Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Wojciech Lepich

Nr albumu: 1146600

Rozpoznawanie cyfr przez sieć neuronową zaimplementowaną na układzie FPGA

Praca licencjacka
na kierunku Informatyka

Praca wykonana pod kierunkiem
dr. Grzegorza Korcyla
z Zakładu Technologii Informatycznych

Kraków 2020

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

.....	.....
Kraków, dnia	Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

.....	.....
Kraków, dnia	Podpis kierującego pracą

Spis treści

1 Wstęp

Tutaj wstęp

2 Teoria

2.1 Architektura FPGA

Field-programmable gate array (FPGA) to układy scalone, które mogą być elektronicznie przeprogramowane bez potrzeby demontażu samego układu z urządzenia. W porównaniu do układów ASIC znacznie taniej zaprojektować pierwszy działający układ. Elastyczna natura układów FPGA wiąże się z większym zużyciem powierzchni krzemu, opóźnień oraz zużycia energii. (FPGA architecture: survey and challenges)

Dodać przypis

Podstawowa struktura układów FPGA składa się z różnych bloków logicznych, które mogą być łączone zależnie od projektu. Przykładami takich bloków są: DSP (jednostka przeprowadzająca obliczenia dodawania/mnożenia), LUT (look-up table, de facto tablica prawdy dowolnej funkcji boolowskiej), Flip Flop (przechowują wynik LUT), BRAM (block RAM, pamięć dwuportowa, jest w stanie przechowywać względnie dużą ilość danych).

Układy FPGA przeważnie pracują na kilku-, kilkunastukrotnie niższych częstotliwościach niż CPU. Wysoką wydajność zawdzięczają zrównolegleniu obliczeń.

Przypis

2.2 Przetwarzanie obrazu

Cyfrowe przetwarzanie obrazu jest problemem wymagającym dużych mocy obliczeniowych ze względu na ilość danych do przetworzenia. Nieskompresowany kolorowy obraz z pikselami w formacie RGB (po 8 bitów na kolor) o wysokości 720 pikseli i szerokości 1280 pikseli to 22118400 bitów ($\approx$ 2,5MB). Obraz przetwarzany w czasie rzeczywistym, na przykład z kamery, zwielokrotnia tę liczbę o liczbę klatek na sekundę (przy trzydziestu klatkach na sekundę liczba danych rośnie do około 79 megabajtów na sekundę). Należy również pamiętać, że dane są dwuwymiarowe co jest ważne przy problemach związanych z rozpoznawaniem wzorców, klasyfikacją przedmiotów na obrazie, filtrowania w celu rozmazania lub wyostrzenia obrazów, itp.

2.2.1 Formaty pikseli

Jest wiele modeli przestrzeni barw (a co za tym idzie, sposobów kodowania pikseli) między innymi:

- RGB, używany w aparatach, skanerach, telewizorach
- CMYK, używany w druku wielobarwnym
- HSV
- YUV

Składowe dwóch ostatnich przestrzeni barw oddzielają informację o jasności od informacji o kolorach. Model barw YUV składa się z kanału luminacji Y oraz kanałów kodujących barwę U oraz V, są to kolejno składowa niebieska i składowa czerwona. W projekcie użyty jest format pikseli YUY2 (znany też pod nazwą YUYV), w którym

na dwa piksele przypadają 32 bity. Licząc od najstarszego bitu pierwsze osiem bitów przypada na Y0, to jest luminacja pierwszego piksela, następne osiem bitów na U0, kolejne osiem bitów to luminacja drugiego piksela, a pozostałe bity to składowa czerwona V0. Dla obydwóch pikseli składowe U i V są wspólne. Co istotne w projekcie, łatwo oddzielić luminację, która jest używana w przetwarzaniu obrazu.

Wstaw obrazek ze schematem, bo to trudne do zrozumienia

2.3 Sieci neuronowe

Z czym się to je, jak działają, jak wygląda trenowanie

3 Opis projektu

3.1 Zarys projektu

Jakich narzędzi korzystałem, co chcę osiągnąć, elo 420.

3.2 Platforma

Zcu104 cośtam. Oparty o reVISION ™. Zdjęcia

3.3 Sieć neuronowa

Jak jest zbudowana, jak uczona, dlaczego taka a nie inna; problemy z LeNet-5

3.4 hls4ml

A co to za framework i dlaczego taki fajny, dostosowanie precyzji z profiling Tutaj też dopisz o dostosowaniu sieci, tzn. o progu

3.5 Gstreamer

Do czego służy, jaki zbudowałem pipeline

3.6 Używanie sieci

Czyli synteza sieci i zrobienie z niej biblioteki statycznej „.a”

3.7 Część neuralnet

Czytanie obrazu, podział na część luma i chroma, wywołanie funkcji sieci, zapis z powrotem, synteza do biblioteki dzielonej „.so”

3.8 Część gstsdxnet

De facto plugin gstreamera, w którym są wywoływane funkcje z biblioteki dzielonej neuralnet.so,

3.9 Małe podsumowanie

4 Wyniki i dyskusja

4.1 Ewaulacja modelu

Wyniki z samego pythona z danymi testowymi z mnista

4.2 Symulacja

Tutaj wyniki z symulacji z danymi testowymi z mnista

4.3 Dane rzeczywiste

No i tutaj wyniki z kamerki. Myślę, że nie trzeba robić jakichś spisów wielkich, wystarczy opisać co dobrze odczytało, co źle, dlaczego, jakie sieć ma problemy, przygotowanie danych (zdjęcia!), jak poprawić skuteczność działania (LeNet xD)

5 Podsumowanie

W projekcie zostało zrobione to i to. Wyszło to tak i tak. Problem sprawiło tamto i owamto. Można to poprawić w ten sposób. Można część funkcjonalności z pipeline przenieść na fpga (sam pisałem, że przetwarzanie obrazu na fpga jest gicior).