Санкт-Петербургский государственный университет

Математико-механический факультет
Кафедра системного программирования

Холод Николай Григорьевич

Разработка на ПЛИС нейронной сети, реализующей распознавание объектов интереса на изображениях

Магистерская диссертация

Научный руководитель:
ст. преп. Смирнов М. Н.

Рецензент:
гл. науч. сотр. АО НПП АМЭ, к. т. н. Крюков С. Н.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics and Mechanics Faculty
Software Engineering Chair

Kholod Nickolay

FPGA implementation of a neural network for recognition of objects of interest on images

Master's Thesis

Scientific supervisor:
Senior Lect. M .N. Smirnov

Reviewer:
CRO JSC SPE AME, PhD S. N. Kryukov

Saint-Petersburg
2019

Оглавление

1. Введение

В настоящее время индустрия технического (компьютерного) зрения, т.е. технологии разработки машин и алгоритмов, способных производить автоматическую обработку изображений разной сложности, стремительно развивается и становится неотъемлемой частью современного мира. Одной из часто встречающихся задач компьютерного зрения является задача классификации изображений — соотнесение изображения к какому-либо классу в зависимости от представленных на нем объектов.

Существенным прорывом в области компьютерного зрения и распознавания изображений стало появление сверточных нейронных сетей. Благодаря им, в течение последних нескольких лет точность автоматического распознавания изображений приблизилась к точности распознавания человека, а в некоторых задачах даже превысила её, вытеснив во многих областях классические алгоритмы компьютерного зрения. Однако, вместе с тем, глубина и сложность сетей увеличились в несколько десятков раз, что повлияло на их вычислительную сложность, которая достигла нескольких гигафлопов.

Отдельный класс задач технического зрения связан со встраиваемыми системами — системами, установленными на каком-либо автономном устройстве, например квадрокоптере или мобильном роботе. Встраиваемые системы предъявляют особые требования к аппаратному обеспечению и алгоритмам. В связи с тем, что система размещается внутри сложного автономного устройства, следующие факторы являются ключевыми.

- Во-первых, в связи с отсутсвием внешнего питания важно, чтобы система имела минимальное собственное электропотребление.
- Во-вторых, при установке системы на воздушные носители, критически важными параметрами системы становятся её габариты и вес.
- В-третьих, необходимо, чтобы использовалась нетеплонагружен-

ная аппаратура, которая не требует специальных схем охлаждения.

В зависимости от назначения встраиваемой системы, к ней могут предъявляться дополнительные требования по работоспособности в различных средах, высокому гарантированному сроку службы и т.д. В связи с тем, что встраиваемые системы обрабатывают сигналы в реальном времени при ограниченности вычислительных ресурсов, к алгоритмам предъявляются строгие требования по производительности.

Графические процессоры (GPU), которые чаще всего используются в качестве вычислителей для сверточных нейронных сетей, не соответствуют данным требованиям по нескольким причинам. Во-первых, GPU не предназначены к эксплуатации при постоянных выбрациях и ударных нагрузках. Во-вторых, графические процессоры отличаются значительным электопотреблением. В-третьих, требуется особая схема отвода тепла с GPU. Кроме того, практически все графические процессоры, присутствующие на рынке, являются импортными и не могут использоваться в приборах, имеющих требование использования только отечественных комплектующих.

Поэтому для запуска нейронных сетей на встраиваемых системах необходимо использовать другие вычислительные устройства. Чаще всего в качестве вычислительного устройства используются системы на кристалле (System on Chip, SoC) — элетронные схемы, содержащие в себе один или несколько микропроцессоров, а также банк памяти, блоки стандартных интерфейсов для подключенния внешних устройств, и выполняющие функции целого вычислительного устройства. Во многих системах на кристалле в качестве компонента присутствуют программируемые логические интергральные схемы (ПЛИС), а именно программируемая пользователем вентильная матрица (field-programmable gate array, FPGA). Особенностью SoC, содержащих FPGA, является то, что большая часть памяти устройства находится в банках памяти, тогда как собственная память ПЛИС мала и составляет порядка нескольких тысяч килобит. Доступ к внешним банкам памяти происходит дольше, чем обращение к встроенному ОЗУ, поэтому уменьшение количества

памяти, потребляемой нейронной сетью, способно увеличить скорость её работы на ПЛИС.

FPGA помимо логических блоков и блоков коммутации содержит еще и блоки цифровой обработки сигналов (digital signal processor, DSP), которые позволяют быстро производить операции над вещественными числами. Несмотря на наличие данных блоков, максимальная пиковая производительность логических операций на ПЛИС больше, чем у операций с плавающей точкой. Поэтому лучше всего для реализации на FPGA подходят бинарные нейронные сети, поскольку их выполнение можно реализовать используя преимущественно логические операции, при этом каждый вес такой сети требует всего один бит для хранения, что позволяет одновременно хранить все веса на устройстве.

Существуют несколько готовых решений, позволяющих запускать нейронные сети на ПЛИС. Данные решения разрабатываются основными производителями ПЛИС и поэтому ориентированы только на устройства собственного производства, например
OpenVino для устройств Intel и PyNQ для FPGA компании Xilinx. Так как в рамках данной работы в качестве платформы использовалась плата DE1-SoC с Cyclone V SoC производства Intel, то в качестве возможного решения рассматривался OpenVino. Однако разработчиками OpenVino заявлена поддержка только Intel Arria 10 FPGA GX, которая имеет большее количество памяти, более быстрые DSP-блоки, однако данная плата в 5 раз дороже платы DE1-SoC. Поэтому OpenVino не позволяет достичь приемлемого времени работы нейронной сети на более доступных и популярных в России устройствах.

В связи с описанными выше причинами возникает задача реализации сверточной нейронной сети на ПЛИС с учетом требований по скорости работы и количеству потребляемой памяти. Данная работа посвящена проектированию и обучению сверточной нейронной сети, сочетающей эффективный алгортим работы бинарной нейронной сети, но достигающей лучшей точности расспознавания по сравнению со сверточными сетями данного типа.

2. Постановка задачи

Целью данной работы является выработка и апробация подхода к построению нейросетевого классификатора изображений, предназначенного для работы на ПЛИС. Для достижения этой цели в рамках работы были сформулированы следующие задачи.

- Изучить, какие типы нейронных сетей используются в существующих реализациях на ПЛИС.
- Разработать архитектуру нейронной сети, допускающую эффективную реализацию процедуры распознавания.
- Произвести обучение и подбор оптимальных гиперпараметров нейронной сети на основе имеющихся данных.
- Реализовать модули, необходимые для запуска обученной нейронной сети на ПЛИС.
- Произвести тестирование и измерение производительности реализованной сети.

3. Литобзор

3.1. Введение в предметную область

3.1.1. Сверточные нейронные сети

Сверточные нейронные сети представляют собой глубокие сети прямого распространения, предназначенные главным образом для распознавания изображений. Структура сверточных нейронных сетей представлет собой последовательность слоев, каждый из которых принимает на вход набор данных, называемый картой признаков, обрабатывает его и выдает новый набор карт признаков. Основными типами слоев в сверточной нейронной сети являются слои свертки, активации, субдискретизации (пулинга), батч-нормализации.

Основой свёрточных нейронных сетей являются слои свертки. Веса данного слоя представляют собой набор многомерных изображений-фильтров с числом каналов равным количеству входных карт признаков. Применение одного трехмерного фильтра к трехмерным входным данным дает в результате одну двумерную карту признаков. Как правило, пространственные размерности фильтра небольшие и равны между собой (1x1,3x3,5x5).

Выходные данные свёрточного слоя поступают на вход слоя активации, который представляет собой нелинейную функцию, применяемую поэлементно к данным. Её главное назначение — дать возможность сети изучить нелинейные закономерности. В настоящее время чаще всего в качестве функции активации используется ReLU (Rectifier Linear Unit) или её модификации. Реже применяются функции гиперболического тангенса и сигмоиды.

Слой субдискретизации производит уменьшение размеров карт признаков с помощью взятия максимума в заданной окрестности. По изображению проходит скользящее окно и находится максимум входного изображения в окне. При этом, окна вполне могут пересекаться, однако в этом случае шаг окна *stride* должен быть больше единицы.

Сверточные слои обычно выделяют некоторые метапризнаки изоб-

ражения и для того, чтобы принять решение о том, в какой класс отнести то или иное изображение, в сеть добавляют полносвязные слои, которые исполняют роль классификатора признаков. Полносвязный слой представлет собой умножение матрицы входных данных на матрицу весов.

В процессе обучения с помощью метода обратного распространения ошибки за одну итерацию через сеть проходит некоторое фиксированное число эталонов, этот набор эталонов называют мини-батчем. В [1] описан метод ускорения обучения сети, названный батч-нормализацией, который заключается в нормировке слоев внутри мини-батча и накоплении информации о среднем значении мини-батча и дисперсии, которые используются для нормировки карт признаков на этапе тестирования сети. Данная операция выносится в отдельный слой и может быть описана формулой.

$$Y = \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \alpha$$

3.1.2. High level Synthesis compiler

На протяжении многих лет основным инструментом для проектирования логики ПЛИС являлись языки описания аппаратуры (hardware description language, HDL). Примерами данных языков являются Verilog и VHDL. В данных языках существует подмножество инструкций, называемых синтезируемыми, а написанный на этом подмножестве код называют RTL (register transfer lever, схема на уровне передач данных между регистрами). RTL код преобразуется в логически эквивалентное описание netlist, на основе которого создается прошивка.

Однако недостатками HDL языков являются высокие требования к квалификации разработчика (поскольку HLD языки сильно отличаются от языков програмирования высокого уровня) и необходимость сложного процесса переноса алгоритмов, написанных на языках программирования высокого уровня, на данные языки. Поэтому начиная с 2000-ых годов, появляется технология высокоуровневого синтеза (High level Synthesis, HLS). Она позволяет генерировать код RTL-уровня из

исходного кода, написанного на языках программирования высокого уровня. Данный подход позволяет значительно упростить и ускорить разработку. Иногда сгенерированный с помощью HLS код является менее оптимальным, по сравнению с аппаратным описанием, однако имеется возможность ручной оптимизации сгенерированного кода.

Существует множество HLS компиляторов, например Vivaldo HLS, Intel HLS, Synphony C. В данной работе для реализации нейронной сети использовался Intel HLS компилятор. Он генерирует описание схемы на языке Verilog из исходного кода на языке C++, также доступна симуляция работы схемы и её верификация. Компилятор содержит директивы препроцессора, позволяющие оптимизировать циклы, интерфейсы для обращения к потокам бит, инструкции, указывающие в какую область памяти следует поместить переменную, встроенные типы данных, позволяющие хранить целые числа, состоящие из произвольного фиксированного числа бит и вещественные числа с фиксированной запятой. Преобразование исходного кода в RTL описание производится с учетом количества ресурсов на конкретном устройстве. Также имеется встроенный профилировщик, который позволяет узнать количество ресурсов, затраченных на построение схемы и отдельных её частей, произвести анализ эффективности циклов и обращений к памяти. Данные возможности позволяют существенно упростить разработку программного обеспечения для ПЛИС.

3.2. Обзор методов дискретизации нейронных сетей

3.2.1. Методы бинаризации

В статье [2] был описан способ обучения нейронных сетей с весами, принимающими значения $\{-1, +1\}$. Авторы назвали данный способ BinaryConect. В данной работе используется два способа бинаризации: детерминистический и стохастический. Детерминистическая бинаризация осуществляется простым взятием знака весов. Стохастическая би-

наризация производится исходя из следующих требований:

$$\begin{cases} p(w^b = 1) = \sigma(w) \\ p(w^b = -1) = 1 - \sigma(w) \end{cases}$$

Для того, чтобы избежать неограниченного роста весов по модулю, веса обрезаются так, чтобы они лежали в отрезке $[-1; 1]$. Бинаризованное значение весов используется при прямом проходе сети и при вычислении градиентов вместо вещественных значений весов, однако обновляются именно вещественные веса. Таким образом, дискретизация создает случайный шум, который добавляется к значениям весов и обеспечивает также регуляризирующий эффект.

Продолжением данной работы стала статья [3], в которой подобный подход используется для обучения сети BinaryNet, которая содержит не только бинарные веса, но и бинарные активации. Так как функция знака не является недифференцируемой, то для вычисления градиента используется straight-through estimator [4], суть которого состоит в замене градиента знака на градиент тождественной функции. Для обучения этой сети очень важно использовать батч-нормализацию [1], так как необходимо избежать случаев, когда все активации становятся одного знака и после бинаризации становятся равны друг другу.

В статье [5] описан еще один подход к бинаризации сети, который позволяет обучать как сети с дискретными весами, так и сети с бинаризованными активациями: XNOR-Net. Для весов и, при необходимости, для активаций сети, используется следующее представление $W_i = \alpha B_i$, где $B_i \in \{-1, 1\}$ — битовое представление веса, α — общая для всего слоя константа. На этапе обучения хранятся и обновляются вещественные веса, а бинарное представление весов вычисляется непосредственно перед применением слоя с помощью решения задачи $\|W_i - \alpha B_i\|^2 \to \min\limits_{B,\alpha}$, решением которой является $B_i = sign(W_i), \alpha = \frac{\sum_i |W_i|}{n}$. Для вычисления градиента используется straight-through estimator. Недостатком данного алгоритма является то, что необходимо вычислять весовые матрицы для активаций во время прямого прохода, что требует дополнительных

вычислений.

В настоящее время имеется тенденция к увеличению количества слоев в нейронных сетях. Однако при большом увеличении глубины сверточной сети возникает проблема деградации точности автоматического распознавания сети, для преодоления которой активно используются residual networks. Для таких сетей остается актуальной проблема умешьнения количества памяти, требуемой для хранения весов. Поэтому в работе [6] рассматривается способ обучения таких сетей с бинарными весами. Бинаризация весов производится согласно подходу BinaryConnect, однако на каждом слое веса умножаются на фиксированный масштабирующий множитель, зависящий от гиперпараметров слоя.

Главное преимещество бинарных нейронных сетей перед остальными нейронными сетями заключается в возможности реализации вычислений сверточных и полносвязных слоев с помощью XnorPopcount операций. При вычислениях, отрицательные кодируются нулем, положительные единицей. Тогда благодаря тому, что таблица умножения для чисел -1;+1 совпадет с таблицей истинности для логической операции «отрицание исключающего или» и сумма таких чисел эквивалентна вычислению разницы между числом единичных и нулевых бит, операция скалярного произведения двух битовых векторов длины n может быть выполнена следующим образом:

$$dot(x, w) = 2bitpopcount(Xnor(x, w)) - n.$$

3.2.2. Методы тернаризации

Помимо способов бинаризации весов сети, также существуют способы квантификации весов на 3 и более значений. Так, в работе [7] представлен способ обучения сети с весами, принимающими три значения: +1, 0, 1. По аналогии с бинаризацией, рассматриваются детерменистический способ квантификации с использованием двух порогов и стохастический способ, который заключается в разделении диапазона весов на два интервала и применении стохастической бинаризации на

каждом интервале.

В работе [8] исследовалось влияние квантификации весов на работу рекуррентных нейронных сетей. Эксперименты показали, что бинаризация матрицы весов во всех разновидностях рекуррентных нейронных сетей приводит ко взрыву градиентов и неограниченному возрастанию состояний сети из-за отстутствия близких к нулю значений весов. Добавление 0 ко множеству возможных значений позволяет решить данную проблему. В статье представлен новый способ квантификации значений весов, названный экспоненциальной квантификацией. Логарифмы абсолютных значений весов квантифицируются либо отсечением по порогу, равному 0.5, либо стохастически по следующему правилу:

$$\log(w_i) = \begin{cases} \lceil \log(w_i) \rceil, p_1 = \frac{|W|}{2^{\log_2 |W|}} - 1 \\ \lfloor \log(w_i) \rfloor, p_2 = 1 - p_1 \end{cases}$$

В работе [9] описан способ квантификации весов и активаций сети на три значения. Данный способ можно считать обобщением работы [5] для случая трех возможных значений весов и активаций. Тернаризация производится при помощи отсечения по симметричному порогу, вычисляемому динамически исходя из распределения весов.

$$w_i = \begin{cases} +a^*, w_i > \delta \\ 0, |w_i| < \delta \\ -a^*, w_i < -\delta \end{cases}$$

В статье [10] рассмотрен способ обучения нейронной сети с весами, принимающими три значения и активациями, принимающими два значения. Обучение нейронной сети разделяется на два этапа. На первом этапе обучается нейронная сеть с вещественными весами и активациями со значениями на отрезке [-1,1]. Это достигается путем использования функции гиперболического тангенса в качестве функции активации и функции-обертки для внутреннего представления весов. На втором этапе обученные ранее веса разделяются на три группы: -1, 0, +1, и проиходит квантификация весов. Затем запускается обучение с ис-

пользованием тернаризированных весов, но с обновлением их внутренних вещественных представлений. После каждой эпохи квантицикация вновь повторяется. Этот способ показывает неплохие результаты для классификации бинаризованных входных данных.

Способ обучения нейронных сетей с тернарными весами и активациями, основанный на подходе учитель-ученик, описан в работе [11]. Его смысл в том, что сначала на исходных данных обучается сеть-учитель, а затем сеть-ученик учится иммитировать поведение сети-учителя в каждом слое. Обе сети имеют одинаковую архитектруру, однако сеть-учитель обучается с вещественными весами и тернарными активациями, а сеть-ученик имеет троичные веса и активации. При этом сохраняется следующий инвариант: знаки весов сети-учителя и сети-ученика совпадают. Сеть, обученная данным способом, достигает лучшей точности распознавания, чем обученные по-другому сети аналогичной вычислительной сложности.

3.2.3. Методы произвольной квантификации

В работе [12] было изучено влияние сжатия весов на точность и скорость работы нейронных сетей, и предложена эффективная степень сжатия: метрика сравнения сетей с разной степеью сжатия весов. Для квантификации весов применялась итеративная процедура, включающая в себя обучение сети с вещественными весами и равномерную квантификацию весов. Эффективная степень сжатия определяется как отношение размера такой несжатой сети, которая достигает такой же точности как и данная сжатая, к количеству памяти, необходимому для хранения весов.

Работа [13] рассматривает вопрос квантификации активаций сети. В последнее время, в глубоких нейронных сетях чаще всего используется функция активации Relu. Однако данная функция не ограничивает область возможных значений, тогда как квантификатор является кусочно-постоянной функцией, и его область значений ограниченна. Это приводит к большой ошибке аппроксимации для больших значений активаций. Поэтому в рамках данной работы были предложены

несколько вариантов функций активации, призванные решить данную проблему. В частности, рассматривается кусочная функция вида

$$f(x) = \begin{cases} 0, x < 0 \\ x, x \in [0, 1] \\ 1 + \log(x), x > 1 \end{cases}$$

Результаты экспериментов показали, что её использование на 2 процента повышает точность распознавания квантифицированной сети.

В статье [14] представлен способ обучения нейронной сети с весами и активациями, занимающими произвольное фиксированное количество бит. Особенностью данной сети является возможность сжимать также и градиенты активаций при обучении, что открывает возможности для обучения нейронных сетей на ПЛИС. Проведенные эксперименты в рамкам этой работы показали, что для достижения точности, сопоставимой с точностью небинаризованной модели, достаточно обучать модель с однобитными весами, двухбитными активациями и 10-битными градиентами.

Метод	Веса	Активации	float-point	∇	реализация
BinaryConnect [2]	-1,1	R	да	R	сложение
BinaryNet [3]	-1,1	-1,1	нет	R	XnorPopcount
XnorNet [5]	-1,1	-1,1	да	R	XnorPopcount
BinaryResNet [6]	-1,1	R	да	R	сложение
NN with Few Multiplications [7]	-1,0,1	R	да	R	сложение
Recurrent Ternary NN [8]	-1,0,1	R	да	R	сложение
TernaryNN [11]	-1,0,1	-1,0,1	да	R	сложение
Ternary weight networks [9]	-1,0,1	-1,0,1	да	R	сложение
Bitwise neural network [10]	-1,0,1	-1,1	нет	R	XnorPopcount
HWGQ[13]	int	R	fix-point	R	умножение
DoReFa[14]	int	int	fix-point	int	умножение

Таблица 1: Сводная таблица методов квантификации

3.3. Обзор существующих реализаций нейронных сетей на ПЛИС

Одним из наиболее известных готовых решений для запуска обученных нейронных сетей на ПЛИС является Intel OpenVINO toolkit [15] — набор библиотек и средств оптимизации для разработки программного обеспечения с использованием компьютерного зрения и глубокого обучения. Данное решение позволяет запускать нейронные сети на различных устройствах: CPU, GPU, FPGA. Процесс подготовки нейронной сети для работы на ПЛИС с помощью OpenVINO состоит из трех основых шагов. Первый шаг — это обучение нейронной сети с помощью любой библиотеки глубокого обучения. Второй шаг — оптимизация модели, полученной на предыдущем шаге, которая производится с помощью оптимизатора, входящего в состав OpenVINO. Список оптимизаций включает в себя как общие оптимизации, такие как перенос батч-нормализации в предшествующий сверточный слой, так и оптимизации под конкретное оборудование. Третий шаг — собственно запуск сети на конкретной платформе. Поддерживается запуск не только нейронных сетей с 32-битными весами и активациями, но запуск сетей с ограниченной точностью, в частности с 8-битными целыми весами и активациями. Недостатком данного средства можно считать отсутствие поддержки для старых устройств, так среди FPGA поддерживается только Intel Arria 10.

В работе [16] представлена библиотека FINN, позволяющая запускать бинарный нейронные сети на ПЛИС производства Xilinx. Данная библиотека содержит несколько оптимизаций, которые позволяют более эффективно запускать бинарные нейронные сети на ПЛИС. Во-первых, батч-нормализация совмещена с вычислением функции активации и заменена на отсечение по предварительно посчитанному порогу. Во-вторых, операция пуллинга применяется к уже бинаризованным активациям и сводится к операции логического ИЛИ. Все эти оптимизации существенно уменьшают вычислительную сложность нейронной сети без потери точности распознавания. Для реализации использовал-

ся Vivaldo HLS.

Бинарные нейронные сети позволяют разворачивать глубокие модели на устройствах с ограниченными ресурсами, благодаря замене матричного умножения на XnorPopcount операции, однако точность распознавания бинарных нейронных сетей существенно ниже, чем точность сетей с фиксированной точкой. С целью повышения точности распознавания авторы статьи [17] представили комплекс библиотек для обучения бинарных нейронных сетей и разработки эффективных ускорителей для их выполнения на ПЛИС. Основным нововведением, позволившим повысить точность распознавания бинарных нейронных сетей, является использование многоуровневой остаточной бинаризации, поэтому предложенная архитектура названа ReBNet. M-уровневая остаточная бинаризация устроена следующим образом: имеются M коэффициентов масштаба γ_i, первый уровень бинаризации выполняется по формуле $y_1 = \gamma_1 sign(x)$, ошибка данной аппроксимации равна $r_1 = x - \gamma_1 sign(x)$, последующие уровни выполняют бинаризацию ошибки аппроксимации предыдущего уровня $y_i = \gamma_i sign(r_{i-1})$. Авторы работы представили HLS библиотеку для реализации сети архитектуры ReBNet на ПЛИС. Главное отличие данной библиотеки, от реализации сетей в FINN, заключается в устройстве блока матричного умножения битовых матриц с помощью XnorPopcount операций. Данный блок в RebNet включает в себя M аккумуляторов для хранения промежуточных результатов вычислений слоев и подмодуль многоуровневой бираризации, реализованный с использованием DSP-блоков. В рамках работы были проведены эксперименты с разным числом уровней бинаризации, которые показали, что при M=1 пропускная способность реализованной сети совпадает с пропускной способностью сети, реализованной с помощью FINN, при увеличении числа уровней бинаризации, пропускная способность падает пропорциально числу уровней.

4. Архитектура нейронной сети

4.1. Сжатие весов

Бинаризация весов проводится с помощью алгоритма, описанного в [5]. В процессе обучения поддерживаются вещественные значения весов, которые обновляются с помощью метода обратного расспространения ошибки. Бинаризация весов выполняется перед вычислением слоя и производится по следующей формуле:

$$\widehat{w_{i,j,k,f}} = c_f sign(w_{i,j,k,f}) = \frac{\sum_{i,j,k} |w_{i,j,k,f}|}{\sum_{i,j,k} 1} sign(w_{i,j,k,f}).$$

Как будет показано ниже, весовые коэффициенты c_f не усложняют процедуру запуска слоя в режиме тестирования, вместе с тем увеличивается множество допустимых нейронных сетей.

Для того, чтобы избежать неограниченного роста вещественных весов по модулю, применяется процедура нормировки. Каждый вес делится на среднеквадратичное отклонение весов, но только если оно больше единицы.

Веса первого слоя нейронных сетей не подвергались бинаризации по нескольким причинам. Во-первых, первый слой получает на вход значения пикселей изображения, находящиеся в диапазоне от -0.5 до 0.5, что не позволяет применить основанный на логических операциях алгоритм вычисления. Во-вторых, бинаризация первого слоя заметно влияет на точность распознавания сети, тогда как вычисления, которые выполняет данный слой, ограничиваются сверткой одноканального или трехканального изображений с несколькими ядрами небольшого размера (как правило 3x3), что существенно меньше, чем вычисления в остальных слоях нейронной сети с вещественными весами.

4.2. Сжатие активаций

Бинаризация активаций с помощью алгоритма, используемого в сети XnorNet, требует большого количества вещественных вычислений на

этапе тестирования алгоритма. Поэтому в данной работе использовался следующий механизм бинаризации:

$$y_{i,j,k} = d_k sign(x_{i,j,k}),$$

где $x_{i,j,k}$ — результат свертки, $y_{i,j,k}$ — активции, а d_k — обучаемые положительные весовые коэффициенты. Использование обучаемых весовых коэффициентов вместо вычисляемых динамически позволяет избежать вещественных вычислений во всех слоях бинарной светрки, кроме последнего. В связи с этим, появляется возможность в процессе тестирования сети вычислять активации без учета весовых коэффициентов, и после чего умножить выходы сети на весовые коэфффициенты последнего слоя. Кроме того, весовые коэффициенты сверточных и полносвязных слоев не влияют на реузльтат активации, так как операция взятия знака инвациантна к умножению на положительную константу.

Для инициализации весовых коэффициентов активаций использовались две стратегии. Первая стратегия заключалась в инициализации всех весовых коэффициентов в сети значением 1. Вторая стретегия заключалась в инициализации весовых коэффициентов в слое значением

$$d = \sqrt{\frac{2}{channels}},$$

где $channels$ — количество каналов в тензоре. Не было замечено существенной разницы в точности расспознавания сети при использовании только различной стратегии инициализации весовых коэффициентов.

Для того, чтобы пропускать градиенты через недифференцируемую функцию взятия знака, использовался straight through estimator. Градиент функции знака заменялся на градиент следующей функции:

$$f(x) = \begin{cases} x, x \in [-1, 1] \\ 1 + \log(x), x > 1 \\ -1 - \log(-x), x < -1 \end{cases}$$

Производная данной функции имеет вид:

$$f'(x) = \begin{cases} 1, x \in [-1, 1] \\ \frac{1}{x}, |x| > 1 \end{cases} \quad (1)$$

При стремлении x к бесконечности производная стремится к 0, однако порядок стремления у неё меньше, чем у сигмоиды, прозводная которой равна

$$\frac{d}{dx}(\frac{1}{1+e^{-x}}) = \frac{e^{-x}}{(1+e^{-x})^2},$$

благодаря чему удается уменьшить влияние проблемы затухающего градиента на обучение сети.

4.3. Ансамблирование

Использование ансамблей решающих функций является часто используемым способом улучшения качества алгоритмов машинного обучения, в частности ансамбли бинарных нейронных сетей используются в работе [18]. Наиболее часто используемым методом ансаблирования нейронных сетей является голосование. Ансамбль представляет собой несколько нейронных сетей, обученых независимо друг от друга. Предсказания этих сетей усредняются.

Использование ансамбля бинарных нейронных сетей в классическом понимании этого термина для работы на ПЛИС сопряжено с некоторыми сложностями, а именно количество памяти, требуемое для хранения весов растет пропорционально количеству нейронных сетей в ансамбле, что приводит к невозможности хранить веса всех сетей ансамбля в локальной памяти устройства (on-chip memory).

Рассмотрим структуру бинарной нейронной сети. Первый сверточный слой данной сети выделяет некоторые низкоуровневые признаки входного изображения. Следующие далее слои батч-нормализации и бинаризации производят сравнение величин полученных признаков с некоторым пороговым значением, преобразуя тем самым вещественные признаки в бинарные. Остальная часть нейронной сети производит

классификацию полученных бинарных признаков. При этом, одни и те же вещественные данные можно бинаризовать по-разному, и результирующие бинарные маски могут сильно отличаться друг от друга. Имея алгоритм построения M независимых бинарных масок по заданному вещественному массиву, можно подать каждую маску на вход одной и той же бинарной сети и получить M разных предсказаний. Полученные предсказания можно усреднить и увеличить тем самым точность распознавания сети. Преимуществом данной архитектуры по сравнению с ансаблем бинарных нейронных сетей является то, что в данной архитектуре количество памяти, требуемое для хранения весов сети, не зависит от числа бинарных масок.

Примером алгоритма построения независимых бинарных масок может служить многоуровневая остаточная бинаризация. Первый уровень бинаризации выполняет сравнение входной величины с некоторвым пороговым значением, второй и последующий уровни также производят сравнение с пороговым значением, однако сравнивают модуль разности входных данных предыдущего уровня с порогом предыдущего уровня.

Схема данной архитектуры представлена на рис. 1.

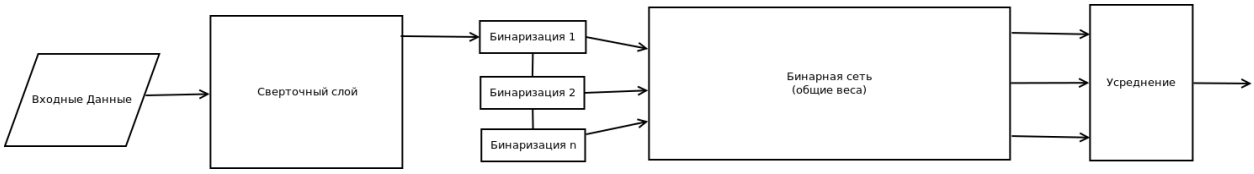

Рис. 1: Схема предложенной архитектуры

Все элементы данной сети обучаются совместно, для каждой ветки данной сети используется своя функция потерь. Итоговая функция потерь для сети равна сумме функций потерь каждой отдельной ветки.

Помимо простого голосования, заключающегося в усреднении предсказаний нескольких веток сети, можно использовать взвешенное голосование, которое предполагает суммирование предсказаний разных веток сети с разными весами. Веса предсказаний обучаются после обучения сети на той части выборки, которая использовалась для валидации при обучении сети. Использование взвешенного голосования позволяет немного увеличить точность распознавания сети.

5. Обучение нейронной сети

5.1. Описание используемых данных

Для обучения нейронной сети использовался набор изображений, представленный АО НПП «Авиационная и морская электроника». Данный набор состоит из 66000 изображений в градациях серого размером 48x48, которые были получены автоматической обработкой более 30 видеозаписей полетов квадрокоптеров. Изображения набора разделены на 4 класса: два класса представляют собой объекты (грузовики и строения) и две класса — фоновые (дороги и деревья). Классы являются несбалансированными, фоновые изображения представляют около 2/3 объема всей выборки. Гистограмма распределения классов в выборке представлена на рис. 3, примеры изображений из обучающей выборки представлены на рис.2.

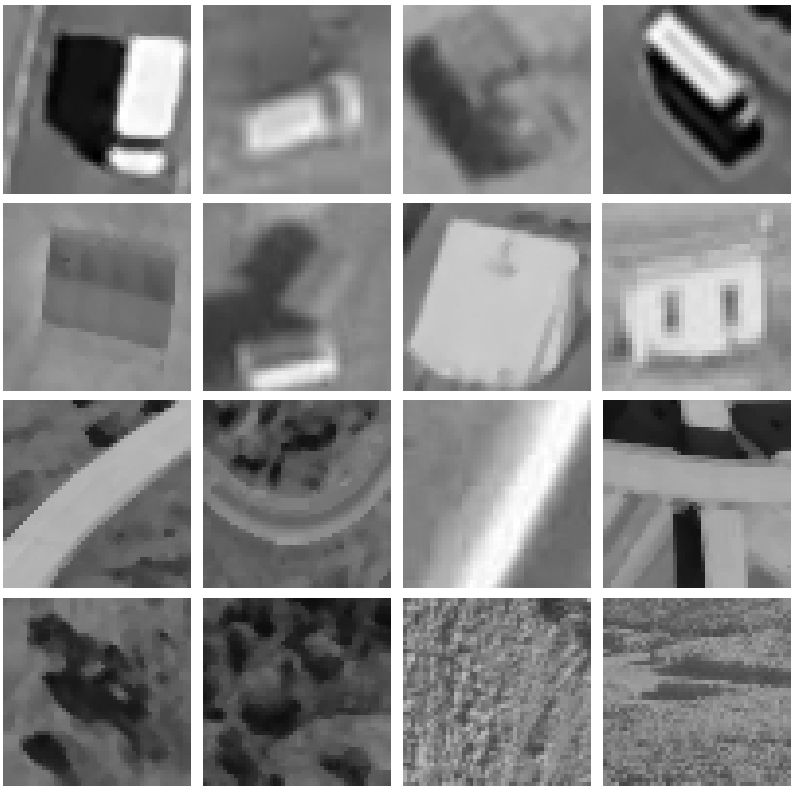

Рис. 2: Примеры изображений из обучающей выборки

Для экспериментов обучающая выборка была разделена на 3 части. Первая часть, состоящая из 54000 изображений, использовалась для

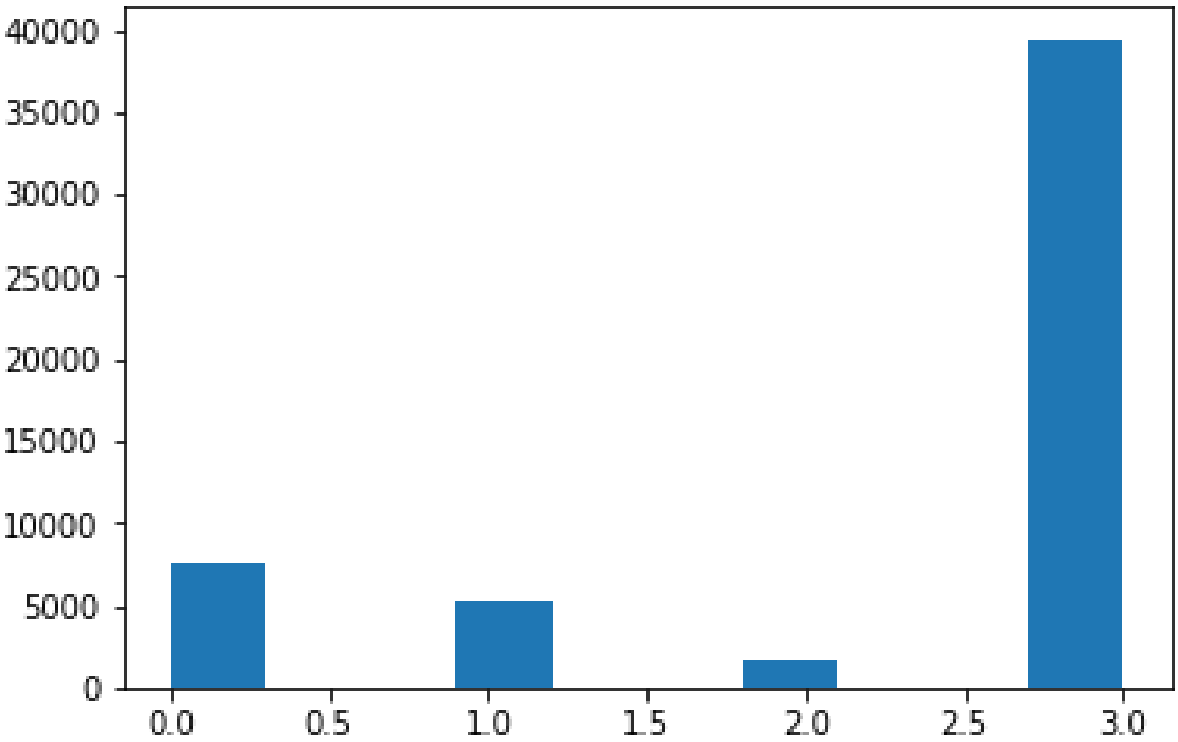

Рис. 3: Гистограмма распределения классов обучающей выборки

обучения сети, вторая, размером в 7200 изображений, применялась для валидаций и подбора параметров, третья — для итогового тестирования. Деление производилось таким образом, чтобы отношение количества эталонных изображений каждого класса оставалось постоянным и равным такому в полной выборке.

5.2. Выбор фукнции потерь и метрик

Несмотря на то, что в выборке преставлены 4 класса, часто стоит задача определить, находится ли на избораженнии некоторый объект интереса, либо изображение представляет собой часть фоноцелевой обстановки, поскольку эта информация может влиять на работу других систем прибора. Довольно часто стоит задача сопровождения расспознанных нейронной сетью объектов. В этом случае, ошибочное определение строения как фрагмента дороги приведет к тому, что объект не станет сопровождаться трекером, а отнесение фрагмента дороги к классу «строения» приведет к сопровождению ложного объекта, что

при большом количестве подобных ситуаций может привести к падению производительности системы, в то время определение строения к классу «грузовики» и фрагмента дороги к классу «деревья» не повлияет на работу трекера.

Чтобы учесть это требование, для обучения была построенна иерархическая функция потерь, учитывающая как качество определения принадлежности к определенному классу, так и точность определения того, имеется ли на изображении объект интереса. Функция потерь $L(y, p)$ имеет следующий вид:

$$L(y, p) = \frac{1}{n} \sum_{i=1}^{n} (L_b(y_i, \widehat{p}_i) + L_m(y_i, p_i))$$

$L_m(y, p)$ представляет собой многоклассовую кросс-энтропию и имеет вид

$$L_m(y, p) = \sum_{j=1}^{K} y_j \log(p_j),$$

где K — количество классов, p — предсказание сети, а y — истинные метки классов. $L_b(y, p)$ представляет собой бинарную кросс-энтропию и имеет вид

$$L_b(y, \widehat{p}) = y_b \log(\widehat{p}) w_b + (1 - y_b) \log(1 - \widehat{p}),$$

где b_p — мера принадлежности , вычисляемая как $b_p = \max(p_1, p_2)$(p_1 и p_2 — предсказанные вероятности принадлежности изображения к классам «строение» и «грузовик»), $y_b = y < 2$ — бинарная метка, принимающая значение 1, если на изображении представлен объект интереса, и 0, в противном случае, w_b — весовой коэфициент, используемый ввиду того, что изображения, на которых представлен объект интереса, занимают небольшую часть выборки.

Для оценки качества распознавания в качествее метрики использовалась площадь под кривой точность-полнота(precision-recall area under the curve).

Точность (precision) представляет собой долю верно определенных положительных объектов. Полнота (recall) представлет собой процент

объектов верно определенных объектов положительного класса среди всех объектов этого класса.

Precision-recall кривая представлет собой график, отражающий зависимость точности алгоритма классификации от полноты при изменении порога принятия решения. Так как точность и полнота являются метриками бинарной классификации, то для случая многоклассовой классификации существует несколько вариаций данной кривой. В данной работе использовалась микро-усредненная (micro average) версия кривой. Пусть имеется задача классификации на k классов. Каждое предсказание превращается в k бинарных предсказаний вида «один класс против остальных». Построенная по полученному набору бинарных предсказаний кривая и является micro average precision-recall кривой.

Precision-Recall AUC представляет собой площадь под графиком precision-recall кривой.

5.3. Подбор гиперпараметров

Для обучения нейронных сетей использовался фреймворк keras [20] с бэкэндом tensorflow [19]. Были реализованны классы для бинарной свертки, бинарного полносвязного слоя и бинаризации активаций. Straight throught estimation был реазизован с помощью функции tf.stop_gradient следующим обзазом. Пусть необходимо сделать так, чтобы градиент функции $f(x)$ заменялся при обратном проходе на градиент функции $g(x)$. Тогда вместо f(x) используется выражение $g(x) + tf.stop\_gradient(f(x) - g(x))$, значение которого равно f(x) и вычислении градиента дифференцируется только первое слагаемое.

В процессе экспериментов было обучено несколько вариантов архитектур нейронных сетей. Часть из них представлены в таблице 2. Использованы следующие обозначения: Cn — сверточный слой с n фильтрами размера 3x3 , BCn — бинарный сверточный слой с n фильтрами размера 3x3, Dense$n$ — бинарный полносвязный слой с n выходами, ResBin — батч-нормализация + остаточная бинаризация, Bin — батч-

нормализация + обычная бинаризация, BN — батч-нормализация.

A1	A2	A3	SA4	A5	A6
C32	C32	C64	C32	C16	C16
ResBin	ResBin	ResBin	ResBin	ResBin	ResBin
BC64	BC32	BC64	BC64	BC16	BC64
Bin	Bin	Bin	Bin	Bin	Bin
Maxpool	Maxpool	Maxpool	MaxPool	MaxPool	MaxPool
BC64	BC64	BC128	BC256	BC32	BC128
Bin	Bin	Bin	Bin	Bin	Bin
BC128	BC64	BC128	BC256	BC32	BC128
Bin	Bin	Bin	Bin	Bin	Bin
MaxPool	MaxPool	MaxPool	MaxPool	MaxPool	MaxPool
BC128	BC128	BC256	BC256	BC64	BC256
Bin	Bin	Bin	Bin	Bin	Bin
BC256	BC128	BC256	BC256	BC64	BC256
Bin	Bin	Bin	Bin	Bin	Bin
MaxPool	MaxPool	MaxPool	MaxPool	MaxPool	MaxPool
Dense256	Dense256	Dense512	Dense512	Dense128	Dense512
Bin	Bin	Bin	Bin	Bin	Bin
Dense512	Dense128	Dense4	Dense4	Dense128	Dense512
Bin	Bin	BN	BN	Bin	Bin
Dense4	Dense4			Dense4	Dense4
BN	BN			BN	BN

Таблица 2: Архитектуры, обучаемые во время экспериментов

Обучение всех вариантов нейронной сети производилось в течение 200 эпох. В качестве оптимизатора использовался алгоритм Adam [21]. В качестве стратегии изменения learning rate в процессе обучения использовалось экспонетциальное убывание с начальным значением learning rate равным 0.002. Предобработка входных изображений заключалась в центрировании и нормирования значений яркостей в диапазоне от -0.5 до 0.5. В качестве аугментации при обучении использовалось зеркальное отражение входных изображений и зашумение их гауссовским шумом со стандартным отклонением $\sigma = 0.02$.

Для каждого варианта архитектуры обучалось 3 варианта сети, отличающиеся числом уровней остаточной бинаризации M, использова-

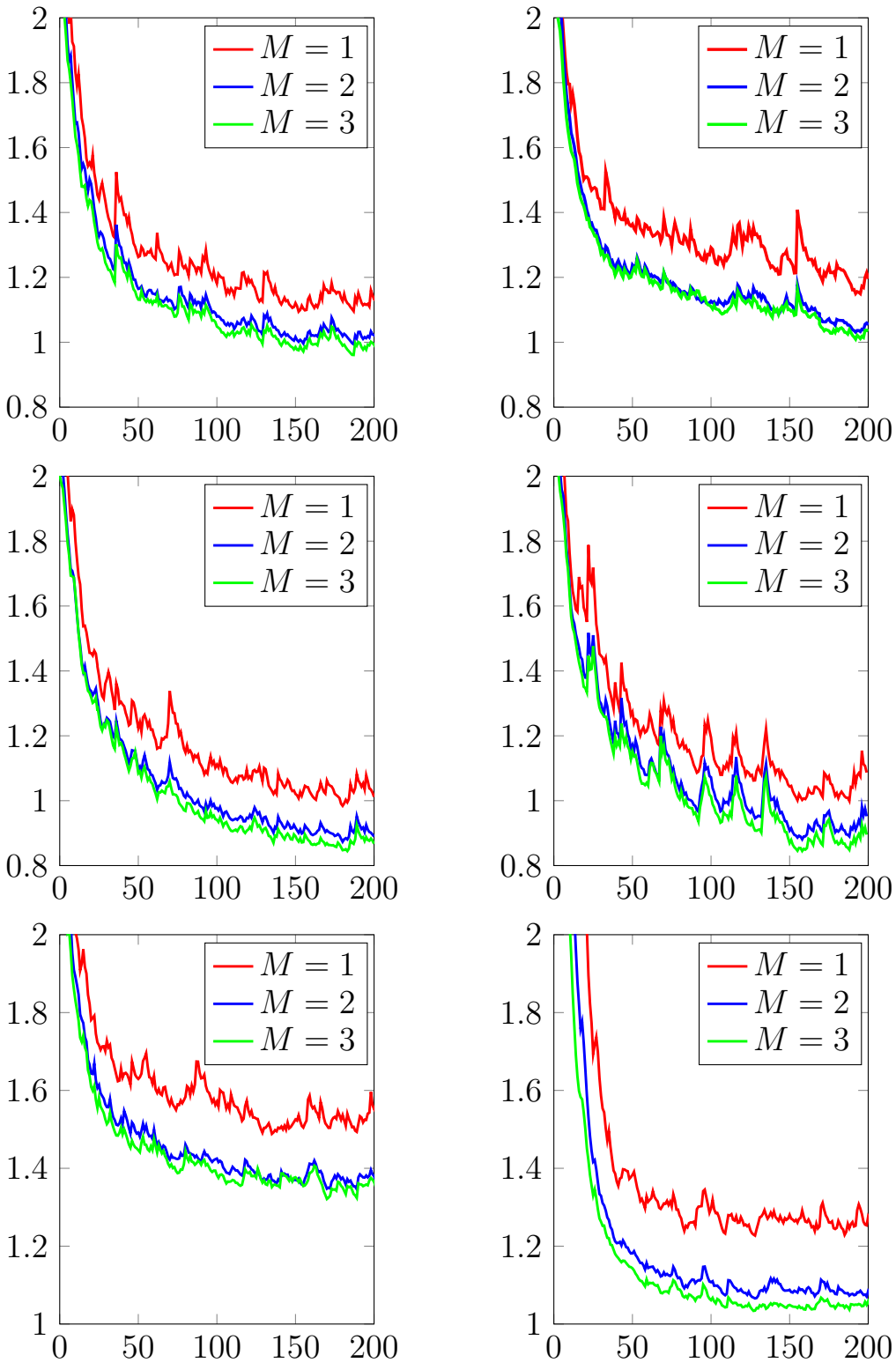

Рис. 4: Графики изменения функции потерь на валидационной выборке

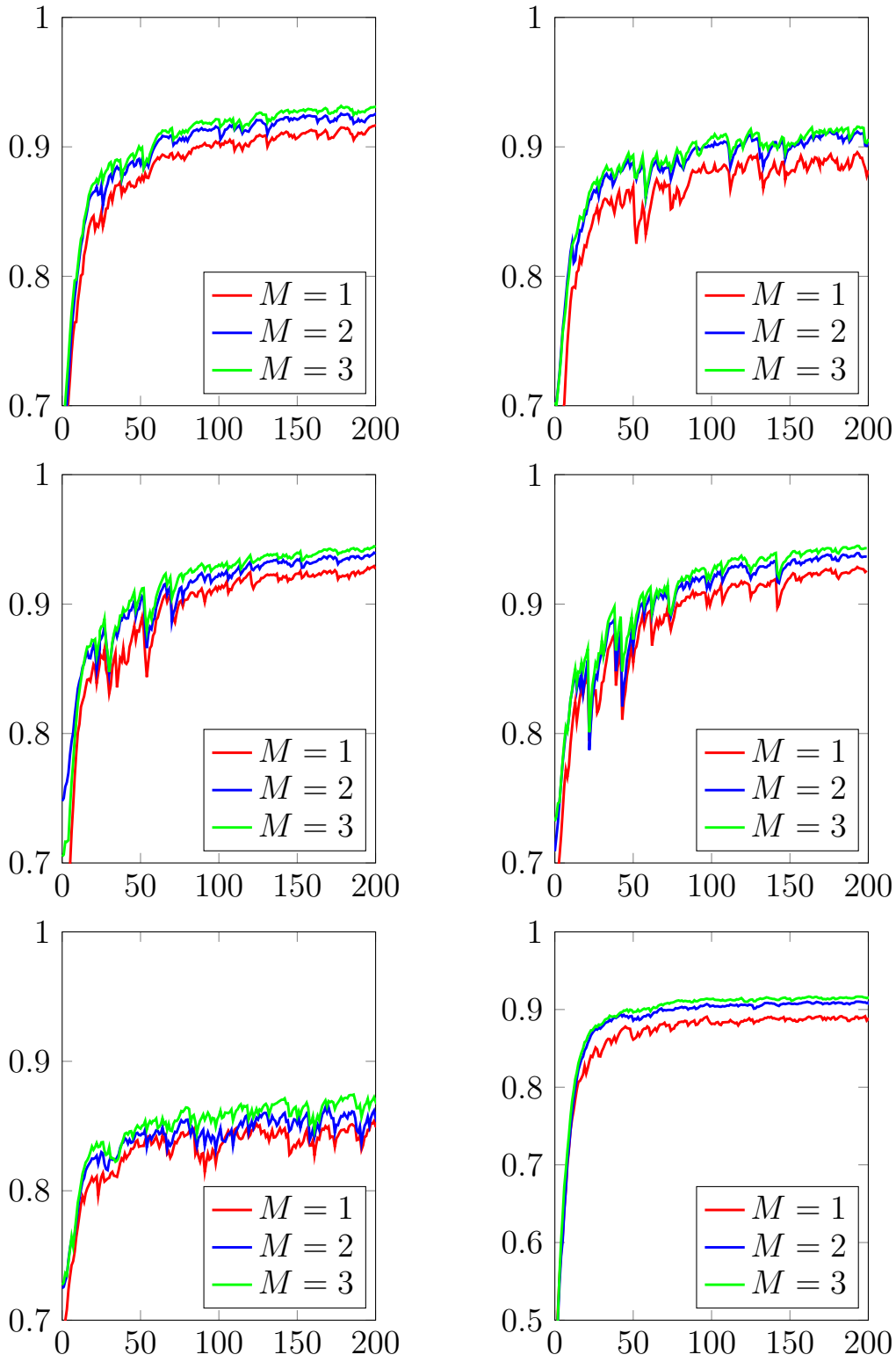

Рис. 5: Графики изменения площади под кривой точность-полнота на валидационной выборке

лись 3 значения: $M = 1$ (обычная бинарная нейронная сеть), $M = 2$ и $M = 3$. Графики кривых функции потерь представлены на рис.4, графики изменения площади под precision-recall кривой представлен на рис. 5. Можно увидеть, что модель с $M = 2$ показывает результаты ощутимо лучше, чем обычная бинарная нейронная сеть. Увеличение параметра M до трех позволяет немного улучшить качество модели, однако прирост уже не такой значительный, как при увеличении M с 1 до 2. Кроме того, усредненние предсказаний сети позволяет уменьшить дисперсию функции потерь, на тех учатках графиков, где при $M = 1$ наблюдаются резкие скачки, скачки для $M = 2$ и $M = 3$ чаще всего заметно меньше.

После обучения лучшая модель каждой архитектуры выбиралась для обучения взвешенного ансамблирования и последующего тестирования. Обучение весов ансабля происходило следующим образом: для всех примеров валидационной и тестовой выборок вычислялись логиты для каждой ветки сети, т. е. выходы сети до превращения их в вероятности с помощью функции softmax. Затем на данных логитах обучалась логистическая регрессия, для которой в качестве обучающей выборки использовались валидационнные логиты, а в качестве тестовой — тестовые.

После получения весов ансамблей, производилось финальное тестирование сети, результаты представлены в таблице 3. Тестировалось не только качество отнесения изображений к правильному классу (с помощью метрики precision-recall auc), но и точность определения того, присутсвует ли объект интереса на изображении, для чего использовалась бинарная версия precision-recall auc, для которой в качетсве вероятности положительного класса использовался $b_p = \max(p_1, p_2)$. Как видно из результатов тестирования, взвешенный ансамбль лучше определяет наличие или отсутствие объекта интереса, чем невзвешенный ансамбль, при этом качество определения класса объекта у взвешенного ансамбля незначительно ухудшилось по сравнению с невзвешенным. Лучшие результаты показали архитектуры 3 и 4 при $M = 3$, однако архитектура 3 имеет меньше обучаемых параметров, поэтому для реализации была

	Величина	A1	A2	A3	A4	A5	A6
M=1	loss	1.05	1.14	0.95	**0.893**	1.468	1.201
	auc binary	0.894	0.842	**0.907**	0.896	0.797	0.853
	auc multiclass	0.921	0.871	0.922	**0.932**	0.857	0.889
M=2	loss	0.92	1.03	**0.81**	0.816	1.278	1.017
	auc binary	0.901	0.875	**0.923**	0.919	0.818	0.874
	auc multiclass	0.929	0.908	0.938	**0.946**	0.884	0.912
M=2, weighted	loss	0.903	0.975	0.824	**0.741**	1.252	0.944
	auc binary	0.901	0.878	0.918	**0.948**	0.801	0.881
	auc multiclass	0.921	0.908	**0.932**	0.928	0.857	0.911
M=3	loss	0.89	1.00	0.782	**0.772**	1.251	0.972
	auc binary	0.916	0.884	**0.932**	0.929	0.832	0.892
	auc multiclass	0.936	0.914	0.943	**0.950**	0.891	0.923
M=3, weighted	loss	0.824	0.947	**0.718**	**0.718**	1.15	0.911
	auc binary	0.917	0.886	**0.931**	**0.931**	0.833	0.892
	auc multiclass	0.931	0.913	**0.951**	**0.951**	0.874	0.917

Таблица 3: Результаты на тестовой выборке

выбрана именно она.

5.4. Сравнение с другими бинарными нейронными сетями

Для сравнения предложенной архитектуры с другими существующими типами бинарных нейронных сетей было проведено обучение сети данной топологии на наборе данных cifar10. Набор представляет цветные изображения размером 32x32 пикселя, принадлежащие одному из 10 классов, собой 50000 изображений составляет обучающая выборка и 10000 изображений предназначены для тестирования. Для валидации модели использовались последние 5000 изображений из обучающей выборки.

Архитектура сети (количество и типы слоев) были взяты такими, какие использовались в экспериментах в работах [16] и [17]. Результаты для сетей FINN и RebNet были взяты из соответствующих статей. Обучение производилось в течение 500 эпох. Обучались сети с разными параметрами M. Помимо сети, используемой для обучения на наборе данных «АМЭ», было обученно также два дополнительных варианта

сети. Первый варинт представляет собой сеть, у которой отличается функция градиента для слоев бинарной активации, вместо функции (1) используется кусочно-линейная функция

$$f'(x) = \begin{cases} 1, x \in [-1, 1] \\ 0, |x| > 1 \end{cases}$$

В другом варианте сети присутствует бинаризация весов первого сверточного слоя.

Сеть	M=1	M=2	M=3
FINN [16]	80.1		
RebNet [17]	80.59	85.94	86.98
Представленая архитектура	84.37	85.84	87.14
взвешенный ансамбль		86.65	87.42
Представленая архитектура , бинарная функция активации из [3]	82.27	85.01	86.12
взвешенный ансамбль		85.62	86.53
Представленая архитектура, бинарные веса первого слоя	80.48	83.26	83.99
взвешенный ансамбль		83.87	84.54

Таблица 4: Доля правильных ответов (accuracy) сетей, обученных на наборе данных cifar10

Таблица 4 данные о проценте правильно классифицированных примеров для обученных сетей. Как видно, использование другой функции для посчета градиента бинаризации заметно влияет на точность распознавания сети. При $M = 1$ и использовании бинаризации первого слоя преложенная архитектура сети совпадает с сетями FINN и ReBNet , поэтому результаты для данных сетей близки друг к другу. Доля правильно соотнесенных объектов для нейронной сети с бинаризованным первым слоем примерно на 2 процента меньше, чем для сети, у которой бинаризация первого слоя отсутсвует.

Таким образом, при использовании бинаризации первого слоя, предсталенная архитектура немного уступает по точности сети ReBNet, однако RebNet использует операции умножения при бинаризации, тогда как представленная архитектура их не использует. В результате сэко-

номленные ресурсы можно направить на вычисление первого сверточного слоя.

6. Реализация поддержки нейронной сети для ПЛИС

Основная вычислительная сложность бинарной нейронной сети заключается в вычислении сверточных и полносвязных слоев. Как уже было сказано выше, скалярное произведение двух битовых векторов длины n может быть вычисленно как:

$$dot(x, w) = 2bitpopcount(Xnor(x, w)) - n.$$

Наличие весовых коэффициентов для карт признаков и ядер свертки принципиально не затрудняет вычисления, так как возможно предпосчитать весовые коэффициенты заранее. Пусть a_i — весовые коэффициенты карт признаков, b_j —- весовые коэффициенты ядер свертки, тогда весовые коэффициенты карт признаков, которые получатся в результате свертки имеют вид $c_j = b_j \sum a_i$.

Слои, выполняющие батч-нормализацию и последующую бинаризацию, производят следующее сравнение:

$$\gamma_i \frac{(\alpha_i x - \mu_i)}{\sigma_i} + \beta_i > 0,$$

где α_i — весовой коэффициент, μ, σ — средние значения и стандартные отклонения, посчитанные во время обучения с помощью скользящего среднего, γ, β — обучаемые параметры батч-нормализации. Объединяя две эти формулы можно записать

$$bitpopcount(Xnor(x, w)) > \frac{1}{2}\left(n + \frac{\mu_i}{\alpha_i} - \frac{\beta_i \sigma_i}{\gamma_i}\right) = thr_i. \qquad (2)$$

Несмотря на то, что вычисленные пороги являются вещественными, их можно округлить, поскольку правая часть неравенства является целочисленным выражением. Поэтому пороги хранятся в виде 16-битных целых чисел.

Таким образом, реализация бинарных слоев нейронной сети главным образом сводится к вычислениям формулы (2). Выполнение оста-

точной бинаризации сводилось к итеративным вычислениям вида

$$b_i^k = r_i^k < thr_i^k$$
$$r_i^{k+1} = |r_i^k - thr_i^k|,$$

где r_i^0 — результат свертки. Данный способ отличается от способа остаточной бинаризации, используемого в [17] тем, что не использует операции умножения.

Полносвязный слой реализован с помощью компоненты умножения битовой матрицы на битовый вектор со сравнением по порогу. Данный модуль получает на вход битовую строку, производит её умножение на битовую матрицу весов и сравнивает результат умножения с пороговым значением, выдавая на выход битовую строку с результатами сравнения. Модуль устроен следующим образом: сначала входные данные считываются с входного потока в локальную память блоками по s бит, затем входные блоки последовательно обрабатываются, путем выполнения параллельных вычислений XnorPopcount операций для p строк матрицы весов одновременно с сохранением промежутого результата в p 16-битных аккумуляторах. Когда входная строка заканчивается, результаты вычислений для p строк сравниваются с соответсвующими пороговыми значениями, и в выходной поток записывается p-битовое число.

Степень параллелизма выполнения матричного умножения достигается за счет варьирования параметров s и p, их увеличение позволяет увеличить пропускную способность модуля, однако приводит также к увеличению количества требуемых для реализации LUT и триггеров. В качестве возможных значении для s использовались 8,16,32,64, поскольку для целых чисел такой длины в пакете intel hls имеются готовые оптимизированные реализации функции подсчета количества бит в числе. Поскольку значение параметра s может не совпадать со значением параметра p предыдущего слоя, была также реализована компонента, объединяющая несколько последовательных слов в одно.

Веса и пороги хранятся в локальной памяти M10K. Так как блок памяти M10K имеет два параллельных порта для чтения или записи,

то для хранения n фильтров (строк матрицы весов) используется $\frac{n}{2}$ банок памяти.

Сверточный слой также реализован с помощью компоненты умножения битовой матрицы на битовый вектор со сравнением по порогу, однако для того, чтобы данную компотенту можно было применить для свертки, входные данные необходимо преобразовать в такую матрицу, что умножение данной матрицы на матрицу весов даст результат эквивалентный свертке входных карт признаков с весовыми ядрами. Для данной операции был реализован модуль скользящего окна, который считывает входные данные с потока и хранит нужное количество строк карты признаков в буфере, и когда буфер оказывает полон, записывает в выходной поток данные из него в нужном порядке. Карты признаков пересылаются через потоки в формате channels last, т.е. после слова, соответствующего координате (x, y, c) идет слово $(x, y, c + 1)$. Это позволяет избежать дополнительных операций для получения результата свертки после умножения.

Слой пуллинга вычисляет максимум в окне заданного размера, однако тот факт, что карты признаков бинарной нейронной сети хранят величины, принимающие значения 0 или 1, позволяет заменить вычисление максимума на операцию «логическое ИЛИ». Выполнение данного слоя производится с помощью двух модулей: модуля скользящего окна и модуля пулинга. Модуль пулинга выполняет агрегирование входных данных с помошью операции «побитовое ИЛИ». Параллельно обрабатываются s каналов, s выбиралось равным параметру p для предыдущего сверточного слоя. Промежуточные результаты вычислений хранились в виде сдвигового регистра из $n = \frac{c}{s}$ s-битных слов, где c — количество карт признаков. При получении на вход нового слова, значения регистра сдвигаются, элемент, который был первым извлекается, выполняется операция побитового «ИЛИ» между ним и сходным словом и результат записывается в конец регистра.

7. Тестирование

Тестрирование нейронной сети производилось на плате DE1–SoC с Cyclone V 5CSEMA5F31C6N. Данные об имеющихся ресурсах устройства представлена в таблице 5.

Логические элементы, тысяч	Триггеры, тысяч	RAM, кб	DSP
85	128	3970	87

Таблица 5: Сведения об имеющихся ресурсах

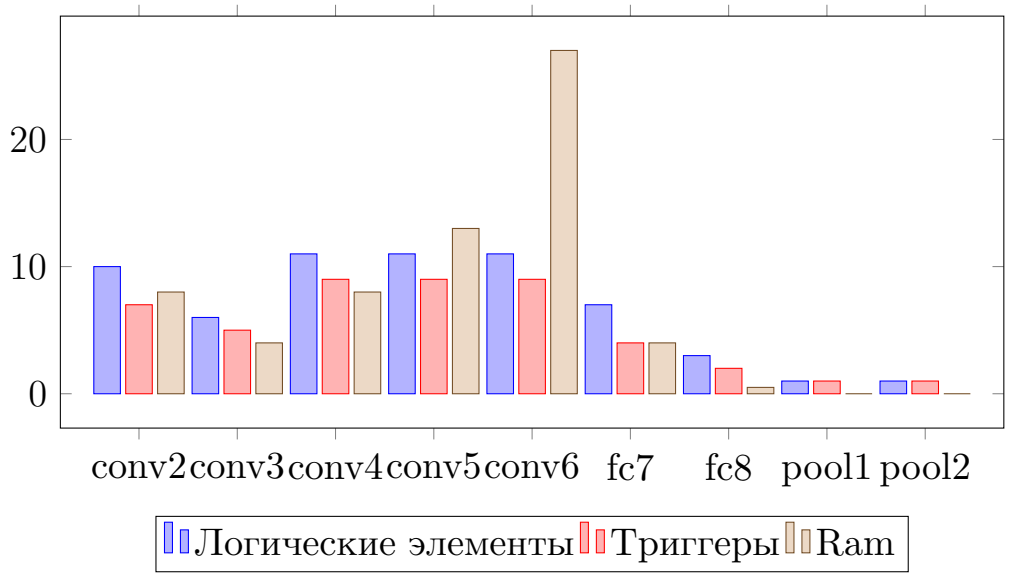

Рис. 6: Количество ресурсов, затраченное на построение модулей

Таблица 6 содержит параметры, которые использовались для распараллеливания вычисления сверточных и полносвязных слоев сети.

Диаграмма 6 содержит информацию о количестве ресурсов, затраченных на реализацию модулей нейронной сети в процентах от общего числа доступных ресурсов.

Для тестирования модулей по отдельности была написанна эталонная реализация вычисления слоев сети на CPU. Для верификации реализованных модулей было проведено следующее испытание: на вход модулю подавались случайно сгенерированные данные и результаты

Слой	s	p
BC64	16	32
BC128	32	16
BC128	32	32
BC256	32	32
BC256	32	32
Dense512	32	16
Dense4	16	2

Таблица 6: Доля правильных ответов (accuracy) сетей, обученных на наборе данных cifar10

выполнения сравнивались с результатами работы эталонной реализации на встроенном в плату процессоре. Модуль разницы между прогнозами сетей составила менее 10^{-5}.

Тестирование производительности всей сети выполнялось на изображениях из тестовой выборки. Среднее время работы составило 0.00112 секунды.

8. Заключение

В рамках данной работы были достигнуты следующие результаты.

- Проведен обзор типов нейронных сетей, используемых в существующих реализациях на ПЛИС, и приведены доводы в пользу использованиия бинарных нейронных сетей.
- Разработана архтектура бинарной нейронной сети, позволяющая увеличить точность распознавания на 0.5 процента, по сравнению с точностью бинарных нейронных сетей с сопоставимой вычислительной сложностью.
- Произведено обучение сети данной архитектуры на данных с датасета «АМЭ» с помощью фреймворка машинного обучения keras.
- Реализован алгоритм прямого прохода бинарной нейронной сети на на ПЛИС с использованием компилятора intel hls.
- Произведено тестирование нейронной сети на наборе данных АМЭ.
- По промежуточным результатам работы представлена к публикуации статья в журнале «Известия ЮФУ ТЕХНИЧЕСКИЕ НАУКИ».

Продолжение данной работы возможно в двух направлениях.

- Использовать подход дистилляции для увеличения точности распознавания нейронной сети.
- Произвести более глубокую оптимизацию модулей поддержки вычислений на ПЛИС.

Список литературы

[1] Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift //arXiv preprint arXiv:1502.03167. 2015.

[2] Courbariaux M., Bengio Y., David J P. Binaryconnect: Training deep neural networks with binary weights during propagations //Advances in neural information processing systems. 2015. P. 3123–3131.

[3] Courbariaux M. et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1 //arXiv preprint arXiv:1602.02830. 2016.

[4] Hinton, Geoffrey. Neural networks for machine learning. Coursera, video lectures, 2012.

[5] Rastegari M. et al. Xnor-net: Imagenet classification using binary convolutional neural networks //European Conference on Computer Vision. Springer, Cham, 2016. P. 525–542.

[6] McDonnell M. D. Training wide residual networks for deployment using a single bit for each weight //arXiv preprint arXiv:1802.08530. 2018.

[7] Lin Z. et al. Neural networks with few multiplications //arXiv preprint arXiv:1510.03009. 2015.

[8] Ott J., Lin Z., Zhang Z., Liu S., Bengio Y. Recurrent neural networks with limited numerical precision //arXiv preprint arXiv:1608.06902. 2016.

[9] Li F., Zhang B., Liu B. Ternary weight networks //arXiv preprint arXiv:1605.04711. 2016.

[10] Kim M., Smaragdis P. Bitwise neural networks.CoRR, abs/1601.06071, 2016.

[11] Alemdar H. et al. Ternary neural networks for resource-efficient AI applications //2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017. P. 2547–2554.

[12] Sung W., Shin S., Hwang K. Resiliency of deep neural networks under quantization.CoRR,abs/1511.06488, 2015.

[13] Cai Z. et al. Deep learning with low precision by half-wave gaussian quantization //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 5918–5926.

[14] Zhou S., Ni Z., Zhou X., Wen H., Wu Y., Zou Y. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR, abs/1606.06160, 2016

[15] OpenVINO toolkit. URL: https://software.intel.com/en-us/openvino-toolkit (дата обращения: 26.05.2019)

[16] Umuroglu Y. et al. Finn: A framework for fast, scalable binarized neural network inference //Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2017. P. 65–74.

[17] Ghasemzadeh M., Samragh M., Koushanfar F. Rebnet: Residual binarized neural network //2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2018. P. 57–64.

[18] Zhu S., Dong X., Su H. Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit? //arXiv preprint arXiv:1806.07550. 2018.

[19] Girija S. S. Tensorflow: Large-scale machine learning on heterogeneous distributed systems // arXiv preprint, 1603.04467, 2016. Software available from tensorflow.org.

[20] Chollet F. et al. Keras. 2015. URL: https://github.com/fchollet/keras (дата обращения: 26.05.2019)

[21] Kingma D., Ba J. Adam: A method for stochastic optimization//ICLR. 2015.