Аппаратная реализация свёрточной нейронной сети с использованием вычислений в системе остаточных классов

Н.И. Червяков [1], П.А. Ляхов [1], Н.Н. Нагорнов [1], М.В. Валуева [1], Г.В. Валуев [1]
[1] Северо-Кавказский федеральный университет,
355009, Россия, г. Ставрополь, ул. Пушкина, д. 1

Аннотация

Современные архитектуры свёрточных нейронных сетей являются весьма ресурсозатратными, что ограничивает возможности их широкого практического применения. В статье предложена архитектура свёрточной нейронной сети, разделённой на аппаратную и программную части для увеличения производительности вычислений. Для реализации свёрточного слоя нейронной сети в аппаратной части использована модулярная арифметика с целью сокращения ресурсозатрат. Предложен численный метод квантования коэффициентов фильтров свёрточного слоя сети для минимизации влияния шума квантования на результат вычислений в системе остаточных классов и определения разрядности коэффициентов. Данный метод основан на масштабировании коэффициентов на фиксированное количество бит и округлении к большему и к меньшему. Используемые операции позволяют уменьшить ресурсы при аппаратной реализации за счёт простоты их выполнения. Все вычисления в свёрточном слое сети выполняются над числами в формате с фиксированной точкой. Программное моделирование с использованием Matlab 2017b показало, что свёрточная нейронная сеть с минимальным количеством слоёв может быть быстро и успешно обучена. Аппаратное моделирование с использованием field-programmable gate array Kintex7 xc7k70tfbg484-2 показало, что использование системы остаточных классов в свёрточном слое нейронной сети позволяет снизить аппаратные затраты на 32,6% по сравнению с традиционным подходом, основанным на двоичной системе счисления. Результаты исследования могут быть применены при создании эффективных систем видеонаблюдения, для распознавания рукописного текста, лиц, объектов и местности.

Ключевые слова: свёрточные нейронные сети, обработка изображений, распознавание образов, система остаточных классов.

Цитирование: **Червяков, Н.И.** Аппаратная реализация свёрточной нейронной сети с использованием вычислений в системе остаточных классов / Н.И. Червяков, П.А. Ляхов, Н.Н. Нагорнов, М.В. Валуева, Г.В. Валуев // Компьютерная оптика. – 2019. – Т. 43, № 5. – С. 857-868. – DOI: 10.18287/2412-6179-2019-43-5-857-868.

Введение

Современное развитие науки и техники подразумевает повсеместное внедрение методов интеллектуального анализа данных. Постоянно расширяется круг задач, требующих использования методов искусственного интеллекта в области обработки изображений: идентификация личности [1], распознавание сцен [2], обработка информации с внешних датчиков в беспилотных наземных и летательных транспортных средствах [3], медицинская диагностика [4] и т.д. Перспективным инструментом для решения задачи распознавания изображений является применение интеллектуальных методов на основе искусственных нейронных сетей [5].

Идея применения искусственных нейронных сетей для обработки визуальной информации предложена в [6] для решения задачи автоматизации распознавания рукописных цифр. Предложенная в указанной статье архитектура получила название сверточной нейронной сети (СНС), и главной её особенностью является объединение свёрточных слоёв, реализующих извлечение визуальных признаков, и многослойного персептрона, реализующего операцию распознавания по данным результатов свёрток. Эволюция данной научной идеи и развитие вычислительной техники привели к тому, что в настоящее время теория СНС и методов её практического применения развивается по пути экстенсивного увеличения количества слоёв СНС, что приводит к высокой вычислительной сложности реализации таких систем. Например, архитектура сети [7], показавшая лучший результат распознавания изображений базы ImageNet в 2010 году, содержит около 650000 нейронов, 60 миллионов настраиваемых параметров и требует 27 гигабайт дискового пространства для обучения. В работе [8] представлена разработка компании Google, которая показала лучший результат распознавания изображений базы ImageNet в 2014 году. Для распознавания изображения данная СНС выполняет более полутора миллиардов вычислительных операций, что мотивировало компанию Google разработать специальный тензорный процессор для оптимизации работы данной СНС [9]. Обобщая сказанное, можно сделать вывод о том, что современные архитектуры СНС являются весьма ресурсозатратными, что ограничивает возможности их широкого практического применения. В настоящее время нет единого подхода к снижению ресурсных затрат при реализации СНС на практике.

Существуют известные библиотеки, которые позволяют автоматизировать процесс проектирования и обучения нейронных сетей. TensorFlow [10] является библиотекой для численных расчётов, но главной це-

лью её создания была помощь в проектировании и обучении моделей машинного обучения [11]. Набор инструментов с открытым исходным кодом под названием LeFlow [12] позволяет разработчику программного обеспечения автоматически преобразовывать численные вычисления Tensorflow в код, написанный на языке описания аппаратуры Verilog с помощью высокоуровневого инструмента синтеза с открытым исходным кодом LegUp. Этот процесс позволяет разработчикам Python создавать прототипы алгоритмов машинного обучения на FPGA, но сгенерированная аппаратная модель будет уступать по производительности модели, оптимизированной вручную [13]. Библиотека Caffe [14] написана на языке C++, но имеет интерфейсы для работы с Matlab и Python. В отличие от Tensorflow, она предназначена только для разработки глубоких нейронных сетей и объединила все современные методы для проектирования СНС. Библиотека FPGA Caffe [15] является версией Caffe с поддержкой FPGA для следующих операций, используемых в СНС: прямая и обратная свёртка, ReLu, max pooling, скалярное произведение векторов. Недостатком представленной библиотеки является поддержка лишь небольшого количества операций [16]. Intel OpenVINO [17] – это набор инструментов для развертывания приложений и решений, которые имитируют человеческое зрение. Этот инструментарий основан на СНС и расширяет рабочие нагрузки компьютерного зрения на аппаратное обеспечение фирмы Intel, оптимизируя производительность. Набор инструментов последней версии программы Intel OpenVINO включает в себя набор инструментов для глубокого обучения Intel Deep Learning Deployment Toolkit. Данный инструментарий поддерживает лишь ограниченное количество FPGA. MATLAB [18] – это высокоуровневый язык и интерактивная среда для программирования, численных расчётов и визуализации результатов. Эта среда обладает мощным инструментом по работе с нейронными сетями. В ней содержится компонент Neural Network Toolbox, который позволяет автоматизировать процесс проектирования и обучения нейронных сетей. В новых версиях продукта также осуществляется поддержка СНС. Кроме того, в Matlab существует возможность конвертирования кода, написанного на встроенном языке программирования, в код на языке описания аппаратуры VHDL, с помощью инструмента HDL Coder. Данная возможность позволяет автоматизировать процесс проектирования FPGA.

Одним из перспективных инструментов для улучшения технических характеристик СНС является система остаточных классов (СОК) [19–23]. В работе [22] предложен метод, использующий фильтры Собеля в свёрточном слое СНС, и его аппаратная FPGA реализация. Авторы показывают увеличение скорости работы устройства и снижение затрат на оборудование в сравнении с реализацией в двоичной системе счисления (ДСС). Недостатком метода, представленного в [22], является фиксация коэффициентов свёрточного слоя, что значительно замедляет время обучения СНС. Чтобы преодолеть недостатки из [22], мы представим в данной статье архитектуру СНС, которая разделена на аппаратную и программную части. В аппаратной части, которая реализует свёрточный слой СНС, предлагается использовать СОК. Мы продемонстрируем эффективность использования предлагаемого подхода с помощью аппаратного моделирования на FPGA Xilinx.

1. Свёрточные нейронные сети

СНС состоит из входного и выходного слоёв, а также из нескольких скрытых слоёв. Скрытые слои СНС состоят из свёрточных слоев, слоёв выборки и полносвязного классификатора, представляющего собой персептрон, обрабатывающий полученные на предыдущих слоях признаки. Рассмотрим часть СНС, отвечающую за выделение признаков, которая состоит из слоя пространственной свёртки и слоя выборки, реализующего процедуру выбора максимальных элементов [6].

Предположим, что на вход СНС поступает изображение I, состоящее из R строк, C столбцов и D цветовых компонент. Например, для изображений в формате RGB $D=3$ и цветовыми компонентами являются уровни красного, зелёного и синего цветов пикселей изображения. В этом случае вход СНС можно описать как трёхмерную функцию $I(x,y,z)$, где $0\le x<R$, $0\le y<C$ и $0\le z<D$ – пространственные координаты, а амплитуда I в любой точке с координатами (x,y,z) – интенсивность пикселей в данной точке. Процедура получения карт признаков показана на рис. 1.

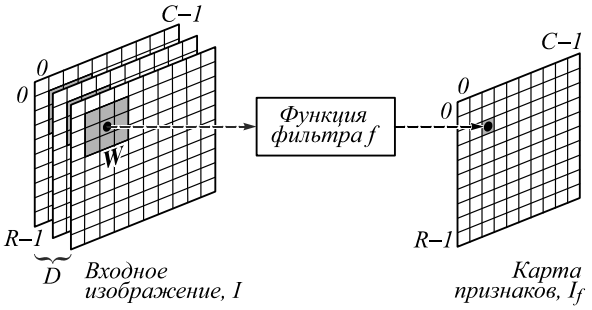


Рис. 1. Процедура получения карты признаков в свёрточном слое СНС

Процедура получения карт признаков в свёрточном слое может быть представлена в виде

$$I_f(x,y)=b+\\ +\sum_{i=-(w-1)/2}^{(w-1)/2}\sum_{j=-(w-1)/2}^{(w-1)/2}\sum_{k=0}^{D-1}W_{i,j,k}I(x+i,y+j,k), \quad (1)$$

где I_f – карта признаков; $W_{i,j,k}$ – коэффициенты 3D-фильтра размера $w\times w$ для обработки D двумерных массивов; b – смещение [22].

СНС обычно использует большое количество фильтров в свёрточном слое. Это приводит к резкому увеличению объёма обрабатываемых данных в сети. Для их уменьшения используется слой выборки,

например, реализующий выборку максимальных элементов в некоторой рассматриваемой окрестности. На рис. 2 схематично показана процедура извлечения максимального элемента с использованием маски фильтра размером $w \times w$ и шагом w. Выход данного слоя передаётся на вход классификатора, который организован как традиционный многослойный персептрон.

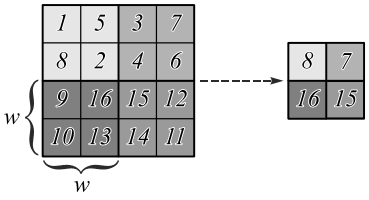


Рис. 2. Принцип работы слоя выборки, реализующего процедуру выбора максимальных элементов

Для улучшения технических характеристик СНС мы предлагаем использовать СОК, краткое описание которой приведено в следующем параграфе.

2. Введение в СОК

В СОК числа представляются в базисе попарно взаимно простых чисел, называемых модулями $\{m_1, ..., m_n\}$, $\text{НОД}(m_i, m_j) = 1$, $i \neq j$. Произведение всех модулей СОК $M = m_1 m_2 ... m_n$ называется динамическим диапазоном системы. Любое целое число может быть единственным образом представлено в СОК с диапазоном M в виде вектора $(x_1, x_2, ..., x_n)$, где $x_i = |X|_{m_i} = X \bmod m_i$ в соответствии с Китайской теоремой об остатках (КТО) [24].

Операции сложения, вычитания и умножения в СОК определяются формулами

$$A \pm B = \left(\left| a_1 \pm b_1 \right|_{m_1}, ..., \left| a_k \pm b_k \right|_{m_n} \right), \quad (2)$$

$$A \cdot B = \left(\left| a_1 \cdot b_1 \right|_{m_1}, ..., \left| a_k \cdot b_k \right|_{m_n} \right). \quad (3)$$

Равенства (2) и (3) показывают параллельную природу СОК, свободную от поразрядных переносов. Основные преимущества представления чисел в СОК [25]:

1. Большие числа кодируются множествами малых остатков, что существенно уменьшает арифметическую сложность вычислений.
2. Независимость остатков числа друг от друга открывает возможность их параллельной обработки, что позволяет значительно ускорить процесс вычислений.
3. Так как остатки числа не зависят друг от друга, ошибка, допущенная в ходе вычислений по одному модулю, не распространяется на другие модули системы. Как следствие, процедуры обнаружения, локализации и коррекции ошибок значительно упрощены.

Основные недостатки представления чисел в СОК [25]:

1. Неэффективное выполнение немодульных операций, таких как деление, извлечение корней, сравнение чисел.
2. Ограниченность действия сферой целых чисел.
3. Отсутствие простых признаков выхода результатов операций за пределы рабочего диапазона системы.

2.1. Выбор модулей СОК

Важной задачей при разработке прикладной системы, использующей вычисления в СОК, является выбор набора модулей $\{m_1, ..., m_n\}$. В большинстве современных работ, посвящённых прикладному применению СОК, используется модуль, равный степени двойки, то есть 2^α, $\alpha \in N$, где N – множество натуральных чисел. Это объясняется тем, что вычисления по модулю 2^α являются наиболее простыми с точки зрения аппаратной реализации, так как они могут быть реализованы в виде обычных арифметических устройств двоичной системы счисления с шириной α бит. Использование модуля 2^α, а также требование попарной взаимной простоты всех модулей СОК приводит к тому, что все остальные модули системы должны быть нечётными, то есть иметь вид $2^\alpha \pm k$, $\alpha \in N$, $k = 1, 3, 5, ...$. Так как в данной статье мы применяем СОК для реализации наиболее ресурсозатратного свёрточного слоя СНС, реализующего вычисления по формуле (1), то наиболее затратными с аппаратной точки зрения элементами арифметического устройства будут модульные сумматоры. Поскольку реализация вычислений по формуле (1) требует суммирования большого количества слагаемых, необходимо использовать многовходовые сумматоры. В настоящее время наиболее эффективные техники суммирования большого количества слагаемых по модулю, аналогичные используемым в традиционной двоичной системе счисления, разработаны лишь для модулей вида $2^\alpha \pm 1$, $\alpha \in N$ [26]. Практическая реализация вычислительного канала по модулю $2^\alpha + 1$, $\alpha \in N$, требует введения дополнительной логики по методу diminished-1 для отслеживания нулевой кодовой комбинации, что является нежелательным явлением при разработке системы с минимальными аппаратными и временными затратами [27]. Таким образом, для проектирования СНС на основе вычислений в СОК целесообразно использовать лишь модули вида $2^\alpha - 1$, $\alpha \in N$, в качестве нечётных модулей системы.

2.2. Преобразование из ДСС в СОК

Рассмотрим набор модулей специального вида $\{2^{p_1}, 2^{p_2}-1, ..., 2^{p_n}-1\}$. Для перевода числа в СОК необходимо вычислить остатки от деления по каждому модулю [24]. Операция вычисления остатка от деления по модулю 2^p осуществляется простой обрезкой p младших бит исходного числа. Для модуля $2^p - 1$ вычисление остатка от деления происходит сложнее. Пусть $X = \overline{X_{g-1} X_{g-2} \dots X_0}$ – g-битное исходное число. Разобьём его на $s = \lceil g/p \rceil$ частей размерности p бит. Для этого дополним X справа 0 до размерности $g' = s \cdot p$, теперь $X' = \overline{X_{g'-1} X_{g'-2} \dots X_0}$. Тогда $Y_0 = \overline{X_{p-1} \dots X_1 X_0}$, $Y_1 = \overline{X_{2p-1} \dots X_{p+1} X_p}$, ..., $Y_s = \overline{X_{g'-1} \dots X_{(s-1)p+1} X_{(s-1)p}}$ – части X'. Представим число X' как

$$X' = Y_0 + Y_1 \cdot 2^p + Y_2 \cdot 2^{2p} + ... + Y_s \cdot 2^{sp}.$$

В случае преобразования с использованием теоретико-числовых свойств мы получаем

$$\left|X'\right|_{2^p-1} = \left|Y_0 + Y_1 + Y_2 + ... + Y_s\right|_{2^p-1}. \quad (4)$$

То есть вычисление остатка от деления по модулю 2^p-1 сводится к сложению p-битных чисел по модулю 2^p-1. Для сложения по модулю 2^p-1 используются сумматоры Когге–Стоуна с циклическим переносом по модулю 2^p-1 (метод, предложенный в [26]).

2.3. Преобразование из СОК в ДСС

Наиболее обобщённый метод для получения эквивалентного взвешенного числа – использование КТО [28]. Вычисление взвешенного числа X из представления в СОК, т.е. $(x_1, x_2, ..., x_n)$, основанном на наборе модулей $\{m_1, m_2, ..., m_n\}$, производится следующим образом:

$$X = \left| \sum_{i=1}^{n} \left| M_i^{-1} \right|_{m_i} M_i x_i \right|_M, \quad (5)$$

где $M_i = M/m_i$ и $|M_i^{-1}|_{m_i}$ – мультипликативный обратный элемент M_i по модулю m_i для $i = 1, 2, ..., n$. Для реализации КТО требуется вычислить остаток от деления на большое число M, реализация этой операции в аппаратном обеспечении приводит к увеличению площади и задержки.

Модифицированная КТО, использующая дробные величины и именуемая приближённой КТО, впервые представлена в работе [29] для определения знака и деления в СОК. Эффективная аппаратная реализация данного подхода основывается на технике сжатия слагаемых и сумматоре Когге–Стоуна по модулю 2^α, представлена в работе [30]. Описанный метод использован для реализации преобразования из СОК в ДСС.

3. Свёртка в СОК

Использование СОК наиболее эффективно в тех случаях, когда вычисления требуют выполнения преимущественно операций сложения и умножения. Формула (1) показывает, что операция свёртки в СОК использует только эти операции. Это означает, что СОК может быть эффективно использована для аппаратной реализации свёрточного слоя СНС. Высокая сложность реализации операции сравнения в СОК затрудняет её применение для реализации слоя выбора максимальных элементов и полносвязного слоя нейронов. Это побуждает предложить подход, заключающийся в разделении СНС на аппаратную и программную части. Предлагается использовать аппаратное устройство для реализации свёрточной части СНС в СОК и использовать программную реализацию для остальных слоёв СНС.

Коэффициенты $W_{i,j,k}$ и сдвиг b из формулы (1) являются константами в обученной СНС. Это означает, что устройство свертки должно реализовывать умножение на константы с последующим сложением результата. Аппаратная реализация вычислений в свёрточном слое СНС по формуле (1) приводит к необходимости квантования коэффициентов $W_{i,j,k}$ фильтра, что влечёт за собой возникновение шума квантования. Смещение b из формулы (1) не оказывает влияния при округлении коэффициентов фильтра, следовательно, оно не связано и с погрешностью, возникающей в результате выполнения свёртки. Пусть фильтр свёрточного слоя имеет размер $w \times w \times 3$, где 3 – количество цветовых компонент. Умножим его коэффициенты на 2^N, где N – параметр масштабирования, и округлим к большему. Результат свёртки в СОК разделим на 2^N и округлим к меньшему. Данный подход обладает следующими преимуществами.

1. Погрешности округлений будут иметь разные знаки и частично компенсировать друг друга. Таким образом, уменьшается влияние шума квантования на результат свёртки.
2. Применение операций округления именно в таком порядке требует меньших ресурсов при аппаратной реализации, чем операции округления к ближайшему целому. Данный факт объясняется тем, что коэффициенты фильтров известны априори и их квантование с округлением вверх может быть выполнено предварительно. Выполнение свёртки осуществляется с использованием арифметико-логических устройств и округление её результата вниз осуществляется простым отбрасыванием дробной части и не требует дополнительных аппаратных и временных затрат.
3. Операции умножения и деления на 2^N в двоичной записи числа соответствуют сдвигу запятой на 2^N знаков вправо или влево соответственно, что существенно упрощает и ускоряет их выполнение.

С увеличением значения параметра масштабирования N уменьшается влияние шума квантования на результат свёртки, но возрастают аппаратные и временные затраты на её выполнение. Возникает вопрос о выборе значения параметра N, эффективного с точки зрения аппаратной реализации на современных устройствах и необходимого для сохранения результатов свёртки приемлемого качества. Качество считается приемлемым, если значение пикового отношения сигнала к шуму (PSNR) достигает 40 дБ. Данное значение описывает разницу между двумя изображениями, незаметную для человеческого глаза [31]. Для ответа на поставленный вопрос оценим влияние шума квантования на результат свёртки [32]. Определим абсолютные погрешности (АП) AE_+ и AE_- округления положительных и отрицательных коэффициентов $W_{i,j,k}$ из формулы (1) соответственно

$$\begin{aligned} AE_+ &= \sum_{W_{i,j,k}>0} \left(\left\lceil 2^N W_{i,j,k} \right\rceil - 2^N W_{i,j,k} \right), \\ AE_- &= \sum_{W_{i,j,k}<0} \left(\left\lceil 2^N W_{i,j,k} \right\rceil - 2^N W_{i,j,k} \right). \end{aligned} \quad (6)$$

Отсюда найдём предельную абсолютную погрешность (ПАП) округления коэффициентов $W_{i,j,k}$ фильтра LAE_1

$$LAE_1 = \max\left\{AE_+, AE_-\right\}. \quad (7)$$

Если через B обозначить максимальное значение яркости изображения, то ПАП результатов свёртки LAE_2 равна

$$LAE_2 = LAE_1 \cdot B. \quad (8)$$

ПАП нормированных результатов свёртки LAE_3 может быть получена из ПАП результатов свёртки LAE_2 после учёта параметра N

$$LAE_3 = \frac{LAE_2}{2^N} = 2^{-N} \cdot B \cdot LAE_1. \quad (9)$$

Если обозначить через $\lambda \in [0,1)$ дробную часть точного результата свёртки, то ПАП округления нормированных результатов свёртки LAE_4 может быть найдена по следующей формуле

$$LAE_4 = LAE_3 + \lambda - \lfloor LAE_3 + \lambda \rfloor. \quad (10)$$

После выполнения операции свёртки точное значение результата редко будет целым числом. Таким образом, значение LAE_4 зависит не только от LAE_3, но и от λ. При $LAE_3 = 0$, LAE_4 в формуле (10) представляет собой неустранимую погрешность. Она коррелирует с LAE_3 и всегда оказывает влияние на точность вычислений исследуемого метода. С увеличением значения параметра N влияние данной погрешности на результат вычислений уменьшается, и на практике по достижении N определённого значения влияние этой погрешности становится пренебрежимо мало.

Итоговая погрешность вычисления свёртки представляет собой ПАП округлённых в меньшую сторону нормированных результатов свёртки LAE_5.

$$LAE_5 = \left|LAE_3 - LAE_4\right|. \quad (11)$$

Формула (11) свидетельствует о том, что погрешность округления к меньшему результатов свёртки LAE_4 частично компенсирует погрешность результатов свёртки LAE_3, основанную на округлении к большему коэффициентов $W_{i,j,k}$ фильтра LAE_1. На определённом этапе увеличения значения N две эти погрешности станут соизмеримы по своей абсолютной величине. На практике на данном этапе результирующая погрешность LAE_5 будет иметь наименьшее значение. При дальнейшем увеличении значения N погрешность LAE_3 начнёт уменьшаться, в отличие от LAE_4, что приведёт к увеличению LAE_5.

Пример. Пусть мы имеем 4 значения яркости пикселей, идущих подряд: 100, 110, 120 и 150. В результате свёртки с одним из фильтров вейвлета Добеши *db*2 с коэффициентами $\widetilde{f_1} \approx -0{,}09151$, $\widetilde{f_2} \approx 0{,}15849$, $\widetilde{f_3} \approx 0{,}59151$, $\widetilde{f_4} \approx 0{,}34151$, мы получим

$$100 \cdot \widetilde{f_4} + 110 \cdot \widetilde{f_3} + 120 \cdot \widetilde{f_2} + 150 \cdot \widetilde{f_1} = 104{,}50962,$$

где $\lambda = 0{,}50962$. Если коэффициенты данного фильтра умножить на 2^6 и округлить к большему, то получим $f_1^* = -5$, $f_2^* = -11$, $f_3^* = 38$, $f_4^* = 22$. В результате свёртки с квантованными коэффициентами имеем $100 \cdot 22 + 110 \cdot 38 + 120 \cdot 11 + 150 \cdot (-5) = 6950$. Разделив данное число на 2^6 и округлив к меньшему, получим $\lfloor 6959/2^6 \rfloor = \lfloor 108{,}59375 \rfloor = 108$. В данном случае погрешность результата свёртки $LAE_3 = 108{,}59735 - 104{,}50962 = 4{,}08413$, погрешность округления результатов свёртки $LAE_4 = 108{,}59735 - 108 = 0{,}59375$, а итоговая погрешность $LAE_5 = 4{,}08413 - 0{,}59375 = 3{,}49038$. Зная, что $LAE_3 = 4{,}08413$ и $\lambda = 0{,}50962$, значения величин LAE_4 и LAE_5 можно определить по формулам (10) и (11) соответственно:

$$LAE_4 = 4{,}08413 + 0{,}50962 - \lfloor 4{,}08413 + 0{,}50962 \rfloor = 0{,}59375;$$

$$LAE_5 = \left|4{,}08413 - 0{,}59375\right| = 3{,}49038.$$

Одним из параметров, влияющих на величину LAE_5, является λ, конкретное значение которого при проведении теоретических расчётов не может быть определено. Его необходимо исключить, учитывая оказываемое им влияние на результат вычислений. Для этого выразим в формуле (11) LAE_4 через LAE_3 и λ согласно формуле (10).

$$LAE_5 = \left|LAE_3 - \left(LAE_3 + \lambda - \lfloor LAE_3 + \lambda \rfloor\right)\right| = \left|\lfloor LAE_3 + \lambda \rfloor - \lambda\right|. \quad (12)$$

Рассмотрим два случая:

1. $\lfloor LAE_3 + \lambda \rfloor - \lambda > 0 \Rightarrow \lfloor LAE_3 + \lambda \rfloor \geq 1$. В этом случае чем больше значение $\lfloor LAE_3 + \lambda \rfloor$, тем больше значение LAE_5. Таким образом, $\lfloor LAE_3 + \lambda \rfloor = \lfloor LAE_3 \rfloor + 1$ и λ представляет собой дополнение дробной части числа LAE_3 до единицы, то есть $\lambda = \lfloor LAE_3 \rfloor + 1 - LAE_3$. Подставив полученное выражение в формулу (12), получим

$$LAE_5 = \lfloor LAE_3 + \lfloor LAE_3 \rfloor + 1 - LAE_3 \rfloor - \left(\lfloor LAE_3 \rfloor + 1 - LAE_3\right) = LAE_3. \quad (13)$$

2. $\lfloor LAE_3 + \lambda \rfloor - \lambda \leq 0 \Rightarrow \lfloor LAE_3 + \lambda \rfloor \leq \lambda \Rightarrow \lfloor LAE_3 + \lambda \rfloor = 0 \Rightarrow LAE_5 = \left|0 - \lambda\right| = \lambda$.

В этом случае чем больше значение λ, тем больше значение LAE_5. При этом

$$\lfloor LAE_3 + \lambda \rfloor = 0 \Rightarrow LAE_3 + \lambda = 1 - \varepsilon,$$

откуда $\lambda = 1 - \varepsilon - LAE_3$. В данных условиях используем AE_3 вместо LAE_3, причём положим её равной нулю

$$LAE_5 = \left|\lfloor 0 + 1 - \varepsilon \rfloor - (1 - \varepsilon)\right| = 1 - \varepsilon, \quad (14)$$

где ε – машинный ноль.

Так как величина LAE_5 представляет собой максимально возможное значение погрешности, то формулу (13) выгодно использовать для $LAE_3 \geq 1$. Таким образом, из формул (13) и (14) можно однозначно определить LAE_5 в зависимости от значений LAE_3 и независимо от конкретных значений λ.

$$LAE_5 = \begin{cases} LAE_3, & LAE_3 \geq 1, \\ 1-\varepsilon, & LAE_3 < 1. \end{cases} \quad (15)$$

Перепишем формулу (15), используя выражение (9).

$$LAE_5 = \begin{cases} 2^{-N} B \cdot LAE_1, & 2^{-N} B \cdot LAE_1 \geq 1, \\ 1-\varepsilon, & 2^{-N} B \cdot LAE_1 < 1. \end{cases} \quad (16)$$

Пиковое отношение сигнала к шуму (PSNR) после выполнения указанных действий может быть найдено по формуле

$$\mathrm{PSNR} = 10\lg\frac{B^2}{LAE_5{}^2} = 20\lg\frac{B}{LAE_5}. \quad (17)$$

Преобразуем формулу (17), используя выражение (16):

$$\mathrm{PSNR} = \begin{cases} 20\lg\dfrac{2^N}{LAE_1}, & 2^N \leq B \cdot LAE_1, \\ 20\lg\dfrac{B}{1-\varepsilon}, & 2^N > B \cdot LAE_1. \end{cases} \quad (18)$$

Формула (18) описывает зависимость PSNR результатов вычисления свёртки по формуле (1) от значения параметра масштабирования N, размера $w \times w \times 3$ фильтра свёрточного слоя и максимального значения яркости изображения B. Практическое применение формулы (18) может быть описано следующим образом: необходимо выбирать наименьшее значения величины N, которое позволяет сохранить приемлемое качество обработанного изображения. Это позволит минимизировать аппаратные затраты на реализацию свёртки. Разрядность r коэффициентов фильтра после квантования определяется по формуле

$$r = \max\{r_{i,j,k}\}, \quad (19)$$

где $r_{i,j,k}$ – количество разрядов, необходимое для хранения квантованного коэффициента $\lceil 2^N W_{i,j,k} \rceil$, определяемое по формуле

$$r_{i,j,k} = 1 + \left\lceil \log_2 \left(\left\| \lceil 2^N W_{i,j,k} \rceil \right\| + 1 \right) \right\rceil. \quad (20)$$

Если нейронная сеть имеет несколько свёрточных слоёв, то ПАП результатов свёртки вычисляется по формуле

$$LAE_{2,d} = \left(\sum_{W_{d,i,j,k}} 2^{N_d} W_{d,i,j,k} + LAE_{1,d} \right) (T_{d-1,,i,j,k} + \\ + LAE_{1,d-1}) - \sum_{W_{d,i,j,k}} 2^{N_d} W_{d,i,j,k} T_{d-1,,i,j,k}, \quad (21)$$

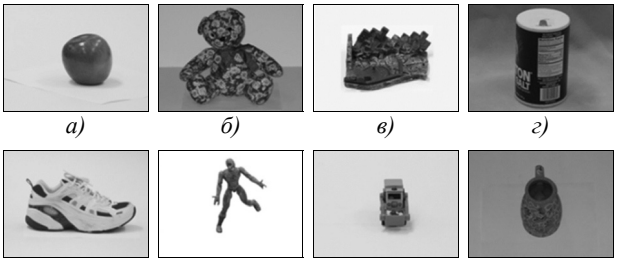
где индекс d обозначает номер свёрточного слоя нейронной сети, а $T_{d-1,i,j,k}$ представляет собой точный результат свёртки в $(d-1)$-м свёрточном слое. В этом случае PSNR вычисляется исходя из выражений (15) и (17) по формуле

$$\mathrm{PSNR} = \begin{cases} 20\lg\dfrac{B_d}{LAE_{3,d}}, & LAE_{3,d} \geq 1, \\ 20\lg\dfrac{B_d}{1-\varepsilon}, & LAE_{3,d} < 1. \end{cases} \quad (22)$$

Далее представлено программное моделирование, в котором используются выведенные формулы. Принцип их работы продемонстрирован на примере конкретного фильтра.

4. Программное моделирование

В качестве экспериментальной базы разработана СНС для распознавания 8 классов изображений из базы изображений Университета Иллинойса [33]. Классы изображений из этого набора данных показаны на рис. 3.



а) б) в) г)

д) е) ж) з)

Рис. 3. Классы изображений базы данных [33]: яблоко (а); медведь (б); камень (в); банка (г); кроссовок (д); человек-паук (е); грузовик (ж); кувшин (з)

Все изображения базы преобразованы к разрешению 256×192 пикселей с использованием алгоритма бикубической интерполяции, встроенного в приложение Adobe Photoshop CS6. Для обучения СНС использовано 161 изображение из базы. Главной целью моделирования являлось сведение к минимуму аппаратных и временных затрат на обучение и реализацию СНС. Для этой цели использовано минимально возможное количество слоёв. Кроме того, в свёрточном слое СНС использована арифметика СОК вместо традиционной двоичной арифметики.

На рис. 4 показана разработанная архитектура СНС. На вход СНС поступает RGB-изображение размера 256×192, два первых слоя отвечают за выделение признаков изображения. Первый слой производит операцию свёртки с использованием 8 фильтров размера 3×3×3 с шагом 3. Результатом вычислений первого слоя являются 8 карт признаков размером 85×64. Второй слой выполняет операцию выбора максимальных элементов с помощью маски размером 2×2 и с шагом 2. 8 карт признаков размером 42×32 являются выходом второго слоя и соединены со входами двух последних слоёв, которые отвечают за классификацию изображения. Третий слой состоит из 10 нейронов, а четвертый слой состоит из 8 нейронов, каждый из которых соответствует определённому классу.

Операция свёртки занимает значительную часть рабочего времени. Для увеличения скорости работы мы разделили архитектуру СНС на аппаратную и программную части. Свёрточный слой реализован аппаратно на FPGA с использованием вычислений в СОК. Поскольку в СОК трудно реализовать операцию сравнения и нелинейную функцию активации, то слой выбора максимальных элементов и полносвязная сеть реализованы в программной части.

Обучение СНС производилось в Matlab R2017b с помощью инструмента Neural Pattern Recognition Toolbox. Вычисления производились на ПК с процессором Intel(R) Core(TM) i7-4790K CPU @ 4.00 ГГц с 16 ГБ оперативной памяти и 64-битной операционной системой Windows 10. Для обучения использовано 161 изображение из 8 классов [33]. Нейронная сеть обучена за 30 итераций в течение 57 секунд.

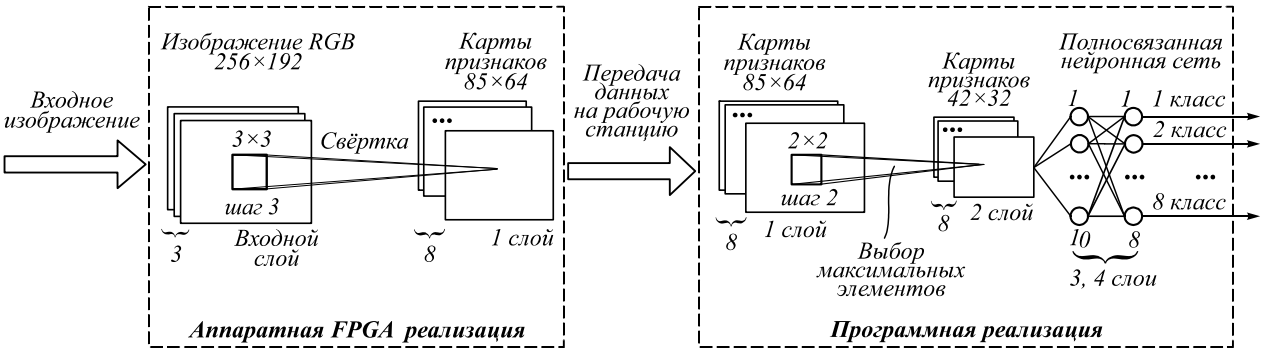


Рис. 4. Предлагаемая архитектура СНС

В программе Matlab R2017b также проведено моделирование для выявления зависимости точности распознавания СНС от параметра N. Результаты моделирования, представленные в табл. 1, подтвердили высокую точность распознавания при использовании теоретически рассчитанного значения N и показали снижение точности распознавания при $N \le 9$.

В табл. 2 показана маска одного из фильтров свёрточного слоя обученной СНС до и после квантования. Для аппаратной реализации значения коэффициентов фильтра приведены к 8-битному представлению. Разрядность $r = 8$ определена по формуле (19) при $N = 11$. Выбранное значение параметра N является наименьшим, при котором результат обработки изображения фильтром размера 3×3×3 гарантированно достигает качества в 40 дБ согласно формуле (18).

Табл. 1. Точность распознавания СНС в зависимости от значений параметра масштабирования N

Параметр масштабирования, N	Точность распознавания СНС, %
32 (в Matlab по умолчанию)	96,97
12	96,97
11	96,97
10	96,97
9	93,94
8	93,94
7	93,94

Далее представлено аппаратное моделирование, в котором используются квантованная маска фильтра из табл. 2.

Табл. 2. Маска одного из фильтров свёрточного слоя обученной СНС

Слой	Маска фильтра до квантования	Маска фильтра после квантования
R	$\begin{pmatrix} -0{,}0394456 & -0{,}0257306 & -0{,}0239846 \\ -0{,}0312350 & -0{,}0403409 & -0{,}0520126 \\ -0{,}0434387 & -0{,}0364724 & -0{,}0520036 \end{pmatrix}$	$\begin{pmatrix} -80 & -52 & -49 \\ -63 & -82 & -106 \\ -88 & -74 & -106 \end{pmatrix}$
G	$\begin{pmatrix} -0{,}0280364 & -0{,}0397294 & -0{,}0528939 \\ -0{,}0423464 & -0{,}0429127 & -0{,}0613428 \\ -0{,}0128505 & -0{,}0548165 & -0{,}0599076 \end{pmatrix}$	$\begin{pmatrix} -57 & -81 & -108 \\ -86 & -87 & -125 \\ -26 & -112 & -122 \end{pmatrix}$
B	$\begin{pmatrix} -0{,}0304336 & -0{,}0289807 & -0{,}0600239 \\ -0{,}0578725 & -0{,}0404073 & -0{,}0482339 \\ -0{,}0492809 & -0{,}0410201 & -0{,}0474128 \end{pmatrix}$	$\begin{pmatrix} -62 & -59 & -122 \\ -118 & -82 & -98 \\ -100 & -84 & -97 \end{pmatrix}$
Смещение	-0,0003320	0

5. Аппаратное моделирование

Аппаратное моделирование произведено на FPGA Kintex7 xc7k70tfbg484-2 в Xilinx Vivado 16.3. Для симуляции использованы параметры моделирования «High Performance Optimized». Цель моделирования – сравнение использования ДСС и СОК.

Результаты аппаратного моделирования представлены только для одного фильтра, но могут быть обобщены на остальные семь, так как имеют одинаковую структуру и отличаются лишь весовыми коэффициентами. Для достижения максимального быстродействия FPGA-реализация фильтров СНС была выполнена в виде комбинационной схемы, без использования памяти. Для реализации операции свёртки использовались параллельно-префиксные сумматоры Когге–Стоуна (*Kogge Stone Adder*) и сумматоры с сохранением переноса (*Carry Save Adder*). Таким образом, могут быть реализованы все восемь фильтров свёрточного слоя рассмотренной СНС. При этом универсальность реализованного подхода позволяет обобщить вывод, полученный для одного фильтра, на

все остальные. Распределение и общее использование аппаратных блоков Slice и LUT Slices при таком подходе выполняется автоматически на основе алгоритмов, встроенных в САПР (в нашем случае Xilinx Vivado 16.3.).

Моделирование операции свёртки произведено для различных модулей вида 2^p и 2^p–1. Результаты представлены в табл. 3 и показывают, что задержка варьируется от 8,805 нс до 17,274 нс.

Табл. 3. Задержка операции свёртки СНС для различных модулей

Модуль	Задержка, нс
2^2–1	8,805
2^3–1	10,474
2^4–1	12,794
2^5–1	13,669
2^6–1	15,187
2^7–1	17,110
2^8–1	15,988
2^9–1	15,085
2^{10}–1	17,274
2^7	13,187
2^8	15,756
2^9	13,755
2^{10}	15,998

Принимая во внимание значения коэффициентов фильтра и необходимость представления отрицательных чисел в СОК, получаем, что динамический диапазон СОК должен удовлетворять условию

$$M \geq 2 \cdot 255 \cdot \max\{0, 2326\} = 1186260, \quad (23)$$

где 255 – максимальное значение яркости изображения; 0 – сумма положительных коэффициентов фильтра; 2326 – модуль суммы его отрицательных коэффициентов. Данное условие, а также данные из табл. 3 позволяют нам выбрать для моделирования СОК, содержащей преобразование из ДСС в СОК, свёртку в СОК и обратное преобразование из СОК в ДСС три набора модулей $\{2^3-1, 2^4-1, 2^5-1, 2^9\}$, $\{2^4-1, 2^9-1, 2^9\}$ и $\{2^5-1, 2^7, 2^9-1\}$, имеющих наименьшую задержку и перекрывающих диапазон M.

Полученные результаты моделирования с использованием СОК и ДСС представлены в табл. 4. Моделирование показало, что использование СОК с набором модулей $\{2^3-1, 2^4-1, 2^5-1, 2^9\}$ позволяет сократить аппаратные затраты на 32,6%, использование набора модулей $\{2^4-1, 2^9-1, 2^9\}$ – на 18,3%, а использование набора модулей $\{2^5-1, 2^7, 2^9-1\}$ – на 25,2% по сравнению с ДСС. Это позволяет сделать вывод, что использование СОК для аппаратной реализации свёрточного слоя СНС более эффективно по площади устройства по сравнению с реализацией в ДСС. Для демонстрации преимуществ предложенного метода проведено сравнение с реализацией функции свёртки одного фильтра СНС, полученной встроенным в Matlab конвертером HDL Coder в код, написанный на языке VHDL. Результаты моделирования показали, что спроектированное на основе известного алгоритма устройство задействует 3303 LUT с задержкой 33,959 нс. Следовательно, предложенный нами подход с использованием СОК $\{2^3-1, 2^4-1, 2^5-1, 2^9\}$ сокращает скорость работы устройства на 27,21% и уменьшает его аппаратные ресурсы на 10,35% по сравнению с автоматически сгенерированным кодом в среде Matlab.

Табл. 4. Результаты аппаратного моделирования свёрточного слоя СНС

Система счисления	Задержка, нс	Занятые LUTs (lookup tables)
СОК $\{2^3-1, 2^4-1, 2^5-1, 2^9\}$	24,717	**2961**
СОК $\{2^4-1, 2^9-1, 2^9\}$	26,245	3587
СОК $\{2^5-1, 2^7, 2^9-1\}$	24,167	3284
ДСС	**18,769**	4392

Программная часть СНС реализована в среде Matlab. Интерфейс передачи данных между FPGA-модулем и «рабочей станцией» – USB 2.0. Время передачи изображения на FPGA-модуль составляет 0,0025 с, а обратная передача полученного результата на «рабочую станцию» занимает 0,0013 с. Данные среднего времени распознавания одного изображения приведены в табл. 5. Таким образом, основываясь на представленных данных о скорости работы системы, можно сделать вывод, что использование предлагаемой аппаратно-программной реализации позволяет сократить среднее время распознавания изображения на 37,06%.

Табл. 5. Среднее время распознавания изображения

Архитектура	Компоненты системы	Время, с
Программная реализация	Слой свёртки	0,0380
	Остальные слои СНС	0,0540
Общее время программной реализации		**0,0920**
Аппаратно-программная реализация	Передача изображения на FPGA	0,0025
	Слой свёртки	0,0001
	Передача результата на «рабочую станцию»	0,0013
	Остальные слои СНС	0,0540
Общее время аппаратно-программной реализации		**0,0579**

Основным выводом по результатам программного и аппаратного моделирования СНС с вычислениями в СОК в свёрточном слое является установление факта снижения ресурсных затрат по сравнению с использованием традиционной ДСС. Разработанная архитектура СНС может найти широкое применение для создания энергоэффективных систем видеонаблюдения, при распознавании рукописного текста, лиц, объектов и местности в различных практических приложениях.

Заключение

В статье представлен метод аппаратной реализации СНС для распознавания образов с использованием вычислений в СОК. Минимизированная конфигурация СНС включает свёрточный слой, слой выбора максимальных элементов и классификатор, который

организован как традиционный многослойный персептрон. Аппаратное моделирование операции свертки показало, что использование предложенного метода, основанного на СОК с набором модулей специального вида, позволяет сократить аппаратные затраты на 32,6% по сравнению с реализацией ДСС. Обобщение полученных результатов на случай больших масок фильтров требует дальнейших исследований на практике. Результаты исследования могут быть применены при создании эффективных систем видеонаблюдения, для распознавания рукописного текста, лиц, объектов и местности.

Благодарности

Работа выполнена при финансовой поддержке базовой части государственного задания (№2.6035.2017/БЧ), Российского фонда фундаментальных исследований (проекты №18-07-00109 А, №19-07-00130 А и №18-37-20059 мол-а-вед), Совета по грантам Президента Российской Федерации (проект СП-2245.2018.5).

Литература

1. **Chen, Y.** Deep and low-level feature based attribute learning for person re-identification / Y. Chen, S. Duffner, A. Stoian, J.-Y. Dufour, A. Baskurta // Image and Vision Computing. – 2018. – Vol. 79. – P. 25-34.
2. **Cheng, X.** Scene recognition with objectness / X. Cheng, J. Lu, J. Feng, B. Yuan, J. Zhou // Pattern Recognition. – 2018. – Vol. 74. – P. 474-487.
3. **Sarikan, S.S.** Automated vehicle classification with image processing and computational intelligence / S.S. Sarikan, A.M. Ozbayoglu, O. Zilcia // Procedia Computer Science. – 2017. – Vol. 114. – P. 515-522.
4. **Qayyum, A.** Medical image retrieval using deep convolutional neural network / A. Qayyum, S.M. Anwar, M. Awais, M. Majid // Neurocomputing. – 2017. – Vol. 266. – P. 8-20.
5. **Zhang, J.** Small sample image recognition using improved convolutional neural network / J. Zhang, K. Shao, X. Luo // Journal of Visual Communication and Image Representation. – 2018. – Vol. 55. – P. 640-647.
6. **LeCun, Y.** Gradient-based learning applied to document recognition / Y. LeCun, L. Bottou, Y. Bengio, P. Haffiner // Proceedings of the IEEE. – 1998. – Vol. 86, Issue 11. – P. 2278-2324.
7. **Krizhevsky, A.** ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G.E. Hinton // Advances in Neural Information Processing Systems. – 2012. – Vol. 25, Issue 2. – P. 1097-1105.
8. **Szegedy, C.** Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – P. 1-9.
9. **Jouppi, N.** Motivation for and evaluation of the first tensor processing unit / N. Jouppi, C. Young, N. Patil, D. Patterson // IEEE Micro. – 2018. – Vol. 38, Issue 3. – P. 10-19.
10. TensorFlow. An end-to-end open source machine learning platform [Electronical Resource]. – URL: https://www.tensorflow.org/ (request date 19.04.2019).
11. **Aghdam, H.H.** Guide to convolutional neural networks: A practical application to traffic-sign detection and classification / H.H. Aghdam, E.J. Heravi. – Springer International Publishing, 2017. – 282 p.
12. danielholanda/LeFlow: Enabling flexible FPGA high-level synthesis of tensorflow deep neural networks [Electronical Resource]. – URL: https://github.com/danielholanda/LeFlow (request date 19.04.2019).
13. **Noronha, D.H.** LeFlow: Enabling flexible FPGA high-level synthesis of tensorflow deep neural networks / D.H. Noronha, B. Salehpour, S.J.E. Wilton // 2018 Fifth International Workshop on FPGAs for Software Programmers (FSP Workshop). – 2018. – P. 1-8.
14. Cafee. Deep learning framework [Electronical Resource]. – URL: https://caffe.berkeleyvision.org/ (request date 19.04.2019).
15. dicecco1/fpga_caffe [Electronical Resource]. – URL: https://github.com/dicecco1/fpga_caffe (request date 19.04.2019).
16. **DiCecco, R.** Caffeinated FPGAs: FPGA framework for convolutional neural networks / R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor, S. Areibi // 2016 International Conference on Field-Programmable Technology (FPT). – 2016. – P. 265-268.
17. Install Intel® distribution of OpenVINO™ toolkit for Linux with FPGA support [Electronical Resource]. URL: https://docs.openvinotoolkit.org/2019_R1/_docs_install_guides_installing_openvino_linux_fpga.html (request date 19.04.2019).
18. MATLAB [Электронный ресурс]. URL: https://matlab.ru/products/matlab (дата обращения 19.04.2019).
19. **Nakahara, H.** A deep convolutional neural network based on nested residue number system / H. Nakahara, T. Sasao // 2015 25th International Conference on Field Programmable Logic and Applications (FPL). – 2015. – P. 1-6.
20. **Nakahara, H.** A high-speed low-power deep neural network on an FPGA based on the Nested RNS: Applied to an object detector / H. Nakahara, T. Sasao // 2018 IEEE International Symposium on Circuits and Systems (ISCAS). – 2018. – P. 1-5.
21. **Manabe, T.** FPGA implementation of a real-time super-resolution system with a CNN based on a residue number system / T. Manabe, Y. Shibata, K. Oguri // 2017 International Conference on Field Programmable Technology (ICFPT). – 2017. – P. 299-300.
22. **Chervyakov, N.I.** Increasing of convolutional neural network performance using residue number system / N.I. Chervyakov, P.A. Lyakhov, M.V. Valueva // International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – 2017. – P. 135-140.
23. **Чернов, В.М.** Тернарные системы счисления в конечных полях / В.М. Чернов // Компьютерная оптика. – 2018. – Т. 42, № 4. – С. 704-711. – DOI: 10.18287/2412-6179-2018-42-4-704-711.
24. **Omondi, A.** Residue number systems: Theory and implementation / A. Omondi, B. Premkumar. – London: Imperial College Press, 2007. – 296 p.
25. **Cardarilli, G.C.** Residue number system for low-power DSP applications / G.C. Cardarilli, A. Nannarelli, M. Re // 41st Asilomar Conference on Signals, Systems, and Computers. – 2007. – P. 1412-1416.
26. **Vergos, H.T.** On modulo 2^n+1 adder design / H.T. Vergos, G. Dimitrakopoulos // IEEE Transactions on Computers. – 2012. – Vol. 61, Issue 2. – P. 173-186.
27. **Živaljević, D.** Digital filter implementation based on the RNS with diminished-1 encoded channel / D. Živaljević, N. Stamenković, V. Stojanović // 2012 35th International

Conference on Telecommunications and Signal Processing (TSP). – 2012. – P. 662-666.
28. **Chervyakov, N.I.** Residue-to binary conversion for general moduli sets based on approximate Chinese remainder theorem / N.I. Chervyakov, A.S. Molahosseini, P.A. Lyakhov, M.G. Babenko, M.A. Deryabin // International Journal of Computer Mathematics. – 2017. – Vol. 94, Issue 9. – P. 1833-1849.
29. **Hung, C.Y.** An approximate sign detection method for residue numbers and its application to RNS division / C.Y. Hung, B. Parhami // Computers and Mathematics with Applications. – 1994. – Vol. 27, Issue 4. – P. 23-25.
30. **Matos, R.** Efficient implementation of modular multiplication by constants applied to RNS reverse converters / R. de Matos, R. Paludo, N. Chervyakov, P.A. Lyakhov, H. Pettenghi // 2017 IEEE International Symposium on Circuits and Systems (ISCAS). – 2017. – P. 1-4.
31. **Rao, K.R.** The transform and data compression handbook / K.R. Rao, P.C. Yip. – London, New York: CRC Press, 2001. – 399 p.
32. **Chervyakov, N.I.** Quantization noise of multilevel discrete wavelet transform filters in image processing / N.I. Chervyakov, P.A. Lyakhov, N.N. Nagornov // Optoelectronics, Instrumentation and Data Processing. – 2018. – Vol. 54, Issue 6. – P. 608-616.
33. **Rothganger, F.** Object recognition database / F. Rothganger, S. Lazebnik, C. Schmid, J. Ponce // [Electronic resource] – URL: http://www-cvr.ai.uiuc.edu/ponce_grp/data/objects (request date 19.04.2019).
34. **Chervyakov, N.I.** Effect of RNS dynamic range on grayscale images filtering / N.I. Chervyakov, P.A. Lyakhov, D.I. Kalita, K.S. Shulzhenko // 2016 XV International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY). – 2016. – P. 33-37.

Сведения об авторах

Червяков Николай Иванович, 1935 года рождения, в 1965 году окончил Харьковское высшее командно-инженерное училище. В 1972 г. защитил кандидатскую диссертацию, защитил докторскую диссертацию в 1987 г. Получил звание доцента в 1974 г., звание профессора – в 1989 г. Заведующий кафедрой прикладной математики и математического моделирования ИМЕН ФГАОУ ВО «Северо-Кавказский федеральный университет». Область научных интересов: цифровая обработка сигналов и изображений, искусственный интеллект, модулярная арифметика, цифровые устройства, высокопроизводительные вычисления, криптография. E-mail: *k-fmf-primath@stavsu.ru* .

Ляхов Павел Алексеевич, 1988 года рождения, в 2009 году окончил Ставропольский государственный университет по специальности «Математика», доцент кафедры прикладной математики и математического моделирования ИМЕН ФГАОУ ВО «Северо-Кавказский федеральный университет», кандидат физико-математических наук. Область научных интересов: цифровая обработка сигналов и изображений, искусственный интеллект, модулярная арифметика, цифровые устройства, высокопроизводительные вычисления.
E-mail: *ljahov@mail.ru* .

Нагорнов Николай Николаевич, 1992 года рождения. Аспирант ИМЕН ФГАОУ ВО «Северо-Кавказский федеральный университет». Область научных интересов: цифровая обработка изображений, искусственный интеллект, модулярная арифметика. E-mail: *sparta1392@mail.ru* .

Валуева Мария Васильевна, 1993 года рождения. Аспирант ИМЕН ФГАОУ ВО «Северо-Кавказский федеральный университет». Область научных интересов: цифровая обработка изображений, искусственный интеллект, модулярная арифметика, цифровые устройства. E-mail: *mriya.valueva@mail.ru* .

Валуев Георгий Вячеславович, 1992 года рождения. Аспирант ИМЕН ФГАОУ ВО «Северо-Кавказский федеральный университет». Область научных интересов: цифровая обработка изображений, искусственный интеллект, высокопроизводительные вычисления. E-mail: *elasgreece92@mail.ru* .

ГРНТИ: 28.23.15
Поступила в редакцию 2 марта 2019 г. Окончательный вариант – 19 апреля 2019 г.

Hardware implementation of a convolutional neural network using calculations in the residue number system

N.I. Chervyakov [1], P.A. Lyakhov [1], N.N. Nagornov [1], M.V. Valueva [1], G.V. Valuev [1]
[1] North-Caucasus Federal University, 355009, Russia, Stavropol, Pushkin street 1

Abstract

Modern convolutional neural networks architectures are very resource intensive which limits the possibilities for their wide practical application. We propose a convolutional neural network architecture in which the neural network is divided into hardware and software parts to increase performance and reduce the cost of implementation resources. We also propose to use the residue number system in the hardware part to implement the convolutional layer of the neural network for resource costs reducing. A numerical method for quantizing the filters coefficients of a convolu-

tional network layer is proposed to minimize the influence of quantization noise on the calculation result in the residue number system and determine the bit-width of the filters coefficients. This method is based on scaling the coefficients by a fixed number of bits and rounding up and down. The operations used make it possible to reduce resources in hardware implementation due to the simplifying of their execution. All calculations in the convolutional layer are performed on numbers in a fixed-point format. Software simulations using Matlab 2017b showed that convolutional neural network with a minimum number of layers can be quickly and successfully trained. Hardware implementation using the field-programmable gate array Kintex7 xc7k70tfbg484-2 showed that the use of residue number system in the convolutional layer of the neural network reduces the hardware costs by 32.6% compared with the traditional approach based on the two's complement representation. The research results can be applied to create effective video surveillance systems, for recognizing handwriting, individuals, objects and terrain.

Keywords: convolutional neural networks, image processing, pattern recognition, residue number system.

Citation: Chervyakov NI, Lyakhov PA, Nagornov NN, Valueva MV, Valuev GV. Hardware implementation of a convolutional neural network using calculations in the residue number system. Computer Optics 2019; 43(5): 857-868. DOI: 10.18287/2412-6179-2019-43-5-857-868.

Acknowledgements: This work was supported by the Government of the Russian Federation (State order No. 2.6035.2017/BCh), the Russian Foundation for Basic Research (Projects No. 18-07-00109 A, No. 19-07-00130 A and No. 18-37-20059 mol-a-ved), and by the Presidential Grant of the Russian Federation (Project No. SP-2245.2018.5).

References

[1] Chen Y, Duffner S, Stoian A, Dufour J-Y, Baskurta A. Deep and low-level feature based attribute learning for person re-identification. Image Vis Comput 2018; 79: 25-34.

[2] Cheng X, Lu J, Feng J, Yuan B, Zhou J. Scene recognition with objectness. Pattern Recogn 2018; 74: 474-487.

[3] Sarikan SS, Ozbayoglu AM, Zilcia O. Automated vehicle classification with image processing and computational intelligence. Procedia Computer Science 2017; 114: 515-522.

[4] Qayyum A, Anwar SM, Awais M, Majid M. Medical image retrieval using deep convolutional neural network. Neurocomputing 2017; 266: 8-20.

[5] Zhang J, Shao K, Luo X. Small sample image recognition using improved convolutional neural network. Journal of Visual Communication and Image Representation 2018; 55: 640-647.

[6] LeCun Y, Bottou L, Bengio Y, Haffiner P. Gradient-based learning applied to document recognition. Proc IEEE 1998; 86(11): 2278-2324.

[7] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 2012; 25(2): 1097-1105.

[8] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015: 1-9.

[9] Jouppi N, Young C, Patil N, Patterson D. Motivation for and evaluation of the first tensor processing unit. IEEE Micro 2018; 38(3): 10-19.

[10] TensorFlow. An end-to-end open source machine learning platform. Source: ⟨https://www.tensorflow.org/⟩.

[11] Aghdam HH, Heravi EJ. Guide to convolutional neural networks: A practical application to traffic-sign detection and classification. Springer International Publishing; 2017.

[12] danielholanda/LeFlow: Enabling flexible FPGA high-level synthesis of tensorflow deep neural networks. Source: ⟨https://github.com/danielholanda/LeFlow⟩.

[13] Noronha DH, Salehpour B, Wilton SJE. LeFlow: Enabling flexible FPGA high-level synthesis of tensorflow deep neural networks. Fifth International Workshop on FPGAs for Software Programmers (FSP Workshop) 2018; 1-8.

[14] Cafee. Deep learning framework. Source: ⟨https://caffe.berkeleyvision.org/⟩.

[15] dicecco1/fpga_caffe. Source: ⟨https://github.com/dicecco1/fpga_caffe⟩.

[16] DiCecco R, Lacey G, Vasiljevic J, Chow P, Taylor G, Areibi S. Caffeinated FPGAs: FPGA framework For Convolutional Neural Networks. International Conference on Field-Programmable Technology (FPT) 2016; 265-268.

[17] Install Intel® Distribution of OpenVINO™ toolkit for Linux with FPGA support – OpenVINO Toolkit. Source: ⟨https://docs.openvinotoolkit.org/2019_R1/_docs_install_guides_installing_openvino_linux_fpga.html⟩.

[18] MATLAB [In Russian]. Source: ⟨https://matlab.ru/products/matlab⟩.

[19] Nakahara H, Sasao T. A deep convolutional neural network based on nested residue number system. 25th International Conference on Field Programmable Logic and Applications (FPL) 2015: 1-6.

[20] Nakahara H, Sasao T. A high-speed low-power deep neural network on an FPGA based on the Nested RNS: Applied to an object detector. IEEE International Symposium on Circuits and Systems (ISCAS) 2018: 1-5.

[21] Manabe T, Shibata Y, Oguri K. FPGA implementation of a real-time super-resolution system with a CNN based on a residue number system. International Conference on Field Programmable Technology (ICFPT) 2017: 299-300.

[22] Chervyakov NI, Lyakhov PA, Valueva MV. Increasing of convolutional neural network performance using residue number system. International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) 2017: 135-140.

[23] Chernov VM. Ternary number systems in finite fields [In Russian]. Computer Optics 2018; 42(4): 704-711. DOI: 10.18287/2412-6179-2018-42-4-704-711.

[24] Omondi A, Premkumar B. Residue number systems: Theory and implementation. London: Imperial College Press; 2007.

[25] Cardarilli GC, Nannarelli A, Re M. Residue number system for low-power DSP applications. 41st Asilomar Conference on Signals, Systems, and Computers 2007: 1412-1416.
[26] Vergos HT, Dimitrakopoulos G. On modulo 2^n+1 adder design. IEEE Transactions on Computers 2012; 61(2): 173-186.
[27] Živaljević D, Stamenković N, Stojanović V. Digital filter implementation based on the RNS with diminished-1 encoded channel. 35th International Conference on Telecommunications and Signal Processing (TSP) 2012: 662-666.
[28] Chervyakov NI, Molahosseini AS, Lyakhov PA, Babenko MG, Deryabin MA. Residue-to binary conversion for general moduli sets based on approximate Chinese remainder theorem. International journal of computer mathematics 2017; 94(9): 1833–1849.
[29] Hung CY, Parhami B. An approximate sign detection method for residue numbers and its application to RNS division. Computers and Mathematics with Applications 1994; 27(4): 23-25.
[30] Matos R, Paludo R, Chervyakov N, Lyakhov PA, Pettenghi H. Efficient implementation of modular multiplication by constants applied to RNS reverse converters. IEEE International Symposium on Circuits and Systems (ISCAS) 2017; 1-4.
[31] Rao KR, Yip PC. The transform and data compression handbook. London, New York: CRC Press; 2001.
[32] Chervyakov NI, Lyakhov PA, Nagornov NN. Quantization noise of multilevel discrete wavelet transform filters in image processing. Optoelectronics, Instrumentation and Data Processing 2018; 54(6): 608-616.
[33] Rothganger F, Lazebnik S, Schmid C, Ponce J. Object recognition database. Source: ⟨http://www-cvr.ai.uiuc.edu/ponce_grp/data/objects⟩.
[34] Chervyakov NI, Lyakhov PA, Kalita DI, Shulzhenko KS. Effect of RNS dynamic range on grayscale images filtering. XV International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY) 2016; 33-37.

Author's information

Nikolay Ivanovich Chervyakov (b. 1935) graduated from Kharkov Higher Command and Engineering School in 1965. Since 1972 is the Candidates of Technical Sciences, 1987 – the Doctor of Technical Sciences. Received the title of associate professor in 1974, the title of professor - in 1989. Head of Applied Mathematics and Mathematical Modeling department of North-Caucasus Federal University, Doctor of Technical Sciences. Research interests are digital signal and image processing, artificial intelligence, residue number system, digital circuits, high-performance computing, cryptography. E-mail: *k-fmf-primath@stavsu.ru* .

Pavel Alekseyevich Lyakhov (b. 1988) graduated from Stavropol State University, specialty "Mathematics" in 2009. Assistant Professor of Applied Mathematics and Mathematical Modeling department of North-Caucasus Federal University. Research interests are digital signal and image processing, artificial intelligence, residue number system, digital circuits, high-performance computing. E-mail: *ljahov@mail.ru* .

Nikolay Nikolaevich Nagornov (b. 1992) PhD student of Applied Mathematics and Mathematical Modeling department of North-Caucasus Federal University. Research interests are digital image processing, artificial intelligence, computer arithmetic and digital circuits. E-mail: *sparta1392@mail.ru* .

Maria Vasilyevna Valueva (b. 1993) PhD student of Applied Mathematics and Mathematical Modeling department of North-Caucasus Federal University. Research interests are digital image processing, artificial intelligence, computer arithmetic and digital circuits. E-mail: *mriya.valueva@mail.ru* .

Georgiy Vyacheslavovich Valuev (b. 1992) PhD student of Applied Mathematics and Mathematical Modeling department of North-Caucasus Federal University. Research interests are digital image processing, artificial intelligence, computer arithmetic and digital circuits. E-mail: *elasgreece92@mail.ru* .

Received March 2, 2019. The final version – April 19, 2019.