ОСОБЕННОСТИ РЕАЛИЗАЦИИ НЕЙРОСЕТЕЙ НА ПЛАТФОРМАХ STM32/CORTEX-M ДЛЯ ЗАДАЧ ОБНАРУЖЕНИЯ ЗАКЛАДНЫХ УСТРОЙСТВ ИМПУЛЬСНО-РЕФЛЕКТОРНЫМ МЕТОДОМ

Сурвило И.С., магистрант гр.467241

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Петров С.Н. – канд. техн. наук, доцент

Аннотация. В статье рассматривается применение нейросетевых технологий для обнаружения закладных устройств импульсно-рефлекторным методом. Особое внимание уделено реализации на микроконтроллерах STM32 в парадигме TinyML. Описаны этапы разработки: сбор и разметка данных, выбор архитектуры нейросети (свёрточные, рекуррентные или временные сети), квантование и обрезка моделей. Представлена конфигурация периферии (АЦП, таймер, DMA) для синхронизированного сбора данных. Ключевыми инструментами выступают TensorFlow Lite for Microcontrollers, CMSIS-NN и STM32CubeAI.

Ключевые слова. Обнаружение закладных устройств, импульсно-рефлекторный метод, нейросетевые технологии, TinyML, STM32, добротность, обработка временных рядов, сверточные нейронные сети, CMSIS-NN, TensorFlow Lite for Microcontrollers, STM32CubeAI.

Задача обнаружения закладных устройств с использованием импульсно-рефлекторного метода представляет собой специфическую область радиофизического контроля, где нейросетевые технологии могут значительно повысить точность и надежность обнаружения. Основная идея метода заключается в облучении исследуемой области зондирующими электромагнитными импульсами и анализе отраженных сигналов. Эффективность метода для обнаружения ЗУ обусловлена их конструктивными особенностями, а именно наличием высокодобротных антенных систем. Добротность, или Q-фактор, является ключевым параметром, характеризующим способность колебательного контура эффективно накапливать и отдавать энергию при резонансе. Антенные системы большинства ЗУ работают в узком диапазоне частот, что делает их чувствительными к воздействию зондирующего сигнала, частота которого совпадает с их собственной резонансной частотой. Когда зондирующий импульс попадает на контур с высокой добротностью, в нем наводятся токи с резко возрастающей амплитудой, что приводит к значительному увеличению мощности отраженного сигнала. Полезный сигнал представляет собой сложный временной сигнал, содержащий прямой отклик от резонансного контура ЗУ, форма которого зависит от резонансной частоты, Q-фактора и степени затухания. Чем выше Q-фактор, тем медленнее затухают колебания после воздействия короткого импульса, создавая длинную осциллирующую часть на графике сигнала. Этот уникальный временной профиль является основным диагностическим признаком наличия ЗУ. Входной сигнал для нейросети всегда будет содержать фоновый электромагнитный шум окружающей среды, а также отражения от других металлических предметов и поверхностей. Классические подходы, основанные на простых пороговых значениях или быстром преобразовании Фурье, часто оказываются неэффективными в условиях сложного шумового фона, так как они плохо различают форму сигнала. Исходя из этого нейросеть, особенно архитектуры, предназначенные для обработки временных рядов, становится эффективным инструментом. Сверточная нейронная сеть может выделять локальные признаки формы сигнала, такие как крутизна фронта или период колебаний, в то время как рекуррентные сети могут улавливать временные зависимости между различными участками сигнала [1].

Для успешного обучения нейросетевой модели критически важно правильно сформировать обучающий набор данных, который должен быть максимально разнообразным и покрывать все возможные сценарии. Набор должен состоять из трех основных классов сигналов:

– сигналы наличия ЗУ от различных типов устройств на разных частотах и с разными параметрами Q-фактора, в том числе за препятствиями;

– сигналы отсутствия ЗУ, включающие чистый фоновый шум помещения и отражения от стационарных объектов;

– сигналы шума и помех от других радиоустройств для повышения устойчивости модели.

Методология сбора данных может быть как экспериментальной, так и основанной на моделировании, позволяющем быстро сгенерировать большие объемы данных. После сбора данных требуется их обработка и нормализация: обрезка до фиксированной длины, масштабирование до единого диапазона и разметка по классам. Качество разметки напрямую влияет на итоговую точность модели.

Выбор и последующая оптимизация архитектуры нейронной сети являются центральными этапами разработки решения для обнаружения закладных устройств на микроконтроллерах семейства STM32. Устройства данного класса характеризуются жесткими ограничениями по объему оперативной памяти, флэш-памяти и вычислительной производительности, поэтому применение глубоких и широких моделей, обученных на графических процессорах для облачных задач, абсолютно невозможно. TinyML предлагает комплекс методов для адаптации нейросетевых моделей под эти условия, позволяя запускать модели машинного обучения непосредственно на устройстве с минимальными ресурсами. Учитывая, что

входными данными являются одномерные временные ряды, наиболее перспективны архитектуры, специально разработанные для обработки последовательных данных [2]. Сверточные нейронные сети эффективны для обработки временных рядов, способны автоматически извлекать локальные признаки из сигнала, а легковесные вариации с использованием сверток с разделением глубины значительно сокращают количество параметров и вычислительную сложность. Рекуррентные нейронные сети и их вариации LSTM и GRU изначально предназначены для работы с последовательными данными и способны улавливать долгосрочные временные зависимости, однако могут быть более ресурсоемкими. Временные сверточные сети сочетают лучшие свойства CNN и RNN, используя свертки с расширенными ядрами для получения большого окна восприятия без увеличения глубины сети, и часто демонстрируют лучшую производительность по сравнению с RNN. Главным инструментом в TinyML является квантование – преобразование весов и активаций нейронной сети из 32-битных чисел с плавающей запятой в 8-битные целочисленные представления. Квантование дает три ключевых преимущества: резкое сокращение размера модели в четыре раза, значительное ускорение вычислений за счет аппаратной поддержки SIMD-инструкций в процессорах Cortex-M, а также снижение потребления энергии, что критически важно для автономных устройств.

Существует два подхода к квантованию. Квантование после обучения просто в реализации, но может привести к заметной потере точности. Квантование с учетом обучения, при котором в процессе обучения модели искусственно имитируется эффект квантования, заставляет сеть привыкнуть к работе с целочисленными значениями и обычно приводит к гораздо меньшей потере точности. Современные фреймворки, такие как TensorFlow Lite, предоставляют инструменты для обоих подходов. Помимо квантования, для дальнейшего сжатия модели используются обрезка – процесс удаления незначительных весов из сети, что экономит память и время вычислений. Эти техники часто комбинируются: модель сначала оптимизируется для поиска легкой архитектуры, затем к ней применяется квантование и обрезка. Инструменты, такие как STM32Cube.AI, интегрируют часть этих процессов и позволяют генерировать код для оптимизированных моделей, которые затем выполняются с использованием высокооптимизированных библиотек CMSIS-NN.

Успешная реализация нейросетевой модели для обнаружения закладных устройств на микроконтроллерах невозможна без глубокого понимания аппаратных возможностей целевой платформы и доступного программного инструментария. Экосистема STM32 предоставляет целостное решение для разработчиков, включая широкий спектр 32-битных ARM Cortex-M процессоров, оптимизированные математические библиотеки и мощные средства автоматической генерации кода. Для задач обнаружения ЗУ наиболее интересны высокопроизводительные серии, такие как STM32H7 и более новые STM32N6. Процессоры Cortex-M7 способны работать с частотой до 480-550 МГц, предоставляя значительную вычислительную мощность. Серия STM32N6 идет еще дальше, предлагая встроенные нейронные процессоры, специально спроектированные для ускорения операций нейронных сетей, которые могут выполнять прогнозирование квантованных моделей с высокой скоростью и крайне низким энергопотреблением, освобождая основной процессор для других задач. Самая важная часть аппаратной поддержки TinyML – это наличие оптимизированных математических библиотек CMSIS-DSP и CMSIS-NN [2].

CMSIS-DSP содержит множество функций для цифровой обработки сигналов, таких как быстрое преобразование Фурье, цифровые фильтры и матричные операции, оптимизированные для процессоров Cortex-M с поддержкой операций с плавающей и фиксированной запятой.

CMSIS-NN – это специализированная библиотека, содержащая высокооптимизированные ядра для основных операций нейронных сетей: свертки, полносвязные слои, пулинг и активационные функции. Эти ядра написаны на ассемблере для конкретных моделей Cortex-M и используют все доступные аппаратные возможности для достижения максимальной производительности.

На программном уровне центральное место занимает TensorFlow Lite for Microcontrollers – открытый фреймворк от Google, специально разработанный для запуска моделей машинного обучения на устройствах с ограниченными ресурсами, предоставляющий среду выполнения, управляющую памятью и выполняющую операции модели. TensorFlow Lite for Microcontrollers по умолчанию использует CMSIS-NN в качестве движка для выполнения вычислений на Cortex-M процессорах. Для упрощения разработки STMicroelectronics создала X-CUBE-AI – пакет расширения для STM32CubeIDE, который берет на вход предварительно обученную и оптимизированную модель в формате TensorFlow Lite или ONNX и автоматически генерирует готовый к использованию C-код. Процесс генерации включает создание C-структур, описывающих архитектуру сети, генерацию массивов данных с весами и смещениями, генерацию C-функций для прогнозирования с вызовом ядер CMSIS-NN, а также предоставление простого API.

Процесс разработки нейросетевого приложения для обнаружения закладных устройств на платформе STM32 представляет собой четко структурированную последовательность этапов.

Первый этап – обучение модели в среде высокопроизводительных вычислений на Python с использованием TensorFlow и Keras. На этом этапе выбирается исходная архитектура, показывающая хорошие результаты на обучающем наборе данных, проводится обучение до достижения приемлемого

уровня точности, при необходимости проводится квантование с учетом обучения. По завершении обучения модель сохраняется в нативном формате фреймворка.

Второй этап – оптимизация и преобразование модели с помощью TensorFlow Lite Converter, который конвертирует модель в формат TensorFlow Lite. Именно на этом этапе происходит основное квантование весов и активаций из 32-битных чисел с плавающей запятой в 8-битные целые числа. Конвертер позволяет указать тип входных данных и предоставляет возможность использовать файл с эталонными данными для калибровки, что помогает минимизировать потерю точности. Результатом является файл «.tflite», содержащий архитектуру и параметры модели в целочисленном представлении [2].

Третий этап – генерация кода с помощью инструмента X-CUBE-AI. Пользователь загружает файл «.tflite» в мастер-конфигуратор, инструмент анализирует архитектуру модели, рассчитывает необходимый объем памяти, проверяет возможность размещения модели в памяти целевого микроконтроллера и генерирует набор C-файлов и заголовочных файлов, содержащих массивы с квантованными весами, заготовки функций инференса и API для инициализации, загрузки входных данных и получения результатов.

Четвертый этап – интеграция и разработка приложения. Сгенерированные файлы добавляются в проект микроконтроллера. Основной код приложения инициализирует периферийные модули: АЦП, таймеры и DMA для настройки сбора данных. По сигналу от таймера, синхронизированному с моментом отправки зондирующего импульса, АЦП начинает преобразование отраженного сигнала, данные передаются через DMA напрямую в буфер в оперативной памяти без нагрузки на центральный процессор. Когда буфер заполняется, программа вызывает функцию прогнозирования, сгенерированную X-CUBE-AI, которая копирует данные в буфер ввода нейронной сети и последовательно выполняет все операции, используя оптимизированные ядра CMSIS-NN с целочисленной арифметикой.

После завершения вычислений в выходном буфере находится массив вероятностей для каждого класса, приложение считывает его, определяет класс с максимальной вероятностью и принимает решение.

Для микроконтроллеров STM32 существует стандартная и чрезвычайно эффективная конфигурация периферийных модулей: синхронизированный сбор данных с помощью АЦП, таймера и DMA. Эта конфигурация позволяет минимизировать нагрузку на центральный процессор и обеспечить сбор данных с высокой частотой дискретизации, что критически важно для корректного анализа формы отраженного импульса. Таймер настраивается для генерации регулярных событий с необходимой частотой, которые служат триггером для АЦП, запуская преобразование аналогового напряжения в цифровое значение, и могут использоваться для генерации самого зондирующего импульса, что обеспечивает точную синхронизацию. АЦП конфигурируется для работы в режиме непрерывного сканирования канала. DMA настраивается на чтение регистра данных АЦП и автоматическую запись этих данных в заранее выделенный массив в оперативной памяти, позволяя центральному процессору свободно выполнять другие задачи или находиться в спящем режиме. Только по завершении сбора всей временной последовательности DMA генерирует прерывание, и ЦП передает собранный массив данных на вход нейросетевой модели для анализа. Качество входных данных зависит от разрядности АЦП (чем выше разрядность, тем больше динамический диапазон), скорости АЦП (максимальная частота дискретизации определяет плавность фиксации формы сигнала) и точности тактового генератора (любые флуктуации частоты таймера приведут к некорректной временной шкале). Для дополнительного анализа в частотной области можно использовать функции FFT из библиотеки CMSIS-DSP, преобразующие временной ряд в массив амплитуд и фаз различных частотных составляющих, что может быть полезно для выделения частотных признаков, менее выраженных во временной области. Управление памятью является важным аспектом: буфер для хранения сырых данных АЦП должен быть точно рассчитан на требуемую длительность сигнала, память под веса модели выделяется статически из расчетов X-CUBE-AI, память для промежуточных активаций также выделяется статически и является одним из главных источников потребления RAM.

На данный момент технология TinyML является единственным жизнеспособным подходом для решения поставленной задачи на ресурсоограниченных микроконтроллерах, а центральным технологическим стеком является связка TensorFlow Lite for Microcontrollers, библиотеки CMSIS-NN и инструмента STM32CubeAI. При выборе аппаратной платформы стоит отдать предпочтение моделям с процессорами Cortex-M7 или новейшим Cortex-M85 с встроенным нейронным процессором, обеспечивающим максимальную производительность и энергоэффективность для прогнозирования нейронных сетей.

Список использованных источников:

1. Сурвило, И.С. Импульсно-рефлекторный метод обнаружения закладных радиоустройств / И.С. Сурвило, С.Н. Петров // Технические средства защиты информации : материалы XXIII Международной научно-технической конференции / Белорусский государственный университет информатики и радиоэлектроники [и др.] ; редкол.: О. В. Бойправ [и др.]. – Минск, 2025. – С. 303–307.

2. Йодиче, Дж. TinyML. Книга рецептов / Дж. Йодиче. – М.: ДМК Пресс, 2023. – 298 с.