Journal of The Institute of Electronics and Information Engineers Vol.57, NO.9, September 2020 https://doi.org/10.5573/ieie.2020.57.9.56
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2020-57-9-8

자율주행 자동차에서 DNN 기반 안전 필수 응용의 성능 저하 방지

(Avoiding Performance Degradation of DNN based Safety-Critical Applications in Autonomous Vehicles)

김 도 영*, 김 명 선**

(Doyoung Kim and Myungsun Kim©)

요 약

DNN(deep neural network) 기술의 빠른 발전은 자율주행 자동차에서 뛰어난 성능 향상으로 이어지고 있다. 이러한 시스템에서는 객체 탐지, 차선 이탈 방지, 졸음운전 방지, 주변 이미지 분석 등 여러 가지 DNN 모델 기반의 응용들이 수행된다. 이때 수행되는 DNN 모델들을 호스팅하는 응용들은 높은 우선순위를 갖는 운행에 관련된 안전 필수적인 것들과 중간 혹은 낮은 우선순위를 갖고 운행에는 직접적으로 관련되지 않는 응용들로 이루어진다. 따라서 GPU와 같은 DNN 가속 장치를 서로 다른 우선순위를 갖는 DNN 모델들이 공유해서 사용한다. 또한 하나의 DNN 모델 당 하나의 배치 프로세스 형태로 수행되어, 비주기적으로 도착하는 안전 필수 응용들이 사용하는 DNN 모델들을 최우선적으로 GPU에 할당하기는 어렵다. 본 논문에서는 이러한 문제를 해결하기 위하여 우선순위를 기반으로 DNN 모델의 각 레이어 별로 GPU에 연산을 의뢰하는 프레임워크를 제안한다. 제안된 기법을 실제 상용 보드에 탑재한 후 실험한 결과 높은 우선순위의 DNN 모델들의 성능이 적용 전 대비 최대 43.9% 향상되었다.

Abstract

The rapid development of DNN technology is leading to outstanding performance improvement in autonomous vehicles. In this system, various DNN model-based applications such as object detection, lane departure prevention, drowsy driving prevention, and surrounding image analysis are performed. The applications hosting the DNN models are classified into safety-critical ones with high priority and others with medium or low priority not directly related to driving. Therefore, DNN accelerators such as GPUs are shared and used by DNN models with different priorities. In addition, since it is performed in the form of one batch process per DNN model, it is difficult to firstly allocate the DNN models used by safety-critical applications that arrive sporadically to the GPU. In order to solve this problem, this paper proposes a framework for requesting computation to the GPU for each layer of the DNN model based on priority. As a result of experiments after running the proposed technique on an actual off-the-shelf board, the performance of high-priority DNN models improved upto 43.9% compared to that of before.

Keywords : Deep neural network, Safety-critical, Multi-DNN, Priority scheduling, Latency

Ⅰ. 서 론

스마트폰을 비롯해서 스마트 농업, 헬스케어 로봇, AR/VR, 스마트 팩토리, 자율주행 자동차 등 많은 분야에서 DNN(deep neural network)은 높은 인식 성공률을 제공하고 있다[1, 2]. 이와 같은 응용 분야들에서 사용 되는 시스템은 다양한 종류의 센서를 갖추고 이들의 입력을 받아 사전에 학습시킨 DNN 모델들을 구동시키는

* 학생회원, ** 정회원, 한성대학교 IT융합공학부(Department of IT Convergence Engineering, Hansung University)
©Corresponding Author(E-mail : kmsjames@hansung.ac.kr)
※ 이 성과는 부분적으로 2020년도 정부(과학기술정보통신부)의 재원으로 한국 연구재단의 지원을 받아 수행된 연구임(No. NRF-2020R1G1A1012170). 본 연구는 김명선 교수의 한성대학교 교내학술연구비 지원과제임.
Received ; July 6, 2020 Revised ; August 23, 2020 Accepted ; August 26, 2020

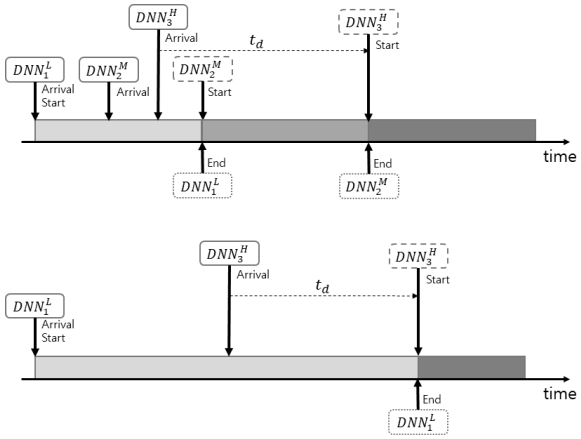

그림 1. 서로 다른 우선순위를 갖는 다중 DNN 모델 연산 시 문제점 예시

Fig. 1. Problem illustration when Multi-DNN models with various priority levels are running.

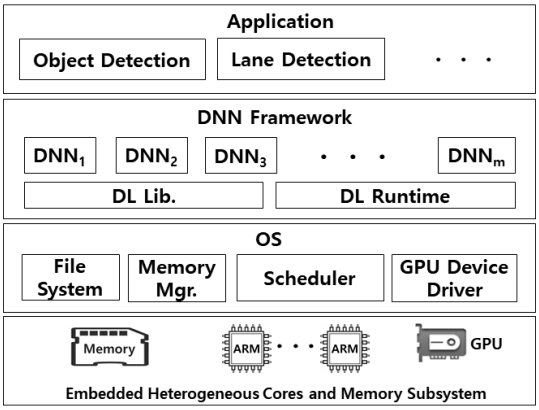

그림 2. 대상 시스템

Fig. 2. Target system.

하드웨어 및 소프트웨어로 구성된 디바이스 모듈 형태를 취한다. 특히 개인의 정보를 보호하고 네트워크 환경에 따른 지연을 피하기 위하여 서버 컴퓨팅에 의존하는 정도를 낮춘 임베디드 시스템에서의 DNN 연산처리 기법은 현재 더욱더 이슈화되고 있다.

임베디드 시스템은 소비전력, 메모리, 크기, 가격 등의 측면에서 제약이 있다. 하지만 다양한 기능과 높은 인식률을 요구함에 따라서 임베디드 시스템에 탑재되는 DNN 모델은 다양화되고 그 개수가 증가되고 있다. 따라서 시스템의 효율성과 인식 정확성을 유지하면서 제한된 시스템 자원 환경에서도 시기적절한(timely) 응답 특성을 보장해야 한다. 특히 자율주행 자동차의 안전 필수(safety-critical) 응용이 사용하는 DNN 모델에게 이러한 응답 특성은 매우 중요하다[3].

DNN 모델들은 입력, 은닉, 출력 층들을 순차적으로(sequential) 수행하고 대부분 모델 당 하나의 프로세스 형태로 할당된다[4~6]. 자율주행 시스템에서는 전방 객체 탐지(object detection)[7~9], 차선 이탈 감지(lane detection)[10], 졸음운전 방지[11], 측면과 후방의 이미지 분석 등에 사용되는 DNN 모델들이 비 주기적(sporadic)으로 다양한 타이밍 요구 사항을 나타내며 연산된다. 이들 중 전방 객체 탐지나 차선 이탈 감지 등의 응용은 안전 필수 응용이며 가장 높은 우선순위로 처리되어야 하고[3] 다른 응용들은 상대적으로 이보다 낮은 우선순위로 처리되어야 한다.

그림 1을 통하여 이러한 상황에서의 문제점을 설명한다. 그림 1에서 DNN_1^L, DNN_2^M, DNN_3^H은 각각 Low, Mid., High의 우선순위를 갖는 DNN 모델 1, 2, 3을 나타낸다. 그림 1의 위 부분 그림은 각 모델의 수행 시간이 같고 모델 1, 2, 3의 순서로 시작되는 경우이다. 안전 필수 응용이 사용하는 모델이 DNN_3^H인 경우 t_d의 지연 시간 후에 시작된다. 그림 1의 아랫부분 그림은 낮은 우선순위의 DNN_1^L의 수행 시간이 길 경우이다. 마찬가지로 DNN_1^L의 수행이 다 끝날 때까지 DNN_3^H이 시작하지 못하여 큰 지연 시간 t_d를 나타내고 있다.

본 논문에서는 안전 필수 응용이 사용하는 DNN 모델의 우선순위를 고려해서 그림의 지연 시간 t_d를 최소화하는 다중 DNN 처리 프레임워크를 제안한다.

Ⅱ. 대상 시스템 및 문제 정의

본 절에서는 본 논문에서 대상으로 하는 시스템을 설명한다. 그리고 본 논문을 통하여 해결하고자 하는 문제를 기술한다.

1. 대상 시스템

그림 2는 본 논문에서 다루는 대상 시스템을 나타내고 있다. 그림 2에서 Application은 차선 감지(lane detection) 및 객체 탐지(object detection) 등과 같은 인공 신경망을 사용하는 응용 프로그램들을 의미한다. DNN 프레임워크는 앞서 설명한 응용 프로그램들이 호출해서 사용하는 각종 딥러닝 모델들과 런타임 및 라이브러리 등으로 이루어진 DNN 연산에 필요한 기본적인 패키지로 이루어져 있다. 대상 시스템이 m개의 딥러닝 모델을 사용하고 DNN_1, DNN_2, … , DNN_m처럼 나타낸다. 대상 시스템은 ARM 코어와 같은 임베디드 시스템에 기본적으로 사용되는 CPU를 사용하며 임베디드 리눅스와 같은 운영체제(OS)를 사용한다. GPU와 같은 DNN 가속기는

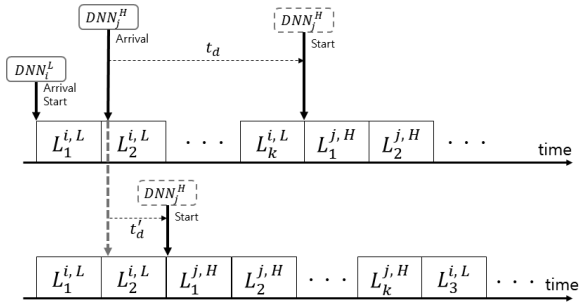

그림 3. 문제 해결 예시
Fig. 3. Problem solving illustration.

운영체제 영역에 존재하는 디바이스 드라이버를 통해서 접근된다.

차선 감지 및 객체 탐지와 같은 ARM코어 상의 호스트 응용 프로그램들은 각자 $DNN_1, DNN_2, \cdots, DNN_m$ 중 필요한 딥러닝 모델들을 인터페이스하고 이 모델들은 NVIDIA(사)의 CUDA[12], cuDNN[13] 등의 라이브러리 및 런타임을 활용하여 운영체제 영역의 GPU 디바이스 드라이버를 접근한다. 이후 GPU는 호스트에서 전달받은 커널을 분산 및 병렬적으로 연산하여 고효율의 DNN 연산을 수행한다. 또한 CUDA 스트림(stream)이라는 기능을 통하여 호스트/GPU 간 데이터 전송과 커널 연산을 중첩시킬 수 있고, GPU 내부에서의 컴퓨팅 자원이 허락되는 범위에서 서로 다른 스트림에 할당된 커널을 동시에 실행할 수 있어 더욱더 높은 병렬 컴퓨팅을 제공한다[14].

2. 문제 정의

앞 절에서 설명한 것처럼 자율주행 시스템에서는 여러 수준의 우선순위를 가진 DNN 모델들이 수행된다. 본 연구에서 풀고자 하는 문제는 우선순위가 높은 응용들이 사용하는 DNN 모델들이 최대한 간섭받지 않고 우선적으로 수행되어 안전성을 보장할 수 있게 함에 있다. 그림 3은 이러한 점을 그림으로 쉽게 설명한다. 그림에서 i, j는 DNN 모델 번호를 나타내고 우선순위는 위 첨자 H(high), L(low)로 표시하였다. 낮은 우선순위를 가진 DNN_i^L가 수행되는 도중에 DNN_j^H가 수행 요구를 받게 되면, 그림의 위 부분에서 알 수 있듯이 DNN_i^L을 구성하는 전체 레이어 $L_1^{i,L}$, $L_2^{i,L}$, $\cdots$, $L_k^{i,L}$가 모두 수행된 후 즉, t_d시간이 지나서야 DNN_j^H의 첫 번째 레이어 $L_1^{j,H}$가 수행됨을 알 수 있다. 본 연구에서는 그림의 아랫부분처럼 $L_2^{i,L}$가 수행된 시점 바로 직후에 $L_1^{j,H}$가 수행하게 하여 높은 우선순위 DNN 모델 연산의 지연을 t'_d로 개선시키고자 한다.

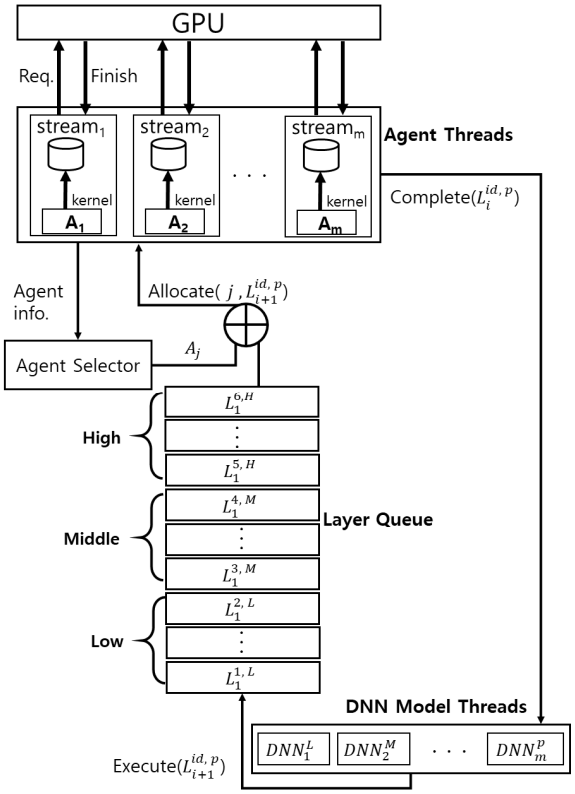

그림 4. 우선순위 기반 다중 DNN 처리 프레임워크
Fig. 4. Priority-based Multi-DNN framework.

Ⅲ. 우선순위 기반 다중 DNN 처리 프레임워크

본 절에서는 다중 DNN 연산 요구에 대하여 우선순위를 기반으로 GPU에서 처리하는 솔루션을 제시한다. 시스템을 모델링하고 이를 통하여 솔루션의 동작 방법을 설명한다.

1. 시스템 구성 및 모델링

그림 4는 본 논문에서 제안하는 다중 DNN 처리 프레임워크의 구조를 보여준다. 프레임워크 전체적으로 세 부분으로 구성되어 있으며 이는 각각 DNN 모델 쓰레드 집합, 우선순위 기반 레이어 큐, 에이전트 쓰레드 집합이다. DNN 모델 쓰레드 집합은 그림 2의 Application 영역 내부의 차선 감지 혹은 객체 탐지 응용이 사용하는 DNN 모델들의 메인 쓰레드 집합을 나타며 $\{DNN_1, DNN_2, \cdots, DNN_m\}$를 나타낸다.

$DNN_{id}^p \in \{DNN_1, DNN_2, \cdots, DNN_m\}$는 k개의 레이어로 구성되고 우선순위 p(H, M, L 중 하나)를 갖는 DNN 모델이며 $1 \le id \le m$ 조건을 만족한다. 또한 이는

$DNN_{id}^{p}=\{L_1^{id,p},L_2^{id,p},\cdots,L_k^{id,p}\}$로 나타낼 수 있다. 레이어 큐(layer queue)는 $L_1^{id,p},L_2^{id,p},\cdots,L_k^{id,p}$ 중 하나의 레이어를 에이전트 쓰레드에 전송하기 전에 대기시키는 역할을 한다. 이때 하나의 레이어 $L_i^{id,p}(1\le i\le k)$는 아직 수행되지 않은 DNN_{id}^{p}의 i번째 레이어를 나타낸다.

에이전트 쓰레드 집합은 $A=\{A_1, A_2,\cdots,A_n\}$으로 표시하고 n개의 CPU상에서 구동되는 쓰레드들로 구성된다. 다시 말해서 $A_j(1\le j\le n)$는 j번째 CPU에서 수행되는 쓰레드이다. A_j는 $L_i^{id,p}\in DNN_{id}^{p}$의 정보를 GPU대신 우선순위 기반 레이어 큐로부터 전달받는 에이전트 역할을 하고 CUDA, cuDNN, CUDA 스트림을 통해서 GPU에 커널 형태로 $L_i^{id,p}$ 연산을 지시한다.

2. GPU기반 다중 DNN 프레임워크의 동작 방법

앞서 기술된 시스템 구성을 위하여 본 연구에서는 먼저 DNN 모델들을 레이어 별로 재구성하여 레어어 큐에 잡(job)의 형태로 전달할 수 있게 하였다.

전체적인 프레임워크의 동작 순서는 먼저 객체 탐지나 차선 감지와 같은 응용들은 각자에게 필요한 DNN_{id}^{p}을 선택한다. 그 후 DNN_{id}^{p}에 해당하는 DNN 모델 쓰레드는 $L_1^{id,p}, L_2^{id,p},\cdots, L_k^{id,p}$ 순서대로 레이어 큐에 전송한다. 이후 현재 시점에서 가장 적절한 에이전트를 선택하고 선택된 쓰레드는 스트림을 통하여 GPU에게 해당 레이어의 연산에 해당하는 커널을 전달한다.

구체적인 동작 방법은 다음과 같다. 에이전트 쓰레드가 GPU로부터 $L_i^{id,p}$ 연산이 완료되었음을 감지하면 Complete($L_i^{id,p}$) 신호를 DNN 모델 쓰레드에 전달한다. 이후 DNN_{id}^{p}에 해당하는 DNN 모델 쓰레드는 다음 수행될 레이어인 $L_{i+1}^{id,p}$의 정보를 담아 Execute($L_{i+1}^{id,p}$) 신호를 형성하여 레이어 큐에 전송하고 자신은 대기 상태인 wait 상태로 천이한다.

레이어 큐는 $L_{i+1}^{id,p}$의 우선순위(p)를 읽어서 해당 우선순위 즉, High, Middle, Low 리스트들 중 하나의 맨 뒤에 삽입된다. 에이전트 선택기(그림 4.의 Agent Selector)는 에이전트 쓰레드들의 상태 정보를 기반으로 가장 적절한 $A_1, A_2,\cdots, A_n$ 중 하나를 선택하고 항상 레이어 큐의 맨 앞에 있는 리스트를 선택된 쓰레드로 전송한다. $L_{i+1}^{id,p}$가 레이어 큐의 맨 앞에 위치하게 되면 Allocate(j, $L_{i+1}^{id,p}$) 신호가 발생하고 $A_j(1\le j\le n)$ 쓰레드로 전송된다.

Algorithm: Layer Queue Management

Input: $L_i^{id,p}$ (i : Layer Index, *id* : DNN id, p : Priority)
LQ (Layer Queue)

Function LQ_push(LQ, $L_i^{id,p}$)
Lock LQ
if p's tail != NULL
  Insert $L_i^{id,p}$ between p's tail and the next of p's tail
  p's tail = $L_i^{id,p}$
else
  p's tail = $L_i^{id,p}$
  if q's tail != NULL /* priority q is one step higher than p */
    Insert $L_i^{id,p}$ between q's tail and the next of q's tail
  else if r's tail != NULL /* priority r is one step higher than q */
    Insert $L_i^{id,p}$ between r's tail and the next of r's tail
  else
    Place $L_i^{id,p}$ at the head of LQ
    Front = $L_i^{id,p}$
Unlock LQ

Function LQ_pull(LQ)
Lock LQ
ret = the current *Front* of the LQ
Front = the next of the current *Front*
return ret
Unlock LQ

그림 5. 레이어 큐 알고리즘
Fig. 5. Layer queue algorithm.

에이전트 쓰레드는 GPU의 병렬처리 기능을 충분히 활용하기 위하여 각 쓰레드별로 CUDA 스트림을 할당한다. 다시 말하면 $A_1,A_2,\cdots,A_n$들은 자신에게 할당된 스트림 1, 스트림 2, ..., 스트림 n을 통하여 할당된 레이어 연산을 수행한다. A_j가 스트림 j를 통해서 $L_{i+1}^{id,p}$ 연산을 완료하면 다시 Complete($L_{i+1}^{id,p}$) 신호가 전달되어 $L_{i+2}^{id,p}$가 레이어 큐에 쌓이고 연산된다. 따라서 임의의 한 순간에 어떠한 DNN_{id}^{p}의 $L_i^{id,p}$와 $L_{i+1}^{id,p}$가 동시에 레이어 큐에 전달되거나 에이전트 쓰레드에 연산이 요구될 수 없어 레이어 연산의 순서가 보장된다. Complete($L_k^{id,p}$) 신호가 수신되면 DNN_{id}^{p}의 마지막 k번째 레이어 연산이 끝났음을 나타내고 추론(inference)의 결과를 최종적으로 출력한다.

3. 우선순위 기반 레이어 큐 관리

그림 5는 DNN의 우선순위가 주어진 환경에서 실행 순서를 결정하는 알고리즘을 나타낸다. 알고리즘은 기본적으로 LQ_push와 LQ_pull 함수로 이루어지고, 각각 우선순위를 고려하여 레이어 큐에 연산 대상이 되는 레이어를 삽입하고 추출하는 역할을 한다. 기본적으로

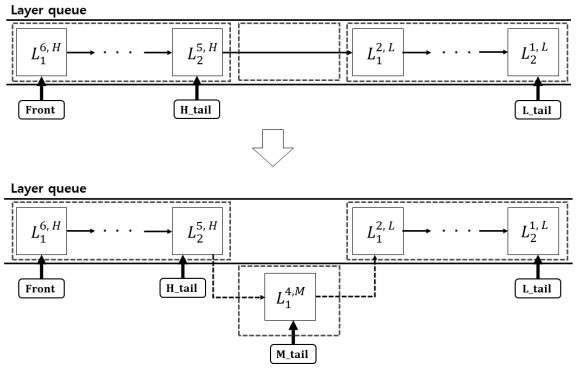

그림 6. 레이어 큐의 동작 예시
Fig. 6. Layer queue operation example.

FIFO(first in first out)로 동작하고 에이전트 쓰레드는 스트림에 매핑하고자 하는 대상을 항상 큐의 Front에서 가져간다. 큐의 Front는 항상 도착한 연산 대상 중 가장 우선순위가 높은 DNN의 레이어이다. 만일 하나 이상의 동일 우선순위 레이어가 큐에 존재한다면 Front는 이들 중 가장 먼저 도착한 항목을 가리킨다.

레이어 큐는 H_tail, M_tail, L_tail 세 개의 tail 포인터를 가지고 있다. 이들은 각각 우선순위가 H, M, L인 레이어 연산 중 가장 늦게 큐에 삽입된 항목들을 가리킨다. 알고리즘의 라인 3-4에서 알 수 있듯이 $L_i^{id,p}$가 큐에 입력되면, $p \in \{H,M,L\}$를 만족하는 p의 tail 포인터를 체크한다. 이때 해당 포인터가 존재하는 경우 바로 그 뒤에 연결된다. 그렇지 않은 경우에는 라인 6-15에서 나타내었듯이 p보다 더 높은 우선순위의 tail 포인터들을 확인하고, 해당 포인터가 존재하는 경우 바로 그 뒤에 연결한 후 자신이 p의 tail 포인터가 된다. 이때 p보다 높은 우선순위의 tail 포인터가 하나도 없을 경우에는 자신이 Front가 된다.

그림 6은 레이어 큐의 기본적인 동작을 예를 들어 설명하고 있다. 그림은 우선순위가 M인 DNN 모델번호 4의 첫 번째 레이어를 나타내는 $L_1^{4,M}$가 큐에 삽입되기 전, 후를 보여준다. 먼저 큐에 H, L 우선순위 레이어들이 존재하고 M에 해당하는 항목이 없을 경우 M_tail은 NULL 포인터를 가리킨다. 이후 $L_1^{4,M}$가 삽입될 때 M_tail이 없으므로 H_tail의 다음을 L 우선순위 첫 번째 리스트에서 새로 삽입된 $L_1^{4,M}$로 수정하고 $L_1^{4,M}$의 다음 포인터는 기존의 L 우선순위 첫 번째 리스트를 가리키게 한다. 마지막으로 M_tail을 $L_1^{4,M}$로 설정한다. 트리 형태의 우선순위 큐를 고려할 수 있으나 본 연구에서는 O(1)의 삽입 시간 복잡도를 위하여 세 개의 우선순위별 tail 포인터를 사용한다.

Ⅳ. 실 험

본 절에서는 우선순위 기반 다중 DNN 처리 프레임워크의 실제 효용성을 증명하는 실험 내용을 다룬다. 실험에 사용된 하드웨어와 소프트웨어를 소개하고 실험 결과와 이의 분석 내용을 기술한다.

1. 실험 대상 시스템

표 1. NVIDIA(사) Jetson AGX Xavier 구성
Table1. Jetson AGX Xavier hardware specifications.

GPU	512-Core Volta GPU/64 Tensor Cores 11 TFLOPS (FP16), 22 TOPS (INT8)
CPU	8-Core ARM v8.2 64-Bit CPU, 8MB L2 + 4MB L3
Memory	32GB 256-bit LPDDR4x 2133MHz - 137GB/s
OS	Linux kernel version 4.9.140
DL Lib.	CUDA 10.0, cuDNN v7

실험에 사용된 시스템은 NVIDIA(사)의 Jetson AGX Xavier 보드이다[15]. 이는 대표적인 상용으로 출시된 자율주행 자동차용 임베디드 시스템이다. 표 1을 통하여 해당 시스템의 구체적인 하드웨어 및 소프트웨어 구성을 알 수 있다.

2. 실험 결과 및 분석

본 연구에서 사용한 DNN 모델은 DenseNet201[16], ResNet152[17], Vggnet16[18], AlexNet[19] 4종류를 사용하였다. 실험 결과를 나타낸 그래프에서 각각의 모델은 Dens, Res, Vgg, Alex 이렇게 나타내었고 H, M, L은 우선순위 High, Mid, Low를 의미한다. 그래프에서 우선순위(H, M, L) 옆 숫자들은 함께 수행된 DNN 모델의 개수를 뜻한다. 예를 들면 Vgg(M5)는 5개의 M 우선순위 VggNet16이 동시에 수행됨을 나타낸다. 마지막으로 Original, Solution은 제안하는 프레임워크 적용 전, 후를 나타낸다.

가. 서로 다른 우선순위의 다중 DNN 환경

그림 7은 다수의 동일 DNN 모델을 사용했을 때 이 중 H 우선순위를 갖는 복수개의 모델들의 평균 수행시간 결과이다. x축은 사용된 복수개의 동일 DNN 모델의 우선순위별 개수이다. 예를 들어 Res(H4, M5, L5)는 시스템에 ResNet152 모델만 H가 4개, M이 5개, L이 5개

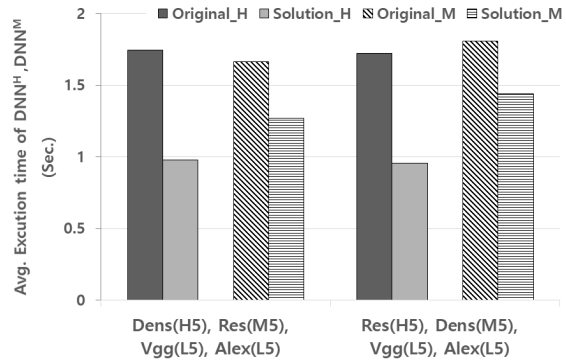

그림 7. 다수의 서로 다른 DNN 모델을 수행할 경우
Fig. 7. Running multiple different DNN models.

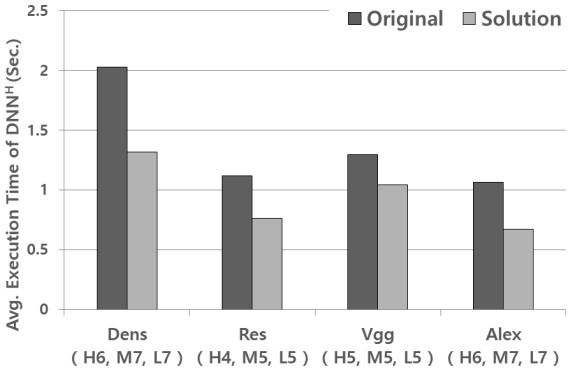

그림 8. 다수의 동일 DNN 모델을 수행할 경우
Fig. 8. Running multiple identical DNN models.

동시에 수행된다는 의미이다. 그림에서 알 수 있듯이 DensNet201, ResNet152, VggNet16, AlexNet 각각 35, 31.7, 19.6, 36.7% 성능 향상되었다. 그림에서 알 수 있듯이 더 많은 M과 L 우선순위의 DNN 모델이 함께수행되는 DensNet201과 AlexNet의 경우 그 성능 향상정도가 더 컸다.

그림 8은 여러 종류의 DNN 모델을 동시에 수행할 때의 결과이다. 좌측 두 쌍의 막대는 각각 DensNet201이 5개의 H, ResNet152이 5개의 M 우선순위를 가질 때 이들의 수행시간 결과이고 우측 두 쌍의 막대는 각각 DensNet201이 5개의 M, ResNet152이 5개의 H 우선순위를 가질 때이다. 좌측 두 쌍의 막대에서 알 수 있듯이 H 우선순위에서는 43.9%, M 우선순위에서는 23.5% 향상된 성능을 나타내었다. 우측 두 쌍의 막대에서는 H는 44.4% 빨라졌고, M은 20.3% 빨라졌다. 그림에서 알 수 있듯이 제안된 솔루션을 적용하면 H 우선순위에서의 성능 향상이 M 우선순위의 경우보다 더 컸다.

실험 결과로 알 수 있듯이 레이어 큐를 활용하고 레이어 연산을 우선순위를 기반으로 수행할 때 높은 우선순위의 DNN 모델들의 성능 향상이 뚜렷하였다.

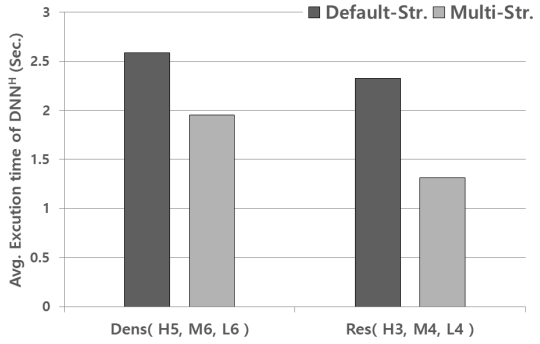

그림 9. 다수의 동일 DNN 모델 사용 시 복수개의 CUDA 스트림 효과
Fig. 9. Performance enhancement through multiple CUDA stream with identical DNN models.

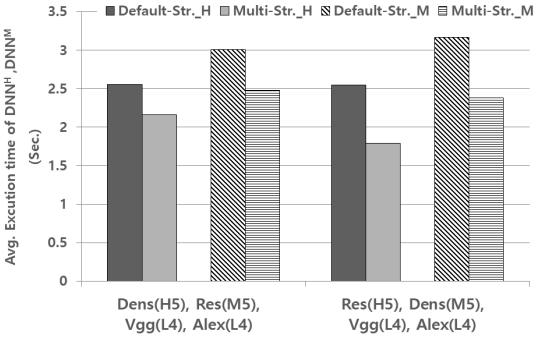

그림 10. 다수의 서로 다른 DNN 모델을 활용한 복수개의 스트림 효과
Fig. 10. Performance enhancement through multiple CUDA stream with different DNN models.

나. 다중 스트림 적용 효과

다음으로 CUDA 스트림[14]을 적용하여 병렬처리 기능을 더 확장했을 때의 효과를 실험하였다. 기본적으로 본 연구에서 제안한 프레임워크를 사용한 상태에서 디폴트 스트림과 복수개의 스트림을 적용한 경우를 비교하며 각각 그림에서 Default-Str., Multi-Str.으로 나타내었다. 복수개의 경우 사용된 스트림 수는 8로 하였다. 먼저 복수개의 동일 DNN 모델을 사용하였다. 그림 9에서 알 수 있듯이 DensNet201의 경우 복수개의 CUDA 스트림을 적용했을 때 H 우선순위 모델 5개의 성능이 평균 24.5% 향상되었고, ResNet152의 경우 H 우선순위 모델 3개의 수행시간이 43.4% 단축되었다.

다음으로 여러 종류의 복수개의 DNN 모델을 사용하였다. 그림 10이 보여주듯이 H 우선순위의 DenseNet201과 M 우선순위 ResNet152는 복수개의 스트림 환경에서 각각 15.4, 29.6% 성능 향상되었고, H 우선순위 ResNet152과 M 우선순위 DenseNet201의 경우에는 각각 17.7, 24.8% 성능 향상을 보였다.

다. 낮은 우선순위 DNN 모델 수에 따른 영향 분석

마지막 실험으로 동일 DNN 모델을 대상으로, H 우선순위를 4개로 고정하고 L 우선순위 개수를 늘려가며 이에 따른 H 우선순위 모델 4개가 받는 영향을 분석하였다. 그림 11과 12는 각각 Densnet201과 ResNet152의 결과를 나타낸다. 먼저 그림 11을 보면 제안된 프레임워크를 적용 전에는 L 우선순위 DNN 모델의 개수가 8일 때보다 12, 16으로 증가함에 따라 32.0, 58.9% 성능 저하가 관측되었다. 하지만 복수개의 CUDA 스트림과 레이어 큐 기반의 제안된 기법을 적용 시 각각 6.5, 10.0%성능 저하가 관측되었다. 그림 12는 ResNet152을 적용한 경우이다. 마찬가지로 L 우선순위 모델의 개수를 6일 때보다 8, 10으로 개수를 증가함에 따라 제안된 기법을 적용하기 전에 18.8, 35.3% 성능 저하가 나타났지만 적용 후 3.5, 8.5%로, 상대적으로 아주 작은 성능 저하만 있음을 알 수 있다.

실험 결과로 알 수 있듯이 제안된 프레임워크 적용 전에는 같이 수행되는 낮은 우선순위의 DNN 모델들의 개수가 증가함에 따라서 높은 우선순위의 모델들이 큰 영향을 받았지만 적용 후에는 그 영향이 크게 감소되었다. 따라서 자율주행 자동차에서 우선순위가 높은 안전 필수(safety-critical) 응용의 성능 고립화(performance isolation) 효과[3]를 기대할 수 있다.

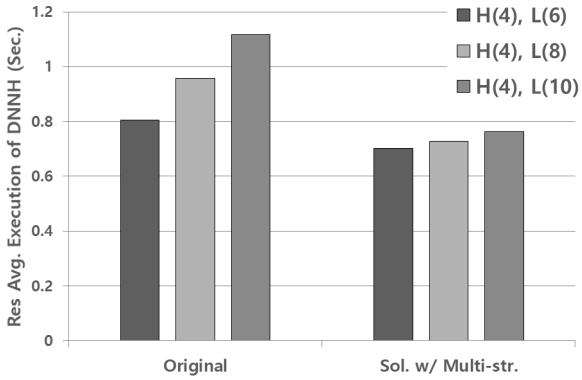

그림 11. 낮은 우선순위 DNN 모델 수에 따른 높은 우선순위 DNN 모델의 성능 저하: Densnet201

Fig. 11. Performance degradation of high priority DNN models according to the number of low priority DNN models: Densnet201 case.

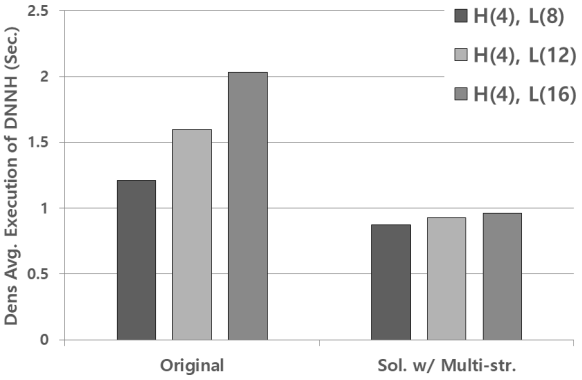

그림 12. 낮은 우선순위 DNN 모델 수에 따른 높은 우선순위 DNN 모델의 성능 저하: ResNet152

Fig. 12. Performance degradation of high priority DNN models according to the number of low priority DNN models: ResNet152 case.

V. 결 론

최근 DNN 기술의 발전은 자율주행 자동차, 농업, 금융, 스마트 팩토리 등 다양한 영역에서의 고도화로 이어지고 있다. 이에 따른 모델 자체의 복잡성과 사용되는 DNN 모델 수의 증가는 피할 수 없다. 자율주행 자동차와 같은 임베디드 시스템에서는 제한된 시스템 자원 하에서 여러 DNN 모델들이 GPU와 같은 DNN 가속 장치를 공유하고 각 DNN 모델들이 배치 프로세스 형태로 수행된다. 따라서 높은 우선순위의 안전 필수적인 응용들이 같이 수행되는 낮은 우선순위 응용보다 항상 최우선적인 DNN 연산을 보장받을 수는 없다. 본 논문에서는 각 DNN 레이어들이 우선순위 기반으로 큐잉되고 동시에 복수개의 CUDA 스트림으로 병렬처리 될 수 있는 프레임워크를 제시하였다. 다양한 조건에서 실험한 결과 높은 우선순위의 DNN 모델들의 성능이 적용 전 대비 최대 43.9% 향상되었음을 알 수 있었다.

REFERENCES

[1] D. Vasisht. Z. Kapetanovic, J. Won, X. Jin, R. Chandra, A. Kapoor, N. sinha, and M. Sudarshan, “FarmBeats: An IoT Platform for Data-Driven Agriculture,” in Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, Boston, USA, 2017.

[2] J. Dyrstad and J. Mathiassen, “Grasping virtual fish: A step towards robotic deep learning from demonstration in virtual reality,” in Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, China, 2017.

[3] M. Kim, “Method for Performance Isolation of Safety-Critical Applications in Mixed-Criticality Systems”, Journal of the Institute of Electronics and Information Engineers, Vol. 56, No. 12, December 2019.
[4] Torch, [Internet]. Available: http://torch.ch/.
[5] Caffe, Deep learning framework by BAIR [Internet]. Available: http://caffe.berkeleyvision.org/.
[6] TensorFlow, [Internet]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf/.
[7] J. Lee, S. Lee and S. Yang, “Model Ensemble for Speed Enhancement in Object Detection”, Journal of the Institute of Electronics and Information Engineers, Vol. 56, No. 6, June 2019, DOI: 10.5573/ieie.2019.56.6.35.
[8] R, Joseph, S. K. Divvala, R. B. Girshick, A. Farhadi, “You only look once: Unified, real-time object detection”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, DOI: 10.1109/CVPR.2016.91.
[9] J. Choi, D. Hwang, J. An and J. Lee, “Object Detection Using CNN for Automatic Landing of Drones”, Journal of the Institute of Electronics and Information Engineers, Vol. 56, No. 5, May 2019, DOI : 10.5573/ieie.2019.56.5.82.
[10] S. Noh, S. Hong and M. Kim, “Rate-Controlled Data-Driven Real-Time Stream Processing for an Autonomous Machine”, Journal of Korea Robotics Society, Vol. 14, No. 4, December 2019.
[11] M. Hajinoroozi et al. Prediction of driver’s drowsy and alert states from EEG signals with deep learning. In IEEE Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2015.
[12] https://developer.nvidia.com/cuda-toolkit
[13] https://developer.nvidia.com/cudnn
[14] https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__STREAM.html
[15] https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit
[16] X. Yu, N. Zeng, S. Liu and Y. Zhang, “Utilization of DenseNet201 for diagnosis of breast abnormality”, in Machine Vision and Applications. vol. 30, Oct. 2019.
[17] L. Nguyen, D. Lin, Z. Lin and J. Cao, “Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation”, in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 2018.
[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proceedings of the International Conference on Learning Representations, San Diego, CA, 2015.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proceedings of the 26th Conference on Neural Information Processing Systems, Lake Tahoe, pp. 1097-1105, 2012.

저 자 소 개

김 도 영(학생회원)
2020년 한성대학교 정보시스템공학 학사 졸업.
<주관심분야 : NPU(인공지능 가속기), AI, 자율주행 자동차>

김 명 선(정회원)
2000년～2002년 LG전자 액세스네트워크 연구소 주임연구원
2002년～2011년 삼성전자 DMC 연구소 책임연구원
2016년 서울대학교 전기컴퓨터공학부 박사 졸업.
2016년～2019년 삼성전자 SR연구소 수석연구원
2019년～현재 한성대학교 IT융합공학부 조교수
<주관심분야 : NPU(인공지능 가속기), Linux kernel, HW/SW Co-desgin 등>