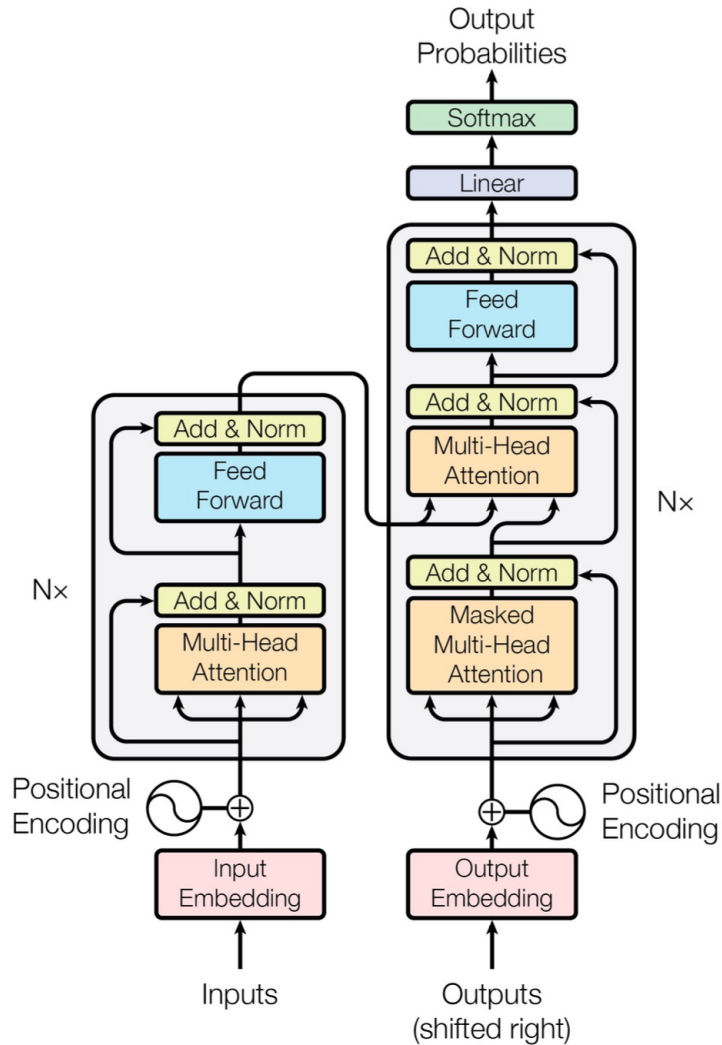

OpenELM- An Efficient Language Model Family

Sachin Mehta Mohammad Hossein Sekhavat Qingqing Cao Maxwell Horton
Yanzi Jin
Chenfan Sun Iman Mirzadeh Mahyar Najibi Dmitry Belenko Peter Zatloukal
Mohammad Rastegari
Apple

2024.04.22

발표자 : 김산

Introduction



Transformer 기반의 구조를 가지고 있는 기존 LLM(Large Language Models)은 isotropic라는 특징을 가지고 있다.
isotropic은 모든 방향으로 동일한 성질을 가졌다는 뜻으로, 여기서 모든 Transformer 계층이 **구조적으로 동일**하다는 것을 의미한다.



- 장점 : 구현과 학습이 간단하다.
- 단점 : 다양한 유형의 정보를 처리해야 하는 경우(문법, 문맥 등) 특정 유형의 정보를 처리하기 힘들 수 있다.


➡ OpenELM은 anisotropic 하다


Introduction


OpenELM vs public LLMs

Model	Public dataset	Open		Model size	Pre-training tokens	Average acc. (in %)
		Code	Weights			
OPT [55]	✗	✓	✓	1.3 B	0.2 T	41.49
PyThia [5]	✓	✓	✓	1.4 B	0.3 T	41.83
MobiLlama [44]	✓	✓	✓	1.3 B	1.3 T	43.55
OLMo [17]	✓	✓	✓	1.2 B	3.0 T	43.57
OpenELM (Ours)	✓	✓	✓	1.1 B	1.5 T	45.93

 [apple/OpenELM](#) 

 like 1.29k

 [arxiv:2404.14619](#)

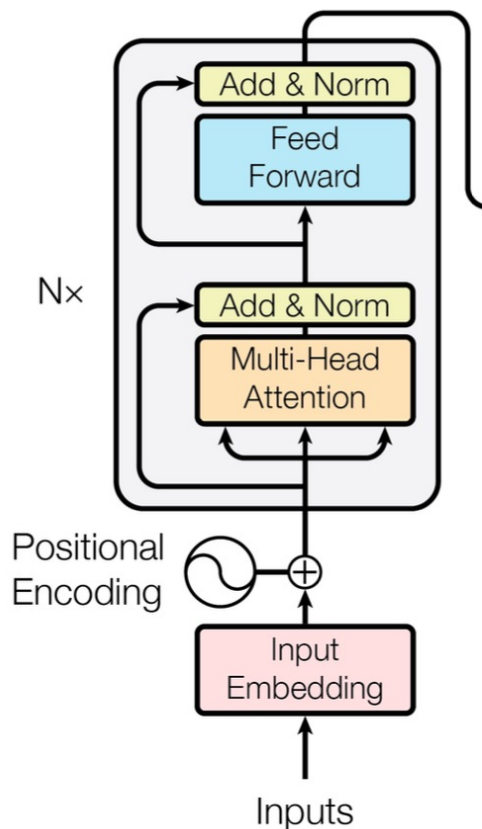
 License: apple-sample-code-license (other)

공개 데이터셋을 사용하여 기존의 LLM보다 더 좋은 성능을 보여준다.

Model size : 270M~

Pre-training

layer-wise scaling



표준 Transformer는 MHA와 FFN으로 구성되어 파라미터를 균일하게 할당한다. OpenELM에서는 layer-wise scaling을 사용해 Head의 개수와 FFN 배율을 조정해 파라미터를 비균일하게 할당한다.

i번째 계층에서 head수와 FFN 배율은 다음 식을 따른다.

$$n_h^i = \frac{\alpha^i \cdot d_{model}}{d_h}, \quad m^i = \beta^i$$

$$\text{where } \alpha^i = \alpha_{min} + \frac{(\alpha_{max} - \alpha_{min}) \cdot i}{N - 1}, \quad (1)$$

$$\text{and } \beta^i = \beta_{min} + \frac{(\beta_{max} - \beta_{min}) \cdot i}{N - 1}, \quad 0 \leq i < N.$$

1. 어텐션 헤드 수(n_h^i)와 FFN 차원(m^i)정의
2. 파라미터 α^i 와 β^i 계산

α^i, β^i : i번째 층에서 어텐션 헤드 수와 FFN 차원을 조절

d_{model} : 모델 차원

d_h : 각 어텐션 헤드의 차원

$\alpha_{min}, \alpha_{max}$: 어텐션 헤드 수를 조절하는 최소 및 최대값

β_{min}, β_{max} : FFN 차원을 조절하는 최소 및 최대 값

N: Transformer의 총 층 수
($i = 0 \sim N-1$)

Pre-training

OpenELM Architecture

1. bias 사용하지 않음
2. RMSNorm 및 RoPE 적용
3. GQA(Grouped Query Attention) 사용
4. FFN(Feed Forward Network)를 SwiGLU FFN으로 대체
5. Flash Attention 사용
6. Llama와 동일한 토큰나이저 사용

Pre-training

OpenELM Architecture

편향 매개변수(bias) 미사용

완전연결레이어(Fully Connected Layer)에서는 각 뉴런이나 층에 사용되는 **편향 매개변수(bias)**를 사용하지 않는다. 이는 **연산량을 줄여** 모델의 **효율성과 성능을 향상**시킨다.

bias는 모델의 출력값이 0이 아니도록 해주고 더 잘 학습될 수 있도록 도와준다.

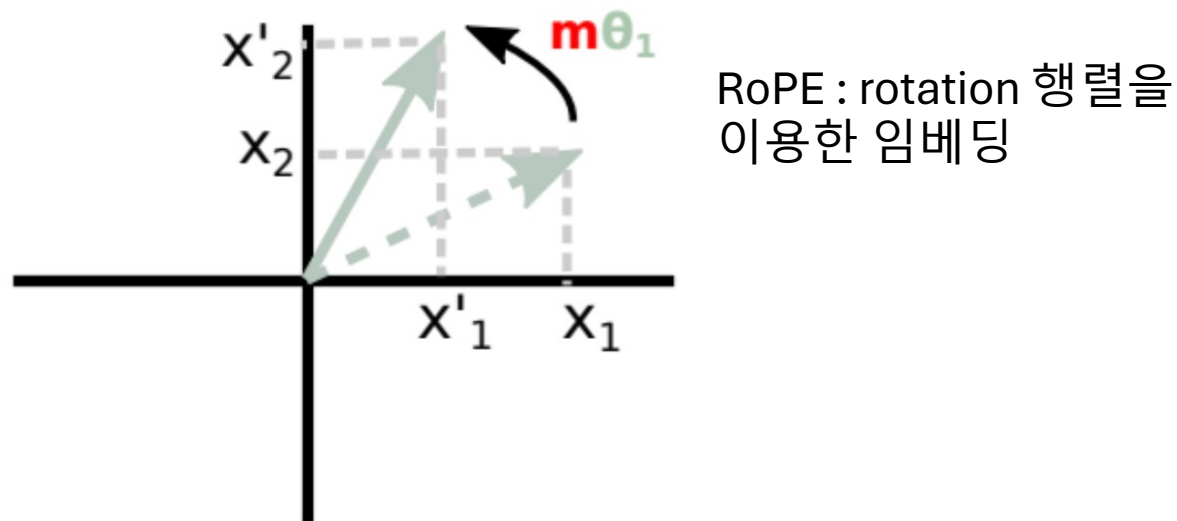
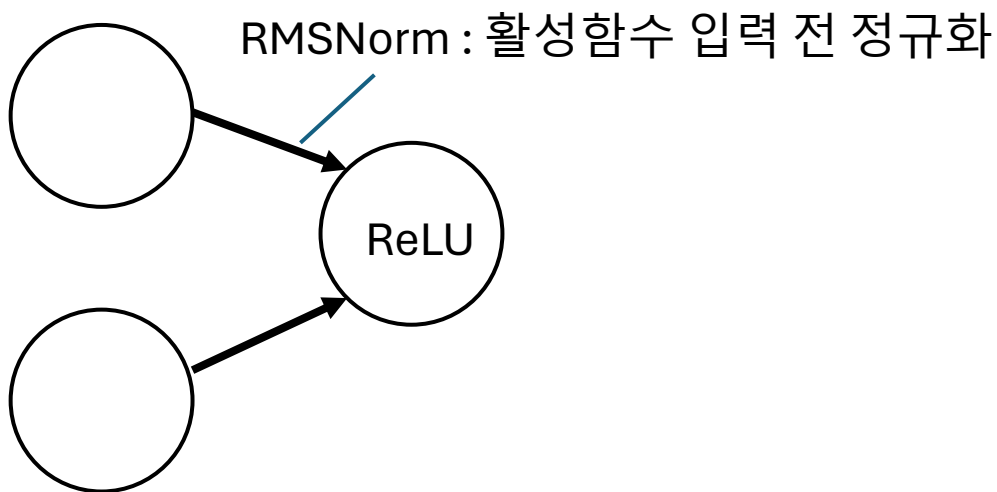
하지만 현대 딥러닝에서는 BatchNorm, layerNorm, RMSNorm 등 **다양한 정규화 기법**이 사용된다. 이런 정규화 기법들로 **충분한 안정화**를 할 수 있으며 bias의 **역할과 중복**될 수 있어 bias가 불필요해질 수 있다. 또한, bias를 제거함으로써 **파라미터 수를 줄이고** 이는 모델의 복잡성을 줄여 **과적합을 방지**할 수 있다.

Pre-training

OpenELM Architecture

사전 정규화 적용

RMSNorm(제곱 평균 제곱근 정규화)을 사용하여 **사전 정규화를 적용**하고, 토큰의 상대적 위치 정보의 인코딩을 위해 RoPE(Rotatory Positional Embedding)를 사용한다.

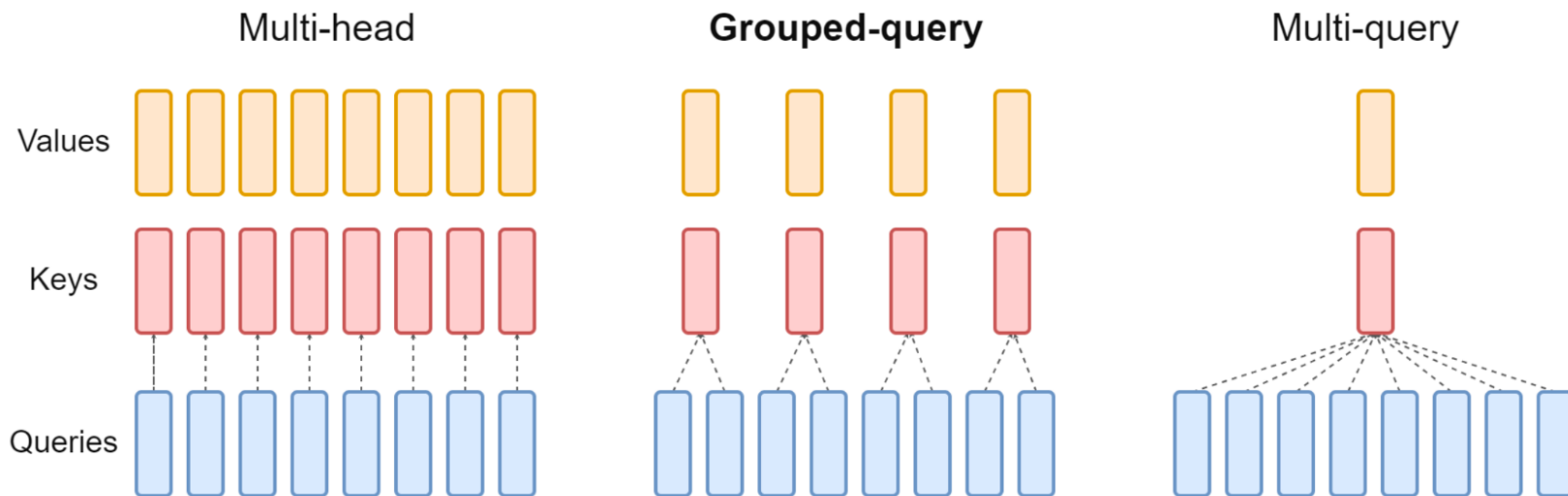


Pre-training

OpenELM Architecture

GQA(Grouped Query Attention)사용

기존의 Transformer에서 사용되는 Multi-Head Attention이 아닌 Grouped-Query Attention을 사용한다. GQA는 입력 텍스트의 **다양한 부분을 동시에 고려**해서 모델이 더 **넓은 문맥을 이해**하고 다양한 종류의 추론 작업을 수행할 수 있도록 한다.



Pre-training

OpenELM Architecture

SwiGLU FFN 적용

SwiGLU = Swish + GLU로 입력값의 특징을 효과적으로 추출하고 정보를 적절하게 필터링하여 전달한다. 모델의 **표현력을 향상**시키고, **학습 능력 개선**에 도움이 된다.

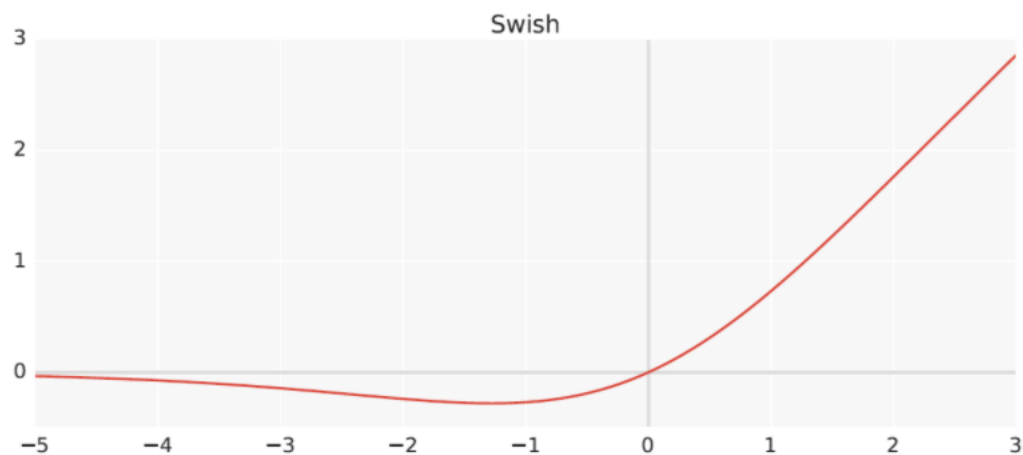
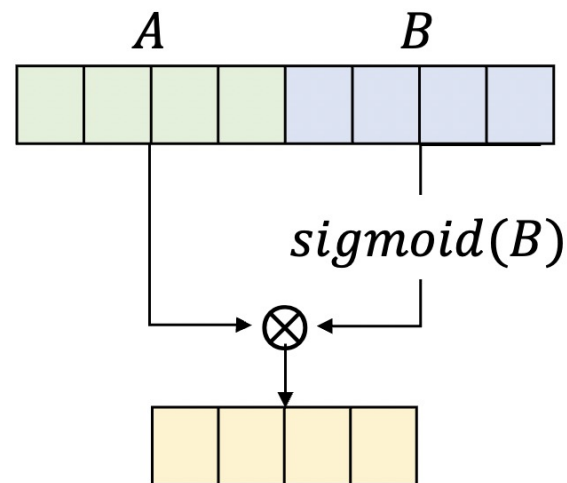


Figure 1: The Swish activation function.



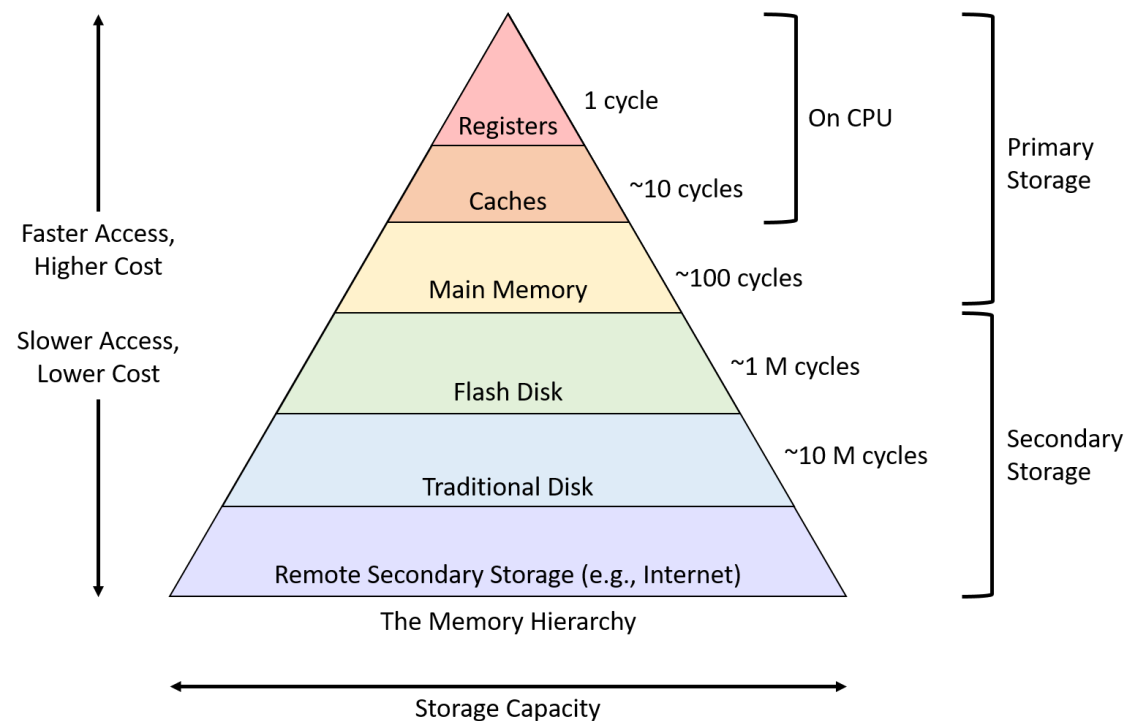
GLU는 입력을 두개의 선형변환으로 나누고 sigmoid함수를 수행한다.

Pre-training

OpenELM Architecture

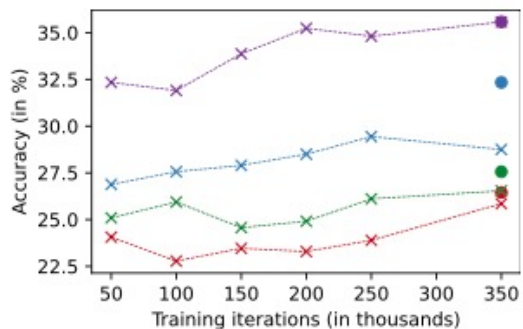
Flash Attention 사용 / Llama와 같은 토큰나이저 사용

Flash Attention은 **dot-product** 계산을 더 효율적으로 하게 도와준다. 특히 대규모 시퀀스에서 attention 메커니즘이 필요할 때 계산 부담을 크게 줄여준다.

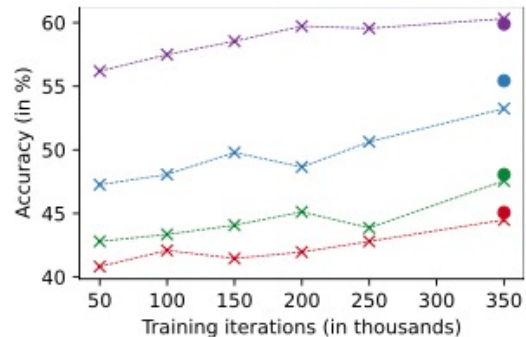


Llama와 같은 토큰나이저를 사용해 사전 학습을 바탕으로 복잡한 언어를 이해 및 생성한다.

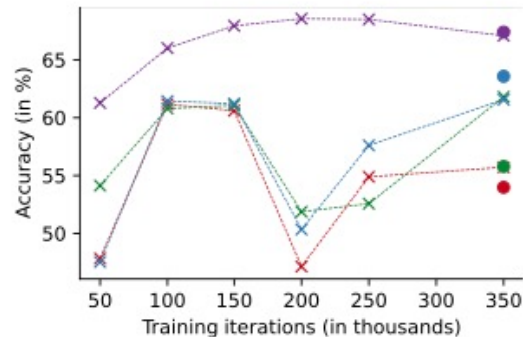
Results



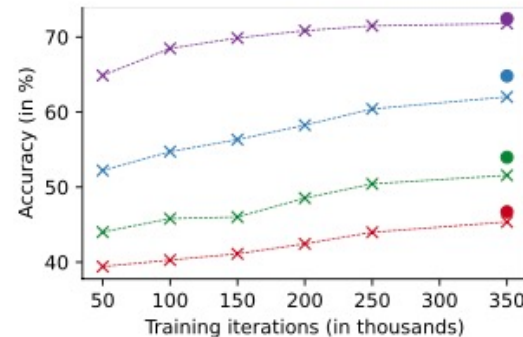
(a) ARC-c



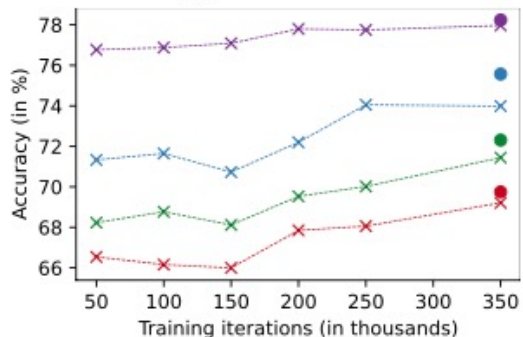
(b) ARC-e



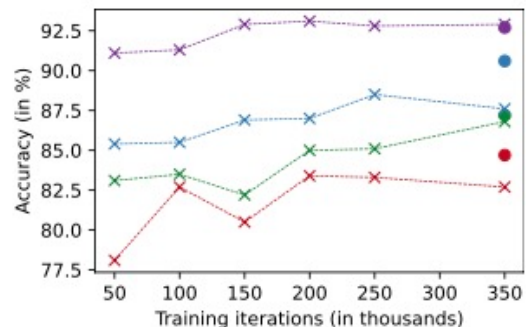
(c) BoolQ



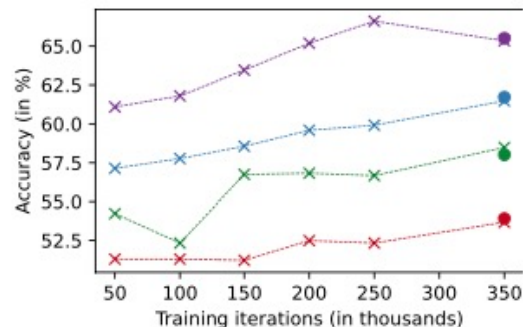
(d) HellaSwag



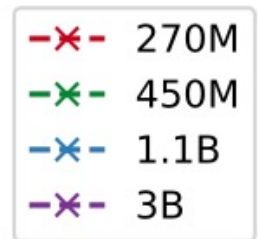
(e) PIQA



(f) SciQ



(g) WinoGrande



OpenELM sizes

zero-shot tasks에서의 성능 평가

시간이 지날수록 성능이 향상되는 것을 확인 할 수 있다.

Results

Model	Model size	Pretraining tokens	ARC-c	HellaSwag	MMLU	TruthfulQA-mc2	WinoGrande	Average
Cerebras-GPT [14]	0.26 B	5.1 B	22.01	28.99	26.83	45.98	52.49	35.26
OPT [55]	0.35 B	0.2 T	23.55	36.73	26.02	40.83	52.64	35.95
OpenELM (Ours)	0.27 B	1.5 T	27.65	47.15	25.72	39.24	53.83	38.72
Pythia [5]	0.41 B	0.3 T	24.83	41.29	25.99	40.95	54.38	37.49
MobiLlama [44]	0.50 B	1.3 T	29.52	52.75	26.09	37.55	56.27	40.44
OpenELM (Ours)	0.45 B	1.5 T	30.20	53.86	26.01	40.18	57.22	41.50
MobiLlama [44]	0.80 B	1.3 T	30.63	54.17	25.2	38.41	56.35	40.95
Pythia [5]	1.40 B	0.3 T	32.68	54.96	25.56	38.66	57.30	41.83
MobiLlama [44]	1.26 B	1.3 T	34.64	63.27	23.87	35.19	60.77	43.55
OLMo [17]	1.18 B	3.0 T	34.47	63.81	26.16	32.94	60.46	43.57
OpenELM (Ours)	1.08 B	1.5 T	36.69	65.71	27.05	36.98	63.22	45.93
OpenELM (Ours)	3.04 B	1.5 T	42.24	73.28	26.76	34.98	67.25	48.90

- 다른 모델과의 성능 평가 비교

비슷하거나 더 적은 model size와 token을 사용했지만 더 나은 성능을 보임

Conclusion

1. 파라미터 수를 많이 줄인 오픈 소스 OpenELM 모델 제시
2. layer-wise scaling 등의 방법을 사용해 효과적으로 파라미터를 줄였다.
3. zero-shot, 모델 간 비교 등 다양한 성능평가에서 좋은 결과를 얻었고
추가적으로 양자화도 수행할 수 있다.