УДК 004.052.32 DOI: 10.31114/2078-7707-2022-1-27-34

Аппаратная реализация нейронной сети для обнаружения объектов в базисе ПЛИС

А.Л. Стемпковский, Р.А. Соловьев, Д.В. Тельпухов, А.Г. Кустов

Институт проблем проектирования в микроэлектронике РАН, г. Москва, ippm@ippm.ru

***Аннотация* — В статье предложена методика по переносу архитектуры современной нейронной сети CenterNet в ПЛИС. Сеть CenterNet является одноэтапным (OneStage) детектором и используется для обнаружения и локализации объектов на изображениях. Данная нейронная сеть имеет хорошие показатели по точности работы и одновременно отличается простотой декодера. В результате исследований удалось достичь очень высокой скорости работы нейронной сети на аппаратном уровне за счёт выбора подходящего для аппаратной реализации энкодера, эффективной аппаратной реализации, как декодера так и последнего слоя с фильтрацией объектов и перехода к вычислениям с фиксированной точкой. При этом качество полученных предсказаний остается высоким.**

***Ключевые слова* — Аппаратная реализация нейронной сети, Программируемые Логические Интегральные Схемы, Арифметика с фиксированной точкой, Двухмерная свертка, Обнаружение объектов.**

I. Введение

Задача по обнаружению объектов в кадре (object detection) является ключевой в современной робототехнике и автономных автомобилях [1]. Эта задача ставится как поиск объектов на изображении и их локализация с помощью прямоугольника, который задан координатами углов и подписан типом объекта из заранее предопределенного списка (например, «человек», «стул», «машина» и.т.д.). На изображении может быть несколько объектов одного типа или объекты могут отсутствовать.

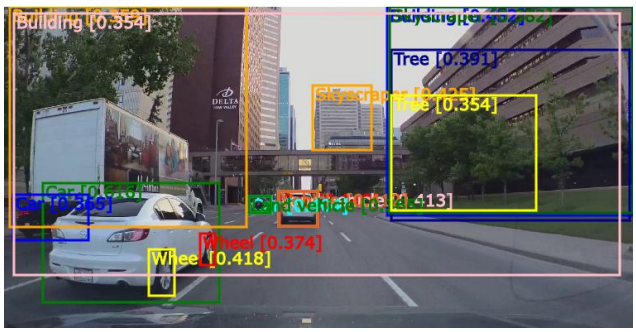

Рис. 1. Задача Object Detection

Для решения этой задачи создано большое число различных нейронных сетей и других подходов. Существует два наиболее распространенных мета-подхода к обнаружению объектов с помощью нейронных сетей: two-shot / two-stage (двухэтапные) и one-shot / one-stage (одноэтапные) детекторы.

Модель двухэтапного обнаружения состоит из двух этапов: предложение региона (region proposal), а затем классификация этих регионов и уточнение прогноза местоположения. Одноэтапное обнаружение пропускает этап поиска региона и сразу дает окончательную локализацию и прогнозирование содержимого. Варианты нейронной сети Faster-RCNN [2] являются популярным вариантом двухэтапного подхода, в то время как single-shot multibox detector (SSD) [3] и YOLO [4] являются популярными представителями одноэтапного подхода.

Главное отличие этих двух подходов заключается в том, что двухэтапные детекторы обычно являются более точными, но работают существенно медленнее. Поэтому для реализации на аппаратном уровне обычно выбирают одноэтапные детекторы [5]. Реализация на аппаратном уровне крайне важна для задач, которые требуют автономной работы устройств (камеры видеонаблюдения, автономное вождение, смартфоны и.т.д.) с поддержкой методов искусственного интеллекта [20, 21].

В последние годы появилось много вариаций одноэтапных детекторов, которые увеличивают скорость работы и при этом достигают очень высокой точности работы. К таким сетям относятся CornerNet [6] и появившаяся вслед за ней CenterNet [7].

II. Описание нейронной сети CenterNet

Нейронная сеть CenterNet была предложена в 2019 году группой ученых из Китая [7]. Она является одноэтапным (OneStage) детектором [3, 4]. Каждый обнаруженный объект на изображении представляется его центром, шириной и высотой.

Сама сеть состоит из энкодера (или по-другому backbone) и декодера. В качестве энкодера может выступать одна из стандартных архитектур для классификации изображений, в частности, в исходной версии CenterNet была использована модель Hourglass-104 [7]. Данная сеть является очень большой, и вследствие этого, оказывается неприменимой для реализации в аппаратуре. В настоящей работе в качестве энкодера была реализована модель MobileNet [8]. В данной модели были удалены слои для классификации и финальный слой пулинга (GlobalAveragePooling).

Декодер имеет структуру, показанную на рисунке 2. В отличие от исходной реализации, активация RELU была заменена на активацию RELU6, что позволило ограничить сверху возможные промежуточные значения на слоях. Это было сделано для удобства последующей квантизации после процесса тренировки.

Так же CenterNet в декодере имеет слой, которого нет в MobileNet - Conv2DTranspose. Этот слой увеличивает размер карт признаков в два раза после каждого применения.

Конец декодера имеет разветвление на 3 ветви. Первая ветвь – это так называемая “тепловая карта”, или Heatmap, предсказывает, есть ли в данной точке объект. Вторая ветвь (название WH) предсказывает размер объекта в данной точке по вертикали и горизонтали (то есть размеры прямоугольника). Третья ветвь (название Reg) предсказывает поправку к центру объекта в пикселях. Это связано с тем, что центры объектов предсказываются на плоскости в 4 раза меньше, чем исходное изображение, и центр объекта может быть смещен на несколько пикселей. Например, если размер входного изображения равен 128 на 128 пикселей, то размер Heatmap с центрами объектов будет 32 на 32 пикселя.

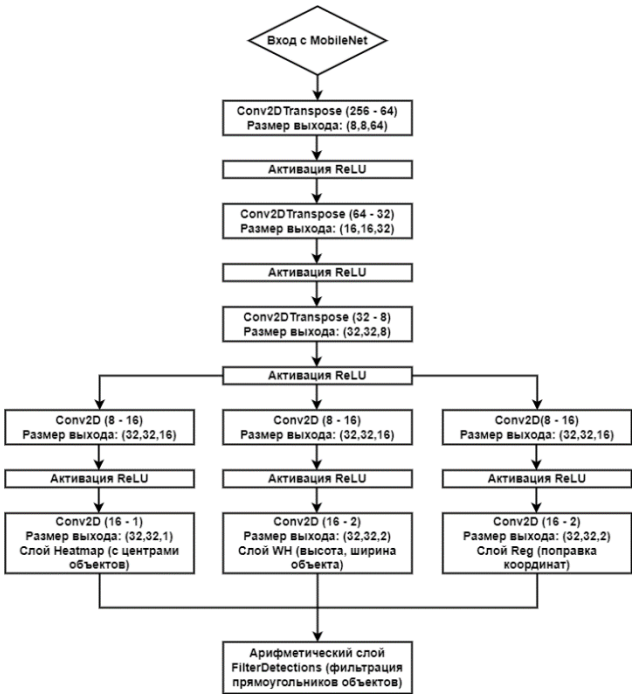

Рис. 2. Декодер CenterNet

На основании предсказанных плоскостей с Heatmap, WH и Reg требуется выполнить поиск прямоугольников с объектами. Этим занимается последний арифметический блок Filter Detection. В аппаратной реализации это отдельный аппаратный блок, структура которого будет описана далее.

III. Тренировка CenterNet

Тренировка CenterNet может быть проведена на любом фреймворке высокого уровня, например, TensorFlow или PyTorch. В данной работе была выбрана задача поиска объектов одного класса, а именно людей в кадре. Использовался датасет Open Images Dataset (OID) от Google [9], который содержит изображения, размеченные на 500 классов. Для решения задачи был выбран один класс, который относится к людям «Person». Аналогично можно использовать любой другой класс или сразу набор классов из OID без существенного усложнения декодера. Как альтернативу OID для обучения нейронной сети можно использовать более компактный датасет COCO [10].

В рамках данной работы было натренировано 2 варианта сети: первый небольшой вариант CenterMobileSmall для отладки маршрута и второй CenterMobileLarge для использования в реальном маршруте. Для CenterNet требуется определиться с количеством фильтров в декодере – чем больше, тем лучше точность, но больше вычислительные затраты. Для CenterMobileSmall было выбрано количество 64 фильтра декодера равным, а для CenterMobileLarge – 128. В качестве энкодера в маленькой сети использовался MobileNet v1 с параметром alpha = 0.25, а в большой сети alpha = 1.0 (то есть число фильтров на слоях энкодера было в 4 раза больше).

Входное изображение имеет размер 128 на 128 пикселей. Так же надо определиться какое максимальное число объектов требуется получить в качестве ответа. Поскольку входное изображение довольно небольшое, было установлено ограничение в десять объектов. Число обнаруживаемых объектов влияет на сложность слоя Filter Detections.

В результате тренировки на CenterMobileSmall удалось достичь значения mean average precision (mAP) [11] на валидации равным 0.3668. На CenterMobileLarge mAP составил 0.4741.

IV. Квантизация весов

После проведения тренировки и получения весов в виде чисел с плавающей точкой (float32) для увеличения скорости работы необходимо преобразовать веса с помощью целочисленного квантования «Post-Training Integer Quantization» [12]. В ходе такого преобразования веса представляются малобитными (обычно 8-битными) числами с фиксированной точкой. Для проведения такого преобразования можно использовать встроенные средства Tensorflow или PyTorch, которые сводят все вычисления к 8-ми битам, но могут значительно снизить точность работы. Второй вариант заключается в использовании своего адаптивного квантизатора, который позволяет найти размерность весов и промежуточных активаций, при которых точность работы нейронной сети остаётся в заданных пределах.

Квантизация позволяет избавиться от дорогостоящих и сложных для реализации операций с плавающей точкой и применять стандартные умножители и сумматоры для целых чисел. Дополнительно появляется возможность увеличить

число параллельно работающих свёрточных блоков, за счёт освободившейся площади на кристалле, что кратно увеличивает производительность.

В данной работе был использован подход, связанный с выбором размерности весов, смещений и активаций таким образом, чтобы точность классификации падала не более чем на 0.5% по сравнению с эталонной моделью с плавающей точкой [13, 15, 16]. Исследования показали, что для сохранения заданной точности хватило 11-12 бит, в зависимости от типа решаемой задачи.

V. Сверточные блоки

Основная вычислительная операция в свёрточных нейронных сетях, в том числе и в CenterNet, это операция 2D-свёртки, которая может различаться размером ядра. Наиболее распространенные варианты — это ядро 3x3 и ядро 1x1. В CenterNet присутствуют только эти два варианта. Базовая формула свёртки с ядром 3x3 для расчета одного пикселя следующего слоя приведена ниже:

$$res = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_4 * w_4 + x_5 * w_5 + x_6 * w_6 + x_7 * w_7 + x_8 * w_8 + x_9 * w_9,$$

где x – пиксель картинки, w – весовой коэффициент.

Наглядно свёртка представлена на рисунке 3. Считается она для каждой выходной карты признаков много раз на базе плавающего окна. Операция содержит только сложения и умножения, и может выполняться параллельно. При этом скорость выполнения свёртки для всего слоя может быть кратно увеличена лишь за счёт увеличения количества доступных аппаратных свёрточных блоков. Поскольку все свёртки во всей сети имеют одинаковую формулу, то для вычислений на каждом слое используются одни и те же аппаратные блоки. Поток вычислений контролирует управляющее устройство, отправляя данные в память каждого из доступных аппаратных блоков.

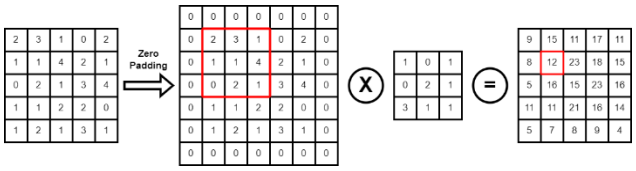

Рис. 3. Пример вычисления операции свёртки

VI. Слой Conv2DTranspose

Слой *Conv2DTranspose* отличается от стандартных свёрточных блоков, которые присутствуют в сети MobileNet. Данный сверточный слой работает по принципу, показанному на рисунке 4. Красным выделены входное и выходное изображения, во входной матрице выделен набор весовых коэффициентов, в выходной – получаемые выходные пиксели. Данный слой используется для двукратного увеличения карты признаков на входе данного слоя.

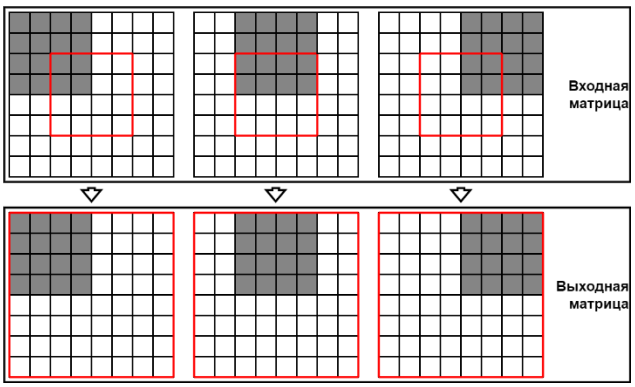

Рис. 4. Схема работы слоя Conv2DTranspose для stride = 2

По входному изображению проходит окно весовых коэффициентов с шагом, заданным в переменной stride. Пиксели карт признаков перемножаются с попавшими на них весовыми коэффициентами, и результаты таких умножений сохраняются в новую область памяти. Если по какому-то адресу уже был сохранен результат, то старое значение результата суммируется с новым. По достижении окном весов конца входных карт признаков, заканчивается расчет на данном слое.

VII. Арифметический слой Filter Detections

Рассмотрим случай сети CenterNet на базе Mobilenet_v1 с размером входа 128 пикселей, одним классом объектов и ограничением не более 10 объектов (людей) на одном изображении. Также установлено дополнительное ограничение на пороговую вероятность ScoreTHR = 0.1. Все объекты с меньшей вероятностью исключаются из рассмотрения.

В этом случай на вход слоя Filter Detection приходит 3 матрицы следующих размерностей:

1) HeatMap (32x32) – вероятность (score, confidence), что в данной точке есть объект.

2) Reg (2x32x32) – поправка положения бокса по x и y (поэтому плоскостей две)

3) WH (2x32x32) – ширина и высота бокса в данной точке (поэтому плоскостей две)

На первом этапе алгоритм получает 10 точек, в которых вероятности максимальны. На данном этапе используется только матрица HeatMap (32x32).

Шаг 1: Сначала ко всем значениям матрицы HeatMap применяется функция $sigmoid(x) = \frac{1}{(1+e^{-x})}$. Следует отметить, что функция *sigmoid* - монотонная и при аппаратной реализации этот шаг может быть пропущен, однако в этом случае необходимо поменяется второй шаг, а также пересчитать ScoreTHR с учетом того, что операция *sigmoid* не применялась. Для этого необходимо решить уравнение:

$$sigmoid(x) = ScoreTHR.$$

Откуда:

$ScoreTHRNew = -\ln((1 - \frac{ScoreTHR}{ScoreTHR})$.

Для ScoreTHR = 0.1 ScoreTHRNew = -2.197224

Шаг 2: К полученной матрице применяется операция прореживания. По всей матрице проходится окно (3x3) и находится максимум в этом окне и сохраняется во вторую матрицу такого же размера HMax. Затем HMax сравнивается с изначальной матрицей HeatMap: в местах, где значения не совпадают, ставятся нули в стандартном случае. Если мы не применяли операцию сигмоида, то ставятся самые малые доступные значения (например -10). Смысл этой операции заключается в прореживании очень близких боксов, которые, скорее всего, относятся к одному и тому же объекту.

Шаг 3: В полученной матрице ищутся позиции десяти наибольших значений, которые превосходят ScoreTHR:

$$xs = [10, 20, 14, \dots, 3]$$

$$ys = [12, 8, 4, \dots, 13]$$

$$score = [0.3, 0.2, 0.15, \dots, 0.0]$$

Шаг 4: Далее необходимо пройти по всем 10 полученным значениям и использовать Reg и WH массивы, чтобы получить координаты 10 боксов по следующим формулам:

$$real\_pos\_x1 = xs + Reg[0][xs, ys] - (\frac{WH[0][xs,ys]}{2});$$

$$real\_pos\_x2 = xs + Reg[0][xs, ys] + (\frac{WH[0][xs,ys]}{2});$$

$$real\_pos\_y1 = ys + Reg[1][xs, ys] - (\frac{WH[1][xs,ys]}{2});$$

$$real\_pos\_y2 = ys + Reg[1][xs, ys] + (\frac{WH[1][xs,ys]}{2}).$$

По результатам первого этапа мы получим набор из 10 боксов претендентов, заданных координатами x1, y1, x2 и y2 и набором их весов в массиве score.

Шаг 5: На следующем шаге необходимо объединить предсказания, которые очень похожи, то есть сильно пересекаются и имеют метрику Intersection over Union (IoU) > 0.5. Этот алгоритм называется Non maximum suspression (NMS) [14].

В этом алгоритме требуется пройти по всем предсказанным боксам, отсортированным по значению score. Для каждого предсказания мы ищем значение IoU с каждым из боксов ниже заданного в списке. Эта метрика вычисляется следующим образом:

```
def bb_intersection_over_union(A, B) -> float:
    xA = max(A[0], B[0])
    yA = max(A[1], B[1])
    xB = min(A[2], B[2])
    yB = min(A[3], B[3])
    # Считаем площадь прямоугольника пересечения
    interArea = max(0, xB - xA) * max(0, yB - yA)
    if interArea == 0:
        return 0.0
    # считаем площадь обоих прямоугольников
    boxAArea = (A[2] - A[0]) * (A[3] - A[1])
    boxBArea = (B[2] - B[0]) * (B[3] - B[1])
    # считаем саму метрику
    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou
```

Если IoU > 0.5 то бокс с меньшим значением *score* исключается из рассмотрения. При аппаратной реализации необходимо избавиться от сложной операции деления в формуле, заменив его на проверку условия:

$$(boxAArea + boxBArea - interArea) * IoU > interArea.$$

Все оставшиеся на этом этапе боксы и есть ответ алгоритма, который можно выводить на экран.

Шаг 6: Поскольку ответ будет получен в рамках размера сети (в нашем случае 128x128), то необходимо пересчитать координаты для вывода на экран, домножив их на соответствующие коэффициенты.

VIII. Результаты различных реализаций

Были получены аппаратные реализации CenterNet для отладочной платформы OpenVINO Toolkit [22] с различным числом сверточных блоков и альфа (коэффициент числа фильтров), результаты которых представлены в табл. 1.

Таблица 1

Результаты аппаратных реализаций

Итог сборки	Выполнено	Выполнено	Выполнено	Не прошло fitter	Выполнено
Число сверточных блоков (для простых/ транспонированных слоёв)	4/1	8/8	8/8	8/8	32/8
Альфа	0.25	0.25	1	1	1
Размер входного изображения (пиксель)	128	128	128	224	128
Точность	0.3663	0.3663	0.4684	0.5227	0.4684
Полный проект в OpenVino					
Logic utilization (in ALMs)	13,517 / 113,560 (12 %)	32,392 / 113,560 (29 %)	37,942 / 113,560 (33 %)	110,550 / 113,560 (97 %)	43,663 / 113,560 (38 %)
Объем занятой памяти (бит)	3,257,312 / 12,492,800 (26 %)	3,716,064 / 12,492,800 (30 %)	6,796,256 / 12,492,800 (54 %)	15,261,696 / 12,492,800 (122 %)	9,352,160 / 12,492,800 (75 %)
DSP блоки	191 / 342 (56 %)	342 / 342 (100 %)	342 / 342 (100 %)	342 / 342 (100 %)	342 / 342 (100 %)

FPS (кадры/с)	8,3	26,6	1,86	-	4,09
CenterNet					
Logic utilization (in ALMs)	10,326 / 113,560 (9 %)	29,739 / 113,560 (26 %)	37,309 / 113,560 (33 %)	104,953 / 113,560 (92 %)	41,942 / 113,560 (37 %)
Объем занятой памяти (бит)	1,613,824 / 12,492,800 (13 %)	2,072,576 / 12,492,800 (17 %)	5,152,768 / 12,492,800 (41 %)	12,812,288 / 12,492,800 (103 %)	7,708,672 / 12,492,800 (62 %)
DSP блоки	165 / 342 (48 %)	342 / 342 (100 %)	342 / 342 (100 %)	342 / 342 (100 %)	342 / 342 (100 %)
Задержка по критическому пути	33.655	34.124	35.181	-	44.809

Из полученных результатов видно, что для большой точности распознавания объектов и большой скорости работы нейросети необходимы значительные ресурсы ПЛИС, которых не всегда хватает, и решить это удаётся с помощью аппаратных оптимизаций.

IX. Обнаружение объектов на видео

VOD работает по принципу объединения (усреднения значений) результирующих матриц CenterNet последних трёх кадров в один набор. Только после этого выполняется слой filterdetections.

В сравнении с исходной реализацией, добавились новые участки памяти для хранения результирующих матриц последних трёх кадров, т.е. три участка размером Sx5x32x32, где S – число бит, необходимое на одно значение матрицы. Данные в этих участках упаковываются по любому отдельно взятому адресу согласно рисунку 5.

Рис. 5. Структура памяти результирующих матриц

Данный подход позволяет избавиться от «мерцания» окна результата поиска объектов на видео при нечёткости кадра, а также отсеять некоторую часть ошибок нейросети.

Сравнение результатов аппаратных реализаций показывает, что при небольшом увеличении затрачиваемых аппаратных ресурсов получаем более точный результат (табл. 2).

Таблица 2

Результаты компиляций нейросетей с и без VOD

	без VOD	с VOD
Logic utilization (in ALMs)	71771 (63%)	71860 (63%)
Объем занятой памяти (бит)	4479249 (36%)	4755729 (38%)
DSP блоки	280 (82%)	285 (83%)

X. Применение MAC-Unit

В Centernet применяются сверточные слои разного типа, и оптимальным минимумом умножителей для нашей аппаратной реализации было выбрано число 16, по размеру самой большой весовой матрицы в сверточном транспонированном слое – 4x4. Сравниваемые далее аппаратные реализации используют данное число умножителей.

Для улучшения проекта было предложено использование умножителя с накоплением (MAC unit) (рис. 6), который представляет собой блок, в котором выполняется операция умножения двух входных значений с последующим суммированием с аккумулятором выходного значения. Применение таких блоков, в сравнении с блоками умножителей размерностью 3x1, 3x3 или более, даёт больше гибкости в аппаратной реализации, например, при реализации сверточных слоёв с матрицами размерностью 1x1.

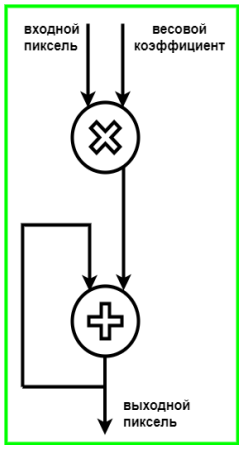

Рис. 6. Схематичное представление блока MAC-Unit

В прежней аппаратной реализации использовались блоки умножителей 3x1, так как это позволяло в сверточном слое 3x3 и depthwise эффективно использовать входные данные без их перезагрузки из памяти. На рисунке 7 представлен механизм применения такого блока умножителей, штрихованной областью отмечены данные, сохранившиеся с прошлой свертки.

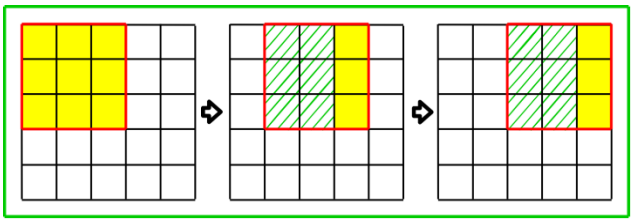

Рис. 7. Схематичное представление работы свертки с буферизацией данных

Но такой блок умножителей оказался не эффективен в применении к сверточным слоям 1x1, так как в реализованной Centernet результат такого слоя получается из суммы произведений, число которых кратно 8, и в результате одно умножение становится не нагруженным из-за применения блока умножителей 3x1. Из-за этого возможна обработка только 8 входных матриц, как показано на рис. 8.

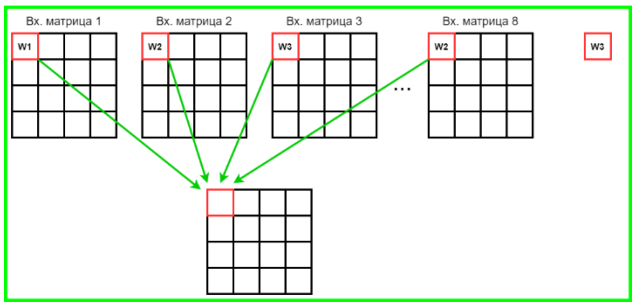

Рис. 8. Схематичное представление использования блока умножителей 3x1 для сверточных слоев 1x1

Применение MAC unit-ов позволяет более грамотно использовать ресурсы. В новой версии аппаратной реализации, построенной на этих блоках, используются все умножители и обрабатываются сразу 16 входных матриц.

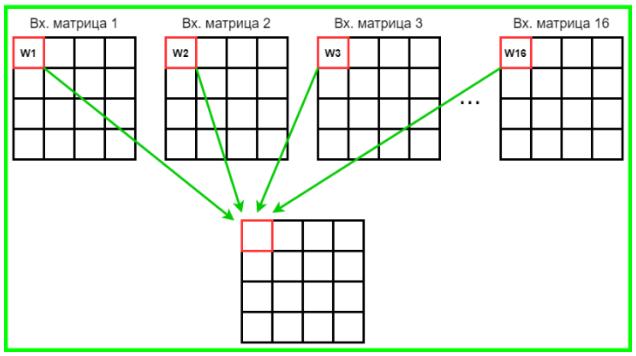

Рис. 9. Схематичное представление использования MAC-Unit блоков для сверточных слоев 1x1

В итоге была получена аппаратная реализация, которая имела значительный прирост скорости работы при незначительном увеличении аппаратный ресурсов. Результаты сравнения аппаратных реализаций представлены в табл. 3.

Таблица 3

Сравнение результатов компиляций нейросетей на базе 16 MAC-Unit и 4x4 сверточного модуля

	16 MAC-Unit	4x4 сверточный модуль
Полный проект в OpenVino		
Logic utilization (in ALMs)	15,179 / 113,560 (13%)	13,517 / 113,560 (12%)
Total block memory bits	3,009,504 / 12,492,800 (24%)	3,257,312 / 12,492,800 (26%)
Total DSP Blocks	188 / 342 (55%)	191 / 342 (56%)
FPS	18.32	8.3
CenterNet		
Logic utilization (in ALMs)	11,932 / 113,560 (11%)	10,326 / 113,560 (9%)
Total block memory bits	1,536,000 / 12,492,800 (12%)	1,613,824 / 12,492,800 (13%)
Total DSP Blocks	162 / 342 (47%)	165 / 342 (48%)
Delay (critical path)	32.472	33.655

XI. Заключение

Нейронная сеть CenterNet показала хорошие результаты в задаче обнаружения объектов в аппаратных реализациях, полученных на базе платформы OpenVino с ПЛИС Xilinx. Она показала высокое быстродействие и качество обнаружения различных классов объектов в видеопотоке. Независимость от ЭВМ даёт полученному устройству мобильность применения, надёжность от вредоносного вмешательства, а также стабильную скорость работы. При этом конструкция полученного устройства позволяет загружать любые веса нейронной сети и, таким образом, менять классы обнаруживаемых объектов, что позволяет применять устройство в различных сферах и менять их по необходимости. Полученные реализации открывают возможность использования нейронных сетей в виде специализированных интегральных схем (ASIC) для создания ещё более быстродействующего и дешевого устройства, выполняющего функцию обнаружения любого типа объектов, под который были натренированы загружаемые веса.

Проект выполнен при финансовой поддержке гранта РНФ № 22-29-00762.

Литература

[1] Szegedy C., Toshev A., Erhan D. Deep neural networks for object detection //Advances in neural information processing systems, 2013, с. 2553-2561.
[2] Ren S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks // Advances in neural information processing systems, 2015, с. 91-99.
[3] Liu W. et al. SSD: Single shot multibox detector // European conference on computer vision, Springer, Cham, 2016, с. 21-37.
[4] Redmon J., Farhadi A. Yolov3: An incremental improvement // arXiv preprint arXiv:1804.02767, 2018, URL: https://arxiv.org/abs/1804.02767 (дата обращения: 22.04.2022).
[5] URL: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (дата обращения: 22.04.2022).
[6] Law H., Deng J. Cornernet: Detecting objects as paired keypoints // Proceedings of the European Conference on Computer Vision (ECCV), 2018, с. 734-750.
[7] Duan K. et al. Centernet: Keypoint triplets for object detection // Proceedings of the IEEE International Conference on Computer Vision, 2019, с. 6569-6578.
[8] Howard A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications // arXiv preprint arXiv:1704.04861, 2017, URL: https://arxiv.org/abs/1704.04861 (дата обращения: 22.04.2022).
[9] Kuznetsova A. et al. The open images dataset v4 // International Journal of Computer Vision, 2020, с. 1-26.
[10] Lin T. Y. et al. Microsoft coco: Common objects in context // European conference on computer vision, Springer, Cham, 2014, с. 740-755.
[11] Everingham M. et al. The pascal visual object classes (voc) challenge // International journal of computer vision, 2010, т. 88, № 2, с. 303-338.
[12] Jacob B. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, с. 2704-2713.
[13] Solovyev, R., Kustov, A., Telpukhov, D., Rukhlov, V., & Kalinin, A. // Fixed-point convolutional neural network for real-time video processing in FPGA. In 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE, 2019, с. 1605-1611.
[14] Solovyev R., Wang W., Gabruseva T. Weighted boxes fusion: Ensembling boxes from different object detection models // Image and Vision Computing, 2021, т. 107, с. 104117.
[15] Solovyev, R. A., Telpukhov, D. V., Kustov, A. G., Rukhlov, V. S., & Isaeva, T. Y. Real-Time Recognition of Handwritten Digits in FPGA Based on Neural Network with Fixed Point Calculations. // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), 2019, № 4, с. 38-42.
[16] Solovyev, R. A., Stempkovsky, A. L., & Telpukhov, D. V. Study of Fault Tolerance Methods for Hardware Implementations of Convolutional Neural Networks. // Optical Memory and Neural Networks, 2019, 28(2), с. 82-88.
[17] Xygkis, A., Soudris, D., Papadopoulos, L., Yous, S., & Moloney, D. Efficient winograd-based convolution kernel implementation on edge devices. // In 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), IEEE, 2018, с. 1-6.
[18] Zhu, X., Dai, J., Yuan, L., & Wei, Y. Towards high performance video object detection. // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, с. 7210-7218.
[19] Han S., Mao H., Dally W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding // arXiv preprint arXiv:1510.00149, 2015, URL: https://arxiv.org/abs/1510.00149 (дата обращения: 22.04.2022).
[20] Kocić, J., Jovičić, N., & Drndarević, V. An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. // Sensors, 2019, 19(9), 2064.
[21] Gavrilov S.V., Zheleznikov D.A., Chochaev R.Z. Simulated Annealing Based Placement Optimization for Reconfigurable Systems-on-Chip // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, 2019, с. 1597-1600.
[22] URL: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=1159 (дата обращения: 22.04.2022)

Hardware Implementation of a Neural Network for Object Detection in FPGA

A.L. Stempkovskiy, R.A. Soloviev, D.V. Telpuhov, A.G. Kustov

Institute for Design Problems in Microelectronics of RAS, Moscow, ippm@ippm.ru

***Abstract* — The article proposes a technique for transferring the architecture of a modern CenterNet neural network to an FPGA. The CenterNet network is OneStage detector and is used to detect and localize objects in images. This neural network has good performance in terms of accuracy and at the same time is distinguished by the simplicity of the decoder. As a result of the research, it was possible to achieve a very high speed of the neural network at the hardware level by choosing an encoder suitable for hardware implementation, efficient hardware implementation of both the decoder and the last layer with object filtering, and switching to fixed-point calculations. At the same time, the quality of the obtained predictions remains high.**

***Keywords* — object detection, neural network hardware, FPGA, Fixed-point arithmetic, 2D convolution.**

REFERENCES

[1] Szegedy C., Toshev A., Erhan D. Deep neural networks for object detection //Advances in neural information processing systems, 2013, pp. 2553-2561.
[2] Ren S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks // Advances in neural information processing systems, 2015, pp. 91-99.
[3] Liu W. et al. SSD: Single shot multibox detector // European conference on computer vision, Springer, Cham, 2016, pp. 21-37.
[4] Redmon J., Farhadi A. Yolov3: An incremental improvement // arXiv preprint arXiv:1804.02767, 2018, URL: https://arxiv.org/abs/1804.02767 (access date: 22.04.2022).
[5] URL: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (access date: 22.04.2022).
[6] Law H., Deng J. Cornernet: Detecting objects as paired keypoints // Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 734-750.
[7] Duan K. et al. Centernet: Keypoint triplets for object detection // Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6569-6578.
[8] Howard A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications // arXiv preprint arXiv:1704.04861, 2017, URL: https://arxiv.org/abs/1704.04861 (access date: 22.04.2022).
[9] Kuznetsova A. et al. The open images dataset v4 // International Journal of Computer Vision, 2020, pp. 1-26.
[10] Lin T. Y. et al. Microsoft coco: Common objects in context // European conference on computer vision, Springer, Cham, 2014, pp. 740-755.
[11] Everingham M. et al. The pascal visual object classes (voc) challenge // International journal of computer vision, 2010, vol. 88, № 2, pp. 303-338.
[12] Jacob B. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2704-2713.

[13] Solovyev, R., Kustov, A., Telpukhov, D., Rukhlov, V., & Kalinin, A. // Fixed-point convolutional neural network for real-time video processing in FPGA. In 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE, 2019, pp. 1605-1611.

[14] Solovyev R., Wang W., Gabruseva T. Weighted boxes fusion: Ensembling boxes from different object detection models // Image and Vision Computing, 2021, pp. 107, art. 104117.

[15] Solovyev, R. A., Telpukhov, D. V., Kustov, A. G., Rukhlov, V. S., & Isaeva, T. Y. Real-Time Recognition of Handwritten Digits in FPGA Based on Neural Network with Fixed Point Calculations. // Problems of Perspective Micro- and nanoelektronic Systems Development, 2019, Issue 4, Pp. 38-42.

[16] Solovyev, R. A., Stempkovsky, A. L., & Telpukhov, D. V. Study of Fault Tolerance Methods for Hardware Implementations of Convolutional Neural Networks. // Optical Memory and Neural Networks, 2019, 28(2), pp. 82-88.

[17] Xygkis, A., Soudris, D., Papadopoulos, L., Yous, S., & Moloney, D. Efficient winograd-based convolution kernel implementation on edge devices. // In 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), IEEE, 2018, pp. 1-6.

[18] Zhu, X., Dai, J., Yuan, L., & Wei, Y. Towards high performance video object detection. // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7210-7218.

[19] Han S., Mao H., Dally W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding // arXiv preprint arXiv:1510.00149, 2015, URL: https://arxiv.org/abs/1510.00149 (access date: 22.04.2022).

[20] Kocić, J., Jovičić, N., & Drndarević, V. An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. // Sensors, 2019, 19(9), art. 2064.

[21] Gavrilov S.V., Zheleznikov D.A., Chochaev R.Z. Simulated Annealing Based Placement Optimization for Reconfigurable Systems-on-Chip // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, 2019, pp. 1597-1600.

[22] URL: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=1159 (access date: 22.04.2022)