靜宜大學資訊工程學系

畢業專題報告書

YOLOv7嵌入式系統加速推論

一張含有 寫生, 卡通 的圖片

自動產生的描述

實驗室名稱:智慧感知與互動技術實驗室

指導教授:翁永昌老師、林浩仁老師

專題學生:葉睿廷

民國112年12月14日

致謝

一年多過去了,隨著專題的完成及落幕,在此誠摯的感謝我的兩位畢業專題教授-翁永昌教授及林浩仁教授，以及一位碩士研究生-洪丞賢。在專題進行的期間，給予我許多寶貴的意見,無私地分享建議,並讓我跟隨著碩士研究生共同研究此專題計畫,讓我在這段期間學習到許多知識及經驗，一步步地解決困難前進。

在步入專題的起點時，我對專題題目及相關的知識毫無頭緒,感謝翁永昌教授帶我初步的認識專題所需要的知識，讓我在探索深度學習影像辨識這部分時，能夠更好的得到相關知識。隨著專題進行，進入實驗室後，熟識了實驗室的碩士研究生，並與其一同進行影像辨識的相關研究，並在2023年與學長發布了第一篇學術論文。

接著要感謝林浩仁教授，在研究的途中，給予我及學長許多研究的方法及相關文獻的探討建議，並讓我在實驗室中除了最初的YOLOv7深度學習模型研究外，認識了可以讓模型佈署於FPGA硬體實作的平台上，讓模型能有更多的研究價值及應用面的價值。最後在學長的指導及建議下，完成了模型的嵌入式推論。在此感謝兩位教授以及學長所提供的設備資源及技術指導，讓我能順利的完成此畢業專題。

目錄

[一、前言 3](#_heading=h.t5nohrt9so9x)

[二、文獻探討 4](#_heading=h.g2m168219v42)

[(一)文獻資料 4](#_heading=h.t9lvvux4vbdo)

[三、系統架構與設計 5](#_heading=h.zhv53ph6vhfo)

[四、安裝與實作流程](#_heading=h.cu5sebhebp9o) 6

[(一)前置作業 20](#_heading=h.5z3tp5prl26u)

[(二)部署監控與品質控管服務 20](#_heading=h.id0e6qaj237u)

[(三)前端、後端、安裝初始化 IPFS 22](#_heading=h.jq9ncipw9kgt)

[(四) Go-ethereum & BlockScout 架設 29](#_heading=h.6snx1ixhowpo)

[(五)智能合約上傳至區塊鏈 33](#_heading=h.ohtt0thahay3)

[(六) MetaMask 錢包設定(智能合約交易需要使用) 38](#_heading=h.t6r5l7r9d8au)

[五、實驗結果與展示 39](#_heading=h.on06i3s7apmz)

[六、結論與未來規劃 47](#_heading=h.sxii6fhnkxew)

[七、參考資料及網站 48](#_heading=h.b3oo4p80em29)

1. 前言

卷積神經網路(CNN)在計算機視覺AI 應用日益廣泛，其中物件偵測應用的準確率有長足的進步。 以FPGA上的DPU實現CNN模型做硬體加速，和使用電腦與GPU比較，具有相對較低的成本與功耗，適合邊緣用算的應用。另外，以FPGA開發而成的半客製化電路，它相較於ASIC節省了大量的開發時間。因此，FPGA具有快速部署及後續更新、維護的優勢。FPGA的另一個特性是可以被反覆修改的架構，在AI算法快速更新的時代，可以根據需要的模型結構來快速調整晶片架構，FPGA平台可以因應不同種類的模型來進行運算。

為了將物件偵測網路模型放入邊緣終端運算裝置，對訓練好的網路模型進行壓縮是必要的考量，常見的模型壓縮技術包含剪枝與量化，如何達到壓縮後的模型在準確度部會嚴重下降，是最重要的目標。壓縮後的模型除了對硬體資源需求降低，也可以加快推論得運行速度、且能節省能量的消耗，如此產生的輕量級模型適合邊緣運算的應用。

基於此研究動機，本專題對於YOLOv7使用硬體套件進行模型訓練、量化及加速，透過QAT(Quantization Aware Training)訓練，使得YOLOv7的推論模型有良好的準確度的同時又降低了原本所需的計算量，本文透過Xilinx 開發的 Vitis-AI套件中的NNDCT進行訓練量化，搭配DPU實作深度學習加速系統，使用FPGA合成卷積神經網路加速器來加速YOLOv6卷積層的計算後於平台內取得結果。

1. 文獻探討

Syed Sahil Abbas Zaidi等人[2]調查了近年來物件偵測模型技術的發展以及各類模型的特色與效能評估，文章內容包含：(1)分析三種物件檢測器並回顧其方法的演變，(2)說明檢測器基礎原理，強調網格設計方法對偵測器效能影響，(3)對具有代表性的網路主幹結構以及輕量化模型進行詳細的分析。此文詳細介紹當前物件偵測神經網路模型，包含YOLO物件偵測模型發展過程與其他檢測模型方法差異、與性能比較。

Chien-Yao Wang 等人[1] 提出YOLOv7物件偵測模型，文中說明歷代YOLO物件偵測模型的差異，藉由由公開的COCO[3]資料集訓練並評測最新的主流YOLO物件偵測模型，包含YOLOv5、YOLOX、PPYOLOE、YOLOR，從文中比較結果可以看出YOLOv7在辨識正確率與執行速率都有極佳的效果，這也是本文選擇在FPGA實現YOLOv7的緣由。

Bradley McDanel [4]等人提出一個全棧優化的框架，用於加速CNN推理，並通過FPGA實現進行驗證。作者共同優化CNN模型、計算架構和硬體實現，以在推理延遲、能量效率、硬件利用率和推理精度等交換空間中實現前所未有的性能。所提出的框架包括一個簡化的CNN架構，僅使用1x1卷積而無需剩餘連接，在移動設置中實現具有競爭力的ImageNet性能，包括量化感知訓練、輸入重塑以提高利用率和FPGA指令。他們使用高效的Selector-Accumulator（SAC）架構來實現任何數字CNN硬件中存在的乘法器累加器（MAC）操作，減少所需的硬件資源和功耗。作者在FPGA上構建了一個高效的CNN推理引擎，通過檢查輸出是否與其Python模擬器的輸出一致來驗證其正確性。

1. 研究方法及步驟

(一)、概念流程規劃

在專題剛確立時，我以YOLOv7為基礎，進行藥物辨識，使模型能夠應用於網站讓使用者能夠辨識藥物及查詢其功效。因為有些使用者是服用慢性藥物，一次會領取長期的劑量，但是當藥物包裝遺失時，藥物混在一起，當使用者需要不需要服用其中一種功用的藥物時，就容易找不到其藥丸為哪種。在第一次口試結束後，我發現模型佈署於平台時，由於模型太過龐大，以及需要很高的計算量。對於伺服器及計算回傳時間都會造成很大的功耗。因為我開始轉換方向，進行模型量化部屬硬體平台的分析研究。

(二)、模型訓練及硬體部署

在2023年5月與碩士研究生共同發表論文後，我在自己的專題除了模型研究外，也開始著手研究硬體部署的部分。

在此專題中的模型部分，PC端使用Ubuntu20.04版本建立整個操作的環境。使用Anaconda建立虛擬環境，並在其中安裝CUDA以及CUDNN讓GPU的RTX 2080Ti可以在訓練模型時得到更好的計算效果。接著在虛擬環境下安裝PyTorch以及Onnx等套件以計算模型推論。

而在硬體部分，使用PYNQ中的ZCU-104(低階)以及ZCU-106(高階)開發版進行硬體的十座，SD卡映像檔版本為官方所提供的v3.0.1版本，並安裝DPU-PYNQ以支援Vitis-AI 2.5.0及PYNQ 3.0。

1. **安裝與實作流程**

PC端:

建立環境

於Ubuntu安裝Anaconda並建立環境python 3.9以及安裝Pytorch 1.12.0。

下載資料

Git YOLOv7官方github後進入該路徑下,接著使用依照官方github安裝requirements.txt檔

安裝CUDNN以及CUDA

依據使用者所使用的顯示卡對應版本。

安裝COCO-datasets

使用bash scripts/get\_coco.sh安裝COCO-datasets的2017資料集，共Training、Validation、Test三個分類以及預標籤好的Label檔。

測試數據。

使用python test.py --data data/coco.yaml --img 640 --batch 32 --conf 0.001 --iou 0.65 --device 0 --weights yolov7.pt --name yolov7\_640\_val 後程式會自行下載yolov7.pt檔(官方預訓練權重檔)進行測試。

開發版端:

取得開發版IP

使用ifconfig得到IP後，於PC網站中輸入該IP位置，進入jupyter介面，密碼為Xilinx。

安裝PYNQ-DPU

於jupyter的終端機中，輸入pip3 install pynq-dpu --no-build-isolation

模型QAT量化

利用nndct API將模型進行QAT訓練量化

模型轉換

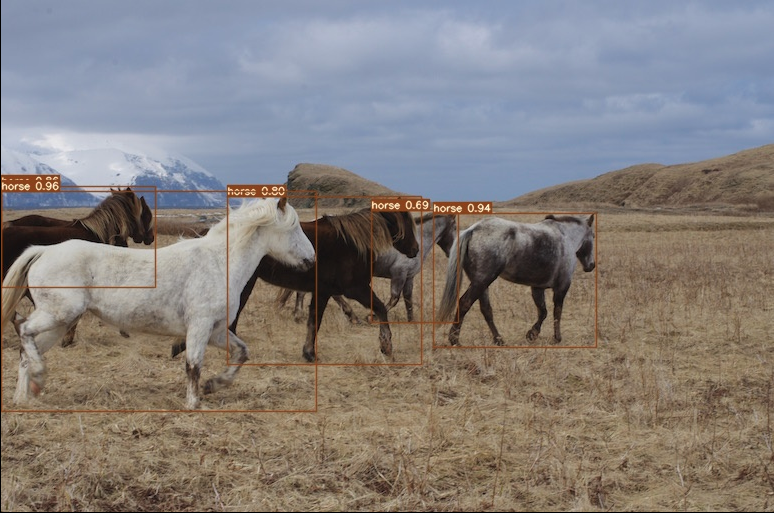
利用export將模型轉換為ONNX格式

模型測試

使用test\_onnx.py檔依照PC端方式進行測試

1. **實驗結果與展示**

在PC端上使用2080TI辨識結果:



在開發版端上使用DPU運算辨識結果:

一張含有 文字, 螢幕擷取畫面, 家畜, 戶外 的圖片

自動產生的描述

於PC端上測試COCO-datasets數據:

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

於開發版端上測試COCO-datasets數據:

一張含有 文字, 螢幕擷取畫面, 字型, 文件 的圖片

自動產生的描述

效能:

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

1. **結論與未來規劃**

本專題成功使用QAT技術進行YOLOv7模型的量化，首先將模型進行QAT量化訓練，接著轉換為ONNX，最後使用OnnxRuntime呼叫DPU對模型進行加速推論。最終，在不會明顯降低準確度的前提下，使得壓縮後的模型能順利部屬到精簡的邊緣運算平台，並獲致較佳的推論執行效率，未來研究方向將搭配剪枝等其它模型技術，近一步壓縮神經網路模型，以獲得更佳的推論執行效能，達到高解析影片實時(real-time)辨識的目標。

1. **文獻資料**
2. Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, Brian Lee, (2022). "A survey of modern deep learning based object detection models", Digital Signal Processing, Volume 126, 2022, 03514, ISSN 1051-2004, <https://doi.org/10.1016/j.dsp.2022.103514>.
3. Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao Institute of Information Science, Academia Sinica, Taiwan,” YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object”, **[v1]** Wed, 6 Jul 2022, <https://arxiv.org/abs/2207.02696>
4. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll´ar, and C Lawrence Zitnick. (2014). “Microsoft COCO: Common objects in contex,” In European conference on computer vision, pages 740–755. Springer, 2014. <https://arxiv.org/abs/1405.0312v3>
5. B. McDanel, S. Q. Zhang, H. Kung, and X. Dong, "Full-stack optimization for accelerating CNNs with FPGA validation," arXiv preprint arXiv:1905.00462, 2019. <https://arxiv.org/abs/1905.00462v1>
6. [Vitis-AI Github](https://github.com/Xilinx/Vitis-AI)