмите на кнопку **Download Linux**». Русский перевод пунктов меню и названий кнопок, если это необходимо для лучшего понимания текста, указывается курсивом в скобках. Например, указанная фраза может заканчиваться следующим образом: «нажмите на кнопку **Download Linux** (*Загрузить Linux*)».

Важные примечания выглядят так.

Примечания выглядят вот так.

Подсказки выглядят вот так.

Часто используемые разделы

В этой книге вы найдете несколько часто появляющихся заголовков. Подробности инструкций по выполнению примеров помещены в следующие разделы.

Подготовка

В этом разделе рассказывается, чего ожидать от примера, и описывается, как настроить программное обеспечение или выполнить предварительные настройки, необходимые для его выполнения.

Как это делается...

В этом разделе приведены шаги, необходимые для выполнения примера.

Дополнительно

Этот раздел содержит дополнительную информацию о примере, чтобы вы лучше ориентировались.

Глава 1

Начало работы с TinyML

Мы делаем первый шаг в мир TinyML.

Глава начинается с обзора этой развивающейся области, в которой рассматриваются возможности и проблемы, связанные с внедрением машинного обучения (ML) в микроконтроллеры с чрезвычайно низким энергопотреблением.

Основная часть этой главы посвящена фундаментальным элементам ML, энергопотреблению и микроконтроллерам, уникальным особенностям TinyML и его отличиям от обычного ML в облаке, настольных компьютерах или даже смартфонах. В частности, раздел «*Программирование микроконтроллеров*» будет иметь существенное значение для тех, у кого мало опыта в программировании встроенных систем.

После представления основных строительных блоков TinyML мы настроим среду разработки для создания простого приложения управления светодиодами, что официально ознаменует начало нашего практического путешествия по TinyML.

В отличие от того, что мы найдем в следующих главах, эта глава имеет более теоретическую структуру, чтобы познакомить вас с концепциями и терминологией быстрорастущей технологии TinyML.

В этой главе мы рассмотрим следующие темы.

- Представление TinyML.
- Краткое описание глубокого обучения (DL).
- Разница между мощностью и энергией.
- Программирование микроконтроллеров.
- Представление Arduino Nano 33 BLE Sense и Raspberry Pi Pico.
- Настройка Arduino Web Editor, TensorFlow и Edge Impulse.
- Запуск скетча на Arduino Nano 33 и Raspberry Pi Pico.

Технические требования

Чтобы выполнить практический пример в этой главе, нам понадобится следующее:

- плата Arduino Nano 33 BLE Sense,
- плата Raspberry Pi Pico,
- кабель micro-USB,
- ноутбук или ПК с Ubuntu 18.04 или Windows 10 на x86-64.

Представление TinyML

Во всех проектах, представленных в этой книге, мы дадим практические решения для **малого машинного обучения**, или, как мы будем его называть, **TinyML**. В этом разделе мы узнаем, что такое TinyML и какие огромные возможности оно открывает.

Что такое TinyML?

TinyML – это набор технологий машинного обучения (ML) для использования интеллектуальных приложений на встраиваемых устройствах с чрезвычайно низким энергопотреблением. Как правило, эти устройства имеют ограниченную память и вычислительные возможности, но они могут воспринимать физические параметры среды с помощью датчиков и действовать на основе решений, принимаемых алгоритмами ML.

Для TinyML собственно ML и платформа, на которой оно разворачивается, – это не просто две независимые сущности, а скорее сущности, которые должны в лучшем случае соответствовать друг другу. На практике разработка архитектуры ML без учета характеристик целевого устройства усложняет развертывание эффективно работающих приложений TinyML.

С другой стороны, было бы невозможно спроектировать энергоэффективные процессоры для расширения возможностей ML таких устройств, не зная задействованных программных алгоритмов.

В этой книге в качестве целевого устройства для TinyML будут рассмотрены микроконтроллеры, а следующий подраздел поможет мотивировать наш выбор.

Почему ML на микроконтроллерах?

Первой и главной причиной выбора микроконтроллеров является их популярность в различных областях, таких как автомобилестроение, бытовая электроника, кухонная техника, здравоохранение и телекоммуникации. В настоящее время микроконтроллеры незаметно присутствуют во всех наших повседневных электронных устройствах.

С появлением интернета вещей (IoT) рынок микроконтроллеров стремительно рос. В 2018 году компания по исследованию рынка IDC (https://www.idc.com) сообщила о 28.1 млрд микроконтроллеров, проданных по всему миру, и прогнозируется рост до 38.2 млрд к 2023 году (www.arm.com/blogs/blueprint/tinyML). Это впечатляющие цифры, учитывая, что на рынках смартфонов и ПК в том же году было продано 1.5 млрд и 67.2 млн устройств соответственно.

Таким образом, TinyML представляет собой значительный шаг вперед для устройств IoT, способствуя распространению малых устройств, способных выполнять задачи ML локально.

Вторая причина выбора микроконтроллеров заключается в том, что они недороги, просты в программировании и достаточно мощны для запуска сложных алгоритмов глубокого обучения (DL).

Однако почему мы не можем перенести вычисления в облако, если мощные серверы гораздо более производительны? Другими словами, зачем нам нужно запускать ML локально?

Зачем запускать ML локально?

Есть три основных ответа на этот вопрос: «задержки», «энергопотребление» и «конфиденциальность».

- *Сокращение задержек*: отправка данных в облако и обратно не происходит мгновенно и может повлиять на приложения, которые должны надежно реагировать в кратчайшие сроки.
- *Снижение энергопотребления*: отправка и получение данных в облако и из него не являются энергоэффективными даже при использовании маломощных протоколов связи, таких как Bluetooth Low Energy (BLE). На следующей блок-схеме мы приводим распределение энергопотребления для встроенных компонентов платы Arduino Nano 33 BLE Sense, одной из двух плат микроконтроллеров, используемых в этой книге:

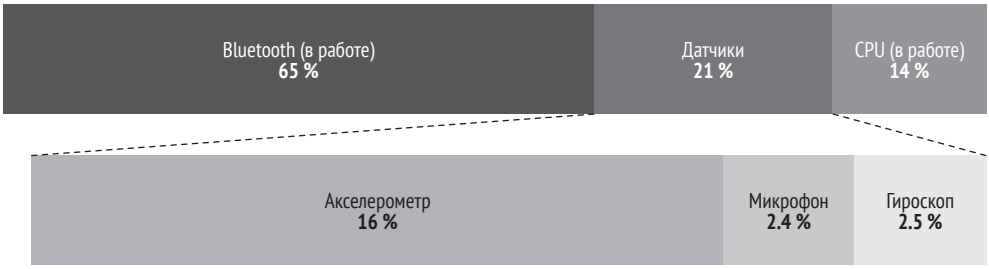

Рис. 1.1 ❖ Распределение энергопотребления платы Arduino Nano 33 BLE Sense

  Как мы можем видеть из распределения энергопотребления, вычисления на процессоре более энергоэффективны, чем связь по Bluetooth (14 % против 65 %), поэтому предпочтительнее вычислять больше и передавать меньше, чтобы снизить риск быстрого разряда батареи. Как правило, радио – это компонент, который потребляет больше всего энергии в типичных встраиваемых устройствах.
- *Конфиденциальность*: локальное управление означает сохранение конфиденциальности пользователей и отказ от обмена конфиденциальной информацией.

Теперь, когда мы знаем о преимуществах использования ML на этих малых устройствах, каковы практические возможности и проблемы, связанные с доведением ML до предела его возможностей?

Возможности и проблемы TinyML

TinyML находит свое естественное применение там, где невозможно или сложно получить питание от сети, и приложение должно работать от батареи как можно дольше.

Если вдуматься, то мы уже окружены устройствами с батарейным питанием, которые содержат ML под капотом. Например, носимые устройства, такие как умные часы и браслеты для отслеживания фитнеса, могут распознавать действия человека, чтобы отслеживать заданные цели в области здравоохранения или обнаруживать опасные ситуации вроде падения на землю.

Эти повседневные вещи являются полноценными приложениями TinyML, потому что они питаются от батарей и используют встроенную память устройств для придания смысла данным, получаемым датчиками.

Однако решения с батарейным питанием не ограничиваются только носимыми устройствами. Существуют сценарии, в которых нам могут понадобиться устройства для мониторинга окружающей среды. Например, мы можем рассмотреть возможность развертывания устройств с батарейным питанием под управлением TinyML в лесу для обнаружения пожаров и предотвращения распространения пожаров на большую площадь.

Существует неограниченное количество потенциальных вариантов использования TinyML, и те, которые мы только что кратко представили, – это лишь некоторые из вероятных областей применения.

Однако наряду с открывающимися возможностями предстоит столкнуться и с некоторыми серьезными проблемами. Проблемы возникают с вычислительной точки зрения, поскольку наши устройства ограничены в памяти и вычислительной мощности. Мы работаем на системах с несколькими килобайтами оперативной памяти и в некоторых случаях на процессорах без аппаратного ускорения операций с плавающей запятой.

С другой стороны, среда развертывания может быть недружелюбной. Факторы окружающей среды, такие как пыль и экстремальные погодные условия, могут помешать правильному выполнению наших приложений.

В следующем подразделе мы представим типичные среды развертывания для TinyML.

Среды развертывания для TinyML

Приложение TinyML может работать как в **централизованных**, так и в **распределенных** системах.

В **централизованной** системе приложению не обязательно требуется связь с другими устройствами.

Типичным примером является **распознавание ключевых слов**. В настоящее время мы легко взаимодействуем со смартфонами, камерами, беспилотными летательными аппаратами и кухонной техникой с помощью голосовых команд. Волшебные слова «*OK*», «*Google*», «*Alexa*» и т. д., которые мы используем для пробуждения умных помощников, являются классическим примером ML-модели, постоянно работающей локально в фоновом режиме.

Приложению требуется работать в системе с низким энергопотреблением без отправки данных в облако, чтобы быть эффективным, мгновенно реагировать и минимизировать энергопотребление.

Обычно централизованные приложения TinyML нацелены на запуск более энергоемких функций и извлекают выгоду из того, что они являются автономными по своей природе, поскольку им не нужно отправлять какие-либо данные в облако.

В **распределенной** системе устройство, т. е. **узел** (*node*) или **узел датчика** (*sensor node*), по-прежнему выполняет ML локально, но также взаимодействует с близлежащими устройствами или хостом для достижения общей цели, как показано на рис. 1.2.

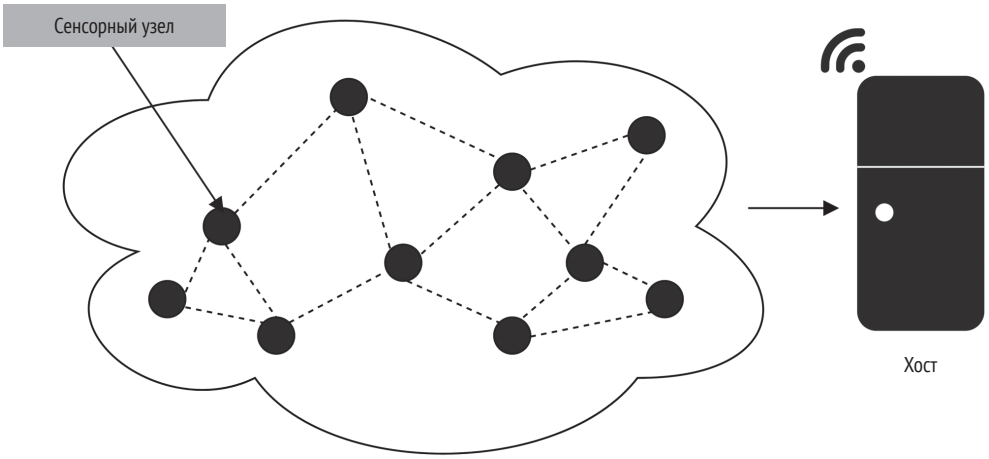

Рис. 1.2 ❖ Беспроводная сенсорная сеть

Поскольку узлы (*node*) являются частью сети и обычно обмениваются данными с помощью беспроводных технологий, мы обычно называем сеть **беспроводной сенсорной сетью** (**wireless sensor network**, **WSN**).

Хотя этот сценарий может показаться невыгодным из-за энергопотребления при передаче данных, устройствам, возможно, потребуется взаимодействовать, чтобы получить значимые и точные данные о рабочей среде. Знание температуры, влажности, влажности почвы или других физических величин для конкретного узла может оказаться неактуальным для некоторых приложений, которым вместо этого потребуются сведения о глобальном распространении этих величин.

WSN может помочь определить, какие участки поля требуют меньше или больше воды, чем другие, и сделать орошение более эффективным. Как мы можем представить, эффективные протоколы связи будут важны для срока автономной службы сети, а TinyML играет определенную роль в достижении этой цели. Поскольку отправка необработанных данных потребляет слишком много энергии, ML может выполнять частичную обработку, чтобы уменьшить объем передаваемых данных и частоту обмена.

TinyML предлагает бесконечные возможности, и **tinyML Foundation** – лучший источник сведений о возможностях, предоставляемых этой быстрорастущей областью ML и встраиваемых систем.

tinyML Foundation

tinyML Foundation (http://www.tinyml.org) – некоммерческая профессиональная организация, поддерживающая и объединяющая мир TinyML.

Для этого tinyML Foundation при поддержке нескольких компаний, включая Arm, Edge Impulse, Google и Qualcomm, выращивает разнообразное сообщество по всему миру: в США, Великобритании, Германии, Италии, Нигерии, Индии, Японии, Австралии, Чили и Сингапуре. В сообщество входят специалисты по аппаратному и программному обеспечению, системные инженеры, ученые, дизайнеры, менеджеры по продуктам и бизнесмены.

Фонд продвигает различные бесплатные инициативы онлайн и офлайн для привлечения экспертов и новичков, поощрения обмена знаниями, налаживания связей и создания более здорового и устойчивого мира с помощью TinyML.

С помощью групп Meetup (https://www.meetup.com) в разных странах вы можете бесплатно присоединиться к сообществу TinyML рядом с вами (https://www.meetup.com/en-AU/pro/TinyML), чтобы всегда быть в курсе новинок TinyML и предстоящих событий.

После представления TinyML давайте более подробно изучим его составляющие. В следующем разделе будет рассказано о том, что делает наши устройства способными принимать интеллектуальные решения: технологии глубокого обучения (*deep learning*, DL).

Краткое описание глубокого обучения (DL)

Машинное обучение ML – это тот компонент, который делает наши малые устройства способными принимать разумные решения. Программные алгоритмы ML в значительной степени полагаются на правильные данные для изучения шаблонов или действий, основанных на опыте. Как мы обычно говорим, данные – это все для ML, это то, что создает или рушит приложение.

В этой книге DL[9] будет рассматриваться как особый класс ML, который может выполнять сложные задачи классификации[10] непосредственно на необ-

---

[9] Концепция глубокого обучения (Deep Learning, DL) – одно из направлений машинного обучения. Как отдельная дисциплина DL оформилась в 2006 году, прежде всего в связи с задачами распознавания образов. Подробнее о Deep Learning на русском языке см. https://habr.com/ru/company/otus/blog/459785/.

[10] Классификацией (*classification*) в контексте машинного обучения называется самая распространенная разновидность задачи распознавания (*recognition*). Чаще всего эти термины означают одно и тоже, но первый термин более строгий. Поэтому в большинстве случаев в переводе сохранена «классификация» иногда с заменой на «распознавание» как более уместное по смыслу текста. Объекты классификации называются классами (*classes*).

работанных изображениях, тексте или звуке. Алгоритмы DL обладают самой высокой точностью, а также в некоторых задачах классификации могут быть лучше, чем люди. Именно эта технология сделала возможным беспилотное вождение, существование виртуальных помощников с голосовым управлением и подняла на невиданную высоту системы распознавания лиц.

Полное обсуждение архитектур и алгоритмов DL выходит за рамки этой книги. Тем не менее в этом разделе будут обобщены некоторые из его основных моментов, которые имеют отношение к пониманию материала следующих глав.

Глубокие нейронные сети

Глубокая нейронная сеть (**deep neural network**, **DNN**) состоит из нескольких вложенных слоев, направленных на изучение паттернов[11]. Каждый слой содержит несколько нейронов, фундаментальных вычислительных элементов для искусственных нейронных сетей (**artificial neural networks, ANN**), сделанных по образцу клеток человеческого мозга.

Нейрон выдает один выходной сигнал посредством линейного преобразования, определяемого как взвешенная сумма входных данных плюс постоянное значение, называемое **смещением** (**bias**), как показано на рис. 1.3.

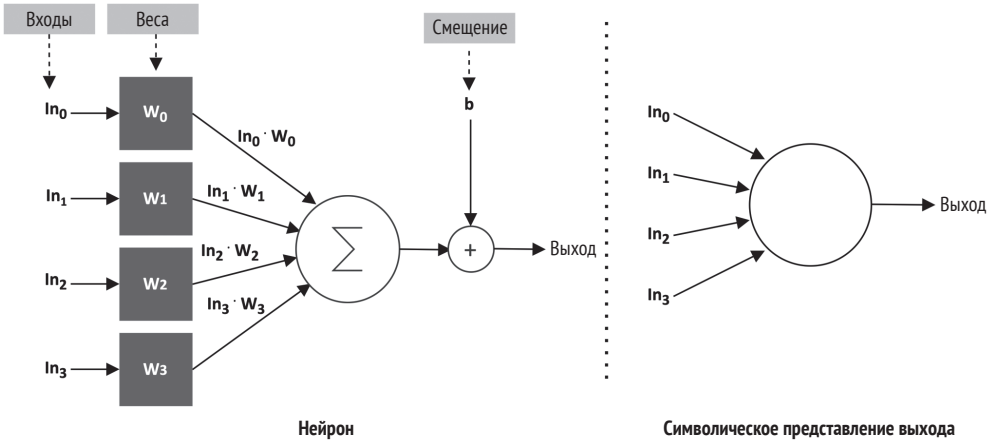

Рис. 1.3 ❖ Представление нейрона

Коэффициенты (W) взвешенной суммы называются **весами** (**weights**).

Веса и смещение получаются после итеративного процесса обучения, чтобы сделать нейрон способным к изучению сложных паттернов.

---

[11] Паттерн (*pattern*) – «шаблон», в машинном обучении термин обозначает группу многомерных объектов в пространстве признаков, указанных «учителем» как имеющие общие черты. Паттерн следует отличать от кластера (*cluster*) – группы объектов, общие черты которых выявляются в пространстве признаков автоматически, без участия «учителя».

Однако нейроны могут решать только простые линейные задачи с помощью линейных преобразований. Поэтому, чтобы помочь сети изучать сложные паттерны, к выходу нейрона обычно подключаются нелинейные функции, называемые **активациями** (**activations**). Активация – это нелинейная функция, выполняющаяся на выходе нейрона:

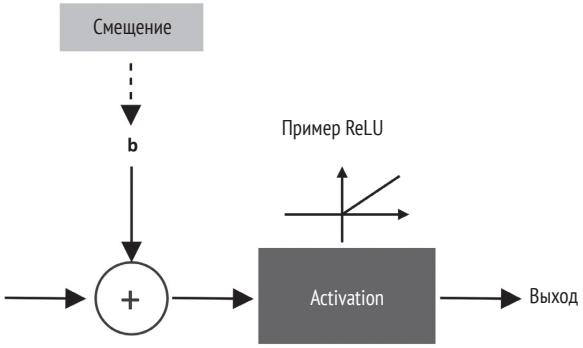

Рис. 1.4 ❖ Функция активации

Широко распространенной функцией активации является спрямленный линейный модуль (**rectified linear unit, ReLU**), описываемый следующим фрагментом кода:

```
float relu(float input) {
  return max(input, 0);
}
```

Его вычислительная простота делает его предпочтительным по сравнению с другими нелинейными функциями, такими как гиперболическая касательная или логистическая сигмоида[12], которые требуют больше вычислительных ресурсов.

В следующем подразделе мы увидим, как нейроны связываются для решения сложных задач визуального распознавания.

Сверточные нейронные сети

Сверточные нейронные сети (**CNNS**) – это специализированные глубокие нейронные сети, преимущественно применяемые для задач визуального распознавания.

Мы можем рассматривать CNNs как эволюцию упорядоченной версии классических полносвязных нейронных сетей с уплотненными (т. е. полностью связанными) слоями.

---

[12] Сигмоида – название класса кривых, общем признаком которых является S-образная симметричная (по обеим осям) форма с асимптотическим приближением к нулю или единице на краях. Подробнее см., например, в статье «Википедии» «Сигмоида».

Характеристикой **полносвязных сетей** (*fully connected*) является подключение каждого нейрона ко всем выходным нейронам предыдущего уровня, как мы можем видеть на следующем рисунке:

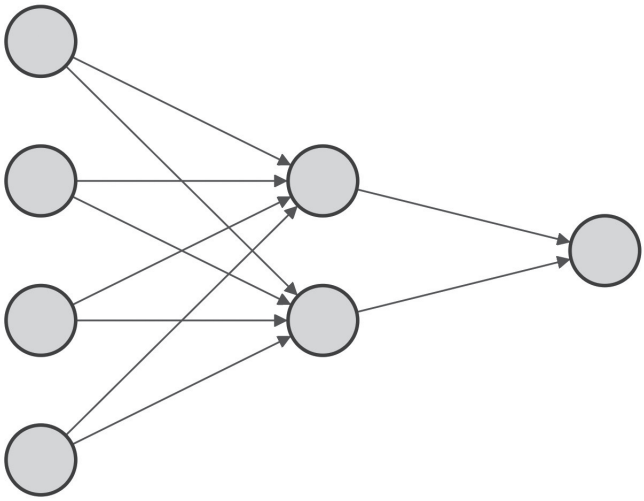

Рис. 1.5 ❖ Полносвязная сеть

К сожалению, этот подход плохо работает для обучения модели классификации изображений.

Например, если бы мы рассматривали изображение в формате RGB размером 320×240 (ширина × высота), нам понадобилось бы 230 400 (320×240×3) весов только для одного нейрона. Поскольку нашим слоям, несомненно, потребуется несколько нейронов для распознавания сложных проблем, модель, скорее всего, будет перегружена, учитывая неуправляемое количество обучаемых параметров.

В прошлом специалисты по обработке данных использовали особенности инженерных технологий для извлечения из изображений уменьшенного набора полезных признаков. Однако этот подход страдал от того, что было сложно выполнять отбор признаков, что отнимало много времени и зависело от конкретной предметной области.

С появлением CNN[13] задачи визуального распознавания улучшились благодаря слоям свертки (*convolution layers*), что делает извлечение признаков частью задачи обучения.

CNN возник в результате изучения биологических процессов в зрительной коре животных. В предположении, что мы имеем дело с изображениями, слой свертки заимствует широко распространенный оператор свертки из обработки изображений для создания набора признаков, поддающегося обучению.

---

[13] CNN (*convolutional neural network*, иначе ConvNet) – сверточная нейронная сеть, специальная архитектура искусственных нейронных сетей, направленная на эффективное распознавание образов. Входит в состав технологий глубокого обучения (DL). Подробнее на русском см. https://habr.com/ru/company/skillfactory/blog/565232/.

Оператор свертки (*convolution*) выполняется аналогично другим процедурам обработки изображений: перемещение окна оператора (фильтра, ядра) по всему входному изображению и применение скалярного произведения между его весами и соседними пикселями, как показано на следующем рисунке:

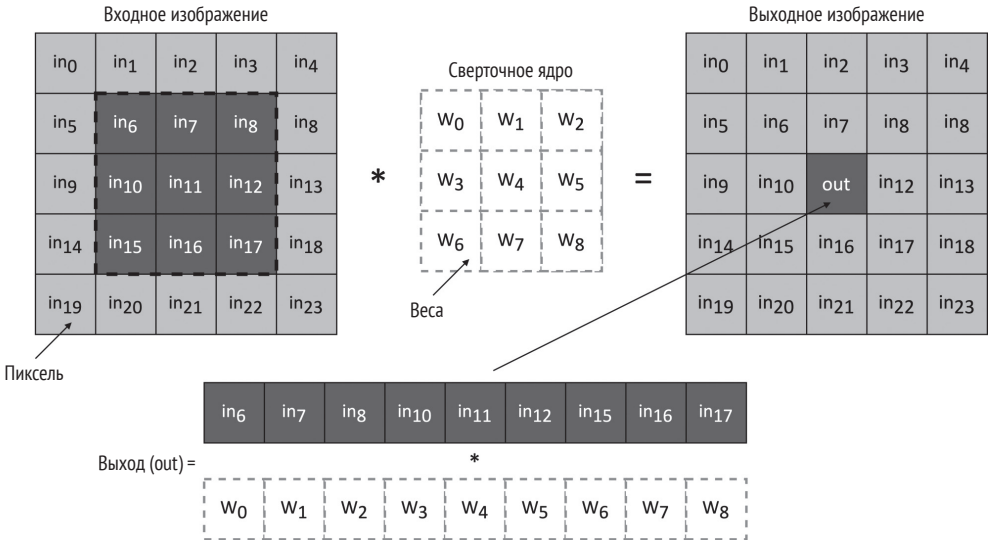

Рис. 1.6 ❖ Оператор свертки

Такой подход дает два существенных преимущества:

- он автоматически извлекает соответствующие функции без вмешательства человека;
- значительно уменьшает количество входных сигналов на один нейрон.

Например, для применения фильтра 3×3 к показанному RGB-изображению потребуется всего 27 весов (3×3×3).

Как и для полносвязных слоев, слоям свертки требуется несколько сверточных ядер, чтобы изучить как можно больше параметров. Следовательно, выходные данные слоя свертки создают набор изображений (**карты параметров**, *feature maps*), обычно хранящийся в памяти в виде многомерного объекта, называемого **тензором**.

При проектировании CNN для задач визуального распознавания мы обычно размещаем полносвязные слои на конце сети для выполнения этапа классификации. Поскольку выходные данные слоев свертки представляют собой набор изображений, как правило, мы используем стратегии подвыборки, чтобы уменьшить объем информации, распространяемой по сети, а затем снизить риск переобучения при подаче полносвязных слоев.

Как правило, существует два способа выполнения подвыборки.

- Пропуск оператора свертки для некоторых входных пикселей. В результате выходные данные слоя свертки будут иметь меньшее количество пространственных измерений, чем входные.

- Внедрение функций подвыборки, таких как объединение слоев (*pooling layers*).

На следующем рисунке показана общая архитектура CNN, в которой на уровне объединенного слоя уменьшается пространственная размерность, а на уровне полносвязного выполняется этап классификации:

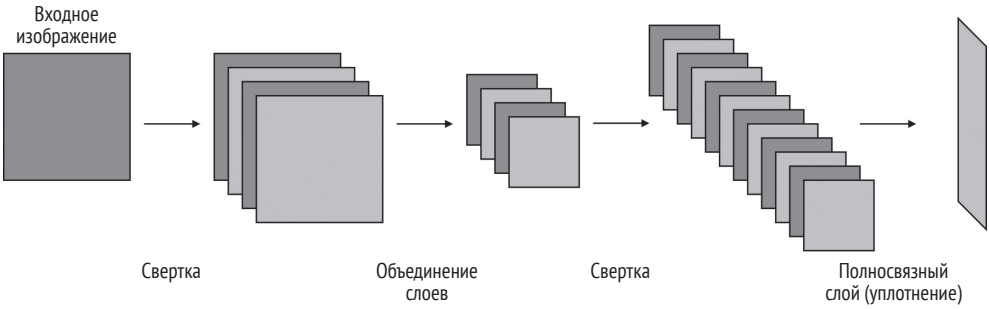

Рис. 1.7 ❖ Общая структура CNN с объединенным (*pooling*) слоем для уменьшения пространственной размерности

Одним из наиболее важных аспектов, которые следует учитывать при развертывании сетей DL для TinyML, является размер модели, обычно определяемый как объем памяти, необходимый для хранения значений весов.

Поскольку наши малые платформы имеют ограниченную физическую память, мы требуем, чтобы модель была компактной, чтобы соответствовать целевому устройству.

Однако ограниченный объем памяти не единственная проблема, с которой мы можем столкнуться при развертывании модели на микроконтроллерах. Например, хотя обученная модель обычно использует арифметические операции с точностью до плавающей запятой, процессоры на микроконтроллерах могут не иметь для этого аппаратной поддержки.

Незаменимым методом для преодоления указанных ограничений является **квантизация** (*quantizing*)[14].

Квантизация

Квантизация – это процесс выполнения вычислений для нейронной сети с более низкой битовой точностью. Широко распространенный метод для микроконтроллеров применяет после обучения квантизацию и преобразует 32-разрядные веса с плавающей запятой в 8-разрядные целочисленные значения. Этот метод позволяет уменьшить размер модели в 4 раза и значительно снизить время ожидания при очень незначительном снижении точности или вообще без него.

[14] Специфический для ML процесс квантизации (*quantizing*, см. далее) не следует путать с квантованием (*quantization*) – процессом разбиения диапазона значений на конечное число уровней.

DL имеет важное значение для создания приложений, которые принимают интеллектуальные решения. Однако ключевым требованием для приложений с батарейным питанием является использование устройства с низким энергопотреблением. До сих пор мы упоминали мощность и энергию в общих чертах, но в следующем разделе посмотрим, что они означают практически.

Разница между мощностью и энергией

Целевое потребление мощности в TinyML, к которой мы стремимся, находится в диапазоне милливатт (мВт) или ниже, что означает в тысячи раз большую эффективность, чем у традиционной настольной машины. Хотя есть случаи, когда мы могли бы рассмотреть возможность использования решений для дополнительного сбора энергии, таких как солнечные панели, это не всегда возможно из-за стоимости и физических размеров.

Однако что мы подразумеваем под мощностью и энергией? Давайте познакомимся с этими терминами, дав общий обзор фундаментальных физических величин, управляющих электронными схемами.

Различие между напряжением и током

Ток – это то, что заставляет работать электронную схему. Ток представляет собой *количество электрических зарядов Q через сечение S проводника за заданное время t*, как представлено на следующем рисунке:

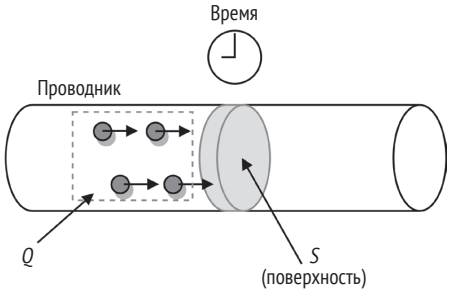

Рис. 1.8 ❖ Ток представляет собой количество электрических зарядов *Q* через сечение проводника *S* за промежуток времени *t*

Ток определяется следующим образом:

$I = Q/t$.

Здесь мы имеем:

- *I*: ток, измеряемый в амперах (А),
- *Q*: количество электрических зарядов в кулонах (К) через сечение *S* за заданное время,

- t: время, измеряемое в секундах (с).

Ток протекает в цепи при следующих условиях.

- У нас есть *проводящий материал* (например, медная проволока), позволяющий пропускать электрический заряд.
- У нас *замкнутая цепь*, т. е. цепь без разрывов, обеспечивающая непрерывный путь для протекания тока.
- У нас есть *источник разности потенциалов*, называемой **напряжением**, определяемым следующим образом:

$$V = V^+ - V^-.$$

Напряжение измеряется в **вольтах** (В или *V*) и создает электрическое поле, позволяющее электрическому заряду протекать по цепи. Любой источник питания, как порт USB, так и аккумулятор, являются источником разности потенциалов.

Символическое представление источника питания приведено на следующем рисунке:

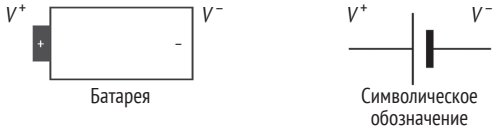

Рис. 1.9 ❖ Символическое обозначение батареи

Чтобы избежать постоянного обращения к V^+ и V^-, мы условно определяем отрицательный вывод батареи как опорный, присваивая ему значение 0 В (GND[15]). Напряжение и ток связаны через **закон Ома**: он гласит, что *ток через проводник пропорционален напряжению и обратно пропорционален сопротивлению проводника*:

$$I = V/R.$$

Все проводники обладают сопротивлением, однако есть специальные компоненты под названием резисторы. **Резистор** – это электрический компонент, используемый для установки значения тока. Он имеет сопротивление, измеряемое в омах (Ом), и обозначается буквой *R*.

Символическое обозначение резистора показано на рис. 1.10.

Резисторы являются важными компонентами любой электронной схемы, и для разновидностей, используемых в этой книге, их значение указано с помощью цветных полос на корпусе. Стандартные резисторы имеют четыре,

[15] Обозначение GND (от *ground* – земля) произошло от названия общего провода схемы, который на заре электронной эры часто «заземлялся» – соединялся с настоящей (электротехнической) землей. В настоящее время для GND более правильным будет употребление названия «общий провод» (а не «заземление»), так как с электротехнической землей его соединять, наоборот, не рекомендуется, кроме особо оговоренных случаев.

пять или шесть полос. Цвет на полосах обозначает значение сопротивления, как показано на рис. 1.11.

Рис. 1.10 ❖ Символическое обозначение резистора

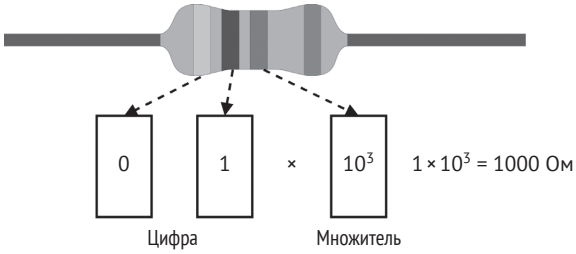

Рис. 1.11 ❖ Пример резистора с четырьмя полосами

Чтобы легко декодировать цветовые полосы, мы рекомендуем использовать онлайн-инструмент магазина «Чип и дип» (https://www.chipdip.ru/calc/resistor?ysclid=l9nxpl83xb883320558).

Теперь, когда мы знаем основные физические величины, управляющие электронными схемами, рассмотрим разницу между мощностью и энергией.

Мощность и энергия

Иногда мы меняем местами слова «мощность» и «энергия», потому что думаем, что это одно и то же, но на самом деле они относятся к разным физическим величинам. Энергия – это способность выполнять работу (например, использовать силу для перемещения объекта), в то время как мощность – это скорость выделения или потребления энергии, т. е. энергия в единицу времени.

С практической точки зрения мощность говорит, например, о том, как быстро разрядится аккумулятор, – более высокая мощность подразумевает более быструю разрядку аккумулятора.

Мощность P и энергия E связаны с напряжением и током по следующим формулам:

$$P = V \cdot I,$$
$$E = P \cdot t.$$

В следующей таблице представлены входящие в эти формулы физические величины мощности и энергии:

Величина	Единица	Значение
P	Ватт (Вт)	Мощность
E	Джоуль (Дж)	Энергия
V	Вольт (В)	Напряжение питания
I	Ампер (А)	Потребление тока
t	Секунда (с)	Время операции

Таблица 1.1. Таблица с указанием физических величин в формулах для мощности и энергии

Для микроконтроллеров напряжение питания составляет порядка нескольких вольт (например, 3.3 В), в то время как потребляемый ток находится в диапазоне микроампер (мкА) или миллиампер (мА). По этой причине мы обычно используем микроватт (мкВт) или милливатт (мВт) для мощности и микроджоуль (мкДж) или миллиджоуль (мДж) для энергии.

Теперь рассмотрим следующую задачу, чтобы ознакомиться с концепциями мощности и энергии поближе. Предположим, у вас есть задача обработки, и у вас есть возможность выполнить ее на двух разных процессорах. Эти процессоры имеют следующее энергопотребление и производительность:

Таблица 1.2. Два процессора с разной производительностью и энергопотреблением

Процессор	Энергопотребление, мВт	Производительность, отн. ед.
PU1	12	8x
PU2	3	1x

Какой процессор вы бы использовали для выполнения этой задачи?

Хотя PU1 имеет в 4 раза более высокое энергопотребление, чем PU2, это не означает, что PU1 менее энергоэффективен. Напротив, PU1 более производительный в вычислительном отношении, чем PU2 (в целых 8 раз), что делает его лучшим выбором с энергетической точки зрения, как показано в следующих формулах:

$$E_{PU1} = 12 \cdot T_1;$$
$$E_{PU2} = 3 \cdot T_2 = 3 \cdot 8 \cdot T_1 = 24 \cdot T_1.$$

Мы можем сказать, что PU1 – наш лучший выбор, поскольку он потребит вдвое меньше энергии от аккумулятора при выполнении тех же самых операций.

Обычно мы используем величину количества арифметических операций, выполняемых на один ватт мощности (**OPS per Watt**), чтобы привязать потребляемую мощность к вычислительным ресурсам процессоров.

Программирование микроконтроллеров

Микроконтроллер (**MCU, МК**) является полноценным компьютером, потому что у него есть процессор (который в настоящее время также может быть многоядерным), система памяти (например, RAM или ROM) и неко-

торые периферийные устройства. В отличие от стандартного компьютера микроконтроллер полностью помещается на одном интегрированном чипе, обладает невероятно низкой мощностью и низкой стоимостью.

Часто путают микроконтроллеры с **микропроцессорами (МП)**, но эти названия относятся к разным устройствам. В отличие от микроконтроллера микропроцессор содержит на одном чипе только процессор (**central processing unit**, **CPU**), требующий внешних подключений к системе памяти и другим компонентам для формирования полностью работающего компьютера.

На следующем рисунке приведены основные различия между микропроцессором и микроконтроллером:

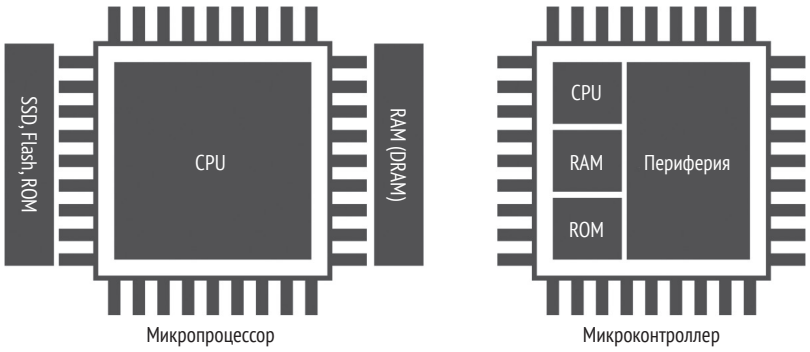

Рис. 1.12 ❖ Микропроцессор в сравнении с микроконтроллером

Как и для всех процессорных блоков, целевое применение влияет на выбор архитектуры.

Микропроцессор выполняет сценарии, в которых задачи заключаются, например, в следующем:

- динамические задачи (меняются в зависимости от действий пользователя или во времени),
- общего назначения,
- интенсивные вычисления.

Микроконтроллер выполняет совершенно другие сценарии, и в следующем списке мы выделим некоторые из важнейших из них.

- *Задачи МК являются одноцелевыми и повторяющимися*.
  В отличие от микропроцессорных приложений задачи для МК, как правило, являются одноцелевыми и повторяющимися, поэтому микроконтроллер не требует перезагрузки программ. Как правило, приложения МК требуют меньших вычислительных затрат, чем микропроцессорные, и не требуют частого взаимодействия с пользователем. Однако они могут взаимодействовать с окружающей средой или другими устройствами.
  В качестве примера вы могли бы рассмотреть термостат. Устройство требует только регулярного контроля температуры и связи с системой отопления.
- *Могут быть ограничения по срокам выполнения*.
  Определенные задачи должны быть выполнены в течение определенного периода времени. Это требование характерно для приложений

реального времени (**real-time applications**, **RTA**), где нарушение временных ограничений может повлиять на качество обслуживания (мягкий режим реального времени) или быть опасным (жесткий режим реального времени).

Автомобильная антиблокировочная система (**ABS**) является примером жесткого RTA, поскольку электронная система должна реагировать в течение определенного периода времени, чтобы предотвратить блокировку колес при нажатии на педаль тормоза.

Для построения эффективного RTA нам требуется устройство с предсказуемой задержкой, поэтому все аппаратные компоненты (процессор, память, обработчик прерываний и т. д.) должны реагировать за точное количество тактов. Изготовители оборудования обычно сообщают в техническом описании о задержке, выраженной в тактовых циклах.

Ограничение по времени требует некоторых архитектурных изменений и ограничений в сравнении с микропроцессором общего назначения. Примером может служить модуль управления памятью (**memory management unit**, **MMU**), который мы в основном используем для преобразования адресов виртуальной памяти, и обычно в составе процессорного модуля микроконтроллеров он отсутствует.

- *Ограничения вследствие низкого энергопотребления*.

  Приложения могут работать в устройстве с батарейным питанием, поэтому микроконтроллер должен быть маломощным, чтобы продлить срок службы источника.

  В соответствии с ограничениями по времени энергопотребление также создает некоторые архитектурные отличия от микропроцессора. Не вдаваясь в подробности аппаратного обеспечения, замечу, что, как правило, дополнительные компоненты на отдельных чипах снижают энергоэффективность. Это основная причина, по которой микроконтроллеры интегрируют как оперативную память, так и своего рода жесткий диск (постоянное запоминающее устройство, ROM) в единую микросхему.

  Как правило, микроконтроллеры также имеют более низкую тактовую частоту, чем микропроцессоры, чтобы потреблять меньше энергии[16].

- *Ограничения по физическому размеру*.

  Устройство может представлять собой продукт небольшого размера. Поскольку микроконтроллер представляет собой разновидность одно-

[16] Это не всегда связано именно с энергопотреблением: рассуждение, подобное приведенному автором выше о количестве операций на ватт мощности, поможет вам понять, что простое снижение тактовой частоты не принесет выгоды в потреблении энергии: те же операции будут выполняться дольше, и общее количество затраченной энергии останется примерно тем же самым. Тактовая частота МК обычно ниже МП по той причине, что МК, как правило, нацелен на выполнение более простых задач, не требующих высоких скоростей вычислений, а более медленные микросхемы не требуют суперсовременного производства с предельными технологическими нормами, и потому существенно дешевле. Кроме того, схемотехнические и конструктивные решения для них при встраивании в аппаратуру оказываются намного проще.

кристальных компьютеров, он идеально подходит для таких устройств. Площадь, занимаемая микроконтроллером, может варьироваться, но обычно находится в пределах нескольких квадратных миллиметров. В 2018 году команда инженеров из Мичиганского университета создала «самый маленький в мире компьютер» размером 0,3 мм с микроконтроллером, работающим на процессоре Arm Cortex-M0+ и системой датчиков без батареи для измерения температуры в живых клетках.

- *Ограничения по стоимости*.
  Все приложения чувствительны к стоимости их производства, а благодаря разработке микросхемы меньшего размера, интегрирующей процессор, память и периферийные устройства, мы делаем микроконтроллеры экономически более выгодными, чем микропроцессоры.

В следующей таблице кратко изложено рассмотренное выше для удобства использования в будущем:

Таблица 1.3. Таблица сравнения микропроцессора с микроконтроллером

Свойство	**Микропроцессор**	**Микроконтроллер**
Приложения	Общего назначения	Конкретного назначения
Арифметический сопроцессор	Может выполнять вычисления с плавающей запятой и с двойной точностью	Преимущественно целочисленные вычисления
RAM	Несколько гигабайт	Несколько сотен килобайт
ROM (или жесткий диск)	Гигабайты и терабайты	Килобайты или мегабайты
Тактовая частота	Гигагерцы	Мегагерцы
Потребление	Ватты	Милливатты и ниже
Операционная система	Требуется	Не обязательна
Стоимость	Десятки-сотни долларов	От нескольких центов до нескольких долларов

Со следующего раздела мы начнем углубляться в архитектурные аспекты микроконтроллеров, анализируя архитектуру памяти и внутренних периферийных устройств.

Архитектура памяти

Микроконтроллеры – это встроенные системы на основе центрального процессора, что означает, что центральный процессор (CPU) отвечает за взаимодействие со всеми его подкомпонентами. Всем процессорам требуется, по крайней мере, память для чтения инструкций и сохранения/считывания значений переменных во время выполнения программы.

В микроконтроллерах мы физически выделяем две отдельные системы памяти для инструкций и данных.

- **Память программ** (**ROM**). Это энергонезависимая память, доступная только для чтения, зарезервированная для хранения выполняемой программы. Хотя ее основная цель – хранить программу, она также

может хранить некоторые постоянные данные. Таким образом, память программ похожа на жесткие диски наших обычных компьютеров.

- **Память данных** (**RAM**). Это энергозависимая память, зарезервированная для хранения и чтения временных данных (переменных). Поскольку это оперативная память, мы теряем ее содержимое при выключении системы.

Поскольку память программ и память данных различаются функционально, для них обычно используются разные полупроводниковые технологии. Это **флеш-технологии** для памяти программ и **статическая оперативная память** (**SRAM**) для памяти данных.

Флеш-память является энергонезависимой и обеспечивает низкое энергопотребление, но, как правило, работает медленнее, чем SRAM. Однако, учитывая преимущество в стоимости[17] по сравнению с SRAM, мы обычно имеем память программ большего объема, чем память данных.

Теперь, когда мы знаем разницу между памятью программ и памятью данных, *где мы можем хранить веса нашей модели глубокой нейронной сети*?

Ответ на этот вопрос зависит от того, имеет ли модель веса с постоянными значениями. Если веса постоянны и не изменяются во время просчета модели, более эффективно хранить их в памяти программ по следующим причинам:

- память программ имеет больший объем, чем SRAM;
- это уменьшает нагрузку на память SRAM, поскольку другие функции требуют хранения переменных или фрагментов памяти во время выполнения.

Мы хотим напомнить вам, что ресурсы памяти микроконтроллеров ограничены, поэтому подобное решение может повлиять на эффективность работы памяти.

Периферийные устройства

Микроконтроллеры предлагают дополнительные встроенные функции, расширяющие их возможности и делающие эти малые компьютеры непохожими друг на друга. Эти функции реализуются через периферийные устройства и имеют важное значение, поскольку именно они позволяют взаимодействовать с датчиками и другими внешними компонентами.

Каждое периферийное устройство имеет специальную функциональность, и оно закреплено за металлической ножкой (выводом) интегральной схемы.

Мы можем обратиться к описанию назначения периферийных выводов в спецификации микроконтроллера, чтобы узнать функциональные возможности каждого вывода. Поставщики оборудования обычно нумеруют выводы против часовой стрелки, начиная с крайнего левого угла корпуса, отмеченного точкой для удобства ориентирования, как показано на следующем рисунке:

---

[17] А также в занимаемой площади на кристалле.

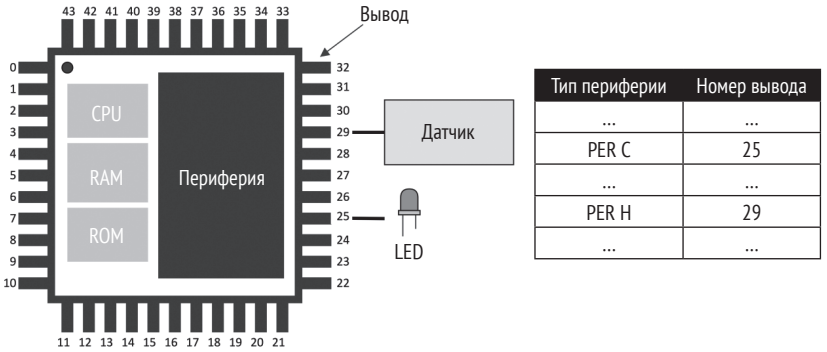

Рис. 1.13 ❖ Назначение выводов.
Выводы нумеруются против часовой стрелки,
начиная с верхнего левого угла, отмеченного точкой[18]

Поскольку периферийные устройства могут быть различных типов, для простоты мы можем сгруппировать их по четырем основным категориям.

Вход/выход общего назначения (GPIO или IO)

GPIO (*general-purpose input/output*) не имеют предопределенного и фиксированного назначения. Их основная функция заключается в установке или считывании двоичных сигналов, которые по своей природе могут находиться только в двух четко определенных состояниях: **высоком** (**HIGH** или 1) или **низком** (**LOW** или 0). На следующем рисунке показан пример двоичного сигнала:

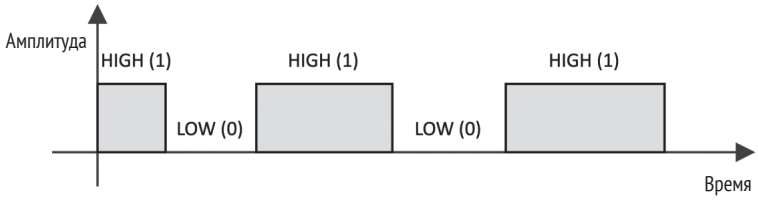

Рис. 1.14 ❖ Двоичный сигнал может находиться только в двух состояниях:
высоком (HIGH или 1) или низком (LOW или 0)

Типичные способы использования GPIO, например, такие:

- включение и выключение светодиода;
- определение того, нажата ли кнопка;
- реализация сложных цифровых интерфейсов/протоколов, таких как VGA.

Периферийные устройства GPIO универсальны и обычно доступны во всех микроконтроллерах.

---

[18] Нумерация выводов микросхем начинается с единицы, а не с нуля, как ошибочно показано на рис. 1.13.

Аналого-цифровые преобразователи

Приложения TinyML, скорее всего, будут иметь дело с изменяющимися во времени физическими величинами, такими как освещенность, звук или температура.

Какими бы ни были эти величины, датчик преобразует их в непрерывный электрический сигнал, интерпретируемый микроконтроллерами. Этот электрический сигнал, который может быть как напряжением, так и током, обычно называют аналоговым сигналом.

Микроконтроллер, в свою очередь, должен преобразовать аналоговый сигнал в цифровой формат, чтобы центральный процессор мог обрабатывать данные.

Аналого-цифровые преобразователи действуют как переводчики между аналоговым и цифровым мирами.

Аналого-цифровой преобразователь (**analog-to-digital converter, ADC, АЦП**) производит выборку аналогового сигнала с фиксированным интервалом времени и преобразует электрический сигнал в цифровой формат.

Цифроаналоговый преобразователь (**digital-to-analog converter DAC, ЦАП**) выполняет противоположную функцию: преобразует внутренний цифровой формат в аналоговый сигнал.

Последовательная связь

Коммуникационные периферийные устройства интегрируют стандартные протоколы связи для управления внешними компонентами. Типичными периферийными устройствами последовательной связи, доступными в микроконтроллерах, являются **I2C**, **SPI**, **UART** и **USB**.

Таймеры

В отличие от всех периферийных устройств, которые мы только что описали, таймеры не взаимодействуют с внешними компонентами, поскольку они используются для запуска или синхронизации событий.

В этом разделе мы завершили обзор ингредиентов TinyML. Теперь, когда мы знакомы с терминологией и общими понятиями, мы можем начать представлять платформы разработки, используемые в этой книге.

Представление Arduino Nano 33 BLE Sense и Raspberry Pi Pico

Плата микроконтроллера – это печатная плата (PCB), объединяющая микроконтроллер с необходимыми электронными компонентами, чтобы сделать его готовым к использованию. В некоторых случаях плата микроконтроллера может содержать дополнительные устройства, предназначенные для конкретных приложений.

В этой книге используются платы микроконтроллеров Arduino Nano 33 BLE Sense (сокращенно Arduino Nano[19]) и Raspberry PI Pico.

Arduino Nano, разработка компании Arduino (https://www.arduino.cc), представляет собой плату, сочетающую микроконтроллер (nRF52840 на базе процессора ARM Cortex-M4) с несколькими датчиками и беспроводную Bluetooth-связь для упрощения разработки TinyML. При разработке на Arduino Nano нам потребуется всего несколько дополнительных внешних компонентов, поскольку большинство из них уже доступно «на борту».

Raspberry Pi Pico, разработка Raspberry Pi Foundation (https://www.raspberrypi.org), не содержит датчиков и встроенного модуля Bluetooth. Тем не менее он оснащен микроконтроллером (RP2040), работающим на двухъядерном процессоре Arm Cortex-M0+, пригодным для уникальных и мощных приложений TinyML. Таким образом, эта плата идеально подойдет для изучения взаимодействия с внешними датчиками и построения электронных схем.

На следующем рисунке показано параллельное сравнение этих двух платформ, чтобы нагляднее показать их различия друг от друга:

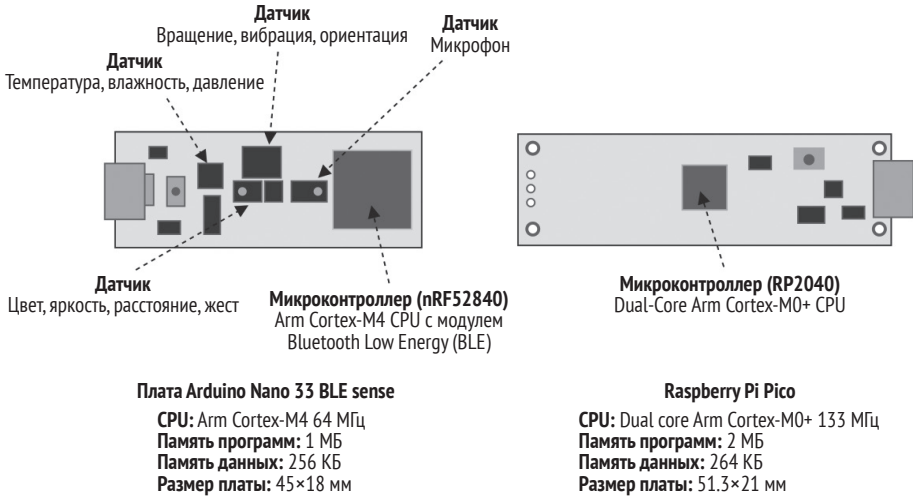

Рис. 1.15 ❖ Arduino Nano 33 BLE Sense и Raspberry Pi Pico

Как мы можем видеть, обе платы имеют небольшие размеры, USB-порт для питания и программирования, а также микроконтроллер на базе ARM. В то же время они обладают некоторыми уникальными функциями, которые делают платы идеальными для различных сценариев разработки TinyML.

[19] Под Arduino Nano без дополнительных пояснений в дальнейшем в этой книге подразумевается именно плата Arduino Nano 33 BLE Sense, которую не следует путать с собственно Arduino Nano – популярной в среде любителей платой Arduino-контроллера на основе ATmega328.

Настройка Arduino Web Editor, TensorFlow и Edge Impulse

Для TinyML нам требуются различные программные средства, охватывающие как разработку ML, так и программирование встроенных устройств. Благодаря компаниям Arduino, Edge Impulse и Google большинство инструментов, рассмотренных в этой книге, основаны на браузере и требуют всего нескольких шагов настройки.

В этом разделе мы представим эти инструменты и подготовим среду разработки Arduino, необходимую для написания и загрузки программ в Arduino Nano и Raspberry Pi Pico.

Подготовка веб-редактора Arduino Web Editor

Интегрированная среда разработки Arduino (**Arduino IDE**) – это программное приложение, разработанное компанией Arduino (https://www.arduino.cc/en/software) для написания и загрузки программ на платы, совместимые с Arduino. Программы написаны на C++ и обычно называются программистами скетчами Arduino.

Arduino IDE делает разработку программного обеспечения доступной и простой для разработчиков, не имеющих опыта в области встраиваемого программирования. Фактически этот инструмент скрывает все сложности, с которыми мы можем столкнуться при работе со встроенными платформами, такими как кросс-компиляция и программирование устройств.

Arduino также предлагает среду IDE на основе браузера (https://create.arduino.cc/editor). Она называется Arduino Web Editor и делает программирование еще более простым, поскольку программы могут быть написаны, скомпилированы и загружены на микроконтроллеры непосредственно из веб-браузера. Все проекты Arduino, представленные в этой книге, будут основаны на этой облачной среде. Однако, поскольку бесплатный тарифный план Arduino Web Editor ограничен 200 с времени компиляции в день, вы можете перейти на платный тарифный план или использовать бесплатную локальную среду разработки Arduino IDE, чтобы получить неограниченное время компиляции.

В следующих главах этой книги мы будем полагать Arduino IDE и Arduino Web Editor взаимозаменяемыми[20].

Подготовка TensorFlow

TensorFlow (https://www.tensorflow.org) – бесплатная программная платформа с открытым исходным кодом, разработанная Google для ML. Мы будем ис-

[20] Отметим, что наименования и размещение пунктов меню в локальной среде Arduino IDE и в онлайновом Arduino Web Editor могут существенно различаться.

пользовать это программное обеспечение для разработки и обучения наших моделей ML с использованием Python в Google Colaboratory.

Colaboratory (https://colab.research.google.com/notebooks), сокращенно **Colab**, – это бесплатная среда разработки на Python, которая запускается в браузере с использованием Google Cloud. Она похожа на записную книжку Jupyter, но имеет некоторые существенные отличия, например:

- среда не нуждается в настройке;
- среда основана на облаке и хостится в Google;
- имеет множество предустановленных библиотек Python (включая TensorFlow);
- интегрирована с Google Drive;
- предлагает бесплатный доступ к общим ресурсам GPU и TPU[21];
- результатами легко поделиться (в том числе на GitHub).

В этом случае библиотека TensorFlow не требует настройки, поскольку Colab поставляется вместе с ней.

В Colab мы рекомендуем включить ускорение с помощью графического процессора на вкладке **Runtime**, чтобы ускорить вычисления в TensorFlow. Для этого перейдите в раздел **Runtime | Change runtime type** и выберите **GPU** в раскрывающемся списке **Hardware accelerator**, как показано на следующем скриншоте:

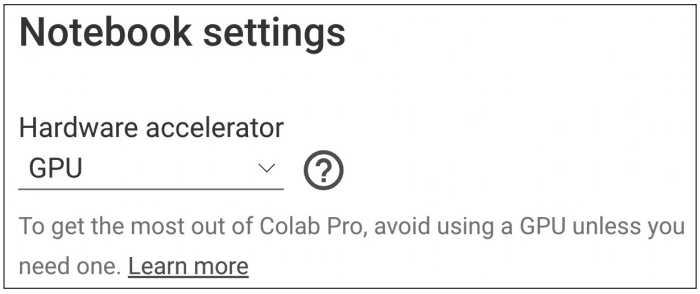

Рис. 1.16 ❖ Включение графического ускорителя

Поскольку ускорение графического процессора является общим ресурсом для других пользователей, доступ к нему из бесплатной версии Colab ограничен.

Вы можете подписаться на Colab Pro (https://colab.research.google.com) для получения приоритетного доступа к самым быстрым графическим процессорам.

[21] GPU (*graphics processing unit*, графический процессор, не путать с обычным процессором CPU) – отдельный процессор, изначально предназначенный для обработки изображений, но хорошо пригодный для выполнения других задач, включающих однотипные параллельные вычисления. TPU (Tensor Processing Unit, тензорный процессор) – разработка Google, специальная ориентированная на задачи машинного обучения. Об особенностях GPU и TPU в задачах ML см. по-русски https://habr.com/ru/post/422317.

TensorFlow не единственный инструмент от Google, который мы будем использовать. Как только мы создадим модель ML, нам нужно будет запустить ее на микроконтроллере. Для этого Google разработала специальную библиотеку TensorFlow Lite for microcontrollers.

TensorFlow Lite for microcontrollers (https://www.tensorflow.org/lite/microcontrollers), сокращенно **TFLu**, – ключевая библиотека программного обеспечения для размещения приложений ML на маломощных микроконтроллерах. Проект является частью TensorFlow и позволяет запускать модели DL на устройствах с несколькими килобайтами памяти. Написанная на C/C++, библиотека не требует операционной системы и динамического выделения памяти.

TFLu не нуждается в настройке, поскольку включена в Arduino Web Editor.

Подготовка Edge Impulse

Edge Impulse (https://www.edgeimpulse.com) – это программная платформа для сквозной разработки ML. Она бесплатна для разработчиков, и через несколько минут вы уже можете запустить модель ML на вашем микроконтроллере. Фактически платформа объединяет инструменты для следующих действий:

- сбора данных с датчиков,
- применения процедур цифровой обработки сигналов к входным данным,
- построения и обучения моделей ML,
- тестирования моделей ML,
- развертывания моделей ML на микроконтроллерах,
- поиска наилучшего блока обработки сигналов и модели ML для вашего варианта использования.

К сведению

Все перечисленные инструменты также доступны через открытые API.

Разработчикам просто нужно зарегистрироваться на веб-сайте, чтобы получить доступ ко всем этим функциям непосредственно через пользовательский интерфейс.

Как это делается...

Вы должны выполнить следующие шаги по настройке Arduino Web Editor.

1. Зарегистрируйтесь в Arduino по адресу https://auth.arduino.cc/register.
2. Войдите в Arduino Web Editor (https://create.arduino.cc/editor).
3. Установите агент Arduino, следуя пошаговой установке по адресу https://create.arduino.cc/getting-started/plugin/welcome.
4. Установите Raspberry Pi Pico SDK.

- Windows.
  i. Загрузите папку *pico-setup-windows* с https://github.com/ndabas/pico-setup-windows/releases.
  ii. Запустите *pico-setup.cmd*.
- Linux.
  i. Откройте Terminal.
  ii. Создайте временную папку:

```
$ mkdir tmp_pico
```

  iii. Измените каталог на свою временную папку:

```
$ cd tmp_pico
```

  iv. Загрузите скрипт настройки Pico с помощью `wget`:

```
$ wget wget https://raw.githubusercontent.com/raspberrypi/pico-setup/
master/pico_setup.sh
```

  v. Сделайте файл исполняемым:

```
$ chmod +x pico_setup.sh
```

  vi. Выполните скрипт:

```
$ ./pico_setup.sh
```

  vii. Добавьте `$USER` в группу удаленного доступа `dialout`:

```
$ sudo usermod -a -G dialout $USER
```

5. Проверьте взаимодействие Arduino Web Editor с Arduino Nano.
   i. Откройте Arduino Web Editor в браузере.
   ii. Подключите плату Arduino Nano к ноутбуку или ПК с помощью кабеля micro-USB.

   Редактор должен распознать плату в раскрывающемся списке устройств и сообщить для **Arduino Nano 33 BLE** имя порта (например, **/dev/tty-ACM0**):

Рис. 1.17 ❖ Ожидаемый результат взаимодействия Arduino Web Editor с Arduino Nano 33

6. Проверьте взаимодействие Arduino Web Editor с Raspberry Pi Pico.
   i. Откройте Arduino Web Editor в браузере.
   ii. Подключите плату Raspberry Pi Pico к ноутбуку или ПК с помощью кабеля micro-USB.

   Редактор должен распознать плату в раскрывающемся списке устройств и сообщить для **Raspberry Pi Pico** имя порта (например, **/dev/ttyACM0**):

Рис. 1.18 ❖ Ожидаемый результат взаимодействия Arduino Web Editor с Raspberry Pi Pico

Мы успешно создали инструменты, которые помогут нам разрабатывать будущие примеры. Прежде чем закончить эту главу, мы хотим протестировать базовый пример на Arduino Nano и Raspberry Pi Pico, чтобы официально отметить начало нашего путешествия в мир TinyML.

Запуск скетча на Arduino Nano 33 и Raspberry Pi Pico

В этом примере мы будем мигать светодиодами Arduino Nano и Raspberry Pi Pico, используя готовый пример мигания из Arduino Web Editor.

Эта программа – аналог «Hello World», управляет встроенным светодиодом, мигающим с помощью периферийного устройства GPIO; через него мы сможем отправиться куда угодно.

Цель этого упражнения – познакомить вас с редактором Arduino Web Editor и помочь понять, как разрабатывать программу с помощью Arduino.

Подготовка

Скетч Arduino состоит из двух функций `setup()` и `loop()`, как показано в следующем фрагменте кода:

```
void setup() {
}
void loop() {
}
```

`setup()` – это первая функция, выполняемая программой, когда мы нажимаем кнопку сброса или включаем питание платы. Эта функция выполняется только один раз и обычно отвечает за инициализацию переменных и периферийных устройств.

После `setup()` программа выполняет бесконечный цикл `loop()`, как показано на следующем рисунке:

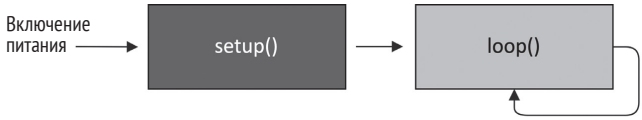

Рис. 1.19 ❖ Схема выполнения скетча

Эти две функции требуются во всех программах Arduino.

Как это делается...

Шаги, описанные в этом разделе, действительны как для Arduino Nano, Raspberry Pi Pico, так и для других плат, совместимых с Arduino Web Editor.

1. Подключите плату к ноутбуку или ПК с помощью кабеля micro-USB. Затем убедитесь, что Arduino IDE сообщает имя и порт устройства.
2. Откройте готовый пример **Blink**: щелкните по **Examples** в меню слева, выберите вкладку **BUILT IN** (т. е. встроенные примеры), а затем **Blink**, как показано на следующем скриншоте:

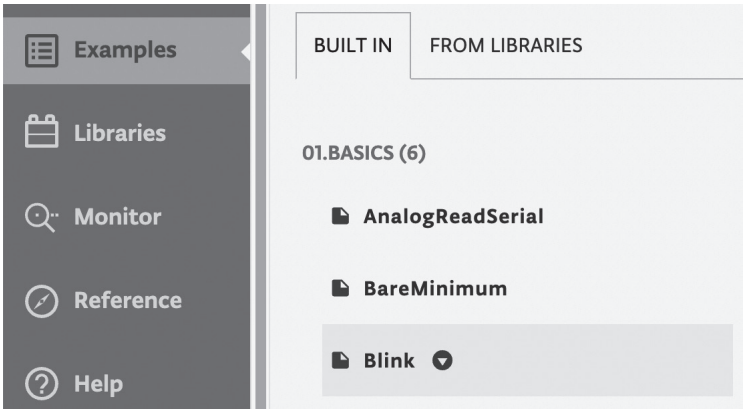

Рис. 1.20 ❖ Пример мигания встроенным светодиодом

После нажатия на название скетча код примера появится в области редактора.

3. Чтобы скомпилировать и загрузить программу на целевое устройство, нажмите на стрелку рядом с раскрывающимся списком плат, как показано на следующем скриншоте:

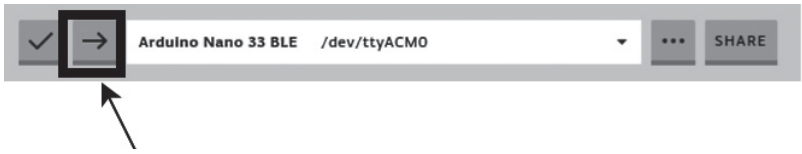

Рис. 1.21 ❖ Стрелка рядом с выпадающим списком плат скомпилирует и прошьет программу на целевом устройстве

По окончании процесса в консоли в нижней части страницы должно появиться сообщение **Done**, а встроенный светодиод должен начать мигать.