TVM 최적화에 미치는 하드웨어 및 프레임워크 영향 U-Net을 중심으로

김영곤
한양대학교 컴퓨터소프트웨어학과
Email: ntzkimy@gmail.com

이종현
한양대학교 디지털의료융합학과
Email: jonghyunlee1993@gmail.com

이찬우
한양대학교 인공지능학과
Email: leechanwoo25@gmail.com

***Abstract*—본 프로젝트는 의료 AI의 도입이 추진되고 있는 상황에서 의료 현장에서 사용 가능한 AI 모델의 가능성을 찾았다. 의료 분야는 충분한 컴퓨팅 리소스를 확보하는 것이 어렵고 보안 이슈로 클라우드 환경 구축이 어렵다. 따라서 본 프로젝트는 의료 현장에서 의료진 개개인이 접근 가능한 PC 등을 엣지 디바이스로 가정하고 엣지 디바이스 내에서의 최적화를 시도하였다. TVM은 다양한 하드웨어와 다양한 딥러닝 프레임워크를 지원하기 때문에 기기가 통일되지 않은 의료 현장에서 적극적으로 사용이 가능할 것이라 판단하였다. 2 X 2 디자인의 실험 결과, TVM의 auto-tuning 이후 추론 속도는 CPU에서는 저하되었으며, GPU에서는 개선되는 양상을 확인하였다. 또한 최적화로 얻는 이득은 Tensorflow에 비해서 Torch가 더 크다는 점을 확인하였다. 본 프로젝트는 최소한의 GPU가 구비된 엣지 디바이스에서 Torch 및 Tensorflow를 통한 AI 모델의 성능 최적화의 이점이 발생한다는 점을 밝혔다.**

1. 서론

최근 AI에 대한 관심의 증가는 산업 전반에 걸친 AI의 보급을 가져왔다. 이는 의료 분야 역시 무관하지 않아 최근 다양한 의료 AI들이 도입되고 있다. X-ray 기반의 COVID-19 진단에 AI를 도입하고자 하는 시도는 이를 잘 보여주는 예이다 . 그러나 의료 현장에서 AI 도입이 쉽지는 않은 문제인데 이는 아래와 같은 원인에서 기인한다.

- 통합된 데이터 플랫폼이 없기 때문에 발생하는 데이터 수집의 어려움
- 기기별, 병원별, 검사자별 능력에 따른 측정 결과의 불일치성
- 보험 수가 시스템으로 인한 AI 기반 의료 시스템 도입의 가치

대부분의 문제는 구조적 차원의 문제였으며, 데이터 3법의 통과, 보건의료 빅데이터 플랫폼의 출범 등 의료 데이터를 통합적으로 다루고자 하는 시도들이 대두되었다. 양질의 데이터 확보가 가능한 플랫폼을 바탕으로 의료 AI에 대한 관심은 다시 높아졌다. 최근 코스닥 상장에 성공한 뷰노는 대표적인 의료 AI 회사로서 의료 AI의 시장의 가능성을 다시금 보여주고 있다. 하지만 의료 AI의 보급을 위해서 해결해야할 가장 큰 문제는 의료 분야의 컴퓨팅 리소스 부족의 문제이다. 대부분의 컴퓨터는 GPU가 없는 CPU 위주의 환경이며, 내부적으로 데이터 서버를 구축하기에는 병원의 규모에 따라 제약이 많다. 이는 데이터를 활용하여 AI를 학습하거나 추론하는 과정에서 상당한 병목이 발생할 수밖에 없음을 의미한다. 따라서 의료 AI의 보편화를 위해서는 부족한 리소스에서도 잘 최적화된 딥러닝 모델이 필수적으로 요구된다.

[4] 는 스마트폰 등으로 대표되는 엣지 디바이스와 클라우드 환경을 최대한 활용하는 최적화 기법을 제안한다. 기존 대부분의 모바일 디바이스는 자신의 획득한 데이터를 3G, LTE, 4G 등의 무선 네트워크 환경을 통해 클라우드 환경으로 전달하는 방식을 택한다. 이는 일반적으로 엣지 디바이스의 컴퓨팅 리소스가 빈약하고 배터리가 작기 때문에 에너지 효율성이 중요하기 때문이다. 하지만 네트워크의 병목이 발생하기 때문에 전체 데이터를 전송하는 것 역시 가장 최고의 성능을 항상 보장하지는 않는다. 본 연구에서는 따라서 일반적인 CNN 모델을 지나며 데이터가 압축된다는 특징에 주목한다. 엣지 디바이스에서 어느 정도 데이터를 압축한 후에 네트워크를 통해 클라우드로 보내는 전략을 통해 네트워크의 병목과 계산 병목을 둘 다 중재하자는 아이디어이다. 에너지 효율화 전략과 계산 성능 효율화 전략 두 개의 회귀 모형을 생성하고 모델의 파라미터와 출력 값의 크기 등을 바탕으로 사용자의 선택에 따른 에너지 효율화 전략과 계산 성능 효율화 전략을 유동적으로 선택할 수 있도록 하였다. 의료 데이터는 민감한 개인정보를 포함하기 문에 외부로의 유출이 엄격히 제한되는 제약이 있다. 문에 앞서 Neurosurgeon이 제시한 것처럼 엣지 디바이스와 클라우드 환경의 밀접한 연계는 쉽지 않다. 많은 병원은 재원이 풍부하지 않아 자체 데이터 서버를 구축하는 것 역시 녹록치 않다. 따라서 필진은 엣지 디바이스에 해당하는 PC 환경에서 모델을 최적화하는 것이 중요한 이슈라고 판단하였다.

아파치 재단의 TVM 프로젝트 (이하 TVM)는 멀티 하드웨어 플랫폼과 멀티 딥러닝 프레임워크에 모두 대응할 수 있는 딥러닝 모델 최적화 프레임워크이다. TVM은 모델의 계산 그래프를 중간 표현(Intermediate Representation)으로 변경하고 이를 각 타깃 하드웨어 맞추어 최적화를 수행하게 된다. IR을 통해 최적화를 수행하기 때문에 TVM은 특정 HW에 종속되거나 특정 딥러닝 프레임워크에 종속되지 않는다는 점이 가장 큰 특징

1. https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

2. https://github.com/apache/tvm

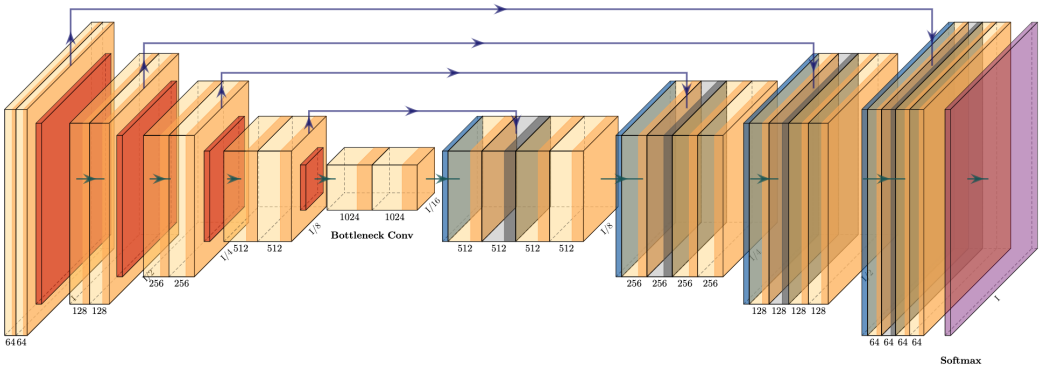

Figure 1: U-Net의 모델 아키텍처. 인코더 - 디코더의 구조를 띠고 있다[1].

으로 꼽을 수 있다. TVM이 제공하는 멀티 HW 플랫폼, 멀티 딥러닝 프레임워크 최적화 기능이 필진은 본 프로젝트의 목적과 잘 부합한다고 판단하였다. 의료 환경은 도입 시기, 장비의 특성 등에 따라서 구형 모델을 이용해서 구동하는 경우도 적지 않으며 GPU의 유무도 보장할 수 없기 때문이다. 또한 특정 기기에 따라서 ARM 프로세서에서 구동되어야할 필요성도 배제할 수 없다.

U-Net은 바이오 메디컬 분야에서 널리 쓰이는 segmentation 모델이다 [5]. U-Net은 convolution layer가 층층이 쌓여있는 U자 모양의 모델이다. 영상을 입력으로 받아 마스크를 출력으로 하는 모델이기 때문에 U-Net은 3 X 3의 convolution 커널 연산이 반복적으로 수행된다. 이 때문에 상대적으로 모델 연산이 무거운 편에 속한다. 이에 필진은 U-Net 모델을 TVM을 이용하여 최적화를 수행하고 이 결과를 비교함으로서 의료 AI의 도입에 있어 TVM 최적화의 성능을 비교할 수 있을 것이라 판단하였다. 이에 하드웨어의 조건(CPU, GPU)과 프레임워크 조건(Pytorch, Tensorflow)에 대한 반복 측정 2 X 2의 실험을 설계하여 TVM을 이용한 최적화 이전과 이후의 실행 시간 비교를 진행하였다.

2. 방법

본 프로젝트는 상술한 바와 같이 반복 측정 2 X 2의 디자인을 이용하여 진행되었다. 실험 조건은 각각 하드웨어와 프레임워크에 해당하며 따라서 다음 4개의 조건에 대한 최적화 이전과 이후의 성능을 비교하였다.

- CPU & Tensorflow
- CPU & Pytorch
- GPU & Tensorflow
- GPU & Pytorch

본 실험은 Linux Ubuntu 16.04 버전 환경과 Anaconda를 이용하여 구축된 Python 3.8 버전 환경에서 진행되었다. CPU는 Intel Xeon Processor (Skylake, IBRS) @2.29 x 16, GPU는 NVIDIA A100을 사용하였다. 비교를 위해 사용한 두 개의 딥러닝 프레임워크는 Tensorflow 2.4 버전과 Pytorch 1.4.1 버전이 사용되었다. TVM 0.8 버전은 공식 Github에서 제공하는 파일을 이용하여 빌드되었다. 학습을 위해 Kaggle 플랫폼에서 제공하는 Brain MRI Tumor segmentation 과제의 데이터를 사용하였다 . 해당 과제는 MRI 이미지와 tumor에 대한 마스크 이미지로 구성되어 있다. 전체 데이터의 20%를 검사 데이터로 할당하였으며, 80% 데이터의 다시 20%는 검증 데이터로 활용하였다. 남은 데이터는 모두 학습 데이터로 이용하였다.

본 프로젝트에서 이용된 모델은 Tensorflow 프레임워크를 이용하여 학습된 모델과 Pytorch 프레임워크를 이용하여 학습된 모델 총 2개가 사용되었다. 두 프레임워크의 모델 구조는 동일하였으며, 최종 성능이 유사할 수 있도록 Augmentation 과정, Optimizer의 하이퍼 파라미터 등을 최대한 유사하도록 디자인하였다. 본 프로젝트에서 학습의 성능은 크게 중요하지 않았는데, 이는 TVM 최적화를 진행하더라도 정확도 측면에서 차이가 발생하지 않기 때문이다. 즉, TVM 최적화는 시간 측면에서만 성능 튜닝이 발생하기 때문에 본 프로젝트에서 정확도는 깊이 있게 고려하지 않았다.

학습이 완료된 모델은 TVM의 Auto-tuner를 이용하여 각각의 타겟 하드웨어 플랫폼에 맞추어 최적화를 진행하였다 (Tuner: XG-Boost; N-Trial: 500; Early Stop: 200; Opt-level: 3). 모델의 성능은 아래 두 가지로 정의하였다.

1) Segmentation 과제의 정확도.
2) TVM 최적화 이전 / 이후 추론 시간.

Segmentation 과제의 정확도는 널리 사용되는 Dice Coefficient Score (DCS)를 사용하였다.

$$DCS = \frac{2|X \cap Y|}{|X| + |Y|}$$

추론 시간는 앞서 사전에 정의된 전체 데이터의 20% 하는 검사 데이터셋을 전체 추론 시간을 데이터의 숫자로 나눈 값으로 정의하였다. 이때 batch size는 1로 설정하였다. 또한 실행 환경 및 순간적 부하의 영향을 최소화하기 위해 동일한 추론 과정을 5번 반복하여 얻은 추론 시간의 평균 값을 사용하였다.

3. https://github.com/HarisIqbal88/PlotNeuralNet/blob/master/examples/Unet/Unet.pdf

4. https://www.kaggle.com/mateuszbuda/lgg-mri-segmentation

3. 결과

본 프로젝트의 실험 결과는 Table. 1에 정리되어 있다. 당연하게도 DSC의 차이는 발생하지 않았다. 이는 TVM이 최적화를 수행하는 논리가 수치를 변경하는 것이 아닌 구조를 효율화하기 때문이다. 본 실험 결과 TVM auto-tuning은 HW에 따른 차이가 발생하였다. CPU는 auto-tuning을 진행한 이후 추론 시간이 증가하는 경향을 보인 반면 GPU는 auto-tuning 이후 추론 시간이 짧아지는 경향을 찾을 수 있었다. 프레임워크의 주효과 보다는 프레임워크와 HW 타겟에 대한 상호작용 효과가 크게 작용한 것으로 나타났다. CPU에서 전반적으로 성능이 저하되었을 때, Torch는 상대적으로 적은 18.06%의 실행 시간이 증가한 반면 Tensorflow는 320.66%가 증가하였다. 반대로 GPU에서 성능 증가는 Torch의 경우, -177.84%인 반면 Tensorflow는 -13.31%로 상대적으로 적은 향상 폭을 보였다.

타겟 HW	프레임워크	성능					
		최적화 수행 이전		최적화 수행 이후		차이 (%)	
		추론 시간	DSC	추론 시간	DSC	추론 시간	DSC
CPU	Torch	0.818674	0.7678	0.999154	0.7678	+18.06	0
	Tensorflow	0.236142	0.7348	0.993360	0.7348	+320.66	0
GPU	Torch	0.008224	0.7678	0.002960	0.7678	-177.84	0
	Tensorflow	0.004530	0.7348	0.003998	0.7348	-13.31	0

Table 1: TVM의 auto-tuning을 이용하여 CPU, GPU에 대한 Torch, Tensorflow 모델 최적화 수행 결과. DSC (Dice Coefficient Score) 성능의 차이는 발생하지 않았으며 성능 차이는 추론 시간에서만 발생하였다.

4. 논의

본 프로젝트는 클라우드 환경을 적극적으로 이용하거나 자체 데이터 서버를 보유하지 못하여 상대적으로 컴퓨팅 리소스가 부족한 의료 현장에서 적용 가능한 의료 AI 모델의 가능성을 파악하고자 딥러닝 모델에 대한 TVM auto-tuning의 성능 개선 정도를 비교하였다. 본 프로젝트에서는 2 X 2 의 실험 디자인을 이용하여 하드웨어 조건 2가지(CPU, GPU), 널리 사용되는 프레임워크 조건 2가지 (Torch, Tensorflow)를 비교하였다. 실험의 결과, CPU에서는 전반적인 실행 속도 성능이 오히려 저하되는 경향성을 찾을 수 있었으며, GPU에서는 실행 속도 개선 효과를 찾을 수 있었다. 또한 TVM auto-tuning의 성능은 딥러닝 프레임워크에 따라서도 다른 양상을 발견하였다. Tensorflow는 최적화의 이후 성능의 Torch 보다 크게 저하되거나 상승하는 폭도 적었다. 하지만 본 프로젝트 이후 여러 가지 발견된 의문들이 존재한다.

- CPU에서의 TVM auto-tuning의 성능 저하 원인
- 동일한 계산 그래프의 형태를 가짐에도 불구하고 발생한 딥러닝 프레임워크에서의 성능 차이

본 프로젝트 진행 이전, 필진은 최적화를 수행하였기 때문에 자연스럽게 auto-tuning 이후의 추론 속도 성능이 개선될 것이라 기대하였다. 그러나 유독 CPU에서 추론 속도 성능이 크게 저하되는 현상이 발견되었으며, Tensorflow의 경우에는 기존의 추론 성능에 비해 3배가 넘는 시간이 소요되었다. CPU에서의 성능 저하는 최초 Tensorflow의 빌드 시간이 상대적으로 길기 때문에 발생한 것이라 추정하였다. 그러나 이 가설은 최적화 수행 이전의 추론 시간이 CPU에서 Torch가 더 길다는 점으로 기각되었다. 한편 최적화 이전의 추론 시간은 CPU와 GPU에서 모두 Tensorflow에서 Torch 보다 짧다는 특징이 발견되었다. 일반적으로 Tensorflow의 모델 빌드 시간이 Torch 보다 긴 것으로 알려져 있으나, 각각의 inference 시간에서 오히려 Torch보다 효율적이었다. 이는 Batch size가 1로 설정되어 있기 때문에 각각의 HW의 퍼포먼스를 최대한으로 활용하지 못한 상황에서 발생한 차이로 추정된다. 최적화 수행 이전과 이후, Tensorflow의 추론 속도 성능이 Torch에 비해서 모두 우수하였으나 추론 시간의 증가폭을 확인하였을 때, Torch는 18.06%만 증가한 반면, Tensorflow는 320.66%가 증가하였음을 볼 수 있다. GPU에서의 성능은 최적화 이전 Tensorflow가 Torch에 비해서 짧았으나 최적화 이후 Torch가 Tensorflow보다 짧아졌음을 확인할 수 있었다. 이는 Torch의 최적화 이후 성능 개선이 177.84%에 달한 반면 Tensorflow는 13.31%만 향상되었기 때문이다. 앞서 최적화 이후의 CPU에서의 성능과 GPU에서의 성능을 종합하여 고려하자면 다음과 같은 결론을 도출할 수 있었다. 제한된 TVM의 auto-tuning을 이용하였을 때, 성능의 향상폭은 Torch로 구성된 모델이 더 적합하였다. 즉, Tensorflow는 CPU에서 성능 감소 폭이 Torch에 비해 더 컸으며 GPU에서의 성능 개선 폭은 Torch 보다 더 작았다. 현재 결과에 따르면 TVM의 auto-tuning은 Tensorflow보다 Torch에 적합한 방식으로 추정된다. 두 번째 의문을 검증하기 위해 본 필진은 TVM의 debugging 기능을 사용하여 각각의 계산 그래프에서 소요되는 시간을 확인하는 작업을 진행하고자 하였다. 동일한 계산 그래프를 가지고 있음에도 불구하고 두 프레임워크에서의 차이가 발생하는 것은 TVM에서 처리하는 방식이 다를 것이라고 가정하였다. 만약 모델의 계산 시간을 레이어 별로 확인할 수 있다면 크게 차이가 발생하는 구간을 발견할 수 있을 것이다. 그러나 불안정한 TVM의 환경으로 인해 여러 차례 디버깅이 가능한 환경 구축에는 실패하여 본 의문을 해소하기 위한 추가적 분석은 진행하지 못하였다.

본 프로젝트에서 발견된 패턴 중 하나는 추론을 여러 차례 반복할 때, 각각의 추론 시간의 편차가 발생한다는 점이었다. 예를 들어, Torch 환경에서 구축된 모델을 최적화 하지 않고 CPU에서 25번 연속으로 실행했을 때, Table. 2와 같은 결과를 얻었다. 실행 환경의 안정성에 따라 편차가 발생하였을 가능성은 배제할 수 없으나 그럼에도 불구하고 최적화 수행 이후, 각 편차가 2배 이상 발생하는 경우가 발생한다는 것을 확인할 수 있다. 예를 들어 22번 시행에서 최적화 이후 추론 성능은 0.006982 초였으나 23번 시행에서 추론 성능은 0.012715으로 큰 편차를 보였다. 그러나 본 프로젝트에서 추론 성능으로 정의한 5번의 성능은 안정적으로 유지되었기 때문에 최종 Torch에서의 추론 성능은 우수한 것으로 간주되었다. 그러나 최적화 수행 이후 편차가 크게 발생하는 시행들이 발생하는 것으로 보아 TVM의 안정성에 대한 추가 검증이 필요할 것으로 보인다. 향후 연구는 CPU와 GPU에서 추론 성능의 차이가 발생하는 이유, 각 프레임워크의 모델이 계산 그래프에서 최적화를 레이어 단위로 디버깅하는 과제, 최적화 이후 안정적이지 않은 성능 편차를 보이는 이유에 대한 고찰이 필요할 것으로 보인다. 그럼에도 본 프로젝트는 의료 현장에서 AI 모델을 활용하기 위한 최적화의 가능성

시행	auto-tuning 이전	auto-tuning 이후
1	0.009751	0.006190
2	0.008802	0.006077
3	0.009093	0.006083
4	0.009569	0.006099
5	0.009820	0.006105
6	0.009830	0.006097
7	0.009253	0.006095
8	0.009762	0.006378
9	0.010656	0.005867
10	0.009312	0.006208
11	0.008855	0.005910
12	0.009057	0.006541
13	0.008941	0.006053
14	0.008926	0.006192
15	0.008848	0.006394
16	0.008911	0.006185
17	0.009471	0.012804
18	0.009862	0.012013
19	0.011011	0.013706
20	0.010455	0.012411
21	0.009348	0.013323
22	0.009542	0.006982
23	0.009954	0.012715
24	0.009887	0.013390
25	0.008778	0.007091

Table 2: GPU 환경에서 실행한 Torch 모델에서의 auto-tuning 이전과 이후의 추론 속도 성능 비교. 총 25번의 반복 추정 결과이며 auto-tuning 이전에 비해서 auto-tuning의 성능 편차가 커짐을 확인할 수 있다.

을 제시했다는 점에 의의가 있다. 부족한 컴퓨팅 리소스의 환경에서 최대한의 성능을 끌어내기 위해 TVM의 auto-tuning은 가능한 대안으로 제시될 수 있으며, 엣지 환경을 최대한 활용해야하는 의료 환경에서 최소한의 GPU 환경이 구축되어 있다면 최적화를 통해 AI 모델의 추론 시간 성능을 이끌어낼 수 있다는 점을 시사한다.

5. 부록

본 프로젝트의 코드 구현은 https://github.com/jonghyunlee1993/AIPlatformOptimization/tree/master/Project_optimizing_U-net에서 확인할 수 있다.

References

[1] TVM official site : https://tvm.apache.org/

[2] PyTorch official site : https://pytorch.org/

[3] Tensorflow official site : https://www.tensorflow.org/

[4] Kang, Y., Hauswald, J.Tang, L. (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ACM SIGARCH Computer Architecture News, 45(1), 615-629.

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.