УДК 004.021

АППАРАТНАЯ РЕАЛИЗАЦИЯ КОМПАКТНОЙ И ЭФФЕКТИВНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ОБУЧАЕМОГО ДВУМЕРНОГО РАЗДЕЛИМОГО ПРЕОБРАЗОВАНИЯ

Кривальцевич Е.А.[1], студент гр.150701

Белорусский государственный университет информатики и радиоэлектроники[1]
г. Минск, Республика Беларусь

Вашкевич М.И. – докт. техн. наук

Аннотация. В работе рассмотрена аппаратная реализация нейронной сети для распознавания изображений на основе обучаемого двумерного разделимого преобразования. Обучение нейронной сети осуществлялось на языке Python с использованием библиотеки PyTorch. Также на языке Python описана эталонная модель вычислительного процесса нейронной сети, использующая представление данных в формате с фиксированной запятой. Осуществлена аппаратная реализация нейронной сети на языке SystemVerilog и произведено её тестирование с использованием отладочной платы Zybo-Z7.

Ключевые слова. Нейронные сети, обучаемое двумерное разделимое преобразование, Verilog, IP-блок.

Введение

В условиях ограничений по объёму памяти, скорости обработки и энергопотреблению — особенно при реализации на встраиваемых устройствах или FPGA — особую актуальность, при решении задачи распознавания изображений, приобретают архитектуры нейронных сетей с минимальным числом параметров при сохранении высокой точности классификации. Одним из подходов к снижению вычислительной сложности нейросетевых моделей является использование двумерного разделимого линейного слоя (LST2D), который позволяет существенно сократить количество умножений и требуемый объём памяти [1].

В данной работе рассматривается аппаратная реализация нейросети с использованием слоя LST2D на базе программируемой логики (FPGA), направленная на достижение компромисса между компактностью модели и точностью распознавания.

Разделимое двумерное преобразование

Двумерное разделимое преобразование применяется в обработке изображений для снижения вычислительной сложности при пространственной фильтрации. Предлагаемое обучаемое преобразование обрабатывает изображение вначале по строкам, а потом по столбцам.

Преобразование LST2D обрабатывает изображение X размера $d_{in} \times d_{in}$ при этом на выходе получается изображение Y размера $d_{out} \times d_{out}$ [1]:

$$Y = LST_{d_{in} \times d_{in}}(X) = \tanh(W_2 \tanh(W_1 X^T)), \tag{1}$$

где W_1, W_2 – матрицы весов слоев FC1 и FC2, соответственно, а d_{in} и d_{out} – это гиперпараметры преобразования, определяющие число обучаемых параметров $N = 2 \cdot (d_{in} + 1) \cdot d_{out}$.

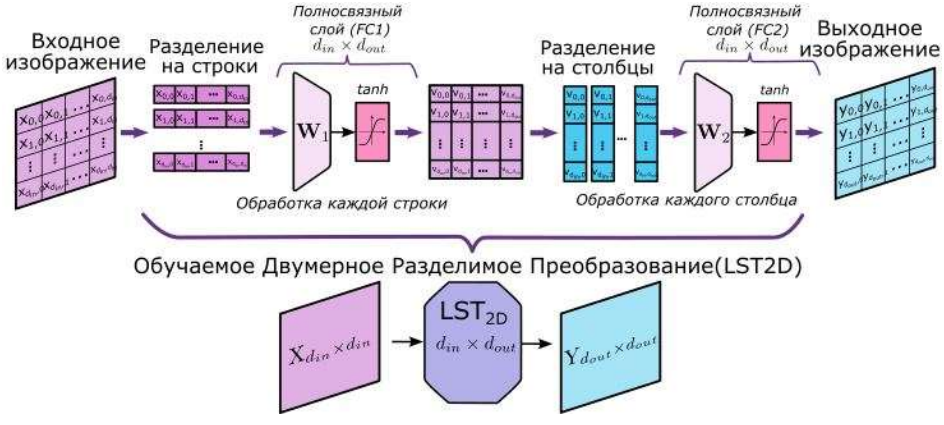

Рисунок 1 – Схема LST2D преобразования[1]

Обучение НС

Обучение НС LST2D осуществляется на основе 60000 тренировочных изображений из базы данных MNIST. Из MNIST загружается 60000 тренировочных и 10000 тестовых изображений размером

28 × 28 пикселей в градациях серого. Из тренировочного набора выделяется 1000 изображений для валидации, оставшиеся 59000 используются для обучения модели.

Перед подачей в нейросеть изображения проходят предварительную обработку, включающую нормализацию значений пикселей. Каждый пиксель исходного изображения представлен числом в диапазоне от 0 до 255. Для упрощения процесса обучения выполняется нормализация данных таким образом, чтобы значение каждого пикселя было в диапазоне от −1 до 1, а стандартное отклонение (σ) составляло 0.5. Это позволяет сделать градиентный спуск более устойчивым и ускорить процесс сходимости модели. Обучение проводится на 300 эпохах, при этом на каждой эпохе вычисляется функция потерь NLLLoss.

На рисунке 2,а показан график функции потерь, которая в течении 300 эпох стабилизируется, приближаясь к значению около 0,2, что указывает на достижение сходимости.

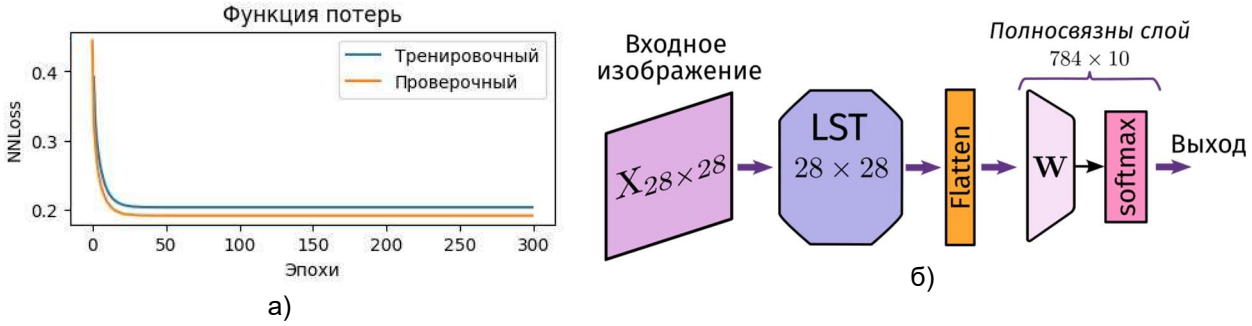

Рисунок 2 – Нейронная сеть: а) функция потерь на обучающем и тестовом наборе; б) архитектура нейронной сети

Эталонная модель

Программная реализация НС выполняется с использованием языка Python. В качестве эталонной модели реализуется нейронная сеть на основе библиотеки fixpoint. В данной реализации используется формат представления данных Q6.7, где 6 бит отведены под целую часть числа, а 7 бит используются для дробной части.

При переводе значений из плавающей запятой в фиксированную используется метод округления к ближайшему минимальному числу (округление вниз). Этот метод позволяет минимизировать ошибки округления и избежать систематического смещения, которое может возникать при округлении к ближайшему числу.

Эталонная модель представляет собой НС с двумя скрытыми слоями и активационной функцией tanh (см. рисунок 2,б). Входные изображения размером 28 × 28 проходят последовательную обработку:

– Первый слой выполняет линейное преобразование входных данных с весами и смещением, после чего применяется функция активации тангенс.

– Второй слой аналогично выполняет линейное преобразование, используя параметры весов и смещений, после чего применяется функция активации тангенс.

– Выходной слой выполняет классификацию, вычисляя вероятности классов с помощью softmax.

Сравнение выходных значений между моделью с фиксированной точностью и моделью с плавающей точкой представлено на рисунке 3.

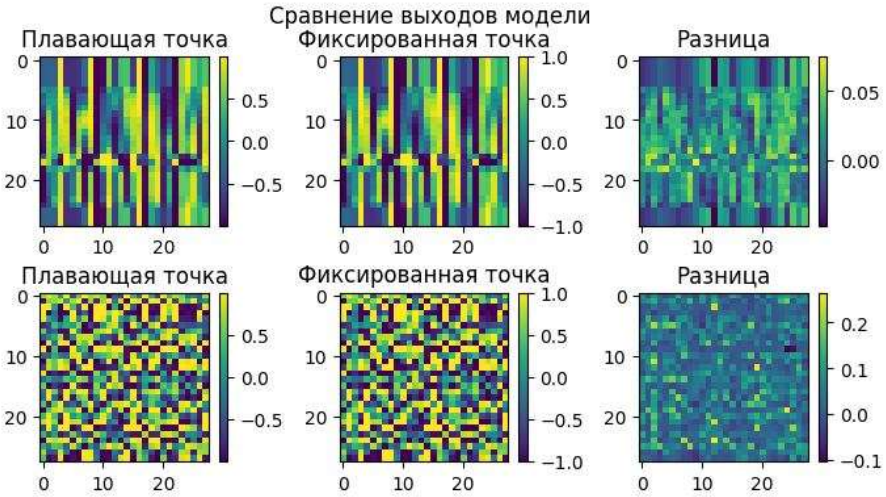

Рисунок 3 – Сравнение результатов вычислений

Реализация на FPGA

Для реализации НС была выбрана отладочная плата Zybo на базе ПЛИС Zynq-7000. Zynq – это система на кристалле (СнК), которая объединяет процессор ARM и программируемую логику FPGA. Для упрощения разработки и тестирования на этой платформе используется дистрибутив Linux PYNQ

(Python productivitY for zyNQ). Для соединения процессорной системы с разработанным IP-блоком используется преобразователь интерфейса uP-AXI4-lite.

Структура разработанной системы для распознавания рукописных цифр на базе платформы Zynq представлена на рисунке 4.

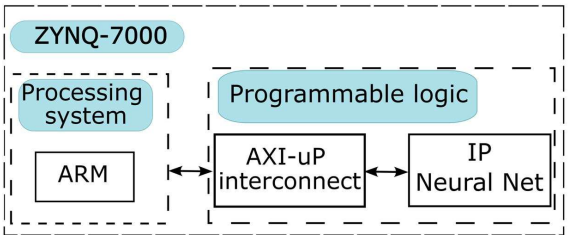

Рисунок 4 – Структура системы для распознавания рукописных цифр

Рассмотрим структуру IP-блока HC, которая представлена на рисунке 5.

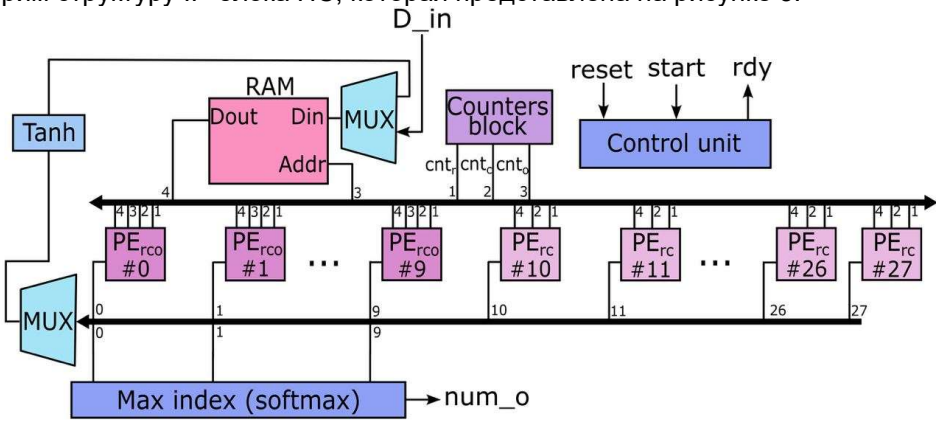

Рисунок 5 – Структура IP-блока HC

Данный компонент состоит из следующих блоков:

1 Блок счетчиков.

2 Блок процессорных устройств (ПУ) первого типа.

3 Блок ПУ второго типа.

4 Блок памяти изображения.

5 Блок softmax.

6 Блок tanh.

7 Блок управляющего устройства.

В блоке счетчиков осуществляется расчёт адреса для памяти весовых коэффициентов, а также формирует адрес для записи и чтения из памяти изображения.

Все вычисление LST слоя, а также финального полносвязного слоя осуществляется на 28 ПУ, которые подразделяются на 2 типа в соответствии с рисунком 6, что удобно для обработки изображений размером 28 на 28 пикселей.

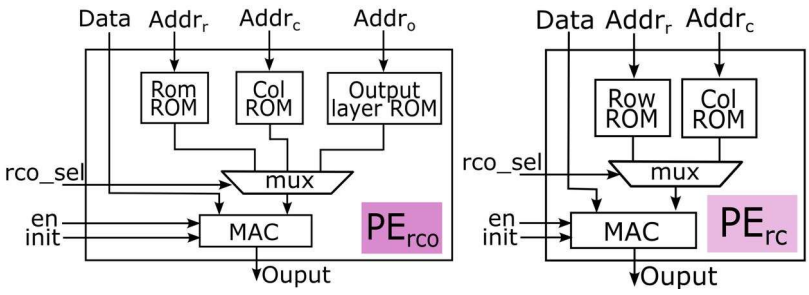

Рисунок 6 – Структура блоков ПУ

Процессорные устройства первого типа используются в количестве 10 штук. Данные ПУ пере используются после вычисления LST слоя в финальном полносвязном слое. Данная ключевая особенность требует хранения дополнительных весовых коэффициентов и передачи дополнительного адресного входа, для управления чтения из данной памяти.

Процессорные устройства второго типа хранят лишь весовые коэффициенты для обработки по строкам и столбцам входного изображения.

В блоке памяти изображения храниться обрабатываемая картинка на всех этапах. Сначала исходные данные, затем результат каждого из FC слоев в LST преобразовании.

В блоке tanh осуществляется аппроксимированный расчет функции активации тангенс гиперболический для LST слоя. Прямая реализация вычисления тангенса будет занимать большое количество вычислительных ресурсов. Поэтому для удобства реализации используется аппроксимация в соответствии с формулой [2]:

$$tanh(x) \approx F(x) = \begin{cases} sign(x), |x| > 2 \\ (1 + \frac{x}{4}) \cdot x, -2 < x < 0 \\ (1 - \frac{x}{4}) \cdot x, 0 < x < 2 \end{cases} \quad (2)$$

Блок softmax осуществляет выбор наиболее вероятного класса. Реализация в соответствии с формулой 3 требует значительных вычислительных ресурсов, поэтому для FPGA реализации осуществляется сравнение всех 10 входных значений и выбор максимального из них.

$$soft\,max = \frac{e^{z_j}}{\sum_{j=1}^{k} e^{z_j}} \quad (3)$$

где z_j – набор числовых значений, e^{z_j} – экспоненциальная функция от z_j, которая переводит числовое значение в положительное.

Управляющее устройство формирует корректные поставки данных между блоками и управляет функционалом в соответствии с требуемым алгоритмом.

Эксперимент и результаты

Экспериментальное тестирование заключается в аппаратной обработке тестового набора изображений и подтверждении релевантности использования данного типа слоя НС для решения задач по распознаванию рукописных цифр на основе компактной архитектуры.

Для тестирования на FPGA осуществляется последовательная обработка 10000 тестовых изображений и построение матрицы спутывания, которая показывает точность распознавания каждой цифры. На рисунке 7 представлена матрица, полученные на FPGA. Общая точность в НС с LST2d слоем и разрядностью Q6.7 составляет 96,6%.

Рисунок 7 – Матрицы спутывания по результатам аппаратного тестирования и эталона

На основании полученных результатов можно сделать вывод, что цифры 3 и 4 распознаются наилучшим образом (точность – 98.6%), хуже всего распознается цифра 5 (точность – 93,2%). Аппаратные затраты, полученные на основе отчетов о размещении проекта на FPGA в среде Xilinx Vivado, представлены в таблице 1. Частота работы устройства 80МГц.

Таблица 1 – Аппаратные затраты на FPGA реализацию

Тип блока	Количество блоков, шт		Использование, %
	используемое	доступное	
LUT как логика	8111	17600	46.09
LUT как память	169	6000	2.82
FF	27	35200	0.08
BRAM	33	60	55.0
DSP	29	80	36.25

Данная архитектура позволяет хранить в памяти 9474 параметра, но значительно меньше, чем в других архитектурах при схожей точности, что показывает релевантность использования данного слоя в архитектурах НС и реализации на FPGA. Сравнение архитектур представлено в таблице 2.

Таблица 2 – Сравнение параметров и точности различных архитектур НС

Авторы	Архитектура	Количество параметров, шт	Точность, %
Huynh[3]	784-40-40-40-10	34960	97.2
Huynh[3]	784-126-126-10	115920	98.16
Westby[4]	784-12-10	9550	93.25
Umuroglu[5]	784-1024-1024-10	1863690	98.4
Liang[6]	784-2048-2048-2048-10	10100000	98.32
LST2D	LST2D	9474	96.6

Выводы

В работе предложен вариант аппаратной реализации устройства распознавания рукописных цифр на базе LST2D преобразования и платформы Zybo. Показано значительное уменьшение количества параметров, в сравнении с другими архитектурами.

Список использованных источников:

1. Vashkevich, M., Krivalcevich E. Compact and Efficient Neural Networks for Image Recognition Based on Learned 2D Separable Transform / M. Vashkevich, E.Krivalcevich // Proc. of 27th International Conference on Digital Signal Processing and its Applications (DSPA). – 2025. – P. 1–5.

2. L. D. Medus, T. Iakymchuk, J. V. Frances-Villora, M. Bataller-Mompean, and A. Rosado-Munoz, "A novel systolic parallel hardware architecture for the FPGA acceleration of feedforward neural networks,"IEEE Access, vol. 7, pp. 76 084–76 103, 2019.

3. T. V. Huynh, "Deep neural network accelerator based on FPGA," in 2017 4th NAFOSTED Conference on Information and Computer Science,2017, pp. 254–257.

4. I. Westby, X. Yang, T. Liu, and H. Xu, "FPGA acceleration on a multilayer perceptron neural network for digit recognition," The Journal of Supercomputing, vol. 77, no. 12, pp. 14 356–14 373, 2021.

5. Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays, 2017, pp. 65–74.

6. S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized neural network on FPGA," Neurocomputing, vol. 275, pp. 1072–1086, 2018.

UDC 004.021

HARDWARE IMPLEMENTATION OF A COMPACT AND EFFICIENT NEURAL NETWORK FOR IMAGE RECOGNITION BASED ON A LEARNABLE TWO-DIMENSIONAL SEPARABLE TRANSFORMATION

Krivaltsevich E.A.

Belarusian State University of Informatics and Radioelectronics[1], Minsk, Republic of Belarus

Vashkevich M.I. – Doctor of Science

Annotation. The paper considers the hardware implementation of a neural network for image recognition based on a learnable two-dimensional separable transformation. The neural network was trained in Python using the PyTorch framework. A reference model in Python is described. Hardware implementation in the Verilog language and testing on a set of MNIST handwritten digits have been implemented.

Keywords. Neural networks, trainable two-dimensional separable transformation, Verilog, IP block.