더 가볍게, 더 멀리: Llama (v1,v2)가 해결한 LLM의 구조적 한계와 효율성

전체 목차

I. 배경

1.1 AI의 기점: 2017년의 혁명
1.2 GPT와 스케일링 전쟁

II. Llama의 과제

2.1 Transformer의 비효율성 극복
2.2 똑똑하게 다듬기: Llama 2 철학

III. 핵심 기술 & IV. 파인튜닝

3.1 RoPE (위치 인코딩의 혁신)
3.2 RMSNorm & Pre-Norm (안정성)
3.3 GQA (속도 및 메모리 혁신)
3.4 SwiGLU (활성함수 최적화)
4.1 파인튜닝 기법 (SFT/RLHF/GAIT)

V. 결론 & Reference

5.1 Llama의 파괴적 혁신
5.2 References: 연대기별 논문 목록

I. AI의 기점: 2017년의 혁명

I-1 2017년의 혁명

'Attention Is All You Need' 논문의 등장

RNN의 순차적 한계를 극복한 **Transformer**

혁신적인 아키텍처는 **현대 LLM의 초석**이 됨

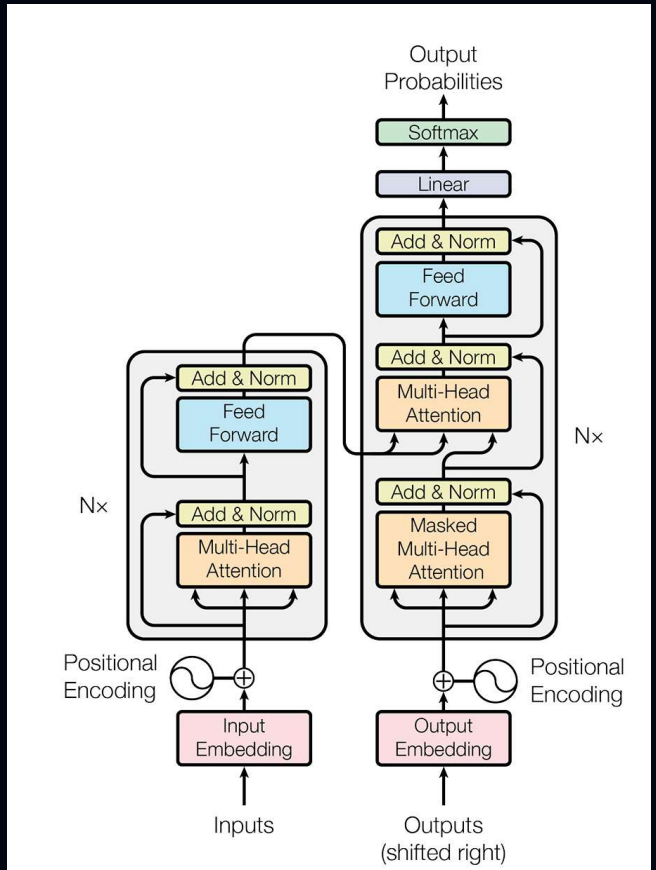


Figure 1: The Transformer - model architecture.

gpt-2(2019)

1.5B

gpt-3(2020)

175B

파라미터 올릴수록 성능 상승

I-2 GPT와 스케일링 전쟁

Transformer Decoder를 기반으로한 GPT등장

2020년 GPT-3: 거대 모델의 성능 입증

"더 많은 데이터, 더 큰 컴퓨팅 파워가 곧 지능이다"

모델 파라미터를 늘리는 스케일링 전쟁 시작

II. 스케일링의 한계: 거대함이 남긴 과제

"단순한 확장은 더 이상 해답이 될 수 없다"

1. 비용의 장벽

2. 학습의 비효율

3. 확장 가능성의 고갈

Llama 2: "더 적은 자원(CPU/GPU)으로 동일 성능을 낼 순 없을까?"

Reasoning Benchmarks: LLaMA vs Others

Model	Size	BoolQ (참/거짓)	PIQA (물리 상식)	SIQA (사회 상식)	HellaSwag (문맥 추론)	WinoGrande (상식 추론)	ARC-e (과학-E)	ARC-c (과학-C)	OBQA (상식 문답)
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

LLaMA-65B는 대부분의 벤치마크에서 9배 큰 모델(PaLM-540B)에 필적하는 성능을 보여줌

III. 핵심 기술

Llama v1

3.1 RoPE

위치 인코딩의 혁신

Llama v1

3.2 RMSNorm (Pre-Norm)

학습 안정성 및 속도 향상

Llama v2

3.3 GQA

추론 속도 및 메모리 최적화

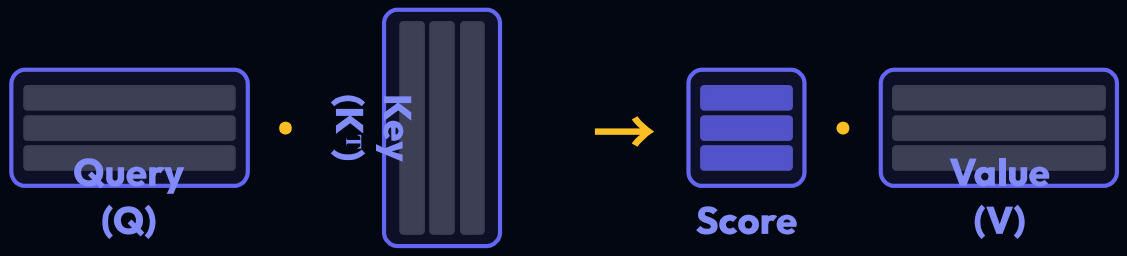
Llama v1

3.4 SwiGLU

표현력 풍부한 활성 함수

Attention: Q, K, V의 조화

Attention은 수많은 단어 중 **의미적으로 중요한 정보**를 찾아내는 과정입니다.



1단계: Q와 K의 연합 (Score)

행렬곱을 통해 단어 간의 **유사도(점수)**를 계산합니다.

2단계: Score와 V의 결합

계산된 유사도를 점수 삼아, 실제 정보인 **Value**를 가중 합산하여 최종 결과를 도출합니다.

왜 위치 정보가 중요한가?

사례 1

"사람 문 개"

사례 2

"개 문 사람"

[사람] · [개] →  VS [개] · [사람] → 

⚠️ 결과가 동일함 (위치 정보 부재)

"두 문장은 동일한 단어 집합{개, 사람, 문}을 가지지만,
단어의 순서에 따라 의미가 완전히 달라집니다."

Transformer의 Attention은 모든 단어를 동시에 병렬 처리하기 때문에,
별도의 **위치 정보(Positional Information)** 없이는 위 두 문장을 구분할 수 없습니다.

III-1. 문제 해결: Positional Encoding(vanilla transformer)

절대적 위치 인코딩

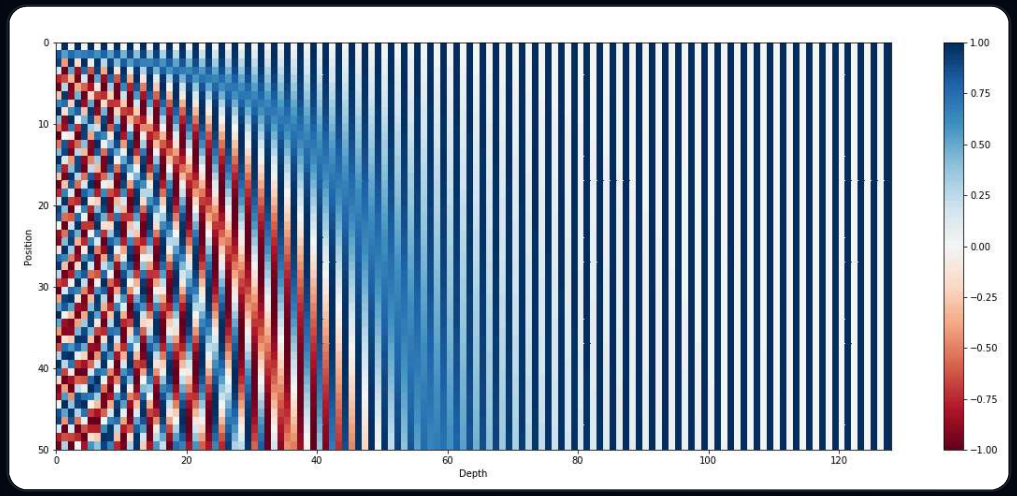
반복되는 **사인/코사인 파동**을 이용해
단어별 고유 위치 정보를 생성했습니다.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d})$$

➡ 이 파형 정보(p)를 기존 단어 임베딩(x)에 **더하여** 위치를 구별하게 합니다.

$$Q, K, V = x + p$$



[위치별 고유 패턴 시각화]
이 패턴 덕분에 모델은 단어들이 배치된 순서를 이해하게 됩니다.

III-1. 절대적 위치 인코딩의 한계

정보를 단순히 더하는(+) 방식은 연산 과정에서 필연적인 **수학적 비효율**을 초래합니다.

[QK 내적 시 발생하는 Mixed Terms]

$$\underbrace{(x_m + p_m)}_{Q_m} \cdot \underbrace{(x_n + p_n)}_{K_n} = \underbrace{x_m \cdot x_n}_{\text{순수한 의미}} + \underbrace{x_m \cdot p_n + p_m \cdot x_n}_{\text{원치 않는 노이즈}} + \underbrace{p_m \cdot p_n}_{\text{순수한 위치}}$$

🧠 학습 부담 및 비효율

노이즈 제거 학습: 모델이 스스로 노이즈를 걸러내는 법을 배우기 위해 **방대한 양의 데이터**가 소모됨 (학습 부담 가중)

자원 낭비: 이는 곧 **학습 속도 저하**와 데이터 아키텍처의 비효율로 직결됨

📏 확장성(Extrapolation) 한계

절대적 위치 번호(1, 2, 3...)에만 의존하는 방식이기에,

학습 시 못 본 긴 문장이 들어오면 모델의 대응 능력이 급격히 상실됩니다.

"본 적 없는 위치 패턴에는 무력함"

III-1. 혁신: "더하지 말고, 회전시켜라"

Llama 2가 채택한 **RoPE(Rotary Positional Embedding)**
는 정보를 더하는 대신, 벡터 공간에서 **회전(Rotation)**을 사용합니다.

[RoPE의 수학적 설계]

$$R_m = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix}$$

$$\theta_i = 10000^{-\frac{2(i-1)}{d}}, \quad i \in \{1, 2, \ldots, d/2\}$$

[내적 보존: 상대적 거리의 도출]

$$(R_m x_m) \cdot (R_n x_n) = x_m^T (R_{n-m}) x_n$$

✅ 결과가 오직 두 단어의
상대적 거리(m-n)에만 의존!

✨ RoPE의 핵심 가치

✓ **의미 보존:** 위치를 더하지 않아 원본 정보 유지

✓ **거리 인식:** '어느 정도 거리'인지 물리적으로 이해

✓ **무한 확장:** 학습 범위 밖의 문장도 처리 가능

III-2. 왜 LayerNorm인가?

Batch Norm (Column)

Idx	1열	2열	3열	4열
S1	**['0']**	'0'	'나는'	'학생'
S2	**['0']**	'0'	'아직'	'학교입니다'
S3	**['기차를']**	'타고'	'집에'	'가요'
S4	**['0']**	'저는'	'집이'	'좋아요'

↓ μ, σ

- ❌ **Time Step별 통계 변화**: 첫 단어와 100번째 단어의 분포 차이 존재 (RNN 가중치 재사용 문제)
- ❌ **가변 시퀀스 길이**: 학습(50단어)보다 긴 테스트(100단어) 입력 시 후반부 통계 부재

Layer Norm (Row)

Idx	1열	2열	3열	4열	
S1	**['0'**	**'0'**	**'나는'**	**'학생']**	→ μ, σ
S2	'0'	'0'	'아직'	'학교입니다'	
S3	'기차를'	'타고'	'집에'	'가요'	
S4	'0'	'저는'	'집이'	'좋아요'	

- ✅ **Time Step 독립성**: 각 단어 내에서만 통계를 계산하여 분포 변화 무관
- ✅ **길이 무관**: 문장 길이에 상관없이 즉시 정규화 가능 (학습/테스트 불일치 해결)

Who uses LayerNorm?

Vanilla Transformer(2017), BERT, GPT-1/2/3, RoBERTa, T5, BART...

➡ 사실상 모든 현대 LLM의 표준 정규화 방식

III-2. LayerNorm 수식 상세

입력 벡터 a에 대해 **평균(μ)**과 **분산(σ)**으로 정규화 후 스케일 조정

$$\bar{a}_i = \frac{a_i - \mu}{\sigma + \epsilon} \cdot g_i + b_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \mu)^2}$$

a_i: 입력, μ: 평균, σ: 표준편차, ϵ: 안정수치
g_i: 이득(gain) 파라미터, b_i: 편향(bias) 파라미터

정규화 (Normalization)

re-Centering: $a_i - \mu$
(평균을 0으로 이동)

re-Scaling: $/\sigma$
(표준편차로 나누어 크기 조정)

복원 (Affine Transform)

g_i (Scale): 데이터 폭 조정
b_i (Shift): 데이터 중심 이동

➡ **정규화로 잃은 표현력 복구**

III-2. RMSNorm: 속도의 미학

(RMS: Root Mean Square, 제곱평균제곱근)

"평균(μ)을 빼는 과정(re-Centering)이 꼭 필요한가?"

LayerNorm (re-Centering O)	**RMSNorm (re-Centering X)**
$\frac{a_i - \mu}{\sigma}$	$\frac{a_i}{\mathrm{RMS}(a)}$
$\sigma = \sqrt{\frac{1}{n}\sum(a_i - \mu)^2}$	$\mathrm{RMS}(a) = \sqrt{\frac{1}{n}\sum a_i^2}$

$$\bar{a}_i = \frac{a_i}{\mathrm{RMS}(a)} g_i$$

(여기서 $\mathrm{RMS}(a) = \sqrt{\frac{1}{n}\sum a_i^2 + \epsilon}$, μ 계산 안 함)

1. Bias(b_i) 제거 이유

"Shift(b_i)는 이미 평균으로
맞춰져 있으므로 **불필요**하다"

2. 결과 (Efficiency)

➡ **re-Centering 생략**으로 연산량 감소
➡ 학습 속도 **10~40% 향상** 효과
➡ **"더 적은 연산, 더 빠른 속도"**

III-2. Norm 위치의 변화: Post vs Pre

기존 (Post-Norm)

```
Sublayer → Add → Norm
```

Post-LN은 2017년에 Transformer가 개발되었을 때 사용된 초기 안정화 방식으로, 연산이 끝난 후 히든 스테이트를 정리해 모델이 흔들리지 않도록 돕습니다.

Llama v1

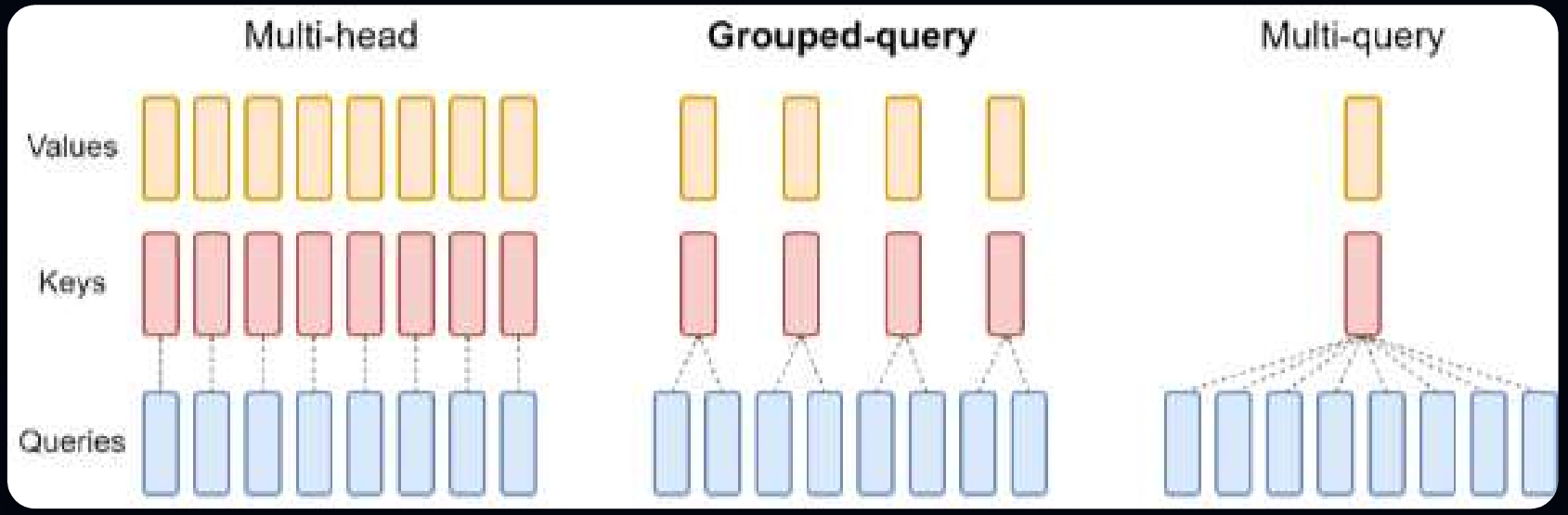
최신 (Pre-Norm)

```
Norm → Sublayer → Add
```

Pre-LN은 연산 전에 입력을 먼저 안정화해 학습이 더 안정적으로 진행되도록 합니다.

"Pre-Norm을 통해 심층 신경망(Deep Network)에서도 학습이 훨씬 안정적이다."

III-3. GQA: 속도의 혁신



High Performance
(Quality)

High Speed
(Efficiency)

Multi-Head (MHA)

품질 좋음 / 메모리 큼
모든 Query가 각자의 KV 가짐
(메모리 폭발)

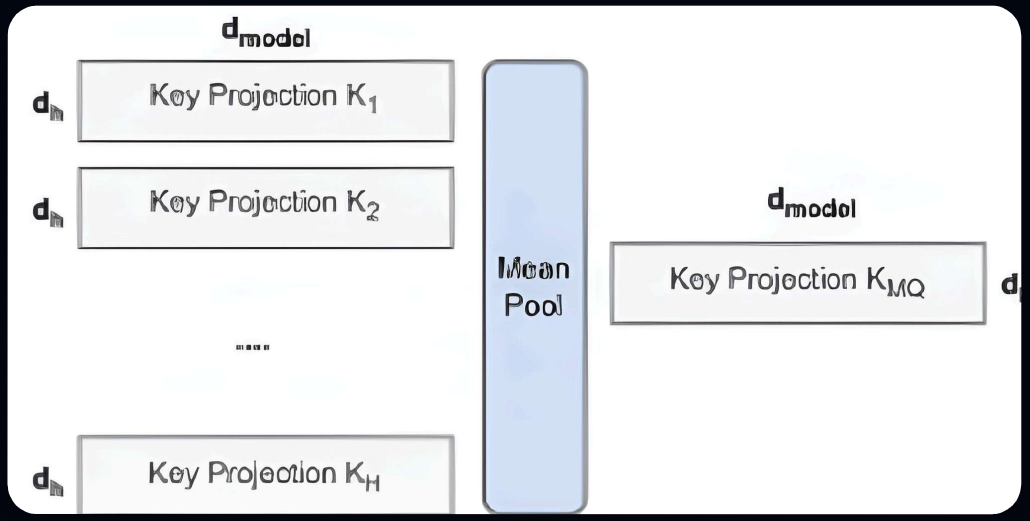
Grouped-Query (GQA)

최적의 균형
Query를 그룹화하여 KV 공유
(속도↑, 성능 유지)

Multi-Query (MQA)

메모리 작음 / 품질 저하
하나의 KV를 모든 Query가 공유
(성능 하락)

III-3. GQA 전환 프로세스 (Uptraining)



[Mean Pooling을 통한 Checkpoint 변환]

1단계: 체크포인트 변환

- **Mean Pooling**: 그룹 내 MHA의 Key/Value 행렬들을 평균내어 초기화 (임의 선택보다 우수)
- **구조 재구성**: MHA(H개) → GQA(G개)로 프로젝션 변경

2단계: 추가 사전 학습

- **설정 유지**: 원본과 동일한 데이터셋/세팅 사용
- **저비용 고효율**: 전체 연산의 **5%**만으로 성능 회복

(GQA (Ainslie et al., 2023))

"결론: 기존 MHA 모델도 적은 비용으로 GQA로 변환하여, 속도와 성능을 모두 잡을 수 있습니다."

III-4. FFN: 활성함수의 진화 (1) ReLU

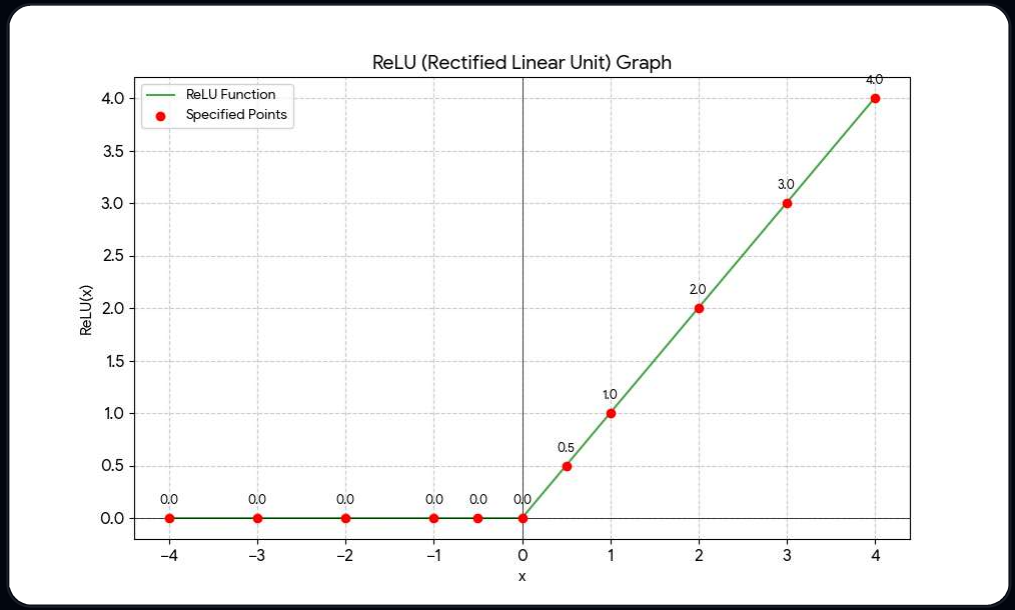
Input: [-4.0, -3.0, -2.0, -1.0, -0.5, 0.0, 0.5, 1.0, 2.0, 3.0, 4.0]

ReLU (Vanilla Transformer)

$$\mathrm{ReLU}(x) = \max(0, x)$$

✅ **단순함**: 연산 비용이 매우 낮음

❌ **Dead ReLU**: 0 이하 입력에 대해 기울기가 0이 되어, 뉴런이 죽는 현상 발생 (학습 불가)



[0 이하의 값은 모두 버림]

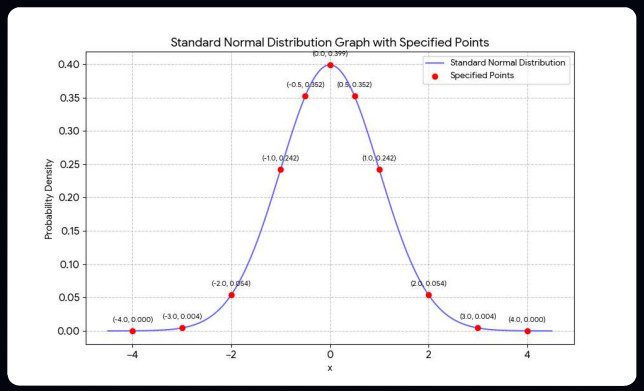
III-4. FFN: 활성함수의 진화 (2) GeLU

GeLU (GPT-3, BERT)

$$\mathrm{GeLU}(x) = x\Phi(x)$$

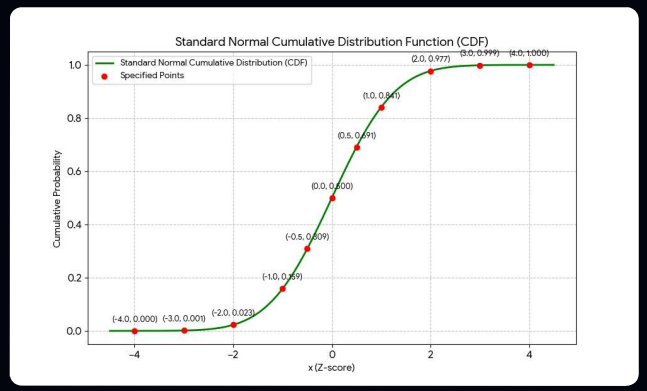
- $\Phi(x)$: **표준정규분포의 누적분포함수(CDF)**

(정규분포 상에서 x 이하의 값이 나타날 **확률**을 가중치로 사용)



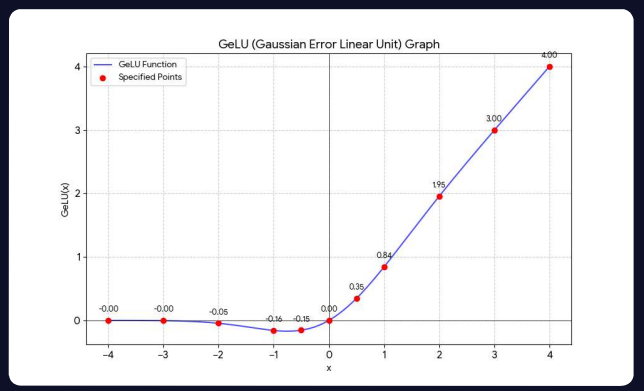
1. 표준정규분포

(평균: 0, 표준편차: 1)



2. 누적분포함수 ($\Phi(x)$)

(x까지의 누적 확률)



3. GeLU 결과 ($x \cdot \Phi(x)$)

(부드러운 비선형성)

III-4. FFN: 활성함수의 진화 (3) SwiGLU

SwiGLU (Llama 2, PaLM)

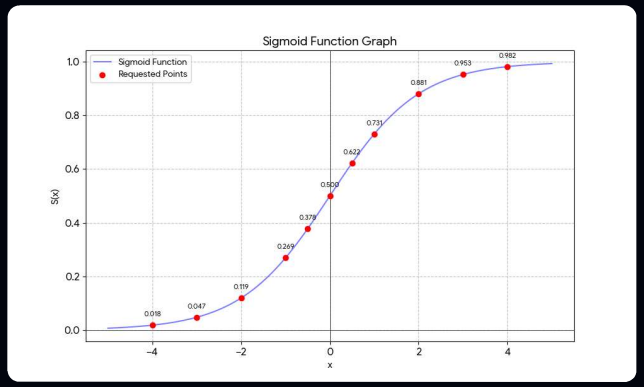
"게이트(Gate)가 달린, 선형 유닛(Linear Unit)"의 변형으로,
Gate 활성함수로 **Swish**를 사용합니다.

$$\text{SwiGLU}(x) = \text{Swish}_\beta(xW + b) \otimes (xV + c)$$

$xW + b, xV + c$: 입력값에 대해 각각 **독립적인 선형변환** 진행
$\otimes$: 요소별 곱 (Element-wise Product)

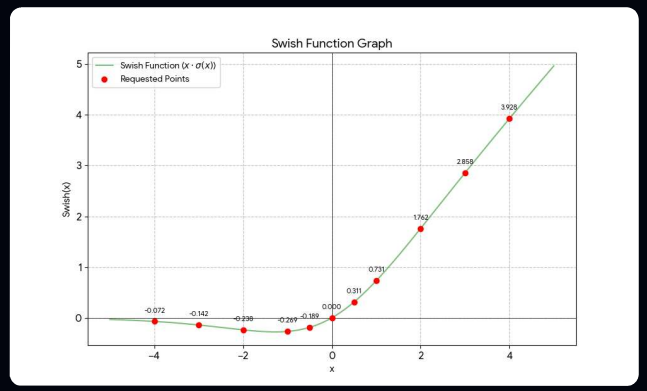
[1] Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



[2] Swish

$$\text{Swish}(x) = x \cdot \sigma(x)$$



SwiGLU 결론

- **쌍선형 성질**: 두 선형 레이어의 곱($W \otimes V$)으로 고차원 데이터 분포를 더 잘 근사함
- **성능 향상**: 실험적으로 단순 비선형 함수보다 **Perplexity(추론 성능)**가 더 뛰어남

IV-1. 파인튜닝 기법

파인튜닝의 목적

🎯 대화형 AI 최적화 | 🤝 인간 선호도 정렬 | ✅ 지시사항 준수 | 🛡 안전성 강화

Llama v1

1. SFT (Supervised Fine-Tuning)

"**지도 학습(Supervised) + 미세 조정(Fine-Tuning)**"

Supervised: 사람이 작성한 **정답(Label)**이 있는 데이터를 교사로 학습

Meaning: 모델에게 답변 내용을 넘어,
질문에 대답하는 **방법(Format)**을 가르침

⚠ **한계**: 단순 모방에 그치며, 고품질 데이터가 항상 **부족**함

Llama v2

2. RLHF (RL + Human Feedback)

"**강화 학습(RL) + 인간 피드백(Human Feedback)**"

Human Feedback: 정답이 없는 문제에 대해 인간의 **선호(Preference)**를 반영

Meaning: 단순 모방을 넘어,
인간의 **가치관(Value)**을 스스로 판단하게 함

IV-2. SFT 기술: GAtt (Ghost Attention)

Llama v2

[문제] Context Loss

대화의 턴이 길어지면...

System: "해적처럼 말해줘"
User: 안녕?
AI: 안녕! (해적 말투 O)
... (20턴 후) ...
User: 점심 뭐 먹지?
AI: 샌드위치 어때요? (해적 말투 X ❌)

➡ 초기 지시(System Prompt) 망각

[해결] GAtt (Ghost Attention)

학습 시 "유령"처럼 지시를 붙임

Turn 1: [System+Msg] ...
Turn 2: **[System]** + [Msg] ...
Turn 20: **[System]** + [Msg] ...

➡ 페르소나/제약조건 완벽 유지하며 지도 학습 데이터 생성

"간단하게 초기 지시사항을 망각하는 문제를 해결"

결론: Llama가 AI 생태계에 가져온 파괴적 혁신

모델의 경량화와 **생태계의 개방성**으로 AI 발전 가속화

1. 규모의 경제를 넘어선 '학습 효율성'의 승리

Scaling Law 도전: "크면 클수록 좋다" vs "효율성이 좋다"

증명: 13B 모델이 10배 큰 GPT-3(175B) 성능 능가

➡ 거대 자본 없는 '효율 중심 AI 시대' 개막

2. 혁신의 촉매제: AI 가속화

개방성(Open Source): 누구나 접근 가능한 고성능 LLM

생태계 확장: 의료, 법률, 코딩 등 다양한 파생 모델 폭발적 증가

➡ AI 연구 및 응용의 비약적 발전