반도체공학회 논문지(TSE: Transactions on Semiconductor Engineering), Vol. 1, No. 1, Oct., 2023
https://doi.org/10.22895/tse.2023.0004 ISSN (On-line) 3022-2028

# 효율적인 Transformer 모델 경량화를 위한 구조화된 프루닝

## Structured Pruning for Efficient Transformer Model compression

류은지 [1], 이영주 [1,+]
(Eunji Yoo[1] and Youngjoo Lee[1,+])

**요 약**

최근 거대 IT 기업들의 Generative AI 기술 개발로 Transformer 모델의 규모가 조 단위를 넘어가며 기하급수적으로 증가하고 있다. 이러한 AI 서비스를 지속적으로 가능케 하기 위해선 모델 경량화가 필수적이다. 본 논문에서는 하드웨어 친화적으로 구조화된(structured) 프루닝 패턴을 찾아 Transformer 모델의 경량화 방법을 제안한다. 이는 모델 알고리즘의 특성을 살려 압축을 진행하기 때문에 모델의 크기는 줄어들면서 성능은 최대한 유지할 수 있다. 실험에 따르면 GPT-2 와 BERT 언어 모델을 프루닝할 때 제안하는 구조화된 프루닝 기법은 희소성이 높은 영역에서도 미세 조정된(fine-grained) 프루닝과 거의 흡사한 성능을 보여준다. 이 접근 방식은 미세 조정된 프루닝 대비 0.003%의 정확도 손실로 모델 매개 변수를 80% 줄이고 구조화된 형태로 하드웨어 가속화를 진행할 수 있다.

**ABSTRACT**

With the recent development of Generative AI technology by IT giants, the size of the transformer model is increasing exponentially over trillion won. In order to continuously enable these AI services, it is essential to reduce the weight of the model. In this paper, we find a hardware-friendly structured pruning pattern and propose a lightweight method of the transformer model. Since compression proceeds by utilizing the characteristics of the model algorithm, the size of the model can be reduced and performance can be maintained as much as possible. Experiments show that the structured pruning proposed when pruning GPT-2 and BERT language models shows almost similar performance to fine-grained pruning even in highly sparse regions. This approach reduces model parameters by 80% and allows hardware acceleration in structured form with 0.003% accuracy loss compared to fine-tuned pruning.

KEY WORDS
Transformer 모델 압축; 구조화된 프루닝; 알고리즘-하드웨어 최적화;

## I. 서 론

딥러닝 모델 중의 하나인 Transformer [1]의 등장으로 AI 기술의 발전 속도가 폭발적으로 증가하고 있다. Transformer 모델은 모델 크기가 커지면 모델의 정확도 성능 역시 비례하여 높아지기에 CNN보다 더 빠른 속도로 모델의 크기를 키워 성능을 높이는 연구가 활발하게 진행되고 있다. GPT-3 [2], HyperCLOVA, PaLM [3] 등의 초거대 언어 모델의 등장으로, 거대 Generative Language Model을 사용한 새로운 AI 서비스의 가능성이 크게 열리고 있으나, 기하급수적으로 늘어난 성능만큼이나 모델 사이즈도 방대하게 커지고 있다. 이미지 처리 AI 기술인 CNN은 모델 훈련에 드는 비용 규모가 2년에 15배씩 증가한 데에 반해 Transformer는 2년에 무려 750배씩 증가하고 있다. 이는 현재 진행형이며 증가 속도는 굉장히 가파르다. 특히 요즘 뜨겁게 주목받고 있는 생성형 AI 모델, ChatGPT의 모델 크기는 대략 1조 8,000억까지 커졌다.

[1]Pohang University of Science and Technology
[+]Corresponding author: Youngjoo Lee, youngjoo.lee@postech.ac.kr
(Received Aug. 31, 2023, Revised Oct. 6, 2023, Accepted Oct. 12, 2023)

하지만 아무리 많은 저장 공간과 처리 속도를 가진 서버일지라도 수십~수백 기가바이트 이상으로 커진 모델을 가지고 학습과 추론을 진행하면 많은 에너지가 소비되어 서버에 큰 부담이 된다. 이러한 서비스를 제공하여 이윤을 내는 기업에서는 엄청난 연산 오버헤드(overhead)로 인한 방대한 서비스 비용이 문제가 된다. 따라서 Transformer 알고리즘의 대중적 성공을 위해서 기존보다 적은 에너지를 사용하는 다양한 최적화 기법들이 활발하게 연구되고 있다.

본 논문에서는 현재 가장 각광 받는 Transformer 알고리즘을 경량화하여 딥러닝 하드웨어의 에너지 및 복잡도 효율을 개선할 수 있는 다양한 연구를 소개한다. 일반적으로 Transformer 알고리즘은 수많은 완전 연결(fully connected)된 입력과 가중치(weight)의 행렬-곱 연산에서 매우 많은 곱셈 연산을 유발하기에, 이 과정에서 소비되는 에너지를 최적화하기 위한 기술 개발이 시급하게 요구되고 있다. 따라서 최근의 Transformer 알고리즘 압축 기술에 관련된 연구들은 방대해지는 모델 크기에서 생기는 모델 복잡도를 낮추면서 알고리즘의 정확도 성능은 최대한 유지하는 기술 개발에 큰 노력을 기울이고 있다. 본 논문에서는 최신 최적화 기술 중에서 프루닝(pruning)[4]에 대해 중점적으로 다루며 이를 통하여 향후 생성형 AI에 기반이 되는 Transformer 압축 기법 개발에 요구되는 기술들을 논의하고자 한다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 Transformer 소개와 Transformer의 다양한 압축 기법들을 살펴보고 대표적인 방법인 프루닝에 대해 자세히 기술한다. 위 기법들을 적용하여 압축된 Transformer 모델의 정확도 성능 결과에 대해 제 Ⅲ장에서 분석하며, 제 Ⅳ장에서는 프루닝 기법이 하드웨어에 적용했을 때 발생할 수 있는 문제와 그에 따른 해결 방안을 다룬다. 마지막으로 Ⅴ장에서는 결론과 함께 보다 효율적인 Transformer 모델 압축을 위한 추가적인 고려사항들을 제안한다.

## II. 본 론

### 1. Transformer model과 연산량

Transformer 기반의 인공 신경망 알고리즘은 기존 RNN의 장기 의존성 문제를 극복하고 더 나아가 CNN보다 개선된 사물 인식률을 보이며, 이를 통해 더욱 신뢰도가 높은 정확도를 생성한다.

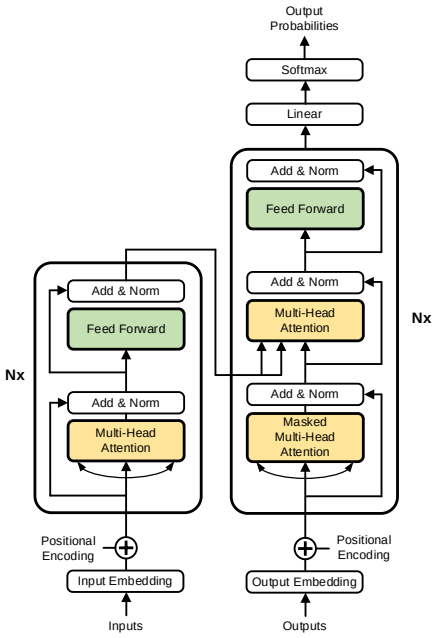


그림 1. Vanilla Transformer[1]의 block diagram

기본적인 Transformer는 번역 작업과 같은 시퀀스 대 시퀀스 작업을 위해 인코더-디코더(encoder-decoder) 구조로 이루어져 있다. 인코더는 입력 문장을 표현하는 역할을 하며, 디코더는 이 표현을 바탕으로 출력 문장을 생성한다. 이러한 특징을 바탕으로 다양한 형태의 Transformer가 제안되었다. 본문 1에서는 여러 형태의 Transformer 종류와 특징에 대해 소개한다.

#### (1) Vanilla transformer 모델

Vanilla Transformer[1]는 그림 1에 따라 기존의 seq2seq 구조인 인코더-디코더를 따르며, 어텐션 매커니즘(attention mechanism) 만으로 높은 성능을 달성한 모델이다. 기존 seq2seq 모델[5]은 인코더가 입력 시퀀스를 하나의 벡터로 압축하는 과정에서 입력 시퀀스의 정보가 일부 손실된다는 단점이 있었지만, Transformer 모델은 어텐션 매커니즘을 통해 위 문제점을 해결할 수 있어 큰 주목을 받았다. Seq2seq 알고리즘과 달리 반복을 회피하고 입력과 출력 사이의 전역 의존성을 그리기 위해 어텐션 매커니즘에 전적으로 의존하여 RNN 및 CNN과 같은 방법보다 훨씬 더 병렬화할 수 있다.

기본적인 Transformer 알고리즘의 연산 과정은 셀프 어텐션 레이어(self-attention layer)와 피드포워드 네트워크 레이어(feed-forward network layer)를 담고 있는 인코더와 디코더 layer가 여러 개로 연결되어 있다. 특히 Transformer 알고리즘에 처음 도입된 셀프 어텐션 매커니즘은 입력 시퀀스 내의 단어 간 관계를 학습하며 병렬 처리가 가능한 구조를 가진다. 따라서 CNN, RNN보다 훨씬 많은 연산량을 처리하지만 시퀀스 내의 모든 단어를 한 번에 처리하므로 병렬화가 용이하여 모델의 학습 속도를 높이는 데 도움을 준다. 또한 멀티 헤드 어텐션(multi-head attention) 기법을 사용하여 셀프 어텐션을 여러 가지 관점에서 수행한다. 이는 입력 데이터를 여러 다른 특징 공간으로 투영하여 각각 다른 측면에서 정보를 학습하고 결합함으로써 더 정밀하고 복잡한 표현을 얻을 수 있도록 도와준다.

### (2) 인코더 기반의 Transformer 모델

BERT(Bidirectional Encoder Representations from Transformers)[6]는 Transformer의 인코더를 여러 층으로 쌓아 올린 구조로 양방향 언어 모델링을 수행한다. Vanilla Transformer와 달리 사전훈련(pre-trained)과 전이 학습(transfer learning)에 중점을 둔 모델로, 대규모 텍스트 데이터로 사전 훈련된 이후 특정 작업에 맞게 미세 조정(fine-tuning)을 한다. Transformer의 인코더는 입력 문장을 표현하는 역할을 하며 대표적인 어플리케이션으로는 문장의 특정 단어에 마스킹하고 알맞은 단어를 찾거나 다음 문장을 예측하고 주어진 문서 내에서 정답을 찾아내는 기계 독해 문제 등이 있다. 이러한 작업을 수행하기 위해서 BERT는 GLUE[7], SQuAD, WikiText 등의 데이터셋을 활용한다.

데이터의 행렬-곱 연산이 수행되는 어텐션 레이어와 피드포워드 네트워크 레이어는 vanilla Transformer와 동일하게 구성되어 있다. BERT는 크게 2가지 모델, base와 large로 존재하며 각각 110M와 340M개의 파라미터 수를 갖는다. 알고리즘과 데이터셋 종류에 따라 가중치의 값 분포와 희소성 경향도에서 차이가 발생할 수 있다.

### (3) 디코더 기반의 Transformer 모델

GPT (Generative Pre-trained Transformer)[2]와 PaLM (Pathways Language Model)[3]는 디코더를 여러 층으로 쌓아 올린 구조로 문장 생성 작업과 같은 생성 태스크에 특화되어 있다. 주어진 문맥 내에서 자연스러운 문장 생성을 목표로 하며, 문장의 흐름을 따라가면서 이전 단어들로부터 다음 단어를 예측하고 생성하는 과정을 수행한다. 생성형 AI 알고리즘은 BERT와 다르게 파라미터의 수가 증가함에 따라 모델의 성능도 상승하기에 모델의 크기가 기하급수적으로 증가하고 있다. GPT와 PaLM은 자동 회귀 언어 모델로, 텍스트를 생성하기 위해 이전 단어의 문맥을 사용한다. 반면에, BERT는 마스크 언어 모델로 텍스트를 이해하기 위해 양방향의 문맥을 사용한다. 이러한 차이로 인해 GPT와 PaLM은 BERT보다 더 많은 파라미터가 필요하다. 또한 생성형 AI의 시장성과 미래 확장성이 훨씬 크기에 모델 크기의 증가 속도가 굉장히 빠르다. GPT-3와 PaLM은 각각 175B와 540B 개의 파라미터 수를 갖는다.

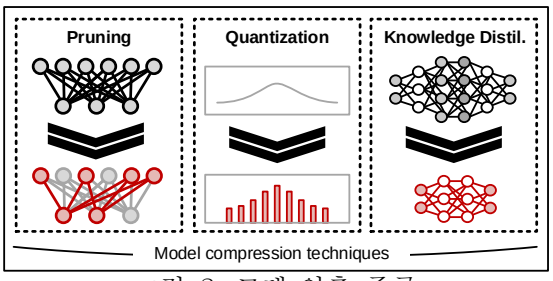

그림 2. 모델 압축 종류

이러한 생성형 AI 알고리즘을 평가하기 위한 데이터셋으로는 LAMBADA[8], WikiText-103 등으로 100만 개가 넘는 단어이며 문맥을 이해해야만 맞출 수 있는 단어들을 포함하고 있다. 그리고 언어 모델의 성능을 평가하는 지표로 perplexity를 사용한다. 이는 언어 모델이 주어진 텍스트를 얼마나 잘 예측하는 지를 측정한다. Perplexity가 낮을수록 다음 단어로 적은 수의 후보만 고려한다는 것으로 더 확신을 두고 다음 단어를 선택할 수 있다.

## 2. 모델 압축을 통한 최적화

### (1) 모델 압축 종류 및 특징

모델 압축은 딥러닝 모델의 성능을 유지하면서 크기를 줄이는 기법이다. 그림 2에 나타낸 바와 같이, 대표적으로 프루닝, 양자화(quantization), 지식 증류(knowledge distillation) 등이 있다.

양자화는 모델의 파라미터나 활성화 함수의 값을 정해진 범위 내에서 표현하는 방법이다. 예를 들어, 32비트 부동소수점 수를 8비트 정수로 바꾸는 것이 양자화의 한 예이다. 이렇게 하면 모델의 메모리 사용량과 연산 속도를 개선할 수 있다 [9]-[12]. 두 번째로 지식 증류는 큰 모델(teacher network)로부터 작은 모델(student network)에 지식을 전달하는 방법이다. 이런 구조를 가지고 여러 번 훈련을 시키며 작은

모델이 큰 모델의 성능을 따라잡거나 뛰어넘을 수 있다. 훈련 비용이 많이 소모되지만 프루닝과 같이 모델의 크기를 줄여 메모리 사용량과 연산 횟수를 감소시킬 수 있다 [13],[14].

프루닝은 파라미터 중에서 중요도가 낮은 것들을 제거하여 모델의 크기를 줄이는 가장 직관적이고 효과적인 방법이다. 모델의 파라미터 수를 줄이기에 모델의 메모리 사용량과 연산 횟수를 모두 감소시킨다.

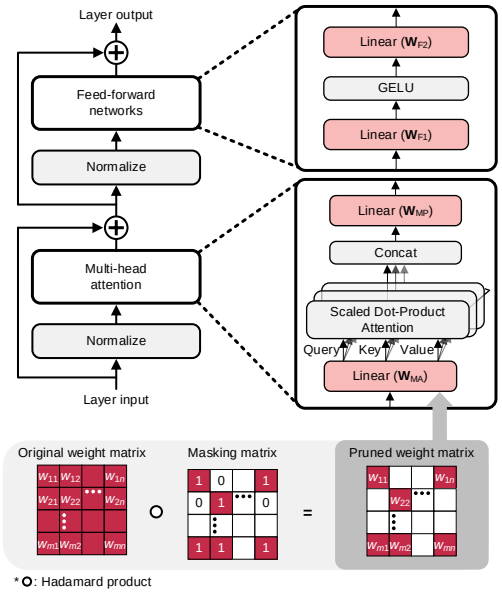


그림 3. Transformer 프루닝 계층 구조

중요도를 판단하는 지표는 다양하게 존재한다. 가중치의 절댓값 크기순으로 줄을 세우거나 학습하는 동안의 가중치의 변화량을 본다거나 혹은 가중치를 제거했을 때 변화되는 모델의 손실 분포 등에 따라 가중치의 중요도를 판단한다 [15]–[17]. 이에 대한 자세한 내용은 본문 2.(2)에서 기술한다.

### (2) Transformer 프루닝 기법

그림 3은 Transformer 모델의 일반적인 계층 구조를 보여주며, 멀티 헤드 어텐션($W_{MA}$)과 멀티 헤드 프로젝션($W_{MP}$) 및 2개의 피드포워드 네트워크( $W_{F1}$ , $W_{F2}$ )의 가중치[1]를 나타내는 4개의 주요 가중치 행렬이 있음을 보여준다. 임베딩 크기 d의 경우 행렬 치수는 각각 3d × d, d × d, 4d × d, d × 4d로 전체 Transformer 복잡성[9]을 가진다. 미세 조정 절차와 결합하여, 프루닝 기반 가중치 희소성(weight sparsity)은 원래 모델 정확도를 유지하면서 모델 복잡성을 줄이기 위해 널리 사용되는 방법이다. 그림 3에 나타낸 바와 같이, 프루닝 방법은 덜 중요한 가중치 위치를 나타내기 위해 각 가중치 행렬에 대한 마스킹 행렬을 생성하고, 프루닝되지 않은 가중치는 마스킹 및 가중치 행렬의 Hadamard 곱셈을 수행한 후 0이 아닌 가중치를 수집하여 얻는다. 단순성을 위해, 본 연구에서는 Transformer 계층에서 가중치 행렬에 해당하는 4개의 마스킹 행렬을 정의하며, 이는 $M_{MA} \in \{0,1\}^{3dxd}$, $M_{MP} \in \{0,1\}^{dxd}$, $M_{F1} \in \{0,1\}^{4dxd}$, 그리고 $M_{F2} \in \{0,1\}^{dx4d}$로 표현된다.

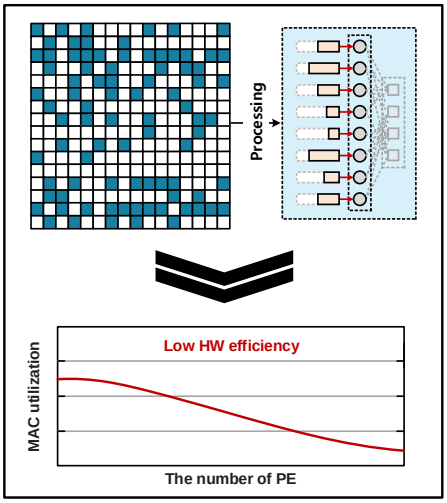


그림 4. 미세 조정된 프루닝의 문제점

각 마스킹 행렬을 구성하기 위해 본문 2.(1)에서 언급했듯이 여러 가지 기법들이 존재한다. 기초적인 방법인 크기 기반(magnitude) 프루닝 기법은 가중치 값이 0에 가까운 것을 먼저 제거한다[4]. 그리고 [18]에서 나온 움직임 기반(movement) 프루닝은 훈련 과정 동안의 가중치 변화량이 가중치 중요성을 나타낸다는 것을 보여주는데, 이는 주어진 프루닝 비율인 $\delta$에 대해 더 정확한 압축 모델을 제공하는 프루닝 방법이다. 마지막으로 가중치 행렬에서 헤시안 행렬의 고유값과 고유벡터를 이용하여 중요도가 낮은 값을 0으로 만드는 방법이 있다 [19],[20]. 해당 값을 얻기 위해서는 Optimal Brain Surgeon 프레임워크에 기반한 근사 이차 정보를 활용해야 한다. 정확하고 높은 성능을 확보할 수 있지만 이차 근사를 수행으로 드는 연산 복잡도가 크다. 그림 4에 따르면, 기본 프루닝은 미세 조정된(fine-grained) 접근 방식으로 간주하기 때문에 일반적으로 무작위로 희소한 가중치 행렬을 관찰하여 정규화된 데이터 수준 병렬에 초점을 맞춘 기존 컴퓨팅 플랫폼의 하드웨어 효율성을 심각하게 저하시킨다. 일부 하드웨어 친화적인 프루닝 방법은 에너지 효율적인 CNN 처리 [21], [22]를 위한 구조화된 패턴을 생성하지만, 정확도 저하가 심각하기 때문에 소형 모델을 만들기 위해 $\delta$를 증가시키는 것이 보통 제한된다.

### (3) 구조화된 프루닝의 중요성 및 제안

기존 CNN 모델을 대상으로, 구조화된 프루닝 방법은 채널별 프루닝[21] 및 벡터별 프루닝[22]와 같은 원하는 프루닝 수준을 조사하여 널리 연구되어 왔다. 일반적으로, 주어진 프루닝 비율 $\delta$에 대해, 미세 조정된 프루닝은 원래의 프루닝 되지 않은 네트워크와 비교하여 가장 가까운 정확도를 제공하며, 프루닝 구조의 크기를 증가시키면 모델 정확도를 저하시킬 가능성이 더 크다. 따라서 적절한 프루닝 구조를 찾기 위해 프루닝되지 않은 가중치의 위치를 주의 깊게 분석하는 것이 중요하다. 최근의 Transformer 모델의 경우, 여러 연구에서 어텐션 비용 [23], [24]을 완화하기 위한 헤드 레벨(head-level) 중요성을 정의할 수 있다는 것을 밝혀냈다. 그러나, 공격적인 헤드 레벨 프루닝은 모델 성능을 크게 저하시켜 $\delta$ < 0.3이 원래의 정확도 [23], [24]를 유지하도록 하는 데 최선이다.

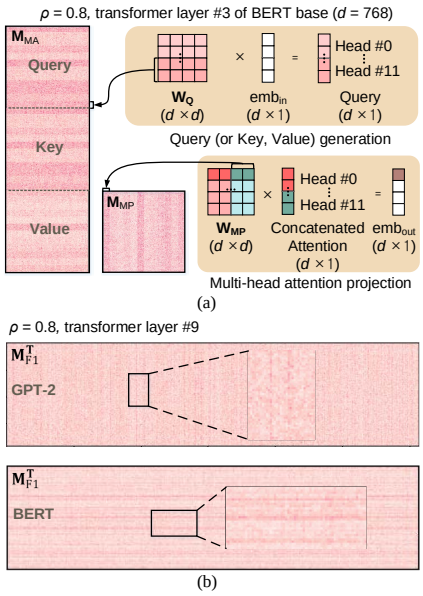


그림 5. Transformer 마스킹 패턴

또한, 헤드 단위의 프루닝이 급격한 성능 저하와 낮은 전체 프루닝 비율을 야기시키므로 이후 전체 가중치 영역에 블록 단위의 프루닝을 적용하는 방법 [25]이 제안되었다. 기본적으로 미세 조정된 프루닝을 바탕으로 도출된 모델 파라미터에 블록 단위로 마스크를 적용한다. 이는 멀티 헤드 어텐션 영역뿐만 아니라 피드포워드 네트워크 영역에도 적용하여 전체 프루닝 비율을 높여 SQuAD v1에서 BERT 모델을 2.4배 빠르고 74% 작게 만들면서 F1 점수는 1%만 감소시키는 등의 결과를 보여준다.

최신의 움직임 기반 프루닝에서 비롯된 미세 조정된 마스킹 행렬을 기반으로 본 연구는 고유 마스크 패턴을 세심하게 분석하여 프루닝 구조에 대한 올바른 결정을 내린다[26]. 프루닝된 GPT-2의 경우($\delta$ = 0.8), 그림 5(a)는 $M_{MA}$와 $M_{MP}$의 마스킹 패턴을 시각적으로 보여준다. 미세 조정된 프루닝 패턴은 $M_{MA}$와 $M_{MP}$에 대해 각각 수평적 및 수직적으로 지배적이며, 이는 다중 헤드 어텐션 연산의 처리 단계에 의해 도출된 합리적인 관측치임이 분명하다. 보다 구체적으로, 프루닝되지 않은 가중치 행렬 $W_{MA}$는 3개의 내부 d×d 쿼리($W_Q$), 키($W_K$) 및 값( $W_V$ ) 행렬을 수직으로 적층하여 구성되며, 각각은 그림 5(a)에 나타낸 바와 같이 $M_{MA}$의 수평적 진일보로 이어지는 여러 개의 헤드를 수직으로 포함한다. 반면, 다음 마스킹 패턴 $M_{MP}$는 강한 어텐션 부분이 그림 5(a)에 나타낸 바와 같이 훈련 단계 동안 $W_{MP}$의 해당 열을 강조하기 때문에 수직 진일보를 가진다.

또한 피드포워드 네트워크를 조사하여 주요 프루닝 방향을 찾아보면, 여기서 마스킹 행렬 $M_{F1}$은 프루닝 0.8된 GPT-2 및 BERT 모델에 대해 전치 상태(transpose)로 그림 5(b)에 시각화된다. 이제 서로 다른 Transformer 모델에 대해 분명히 두 가지 다른 방향이 있다. 즉, GPT-2는 수평선이 있는 반면 BERT는 수직선으로 표시된다. 실제로 GPT-2의 $W_{F1}$ 은 $M_{F1}$ 에서 행별 패턴을 유도하는 토큰 생성 단계에 개념적으로 사용된다. 반면 BERT의 $W_{F1}$ 은 기본적으로 $M_{F1}$에서 열별 패턴을 만드는 토큰 선택 프로세스를 수행한다. $M_{F2}$ 의 경우 마지막 가중치 행렬 $W_{F2}$가 열 별 가중치 합에서 출력 임베딩 벡터의 각 요소를 만드는 데 사용되기 때문에 프루닝된 GPT-2 및 BERT 모델 모두 수직 정렬 마스킹 패턴을 보여주는 것으로 관찰된다. 이렇게 관측된 가중치 마스킹 패턴을 바탕으로 모델 성능이 유효하게 떨어지지 않는 선에서 구조 크기를 최대한으로 높이는 절충안을 찾아야 한다. 더불어, 마스킹 매트릭스의 방향 강도는 여러 가지 방법을 통해 정량적으로 평가할 수 있으며, 각 미세 조정된 마스킹 매트릭스의 주요 가지치기 방향을 파악해야 한다. 가장 단순하고 직관적인 방법은 수직 및 수평 방향의 분산을 계산하고, 이 두 방향 간의 차이를 통해 마스킹 매트릭스의 방향 강도를 평가하는 것이다. 이러한 방법은 간단하면서도 효과적이며, 보다 정교한 정량적 평가 방법을 고안할 수도 있다. 이렇게 얻은 마스킹 매트릭스의 방향 강도를 활용하여 모델 성능을 현저하게 저하시키지 않고 구조 크기를 극대화하기 위한 타협적인 해결책을 찾아 나가야 한다.

## III. 결과 분석

프루닝 전략에 의해 영향을 받는 ML 모델 품질을 공정하게 비교하기 위해 Ⅱ장 본문 2 에서 설명한 미세 조정된 프루닝[18], 헤드 레벨 프루닝[23], [24], 블록 레벨(block-level 프루닝[25] 및 혼합 벡터 레벨(mixed vectors-level)[26] 프루닝을 포함한 다양한 접근 방식을 테스트했다. 사례 연구에서는 서로 다른 응용 프로그램을 가진 두 가지 Transformer 모델이 사용되었다. 1) SQuAD v1.1 의 BERT 와 2) Wikitext-2 의 GPT-2. 표 1 은 모든 모델이 16 비트 부동 소수점을 기반으로 하는 서로 다른 프루닝 비율에 대한 실험 결과를 요약한다.

표 1. 프루닝된 Transformer의 성능표

| Pruning method | Pruning ratio | BERT on SQuAD v1.1 (F1 score) | GPT-2 on Wikitext -2 (PPL) |
|---|---|---|---|
| Baseline | 0 | 87.54 | 21.3 |
| Fine-grained [18] | 0.60 | 82.72 | 24.2 |
| | 0.70 | 82.41 | 27.4 |
| | 0.80 | 81.88 | 30.1 |
| Head-level [23] | 0.13 | 87.67 | 21.9 |
| | 0.22 | 86.45 | 27.3 |
| | 0.24 | 82.29 | 39.7 |
| Block-level [25] | 0.60 | 82.62 | 32.3 |
| | 0.70 | 81.85 | 49.1 |
| | 0.80 | 78.28 | 66.7 |
| Mixed vectors -level [26] | 0.60 | 84.27 | 24.3 |
| | 0.70 | 83.37 | 26.3 |
| | 0.80 | 82.82 | 30.2 |

0.6 이상의 프루닝 비율을 적용하면 최신 미세 조정된 프루닝이 프루닝되지 않은 버전과 비교하여 허용 가능한 모델 성능을 달성한다. 헤드 레벨 프루닝은 덜 중요한 head 를 완전히 제거하여 가장 구조화된 패턴을 생성하지만 표 1 과 같이 품질 저하가 심하여 프루닝 비율을 0.3 이상으로 높이기가 어렵다. 블록 레벨 프루닝은 미세 조정된 프루닝의 파라미터 중요도를 바탕으로 전체 가중치 영역에 대해 블록 단위의 마스킹을 형성하지만, 구조화된 크기가 커서 품질 저하가 상당하다.

마지막으로 혼합 벡터 레벨 프루닝은 제안된 방향 강도를 확인하여 프루닝 방향과 강도를 모두 고려하는 혼합 길이 벡터로 구조화된 프루닝 패턴을 구성했다. 매트릭스 특성을 반영하는 적응적 프루닝 방향으로 인해 여전히 미세 조정된 프루닝과 동등한 모델 성능을 제공할 수 있다. 혼합 벡터 레벨 프루닝은 구조 크기를 증가시키면서도 모델 품질을 유지한다. 예를 들어 혼합 벡터 레벨 프루닝은 결과와 거의 동일한 GPT-2 가중치의 20%만 유지하면서 PPL 30.2 를 달성한다. 허용 가능한 모델 성능으로 큰 프루닝 비율($\delta > 0.6$)을 지원함으로써 혼합 벡터 레벨 프루닝은 처음으로 매트릭스 특성을 활용하여 조정 가능한 구조화 패턴을 성공적으로 활용하여 전용 희소성 인식 처리로 하드웨어 효율성을 향상시킬 수 있다.

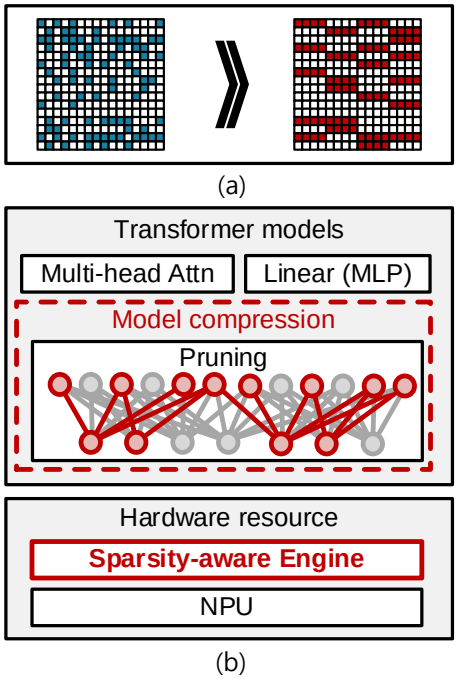


그림 6. 전용(dedicated) 하드웨어의 필요성

## IV. 후속 연구

행렬-곱으로 표현되는 Transformer의 주요 연산들은 프루닝 작업이 적용될 경우 해당 행렬의 내부가 대부분 0인 희소 행렬(sparse matrix)의 형태를 보이게 된다. 이를 일반적인 하드웨어에 적용하면 희소 패턴을 푸는 오버헤드가 자연적으로 발생한다. 대신 희소 행렬을 CSR (Compressed Sparse Row) 또는 CSC (Compressed Sparse Column) 방식으로 저장하여, 데이터 저장 공간을 줄일 수 있다. CSR 또는 CSC는 원래의 행렬을 다음과 같이 열 인덱스 값, 행 압축정보 [최초 시작행 번호, 시작행의 데이터 개수, 두 번째 행의 데이터 개수, … , 마지막 행의 데이터 개수], 데이터의 배열로 나타내는 것으로, 0이 아닌 값의 개수를 a,

열의 개수를 n이라 했을 때 2a + n + 1개의 데이터로 원래의 희소 행렬을 모두 표현할 수 있다. 최근의 연구 결과는 그림 6과 같이 프루닝과 CSR/CSC 기법을 적절하게 사용할 경우, Transformer의 인식률을 거의 떨어뜨리지 않고 주요 파라미터를 저장하기 위한 메모리의 크기를 4배만큼 줄일 수 있음을 보고하고 있다 [27]. 하드웨어 수준의 성능을 저하시키지 않고 실질적인 프루닝된 Transformer 모델을 작동하기 위해서는 압축 해제기(decompressor), 제로 스킵 컨트롤러(zero-skip controller), 매칭 장치(matching machine), 온라인(또는 오프라인) 스케줄링 엔진(online scheduling engine) [28]-[32] 등 전용 하드웨어 장치가 필요하다. 부하 분산 오프라인 가중치 스케줄링(offline load balancing weight scheduling) 및 집합 연관 PE 관리(set-associated PE management)와 관련된 고급 매칭 알고리즘을 활용하는 [28]의 최신 Transformer 가속기를 고려해야 한다. 그러나 세분화된 매칭 절차로 인해 기본 가속기에 구조화된 프루닝 패턴을 직접 적용하여 처리 효율을 더 높이기는 어렵다. 따라서 하드웨어 친화적인 구조 프루닝의 이점을 충분히 활용하기 위해서는 주어진 구조 가중치 패턴 전용 알고리즘 친화적 가속기 설계의 개발이 강하게 요구된다.

## V. 결 론

본 논문에서는 Transformer의 효율적인 압축을 위한 다양한 프루닝 기법에 대하여 소개하였다. Transformer 알고리즘의 경량화와 그에 맞는 전용 딥러닝 하드웨어를 사용하여 하드웨어 에너지 및 복잡도의 효율을 개선할 수 있다. 기존의 미세 조정된 프루닝을 통해 얻은 가중치의 중요도를 파악하여 마스킹 패턴을 분석하고 하드웨어 친화적인 구조 패턴을 도출해 내는 것이 필요하다. 이러한 연구들은 폭발적으로 성장하고 있는 Transformer 모델을 더 효율적으로 활용하고 에너지 소비를 최적화하는 방향으로 나아가는 노력을 보여주고 있다. Transformer의 정확도를 높은 상태로 유지하는 한도 내에서 앞으로도 보다 공격적인 압축률을 달성하는 방식이 Transformer를 위하여 제시될 것으로 예상된다. 그 과정에서 중요한 가중치 내의 하드웨어 친화적인 구조화를 만들어 내거나 많은 데이터 손실을 허용하는 압축 기술들이 지속해 연구될 것으로 생각된다.

## Acknowledgement

본 논문은 한국정부(MSIT)가 지원하는 정보통신기술 기획평가원(IITP) 보조금 지원(제 2021-0-00779 호, 개인정보보호 기반 하드웨어를 보장하는 초고속 암호화 데이터처리기술개발)과 정부(과학 기술 정보 통신부)의 재원으로 한국연구재단-차세대 지능형 반도체 기술개발사업(소자)의 연구(RS-2023-00258227) 그리고 2023 년도 정부(과학 기술 정보통신부)의 재원으로 정보통신기획평가원(RS-2023-00229849,IoT Intelligence 용 eFLASH 파운드리 공정 기반 MPU/Connectivity/경량 신경망 통합 반도체 개발)의 지원을 받아 수행된 연구이다.

## 참고 문헌

[1] A. Vaswani et al., “Attention is all you need,” in Proc. of NeurIPS, 2017.

[2] T. Brown et al., “Language models are few-shot learners,” in Proc. of NeurIPS, 2020, pp. 1877–1901.

[3] A. Chowdhery et al., “Palm: Scaling language modeling with pathways,” arXiv preprint arXiv:2204.02311, 2022.

[4] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” arXiv preprint arXiv:1510.00149, 2015.

[5] Sutskever, I., Vinyals, O., & Le, Q. V.. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in Proc. of NAACL, 2019, pp. 4171–4186.

[7] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. "GLUE: A multi-task benchmark and analysis platform for natural language understanding." arXiv preprint arXiv:1804.07461 (2018).

[8] Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., ... & Fernández, R. "The LAMBADA dataset: Word prediction requiring a broad discourse context." arXiv preprint arXiv:1606.06031 (2016).

[9] G. Park, B. Park, S. J. Kwon, B. Kim, Y. Lee, and D. Lee, “nuqmm: Quantized matmul for efficient inference of large-scale generative language models,” arXiv preprint arXiv:2206.09557, 2022.

[10] Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. "Llm. int8 (): 8-bit matrix multiplication for transformers at scale." arXiv preprint arXiv:2208.07339 (2022).

[11] Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., & He, Y. "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers." Advances in Neural Information Processing Systems 35 (2022): 27168-27183.

[12] Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. "Smoothquant: Accurate and efficient post-training quantization for large language models." International Conference on Machine Learning. PMLR, 2023.
[13] Gou, Jianping, et al. "Knowledge distillation: A survey." International Journal of Computer Vision 129 (2021): 1789-1819.
[14] Gu, Y., Dong, L., Wei, F., & Huang, M. "Knowledge Distillation of Large Language Models." arXiv preprint arXiv:2306.08543 (2023).
[15] Frantar, E., & Alistarh, D. "SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot." (2023).
[16] Ma, X., Fang, G., & Wang, X. "LLM-Pruner: On the Structural Pruning of Large Language Models." arXiv preprint arXiv:2305.11627 (2023).
[17] Zhang, M., Shen, C., Yang, Z., Ou, L., Yu, X., & Zhuang, B. "Pruning Meets Low-Rank Parameter-Efficient Fine-Tuning." arXiv preprint arXiv:2305.18403 (2023).
[18] V. Sanh, T. Wolf, and A. Rush, “Movement pruning: Adaptive sparsity by fine-tuning,” in Proc. of NeurIPS, 2020, pp. 20 378–20 389.
[19] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In IEEE International Conference on Neural Networks, 1993.
[20] Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal Brain Compression: A framework for accurate post-training quantization and pruning. arXiv preprint arXiv:2208.11580, 2022.
[21] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. of ICCV, 2017, pp. 1389–1397.
[22] M. Zhu, T. Zhang, Z. Gu, and Y. Xie, “Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus,” in Proc. of MICRO, 2019, pp. 359–371.
[23] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, “Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned,” in Proc. of ACL, 2019, pp. 5797–5808.
[24] P. Michel, O. Levy, and G. Neubig, “Are sixteen heads really better than one?” Proc. of NeurIPS, vol. 32, 2019.
[25] Lagunas, François, et al. "Block pruning for faster transformers." arXiv preprint arXiv:2109.04838 (2021).
[26] E. Yoo, G. Park, J. Min, S. Kwon, B. Park, D. Lee, and Y. Lee*, "TF-MVP: Novel sparsity-aware transformer accelerator with mixed-length vector pruning," Design Automation Conference (DAC), San Francis-co, CA, USA, July 2023.
[27] J. Park, H. Yoon, D. Ahn, J. Choi, and J.-J. Kim, “Optimus: Optimized matrix multiplication structure for transformer neural network accelerator,” Proc. of MLSys, pp. 363–378, 2020.
[28] A. Parashar et al., “Scnn: An accelerator for compressed-sparse convolutional neural networks,” ACM SIGARCH computer architecture news, vol. 45, no. 2, pp. 27–40, 2017.
[29] S. Zhang et al., “Cambricon-x: An accelerator for sparse neural networks,” in Proc. of MICRO. IEEE, 2016, pp. 1–12.
[30] S. Moon, H. Lee, Y. Byun, J. Park, J. Joe, S. Hwang, S. Lee, and Y. Lee*, "FPGA-based sparsity-aware CNN accelerator for noise-resilient edge-level image recognition," IEEE Asian Solid-State Circuits Confer-ence (A-SSCC), Macao, China, Nov. 2019, pp. 205-208.
[31] H. Kwon, Y. Byun, S. Kang, and Y. Lee*, "CHAMP: Channel merging process for cost-efficient highly-pruned CNN acceleration," IEEE Transactions on Circuits and Systems I: Regular vol. 69, no. 8, pp. 3308-3319, Aug. 2022.
[32] Y. Byun, S. Moon, B. Park, S. Kwon, D. Lee, G. Park, E. Yoo, J. Min and Y. Lee*, “Sparsity-Aware Memory Interface Architecture using Stacked XOR-Net Compression for Accelerating Pruned-DNN Models,” Proceedings of Machine Learning and Systems, Miami, FL, USA, June 2023.

류 은 지 (Eunji Yoo), 정회원

2017년 2월 : 광운대학교 전자전기공학과 졸업
2017년 1월~2021년 12월 : ㈜뷰웍스 엔지니어
2022년 2월~현재 : 포항공과대학교 전자전기공학과 석사과정

<관심분야> Embedded System Architecture, Deep compression

이 영 주 (Youngjoo Lee), 정회원

2008년 2월 : KAIST, 전기 및 전자공학과 학사 졸업
2010년 2월 : KAIST, 전기 및 전자공학과 석사 졸업
2014년 2월 : KAIST, 전기 및 전자공학과 박사 졸업
2014년 5월~2015년 2월 : 벨기에 IMEC 연구원
2015년 3월~2017년 2월 : 광운대학교 전자전기공학과 조교수
2017년 2월~2021년 2월 : 포항공과대학교 전자전기공학과 조교수
2021년 3월~현재 : 포항공과대학교 전자전기공학과 부교수

<관심분야> Embedded SoC 설계 및 최적화