NVIDIA Jetson Nano에서 Deep Neural Network의 WCET 분석

2017104017 이준용

1. 서론

Real-Time System은 특정한 시간 내에 결과를 보장해야 하는 하드웨어나 소프트웨어 시스템이다. 시간내에 결과값을 보장하지 못할 경우 사용자의 안전과 같은 심각한 문제를 초래하는 hard real time system과 그만큼은 아니지만 효율성의 문제를 초래하는 soft real time system 으로 구분할 수 있다. Hard real time system의 예로는 자율주행차에 들어가는 소프트웨어를 들 수 있고, Soft real time system의 예로는 키보드 입력이나 화면 출력과 같은 소프트웨어를 예로 들 수 있다.

Real-Time System은 앞서 언급한 것처럼 특정한 시간 내에 결과를 보장해야 하는 시스템이다. 이 때 이 프로그램의 실행시간은 캐시 메모리의 사용이나 파이프라인, branch prediction 결과가 다를 수 있다는 이유로 달라질 수 있다. 이 때 Real-Time System에서 WCET(Worst Case Execution Time)를 분석한다는 것은 캐시 미스가 발생하거나 Branch Prediction이 실패하는 등의 최악의 경우의 실행시간을 구하는 것이다. 이 시간이 프로그램이 동작하는데 가장 오래 걸리는 시간일 것이고, 이 시간을 Deadline으로 설정한다면 Real-Time System의 조건을 만족할 수 있게 된다. 따라서 Real-time System에서 WCET 분석은 반드시 필요하다.

Jetson Nano라는 임베디드 시스템에서의 DNN Execution을 연구하는 이유는 데스크탑에서의 실행시간 분석은 실제 프로그램 동작에 필요하지 않은 여러 함수나 데이터들, 예를 들면 키보드 마우스 인터럽트나 모니터 출력으로 인해 부정확한 결과를 초래할 수 있다. 또한 대표적인 Real-Time System인 자율주행 시스템에서는 여러 센서들로부터 비동기적으로 수집된 데이터들을 처리해야하고, 딥러닝 모델을 통한 객체 인식이 필수적인 기능이다. 따라서 GPU를 통한 뉴럴 네트워크의 빠른 연산이 가능한 NVIDIA Jetson에서 본 연구를 진행하고자 한다.

한편, 프로그램의 실행시간을 분석하는 Tool로 PyTorch에서 프로파일러 API를 제공한다. 해당 API를 통해 딥러닝 모델의 성능을 분석하고 모델의 각 layer별 실행시간을 파악할 수 있다. 이 툴을 활용하여 임베디드 시스템인 NVIDIA Jetson Nano 위에서 4~5개의 미리 훈련된 Deep Neural Network을 실행시켜 실행시간의 분포를 파악하고 excel을 활용하여 실행시간 분포 그래프를 그려본다. 이를 통해 WCET를 분석하는 것이 본 프로젝트의 목표이다.

2. 연구 목표

2.1 NVIDIA Jetson

NVIDIA Jetson은 GPU, CPU, 메모리, 전원 시스템, 고속 인터페이스, 센서와 SDK 등을 포함하는 임베디드 애플리케이션을 위한 플랫폼이다. Jetson은 다른 NVIDIA 플랫폼에서 사용되는 AI 소프트웨어나 클라우드 네이티브 워크플로와 호환되고, AI나 컴퓨터 비전 분야를 불과 5~10와트의 저전력으로 효율적인 성능을 제공하는 컴퓨터이다.

본 프로젝트에서는 NVIDIA Jetson에서 제공하는 SDK를 사용하여 임베디드 시스템에서 OS를 설치해보고, 해당 시스템에서 딥러닝 모델의 성능을 시험해보고 실행 시간을 예측해보는 경험을 할 수 있을 것이다.

2.2 딥러닝 모델

NVIDIA Jetson 위에서 훈련된 딥러닝 모델을 4~5개 정도 반복 훈련시켜 각각의 딥러닝 모델의 성능을 분석하는 경험을 할 수 있다. 훈련된 딥러닝 모델은 PyTorch Hub에서 제공한다. 현재 사용할 딥러닝 모델은 객체 인식 딥러닝 알고리즘의 YOLOv5[1], ResNet[2], GoogLeNet[3], Deeplabv3[4]을 사용할 예정이다.

3. 관련 연구

Hardware Architecture와 컴파일러에 따라 Deep Neural Network를 활용해 초기 WCET를 파악할 수 있는 모델을 연구했다는 논문[5]이 있다. 해당 논문에서는 PyTorch 라이브러리를 사용하여 프로그램을 컴파일하지 않고 WCET을 파악하는 네트워크를 만들었다. 이 네트워크는 정확도가 떨어져 실제로 사용되기에는 무리가 있지만, 시스템 개발의 초기 단계에서 하드웨어 구성을 위한 가이드라인을 제공할 수 있다.

4. 현재 진행 상황

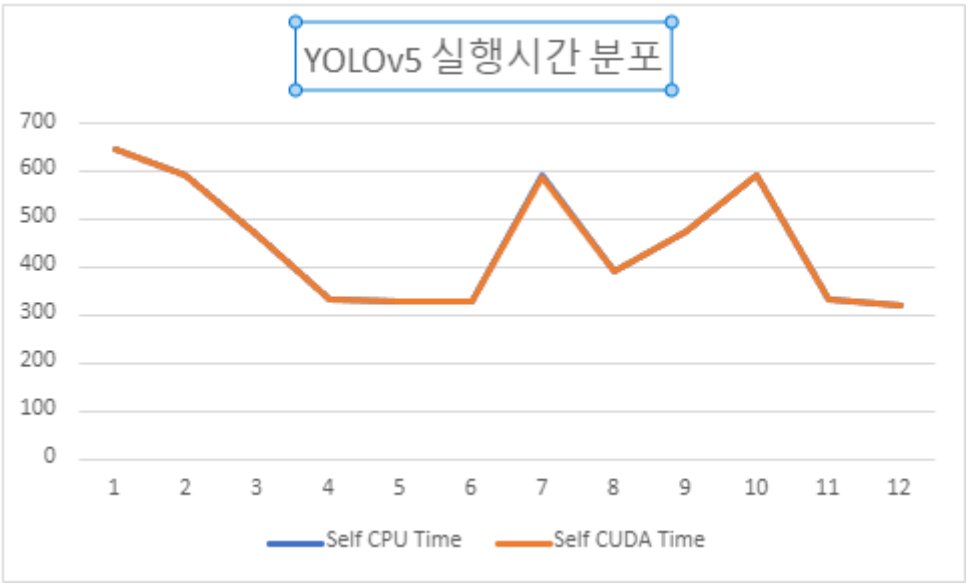
4.1 Real-Time System에 관한 기초 학습

교수님께서 제공해주신 Real Time embedded systems 강의 자료와 buttazzo의 'Hard real-time computing systems: predictable scheduling algorithms and applications'의 Real Time system 소개글을 참고하여 Real Time System에 대해서 공부하였다.

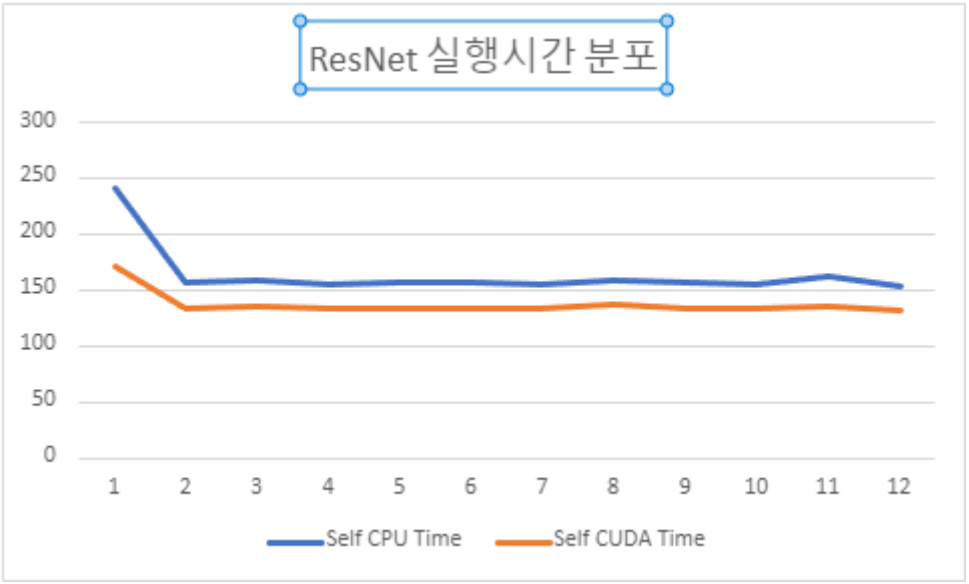
4.2 PyTorch Profiler를 사용한 실행시간 분석

PyTorch에서 제공하는 프로파일러 API를 사용하였다. 해당 API의 사용방법은 https://tutorials.pytorch.kr/recipes/recipes/profiler_recipe.html 을 참고하였다. 파이썬은 3.9버전을 사용하였고 아나콘다 가상환경에서 실험하였다.

torch함수에서 cuda()함수를 사용하기 위해서는 NVIDIA 그래픽카드가 설치되어야 하고, cuda 버전에 맞는 라이브러리를 설치해야 한다.



<YOLOv5 profiler 결과>

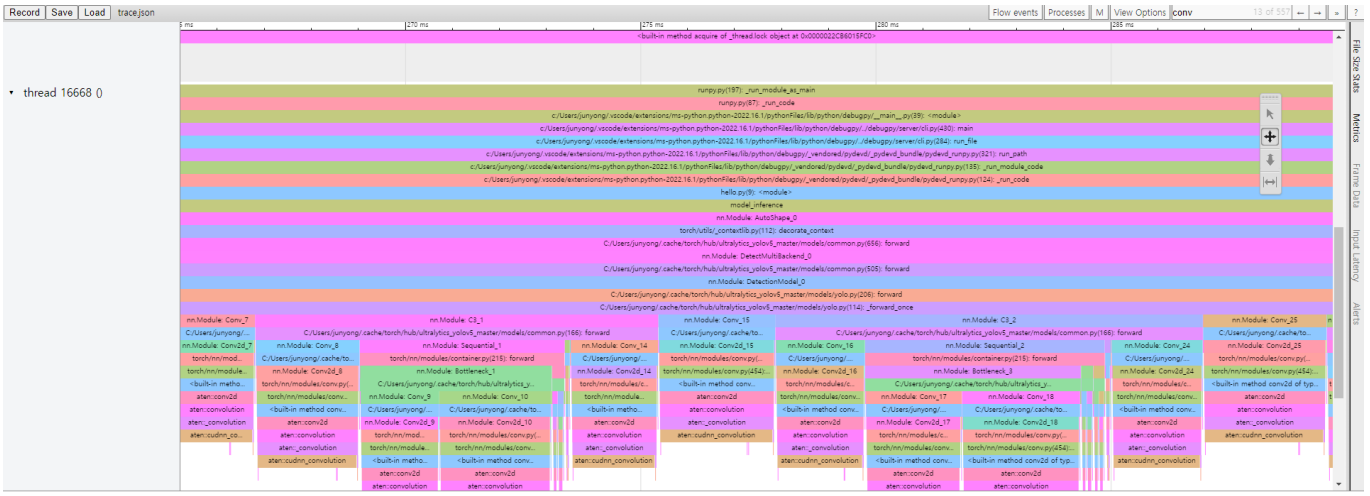


<ResNet profiler 결과>

프로파일러를 통해 YOLOv5 모델과 ResNet 모델을 데스크탑에서 실행시켜 보았을 때 결과는 위의 그래프와 같았다.

먼저 YOLOv5 모델의 실행시간 분석을 보면 CPU로 실행했을 때와 CUDA로 실행했을 때 CUDA가 약간 빠르긴 했지만 실행시간이 유의미하게 차이나지 않은 모습이었다. 또한 12번을 실행시켰을 때 가장 빠른 실행시간이 319.7ms, 가장 오래걸린 실행시간이 589.3ms로 WCET이 BCET에 비해 2배 가까이 차이나는 결과를 확인하였다.

한편 ResNet 모델의 실행시간 분석을 보면 YOLOv5 모델과는 다르게 CPU 실행시간과 CUDA 실행시간이 유의미하게 차이나는 모습을 볼 수 있다. 또한 첫 번째 실행에서 CPU 실행시간 240ms, CUDA 실행시간 171ms로 나타났지만 이외의 모든 실행에서 CPU 실행시간 155ms 근처, CUDA 실행시간 133ms 근처로 일정한 실행시간 분포를 보였다.



<YOLOv5 profiler chrome://tracing결과>

5. 향후 일정

- 매주 월요일 17시 교수님께 진행상황 보고 및 지도사항 전달
- 4월 3일 NVIDIA Jetson 보드 수령
- 4월까지 NVIDIA Jetson 사용방법 숙달
- 4월 28일까지 중간보고서 작성
- 5월 26일 최종보고서 작성완료 및 프로젝트 완료

프로젝트 github주소: https://github.com/jooonyong/CAPSTON2

6. 프로젝트 기대효과

먼저 본 프로젝트를 진행하면서 자율주행 분야나 군사 시스템 등에서 사용될 수 있는 Real Time System이라는 생소한 주제에 대해 공부할 수 있고 해당 시스템에서 Worst-Case Execution Time(WCET), Best-Case Execution Time(BCET) 를 분석한다.
GPU가 장착된 임베디드 시스템에서 딥러닝 모델을 실행시켜보는 경험을 할 수 있을 것이다. 또한 한정된 메모리의 임베디드 시스템 환경에서 딥러닝 모델의 실행시간 분포를 분석해 WCET을 추정할 수 있다.

7. 참고자료

[1] Couturier, Raphaël, Hassan N. Noura, Ola Salman, and Abderrahmane Sider. n.d. “A DEEP LEARNING OBJECT DETECTION METHOD FOR AN EFFICIENT CLUSTERS INITIALIZATION.”

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Deep Residual Learning for Image Recognition.” (12).

[3] Szegedy, Christian, Wei Liu, and Yangqing Jia. 2014. “Going deeper with convolutions.” (9).

[4] Chen, Liang-Chieh, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. “Rethinking Atrous Convolution for Semantic Image Segmentation.” (6).

[5] Kumar, Vikash. 2021. “Deep Neural Network Approach to Estimate Early Worst-Case Execution Time.”

[6] Hard real-time computing systems: predictable scheduling algorithms and applications - GC Buttazzo

[7] https://developer.nvidia.com/embedded-computing

[8] https://pytorch.org/hub/research-models

[9] https://tutorials.pytorch.kr/recipes/recipes/profiler_recipe.html