

# LG CNS One-day LLM

최신 LLM의 이해와 응용

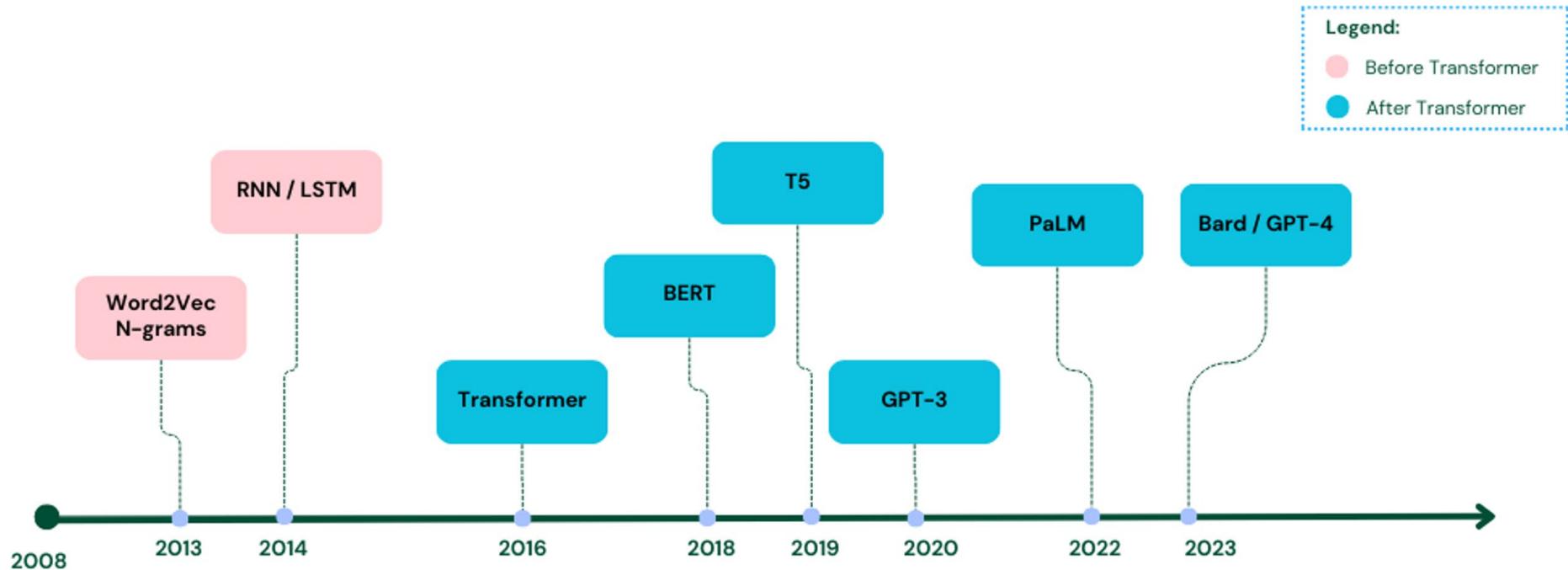
강사 김형욱  
[hukim@artiasolution.com](mailto:hukim@artiasolution.com)

# Contents

- 언어모델의 이해 (3 page)
- Transformers (18 page)
- Berts and GPTs (42 page)
- Prompt engineering (73 page)
- LLM Fine-tuning (84 page)
- Small Language Model (92 page)
- LLM serving framework (97 page)
- LLM trends (101 page)
- Vision transformer (107 page)
- Multimodal : CLIP (121 page)
- Multimodal : LDM (140 page)

# 언어모델의 이해

# Language Model History



# Statistical Language Model : 초기 언어모델

- 언어 모델이란 문장에서 다음 단어가 무엇일지 예측하는 모델
  - 문장의 자연스러움을 판단하거나, 텍스트 생성, 기계 번역 등 자연어 처리에 활용
  - 통계적 언어모델은 단어 시퀀스의 확률 분포를 통계적으로 계산할 수 있음.

$$(1) P(\text{Today is Wednesday}) = 0.001$$

$$(2) P(\text{Today Wednesday is}) = 0.0000000001$$

- N-Gram은 앞선 N-1개의 제한된 단어를 조건으로 사용하는 언어모델

$N = 1$  : This is a sentence *unigrams:*  
this,  
is,  
a,  
sentence

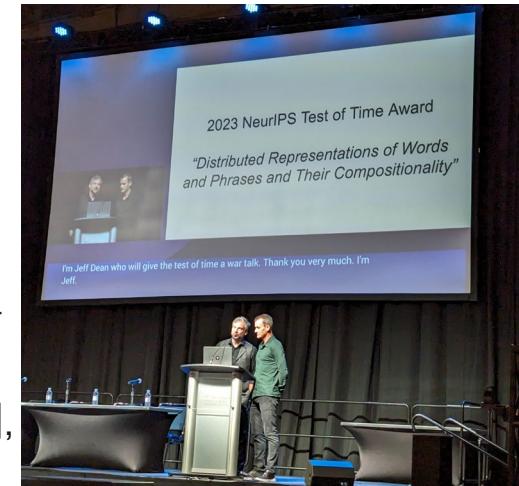
$$\begin{aligned} \text{Uni-gram} &: P(x_i|x_1, x_2, \dots, x_{i-1}) \approx P(x_i) \\ \text{Bi-gram} &: P(x_i|x_1, x_2, \dots, x_{i-1}) \approx P(x_i|x_{i-1}) \\ \text{Tri-gram} &: P(x_i|x_1, x_2, \dots, x_{i-1}) \approx P(x_i|x_{i-2}, x_{i-1}) \end{aligned}$$

$N = 2$  : This is a sentence *bigrams:*  
this is,  
is a,  
a sentence

$N = 3$  : This is a sentence *trigrams:*  
this is a,  
is a sentence

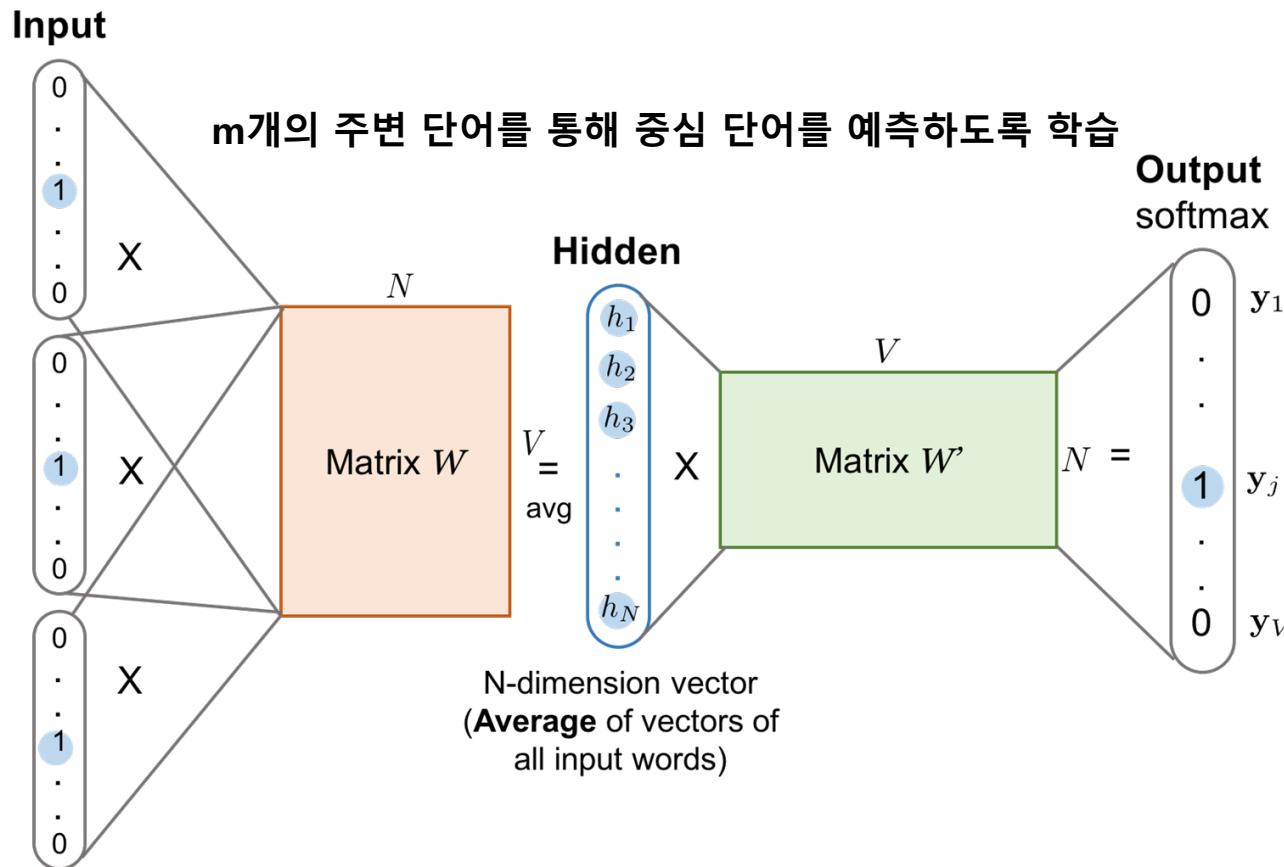
# Word2vec : 진화하는 언어모델

- N-gram 이후 카운트 기반 모델들이 많이 활용
  - 텍스트를 머신러닝으로 분석하기 위해 단순 확률 표현이 아닌 벡터 표현이 필요해짐
  - 카운트 기반 언어모델은 단어의 발생 빈도로 문서의 벡터 표현을 만듦.
    - ✓ Term Frequency-Inverse Document Frequency(TF-IDF)
    - ✓ PMI, PPMI, AP, LSA 등..
- Google 연구진은 단어의 빈도보다 단어 간의 관계를 예측하는 학습 기반의 Word2Vec[1] 개발
  - 주변 단어를 기반으로 중심 단어 예측하는 CBOW와 중심 단어로 주변 단어를 예측하는 SKIP-GRAM이 있음
  - Word2Vec은 각 단어에 대한 밀집된 벡터(dense vector)를 생성해, 의미적 유사성과 문법적 관계를 반영
  - NIPS 2023의 'Test of Time Award' 수상



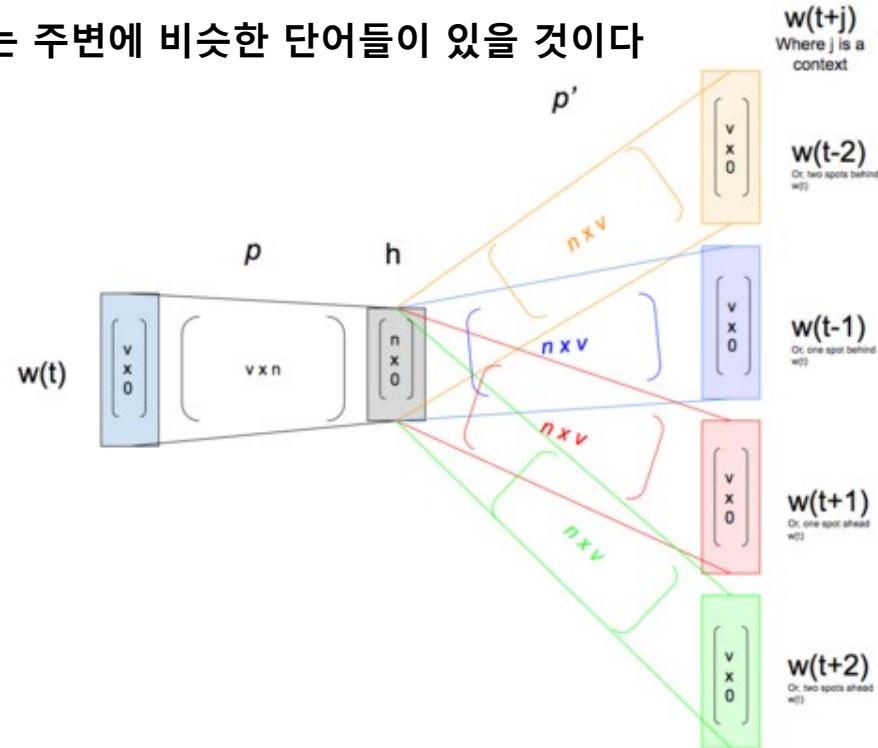
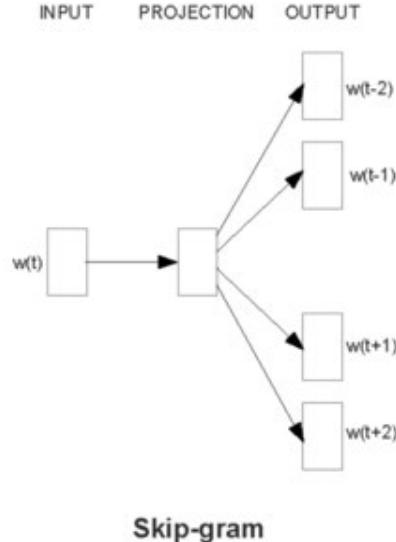
[1] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013).

# Word2Vec : CBOW(Continuous Bag of word)

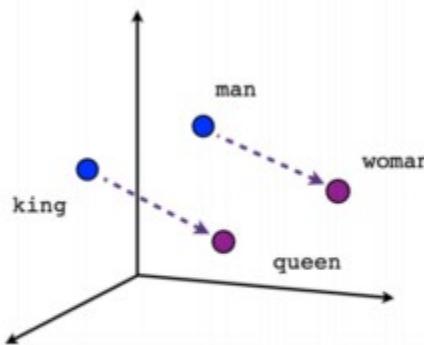


# Word2Vec : SKIP-GRAM

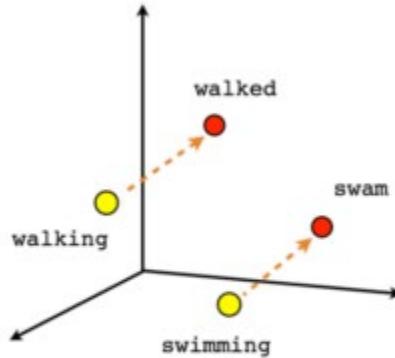
문법적 또는 의미적으로 비슷한 단어는 주변에 비슷한 단어들이 있을 것이다



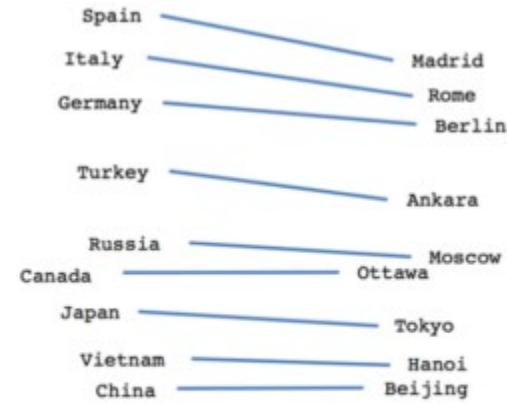
# Word embedding



Male-Female



Verb tense



Country-Capital

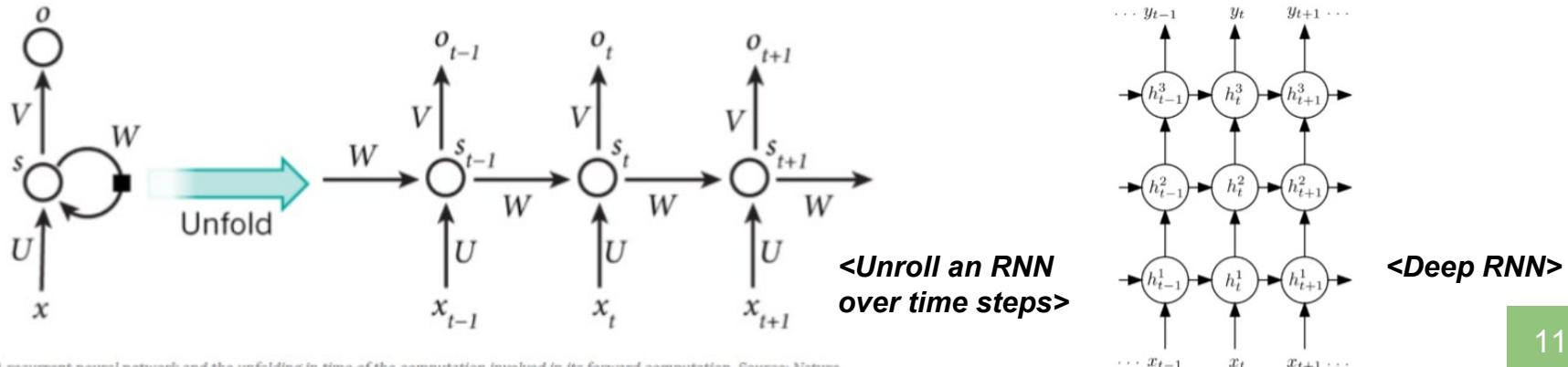
임베딩 공간 안 워드 시멘틱스의 기하학적 표현

# Word2Vec의 중요성

- 분산 표현(Distributed Representation) : 단어의 의미를 다차원 공간의 위치로 표현
  - 단어 사이의 유사성과 의미적 관계 추론 가능
    - ✓ “왕”과 “남자” 사이의 벡터와 “여왕”과 “여자” 사이의 벡터가 유사
- 효율적 학습 : 대규모 텍스트 데이터에서 학습 가능
- 다양한 NLP 작성의 성능 향상
- 분포 가설 입증
  - 비슷한 컨텍스트에서 등장하는 단어들은 비슷한 의미를 가질 거라는 가설
  - 학습 과정에서 직접적으로 단어 사이의 유사성을 비교하지 않더라도, 유사한 컨텍스트에서 자주 함께 등장하는 단어들의 임베딩은 비슷한 방향으로 조정할 수 있음.

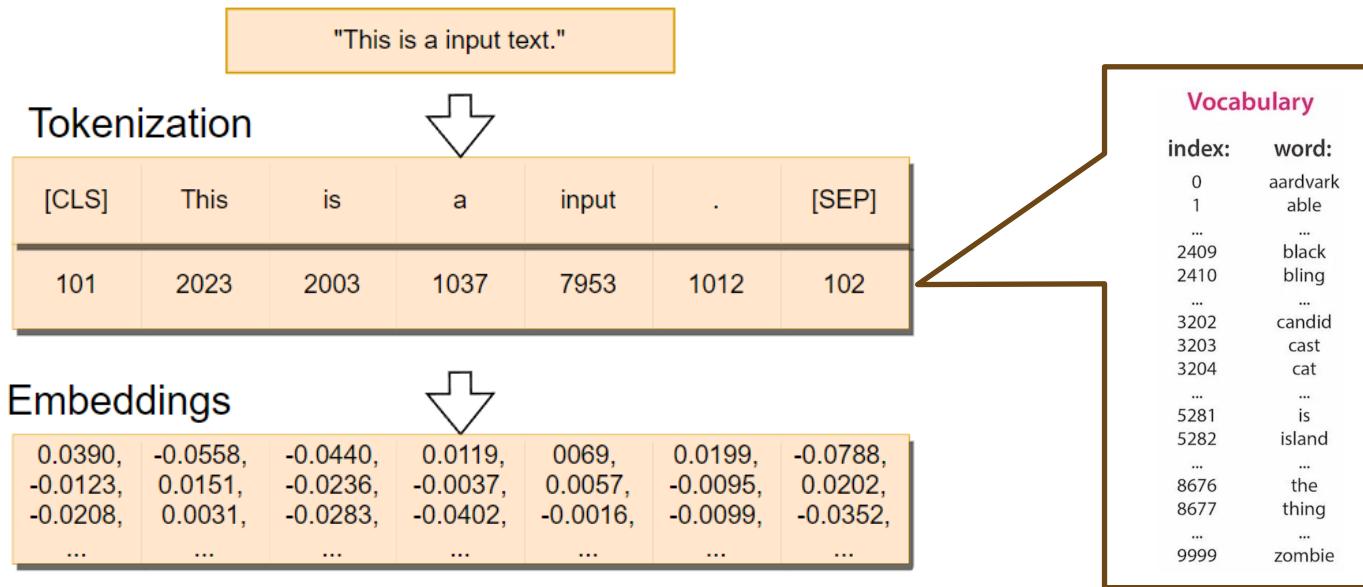
# Neural Language Models, RNNs and LSTMs

- RNN(Recurrent Neural Networks)은 단어의 순서를 고려하여 문맥적 정보를 유지하는 데 강점을 가지는 신경망 구조
  - N-gram과 카운트 기반 모델은 단어의 순서 정보를 고려하지 못함
  - RNN은 이전 단계의 출력을 다음 단계의 입력으로 사용하는 순환 구조를 가지고 있어서, 문장에서 시간적인(순차적인) 의미를 포착함
  - RNN의 Long-term dependency 문제를 보완한 LSTM, GRU같은 변형 모델 개발



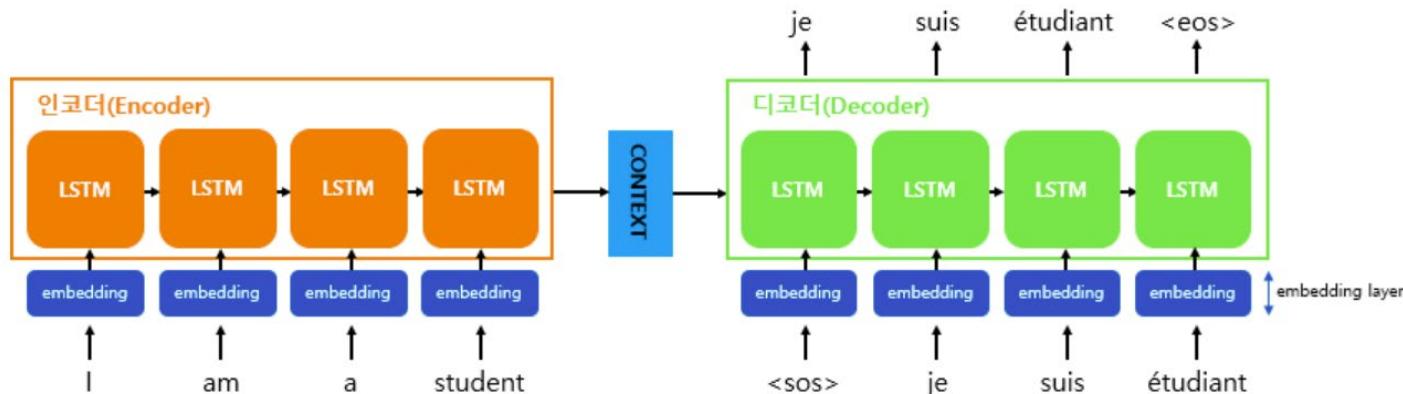
# Tokenization & Embedding

- NLP에서 모델에 텍스트 데이터를 입력하기 위해 텍스트를 수치형으로 변환하는 과정
  - Tokenization: 텍스트를 의미 있는 단위(토큰)로 나누고, 이를 고유한 인덱스 번호(vocabulary 참조)로 변환
  - Embedding: 각 토큰 인덱스를 고차원의 벡터로 변환하여 모델이 처리할 수 있는 형태로 변환



# Seq2Seq and attention mechanism

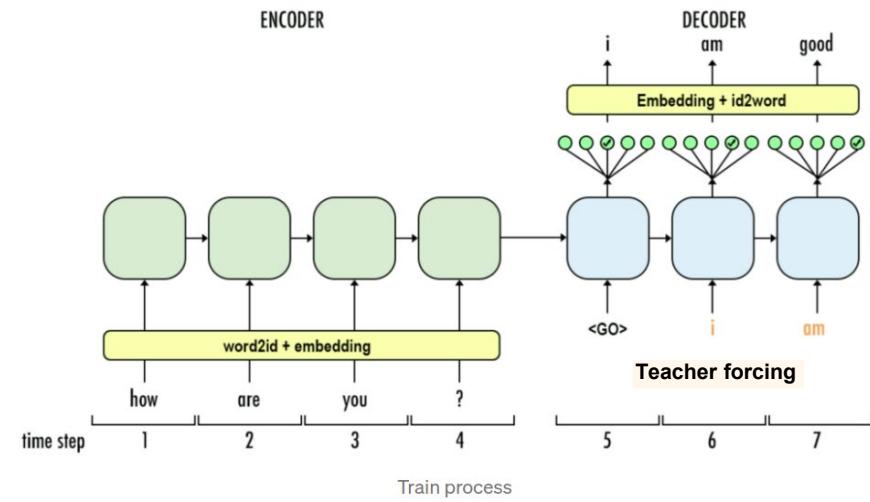
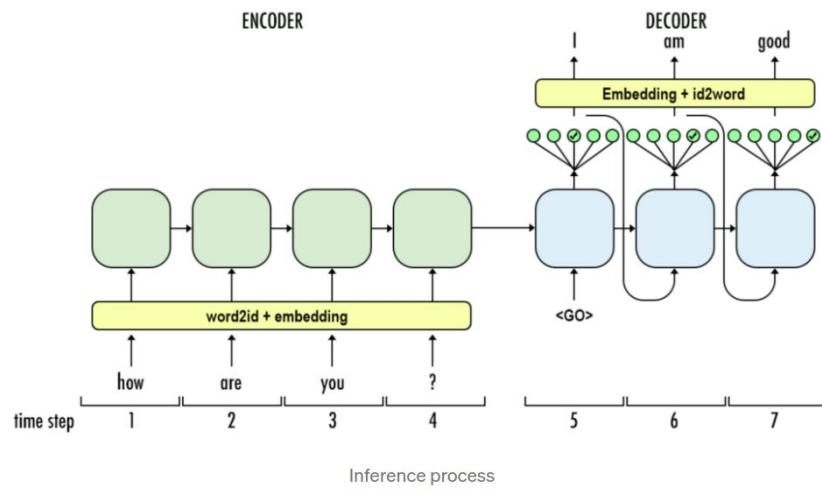
- Seq2Seq[1]는 Sequence to Sequence 모델의 약어로 순서 있는 데이터를 다른 순서 있는 데이터로 변환하는 신경망 모델
  - Seq2Seq는 인코더와 디코더로 구성되며, 입력 시퀀스를 인코더로 context vector로 변환 후 디코더가 이를 토대로 출력 시퀀스를 생성
  - Seq2Seq는 번역, 답변, 챗봇, 요약, 멀티모달 등 활용 가능



[1] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).  
Image Reference : <https://wikidocs.net/24996>

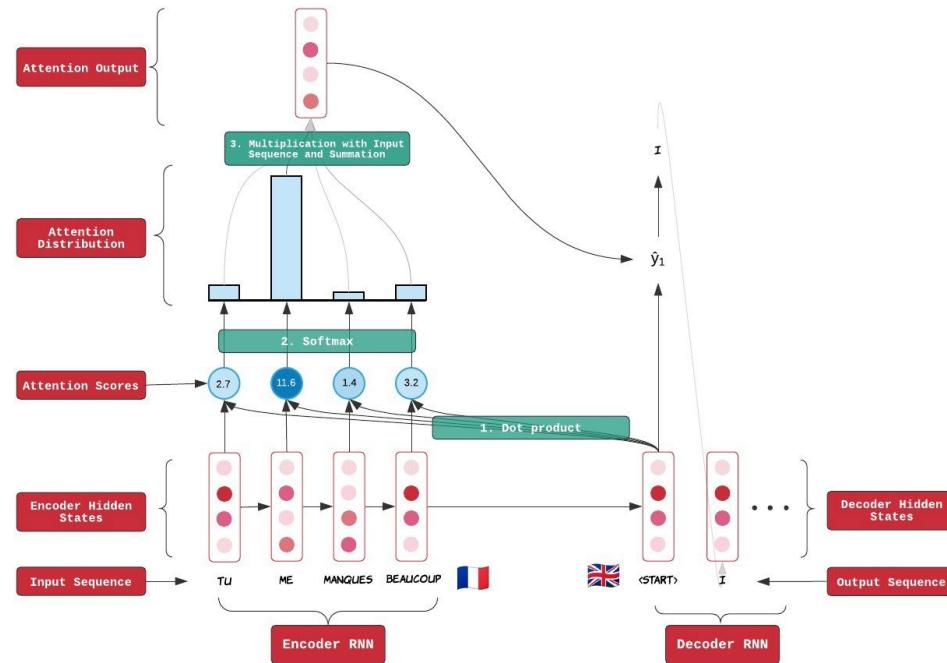
## • 교사강요(Teacher Forcing)

- Teacher Forcing 방식은 Autoregressive 모델의 학습 시 사용됨.
- 초기 모델의 예측은 부정확하므로, 학습 시 모델은 자신의 예측 단어가 아닌 실제 정답 단어를 다음 입력으로 사용하면서 학습함. 이는 초기 학습 단계에서 모델이 올바른 출력을 학습하도록 도움.



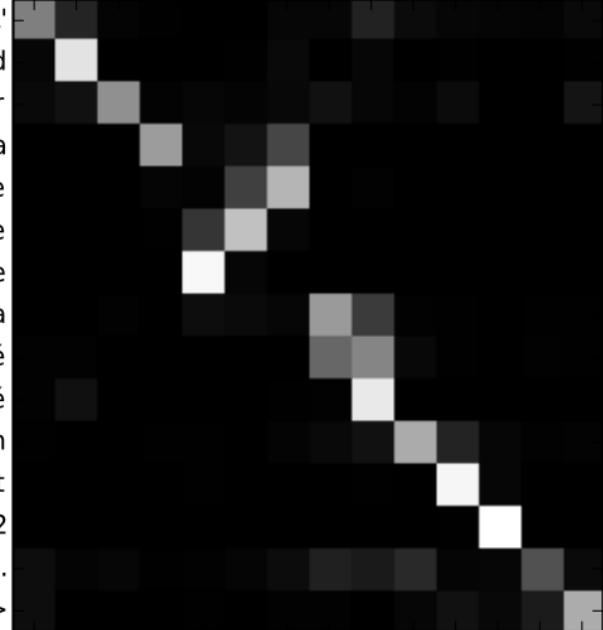
## • Attention mechanism

- Seq2Seq 모델은 입력 정보를 단일 문맥 벡터로 압축하는 방식 때문에 정보 손실이 발생
- 이를 극복하기 위해 Bahdanau et al, ICLR, 2015가 제안한 주의(Attention) 메커니즘이 도입

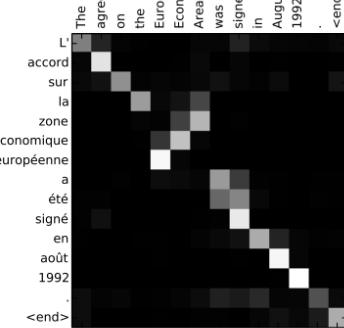


# Attention : visualization

The  
agreement  
on  
the  
European  
Economic  
Area  
was  
signed  
in  
August  
1992  
.  
<end>

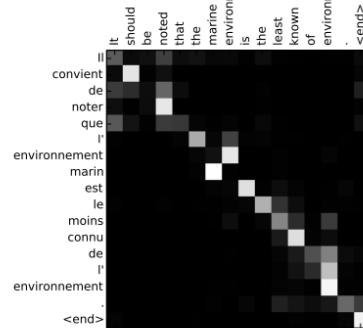


L'  
accord  
sur  
la  
zone  
économique  
européenne  
a  
été  
signé  
en  
août  
1992  
.  
<end>



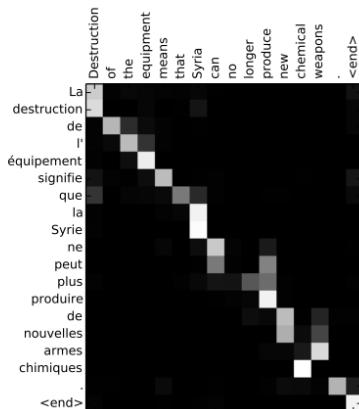
(a)

Il  
convient  
de  
noter  
que  
l'  
environnement  
marin  
est  
le  
moins  
connu  
de  
l'  
environnement  
.  
<end>



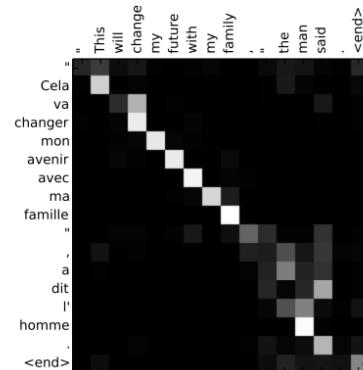
(b)

Destruction  
of  
the  
equipment  
means  
that  
Syria  
can  
no  
longer  
produce  
new  
chemical  
weapons  
.  
<end>



(c)

This  
will  
change  
my  
future  
with  
my  
family  
.  
" " "  
Cela  
va  
changer  
mon  
avenir  
avec  
ma  
famille  
."  
" " "  
a  
dit  
l'  
homme  
.  
<end>



(d)

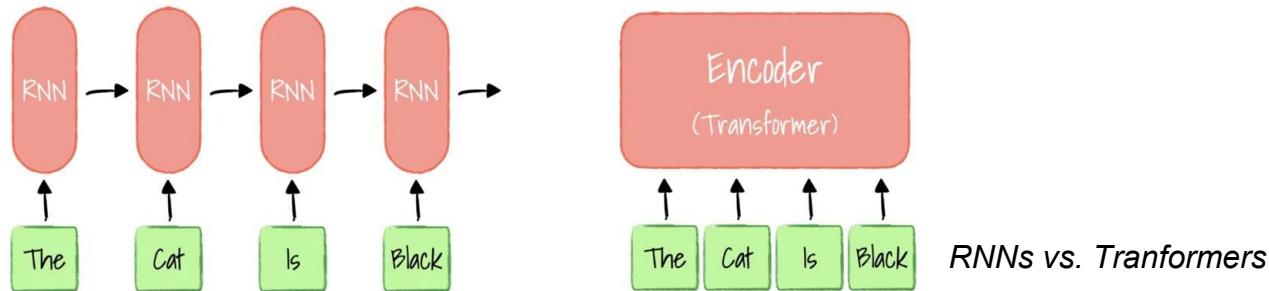
# Summary

- 충분히 많은 텍스트가 있다면 단어 사이의 컨텍스트에 대한 직접적 지도 학습 없이도 학습이 가능함(Word2Vec)
- 텍스트 전처리를 위한 Tokenization & Embedding
- RNN의 장기의존성 문제를 해결하는 Attention mechanism 발견
- Autoregressive language model은 Teacher forcing 방식으로 학습

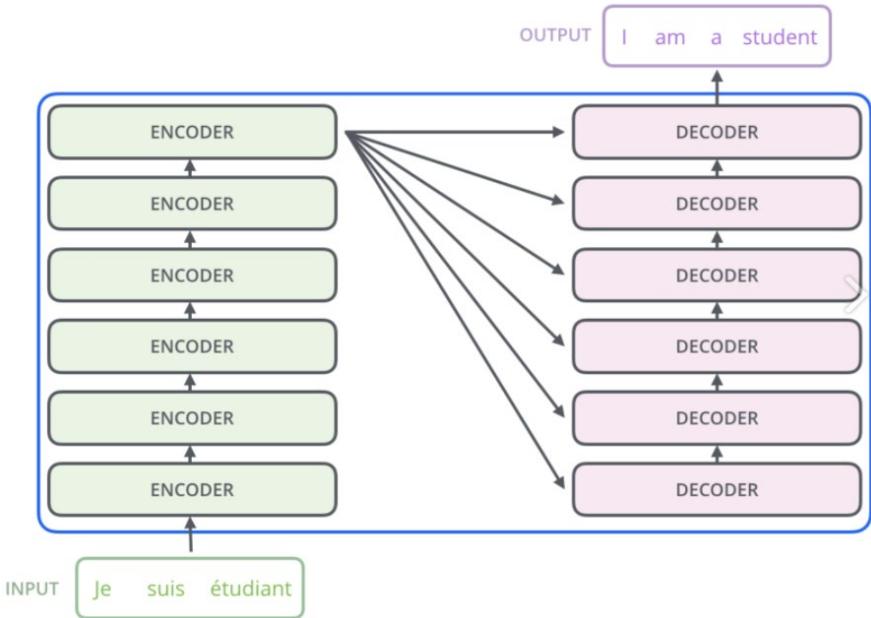
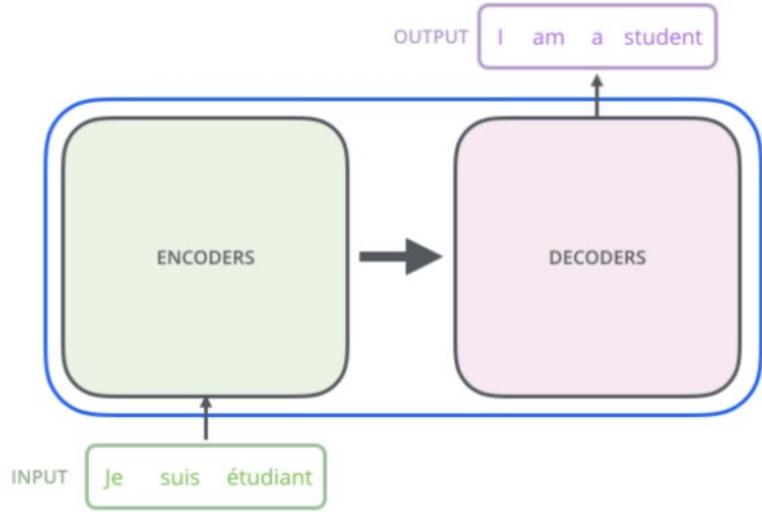
# Transformer

# Transformer : 언어모델의 새 지평

- Transformer [1] 는 Attention block으로 구성된 새로운 신경망 기반 언어모델
  - 병렬 처리 : 입력 시퀀스를 병렬적으로 한 번에 처리
  - 어텐션 메커니즘 : 특정 토큰과 입력 시퀀스의 모든 토큰 사이의 관계를 직접적으로 계산, 장기 의존성 문제 해결 및 컨텍스트 이해 능력 향상
  - 확장성 : 모델의 크기가 증가함에 따라 성능이 비례하여 향상

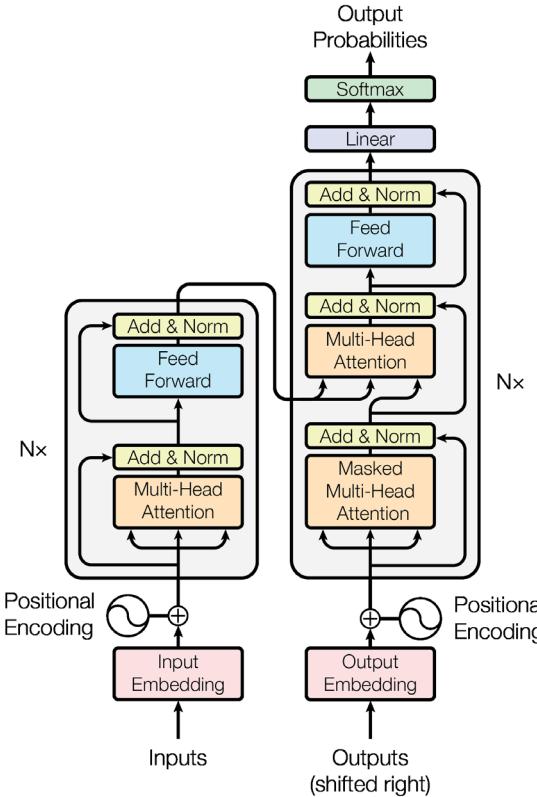


# Transformers' overall structure



<Transformer의 구조 : Encoders-Decoders>

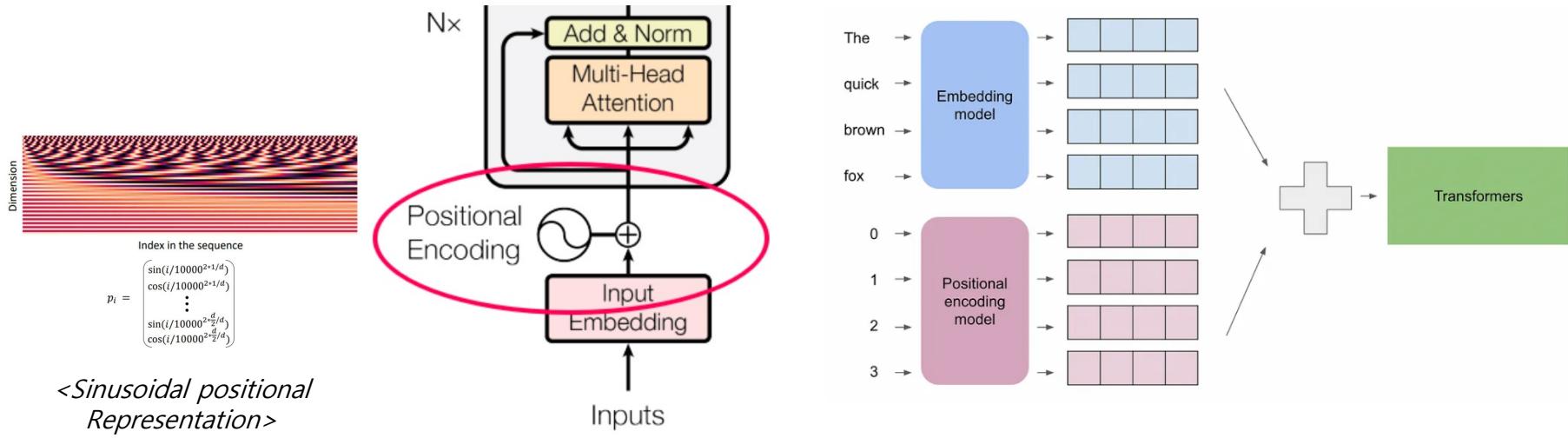
# New layers for Transformer



- **Positional encoding :**
  - 단어의 순서를 고려하지 않는 self-attention 구조를 보완
  - 단어의 상대적 위치를 표현하는 벡터를 단어 임베딩에 인코딩
- **Self-Attention :**
  - 입력 시퀀스의 자체 맥락에 집중하는 메커니즘
  - 입력 시퀀스의 모든 단어 관계 학습
  - 병렬 처리로 빠른 학습 및 추론

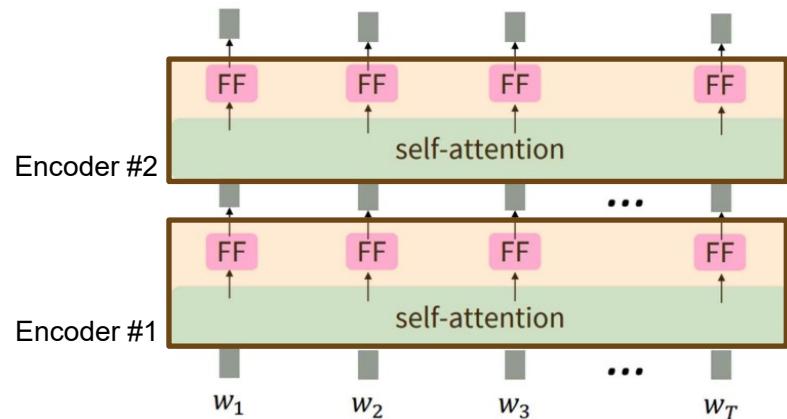
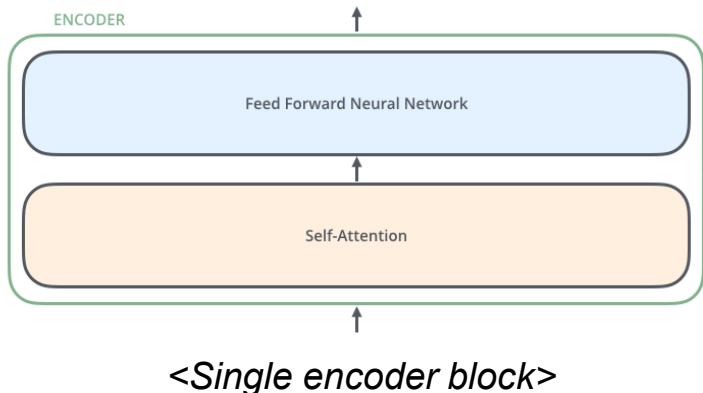
# Positional encoding

- Transformer는 RNN과 달리 시퀀스 내 토큰의 위치를 고려하는 구조가 아님
- Positional encoding은 문제를 해결하기 위해 토큰의 위치를 임베딩에 인코딩



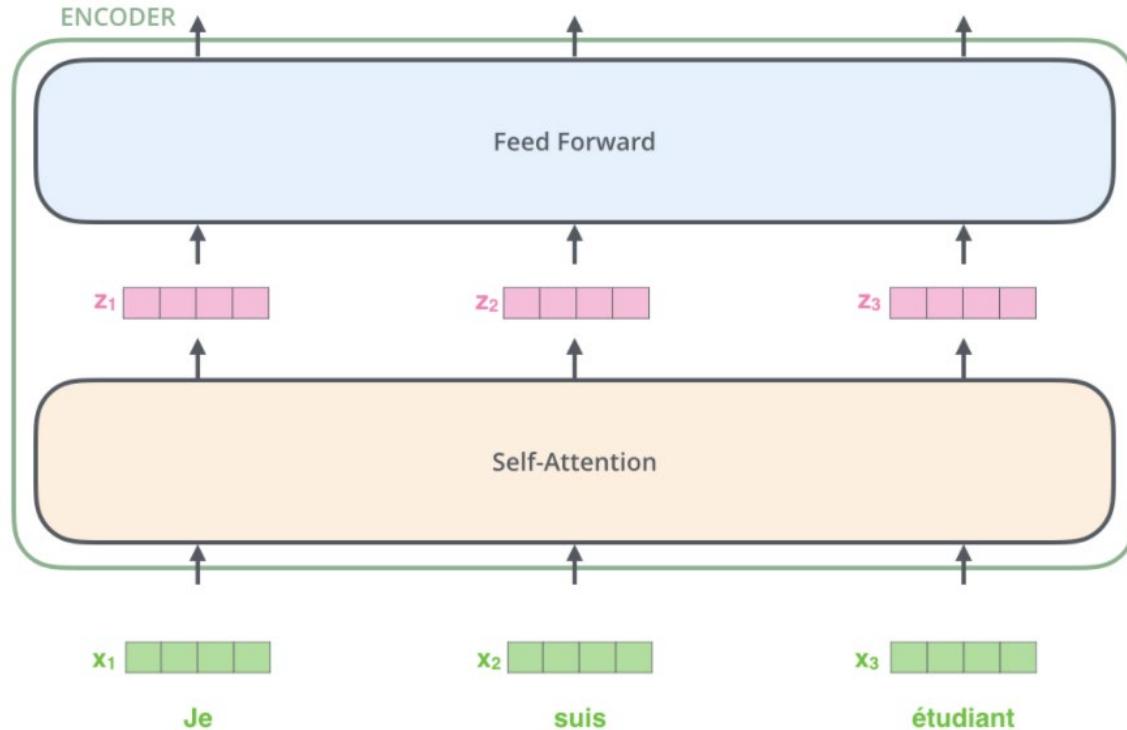
# Encoder block

- Self-attention
  - 하나의 단어를 encode할 때, 입력 내 모든 단어들과 관계(Attention)를 고려.
- Feed Forward Neural network
  - Self-Attention 층을 통과한 출력은 feed-forward 신경망으로 입력.
  - Attention layer에 없는 Nonlinearity 부여
  - 출력은 각 단어에 대해 독립적으로 계산.



*<Layered encoders>*

# Encoder block in-output shape



## <FFNN>

출력 : 64차원 벡터 시퀀스

- *Self-Attention*의 출력은 독립적으로 병렬처리
- 벡터의 차원은 하이퍼파라미터

## < Self-Attention >

출력 : 64차원 벡터 시퀀스

- 리스트의 크기는 문장 길이
- 벡터의 차원은 하이퍼파라미터

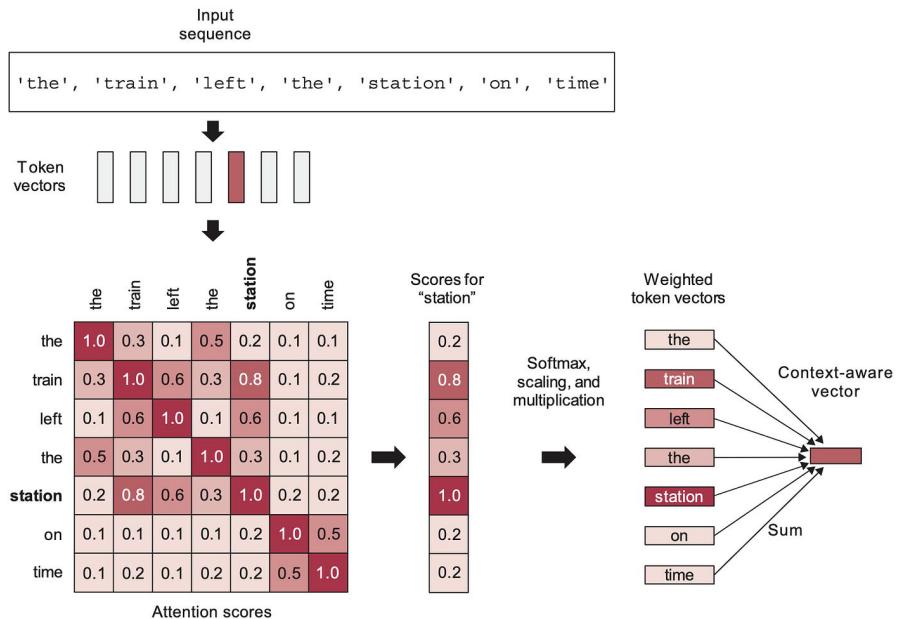
## <Embedding layer>

출력 : 512차원 벡터 시퀀스

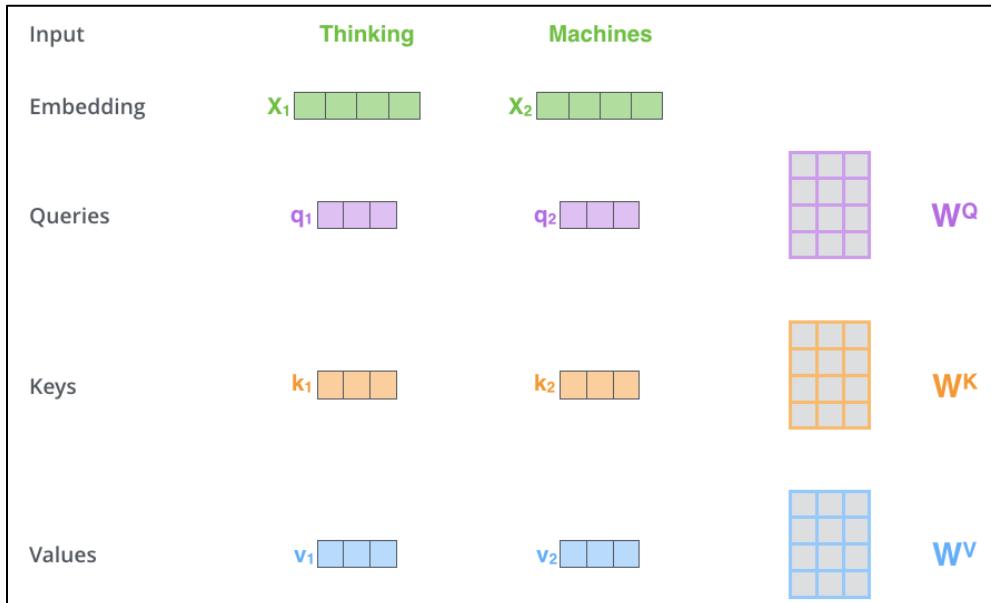
- 리스트의 크기는 문장 길이
- 벡터의 차원은 하이퍼파라미터

# Self-Attention

- Self-attention은 입력된 시퀀스 내에서 각 토큰이 다른 토큰들과 얼마나 관련이 있는지를 계산
- 토큰 임베딩을 선형적으로 재결합하여 새로운 토큰 임베딩 공간을 학습하는 매커리즘
  - 함께 나타나는 토큰은 임베딩 공간에서 서로 가깝게 위치(feat. Word2vec)



# Self-Attention 연산(1)



- ['Thinking', 'Machine'] 시퀀스의 입력
  - 512차원 토큰 임베딩
- 행렬곱으로 Query, Key, Value 벡터 생성.
  - 64차원 Q,K,V 벡터
  - Q, K, V는 attention을 계산하기 위해 도입한 추상적 개념

<Query>

-입력 단어와 다른 단어의 관계를 질의하는 벡터

<Key>

-다른 단어가 입력 단어 사이의 관계를 질의 받는 벡터

<Value>

-입력 단어에 대한 의미 정보를 내포한 벡터

-

<Self-Attention을 위한 Query, Key, Value 개념>

# Self-Attention 연산(2)

Input	Thinking	
Embedding	$x_1$	
Queries	$q_1$	
Keys	$k_1$	
Values	$v_1$	
Score	$q_1 \cdot k_1 = 112$	
	Machines	
	$x_2$	
	$q_2$	
	$k_2$	
	$v_2$	
	$q_1 \cdot k_2 = 96$	



Input	Thinking	
Embedding	$x_1$	
Queries	$q_1$	
Keys	$k_1$	
Values	$v_1$	
Score	$q_1 \cdot k_1 = 112$	
Divide by 8 ( $\sqrt{d_k}$ )	14	
Softmax	0.88	
	Machines	
	$x_2$	
	$q_2$	
	$k_2$	
	$v_2$	
	$q_1 \cdot k_2 = 96$	
	12	
	0.12	

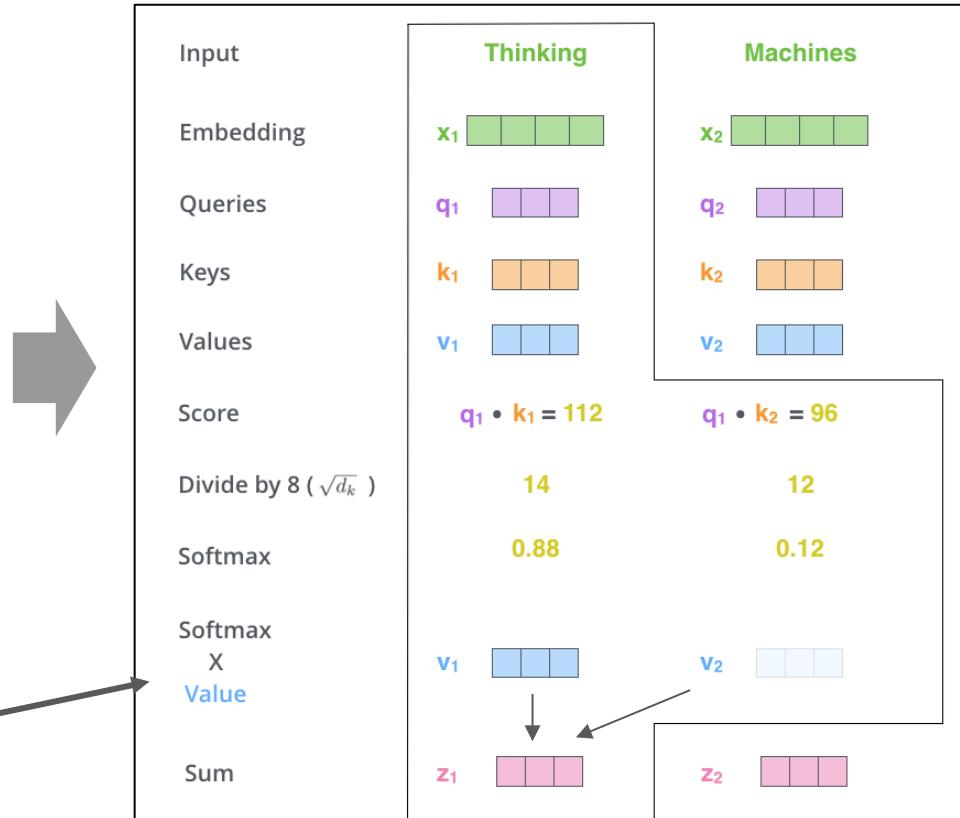
<Query, Key의 내적  
: attention score>

<Softmax를 사용한  
attention distribution 계산>

# Self-Attention 연산(3)

Input	Thinking	
Embedding	$x_1$ [green green green green]	
Queries	$q_1$ [purple purple purple]	
Keys	$k_1$ [orange orange orange]	
Values	$v_1$ [blue blue blue]	
Score	$q_1 \cdot k_1 = 112$	
Divide by 8 ( $\sqrt{d_k}$ )	14	
Softmax	0.88	
Input	Machines	
Embedding	$x_2$ [green green green green]	
Queries	$q_2$ [purple purple purple]	
Keys	$k_2$ [orange orange orange]	
Values	$v_2$ [blue blue blue]	
Score	$q_1 \cdot k_2 = 96$	
Divide by 8 ( $\sqrt{d_k}$ )	12	
Softmax	0.12	

<attention distribution을 사용한,  
Value의 weighted sum>



# Self-Attention 연산(4): matrix form

$$\begin{array}{c} X \quad W^Q \quad Q \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \end{array}$$
$$\begin{array}{c} X \quad W^K \quad K \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \end{array}$$
$$\begin{array}{c} X \quad W^V \quad V \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \\ \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \quad \begin{matrix} \text{---} \\ | \end{matrix} \end{array}$$

<Embeddings에 대해  
Queries, Keys, Values 계산>



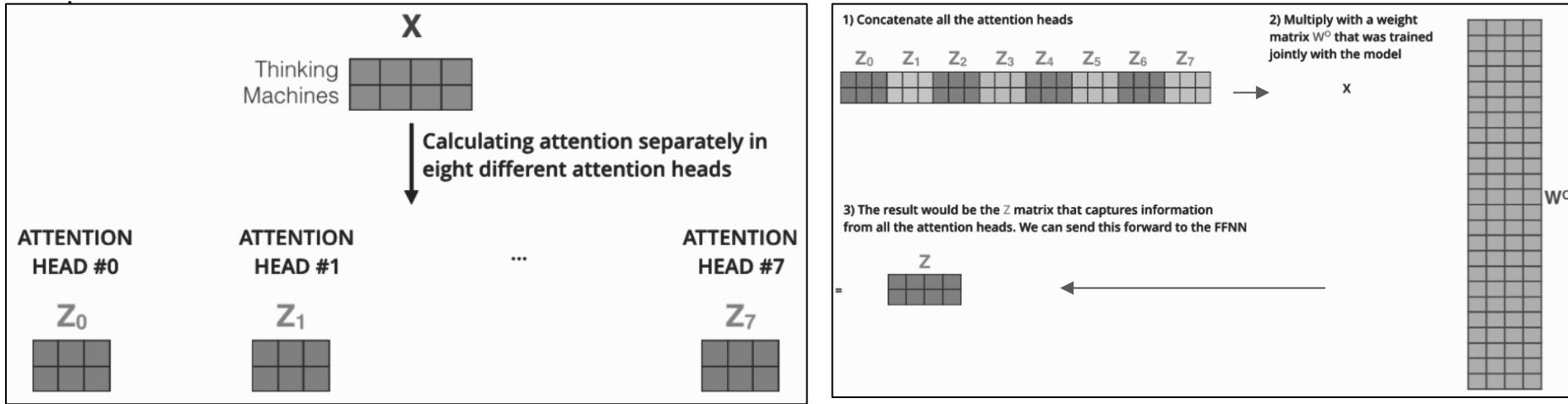
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) = Z$$

각 입력의 Value 벡터의 가중치합이 출력

<행렬곱 형태로 Self-Attention 계산>

# Multi-head self-attention 연산(1)

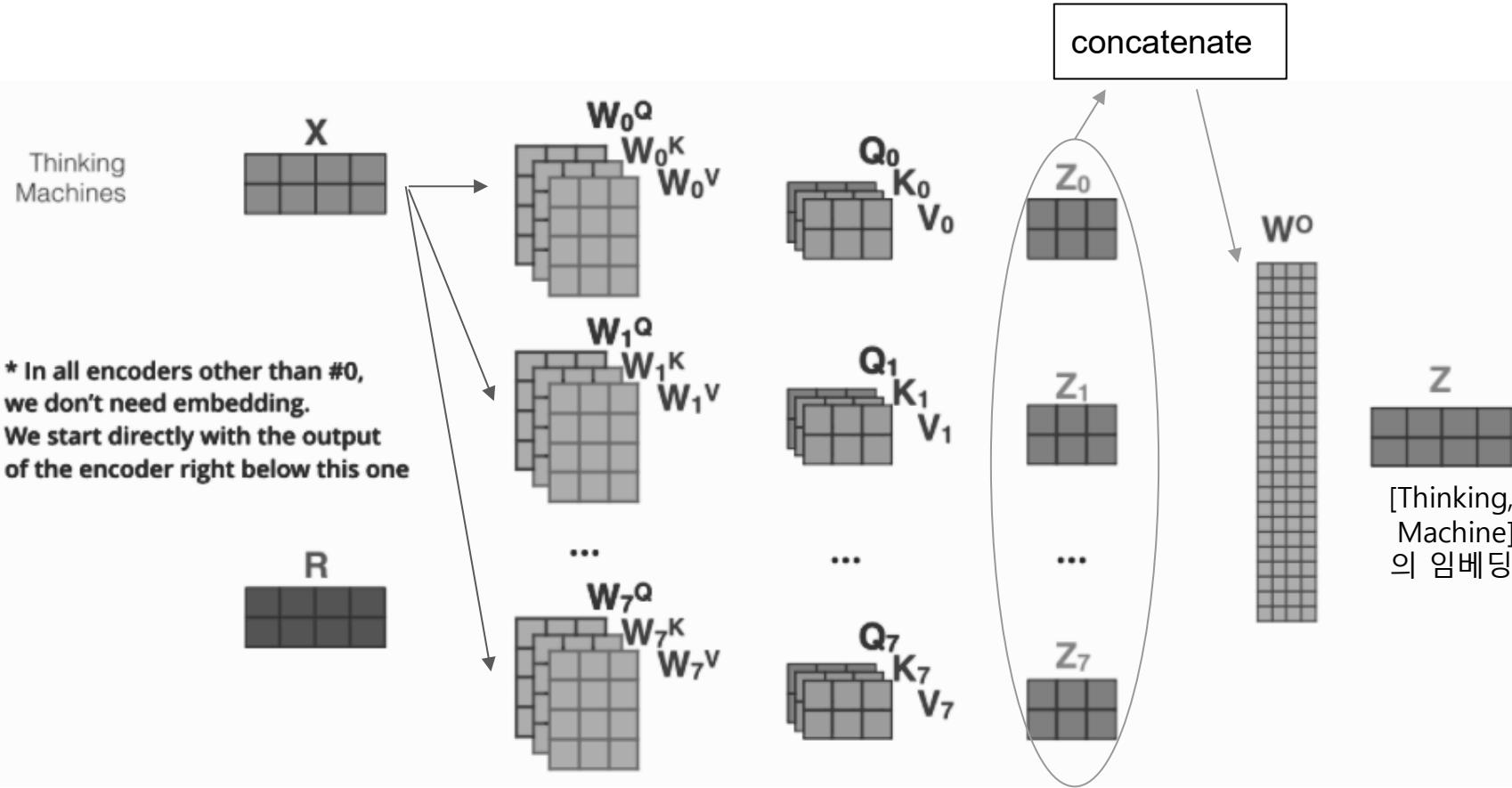
- Multi-head attention은 모델의 표현력과 학습 능력을 개선하기 위해 도입
  - 각 head가 입력 데이터의 다른 특성이나 관점에서 정보를 추출
  - Self-attention 계층이 여러 representation 공간을 다중으로 만들 수 있음.
  - attention head는 독립적으로 계산될 수 있기 때문에, 병렬 처리 하드웨어를 활용 가능



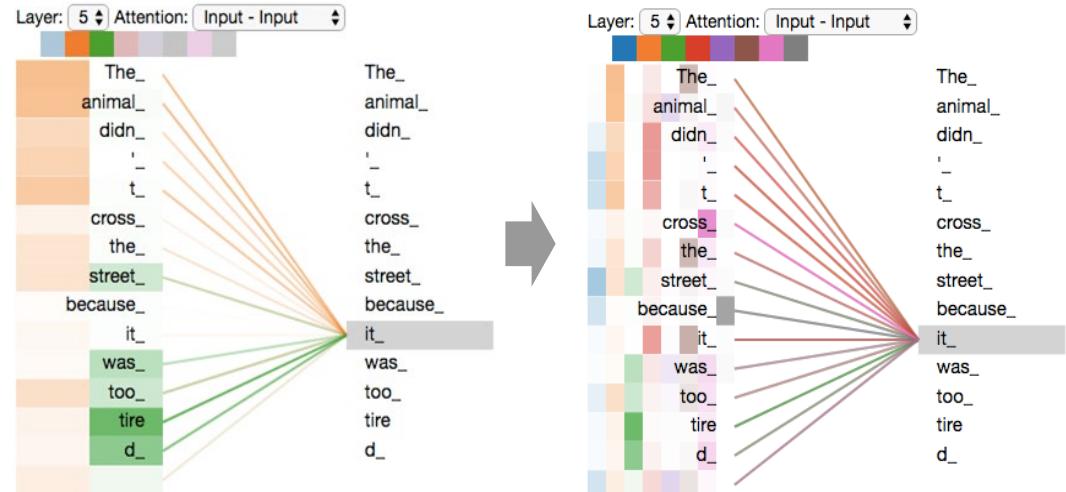
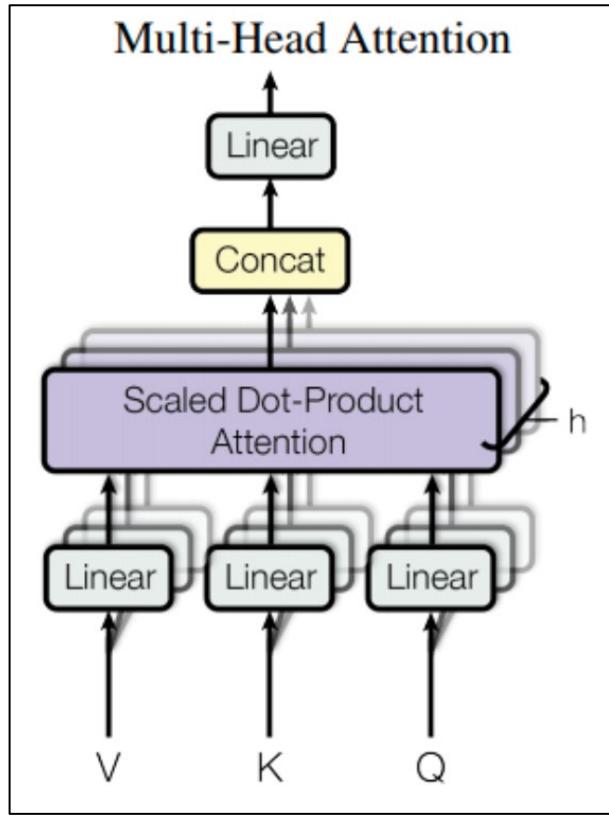
<다중 attention에 의한 출력들>

(1) 출력들은 concatenate  
(2) 행렬곱으로 임베딩 차원에 매핑

# Multi-head self-attention 연산(2)



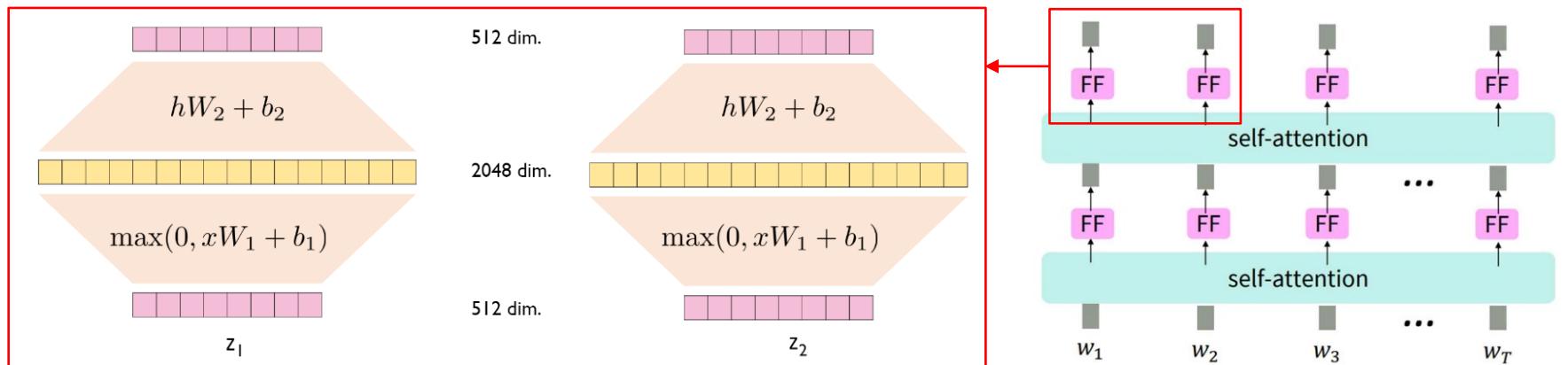
# Multi-head self-attention 연산(3)



- 한 단어가 여러 개의 단어에 동시에 집중
- 하나의 attention이 아니라 여러 개의 attention 사용

# Position-wise Feed-Forward Networks

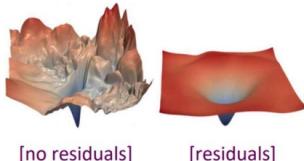
- Position-wise Feed-Forward Network (FFN)은 Encoder와 Decoder 층에 포함된 중요한 구성 요소
  - Transformer 아키텍처의 핵심적인 비선형 변환 메커니즘을 제공
  - "Position-wise"는 시퀀스의 각 위치에 있는 단어 벡터에 독립적으로, 그리고 동일한 연산을 적용한다는 의미



# Residual connection & LayerNorm

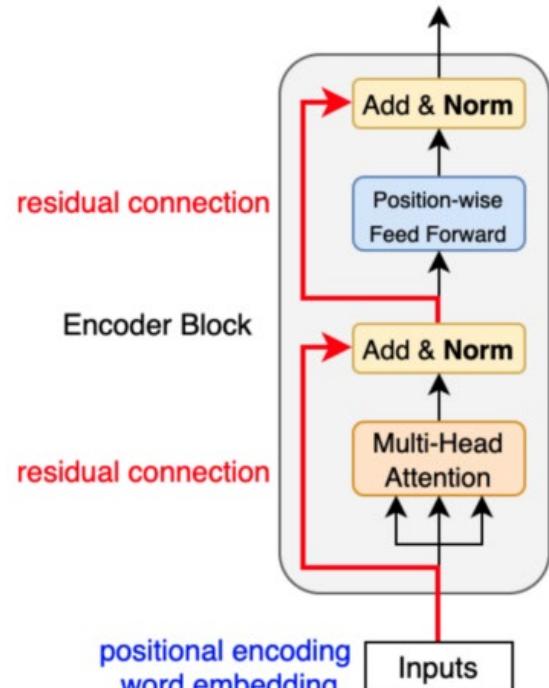
- Add : ResNet의 Residual connection

- 그래디언트 소실 및 폭발 문제 완화
  - 학습 과정의 가속화
    - ✓ 네트워크가 학습해야 할 함수의 복잡성 감소



- Normalize : Layer Normalization

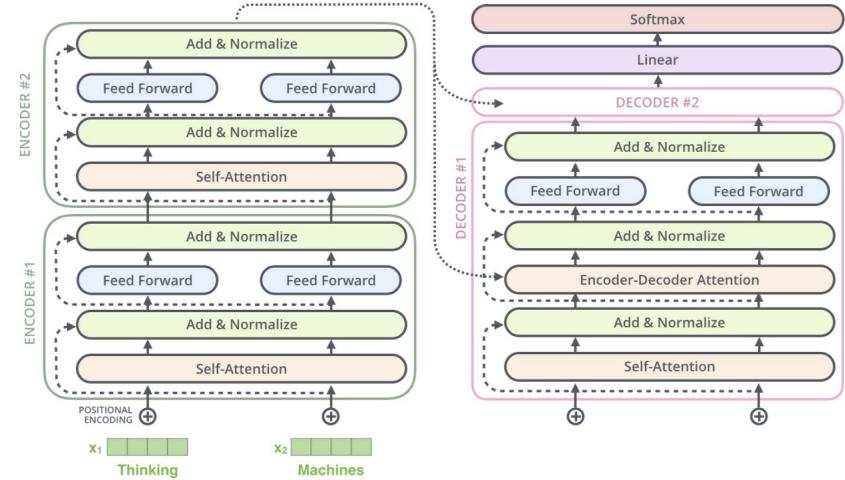
- 학습 안정성 향상 및 학습 속도 개선
    - ✓ 정규화 과정은 입력 데이터의 분포를 일정하게 유지
    - ✓ 입력 데이터의 정규화는 학습률을 높여 학습 속도 개선
  - 배치 크기에 대한 유연성 (vs. batch normalization)
    - ✓ Time-step들의 각 특징에 대해 정규화
    - ✓ 각 샘플 별 독립적으로 적용 가능



< Residual connections,  
Layer Normalization >

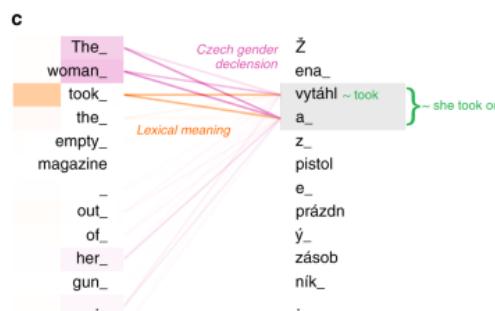
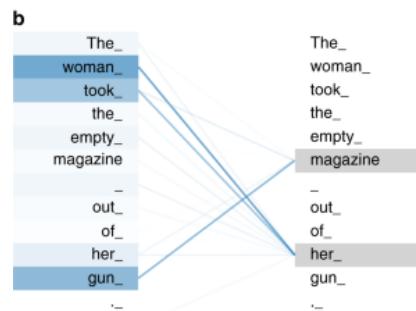
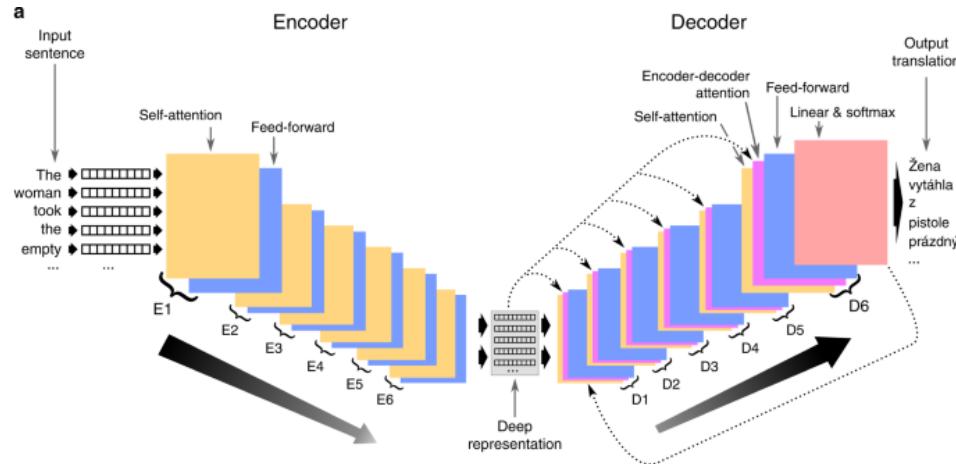
# Decoder block

- Decoder 연산 시 Encoder과 차이점
  - Masked self-attention
    - ✓ Decoder 내에서의 self-attention 메커니즘은 'masked' 형태로 적용
    - ✓ 이는 아직 생성되지 않은 미래의 출력을 참조하지 않도록 하기 위함
  - Cross-attention
    - ✓ Encoder로부터의 출력을 활용하여, 입력 시퀀스와 관련된 컨텍스트 계산
- Decoder는 이러한 메커니즘을 통해 각 time step마다 출력 시퀀스의 한 요소를 순차적으로 생성

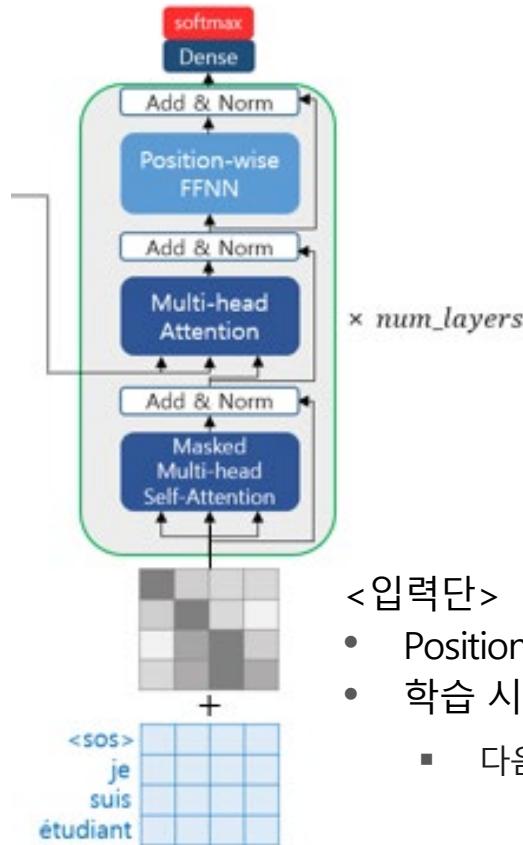


< Decoder에 도입된 두 주요 메커니즘 : Masked Self-attention, Cross-attention >

# Transformer : Decoder

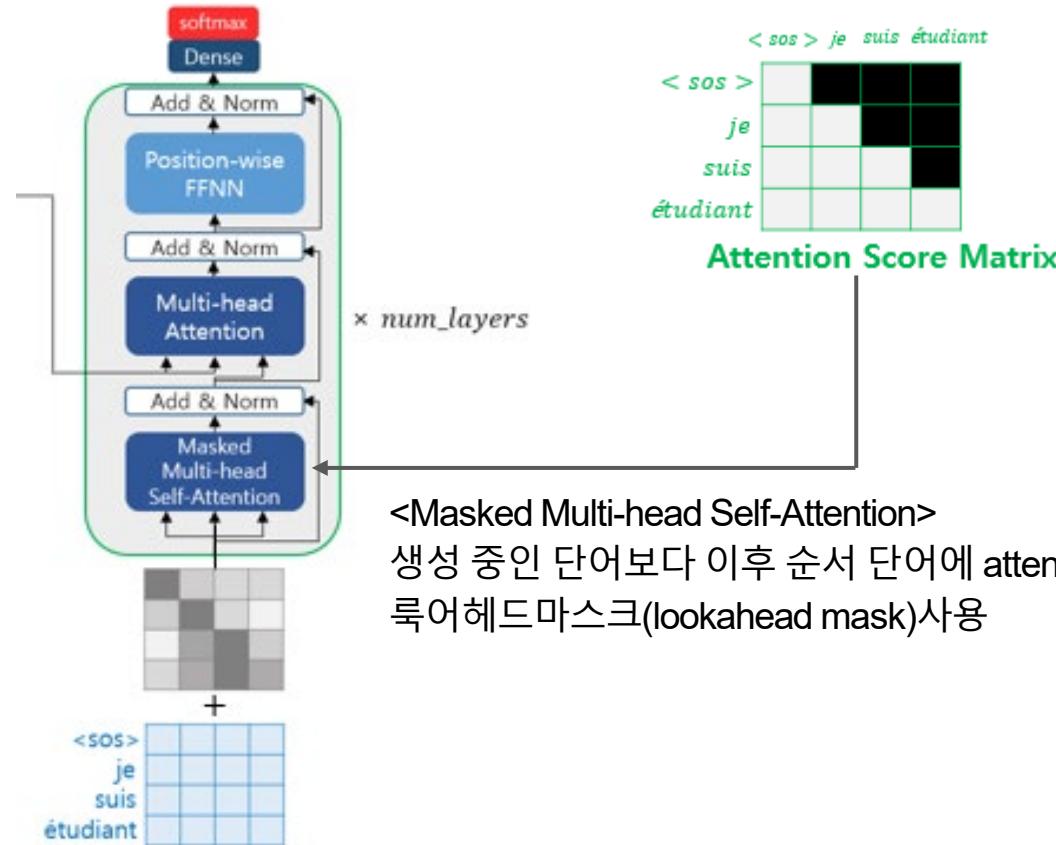


# Transformer : Decoder

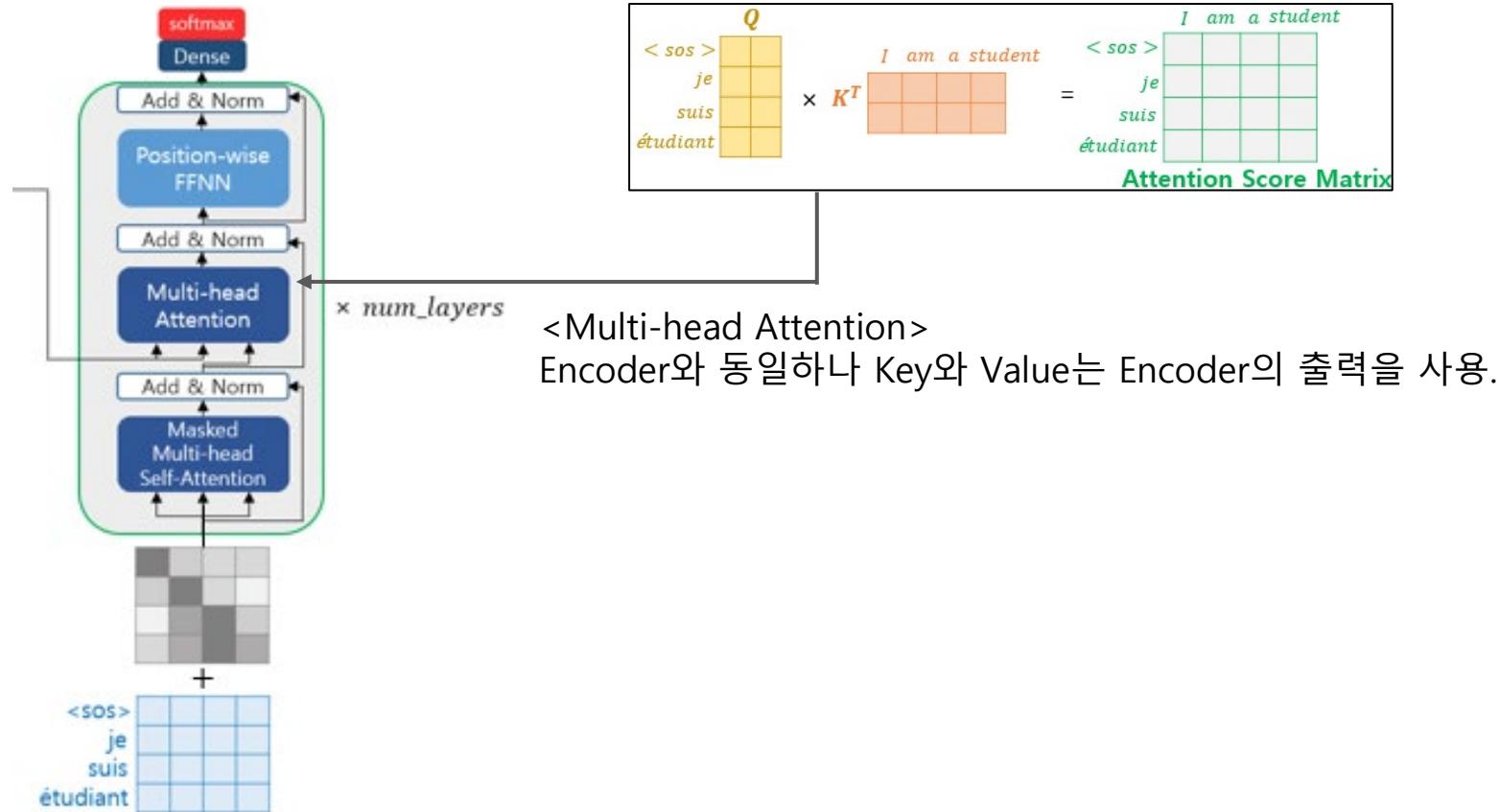


- <입력단>
- Positional encoding 적용
  - 학습 시 Teacher forcing 사용
    - 다음 time step 입력으로 실제 정답으로 알려줌

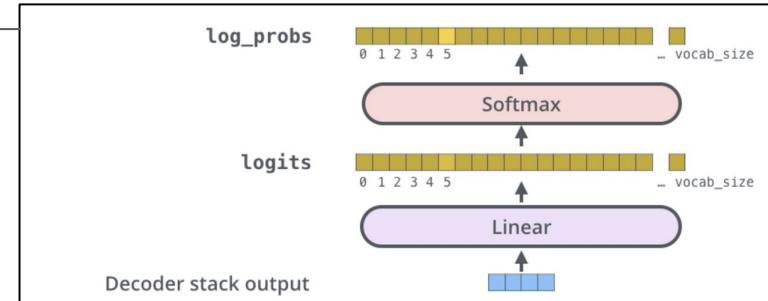
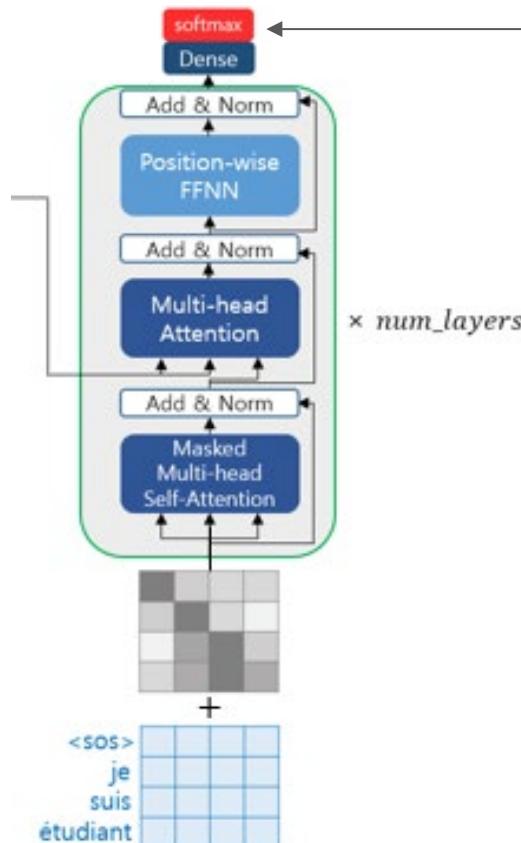
# Transformer : Decoder



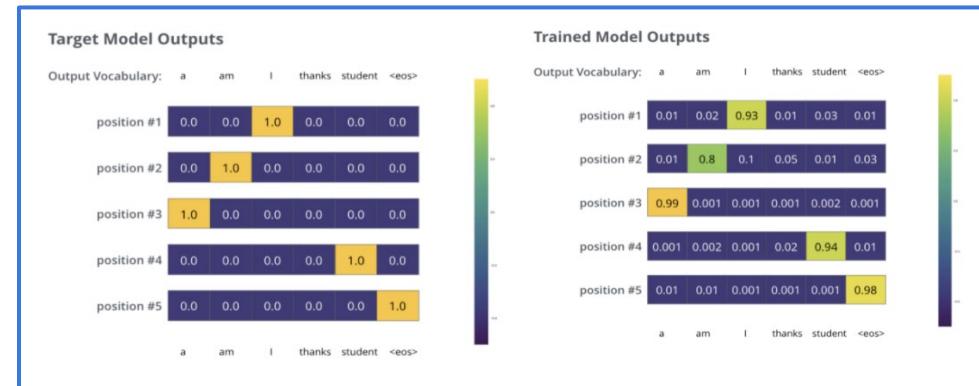
# Transformer : Decoder



# Transformer : Decoder



<Softmax after Dense>  
각 Step에 대한 cross entropy로 분류 학습.



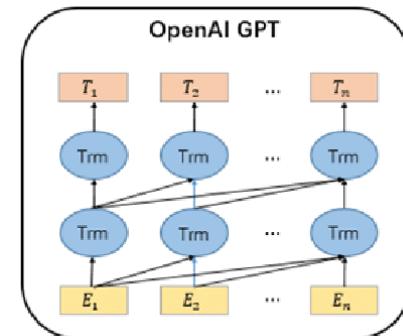
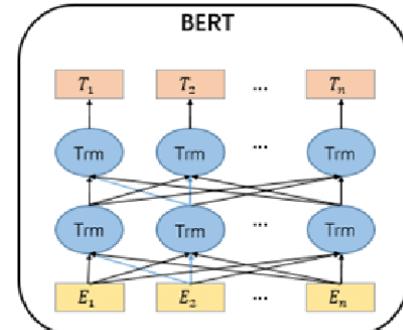
# Summary

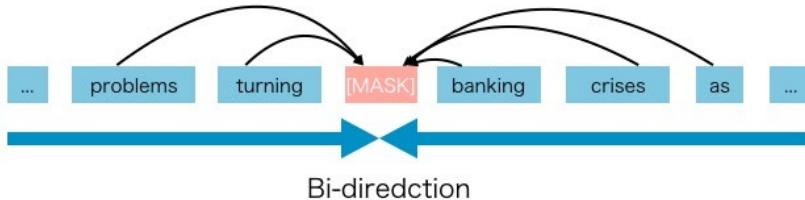
- Transformer는 토큰 시퀀스를 입력 받아 문장 특징을 추출하는 인코더와 문장 생성에 특화된 디코더로 구성
- Self-attention은 토큰 사이의 query, key 상관성으로 가중치를 계산하고 value들의 가중치 합 계산으로 새로운 토큰 표현

# BERTs와 GPTs

# Bert & GPT

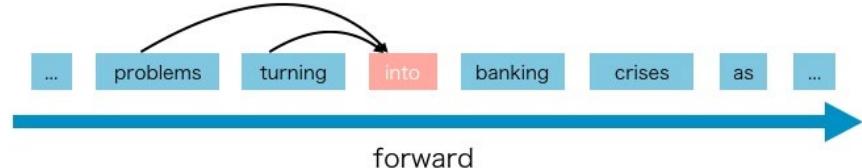
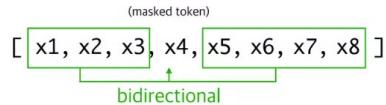
- BERT (Bidirectional Encoder Representations from Transformers), Google AI, 2018년
  - **BERT는 Transformer 아키텍처의 Encoder 부분으로 언어이해에 중점**
  - 양방향성: BERT는 문장의 앞과 뒤를 모두 고려하여 단어의 맥락을 이해하여 더 정확한 단어의 의미를 파악
  - Pre-training과 Fine-tuning의 조합: BERT는 대규모 텍스트 데이터로 사전 학습을 수행한 후, 특정 작업에 맞게 추가 학습(미세 조정)을 통해 다양한 NLP 작업에 적용
- GPT (Generative Pre-training Transformer), OpenAI, 2018년
  - **GPT는 Transformer 아키텍처의 Decoder 부분으로 문장생성에 중점**
  - 단방향성: 초기 GPT 모델은 주로 문장의 이전 단어들만을 고려하여 다음 단어를 예측. GPT-3와 같은 최신 버전에서는 더 발전된 기법을 사용하지만, 기본적으로 생성적 예측에 초점
  - 규모의 확장성: 특히 GPT-3는 엄청난 규모(1750억 개의 파라미터)로 사전 학습되어 다양한 작업을 zero-shot 또는 few-shot 학습으로 처리





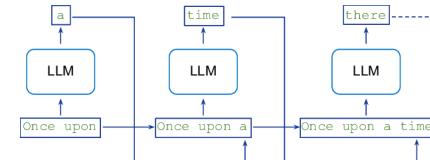
## Auto-encoding language model

$$\max_{\theta} \log p_{\theta}(\bar{x} | \hat{x}) \approx \sum_{t=1}^T m_t \log p_{\theta}(x_t | \hat{x})$$



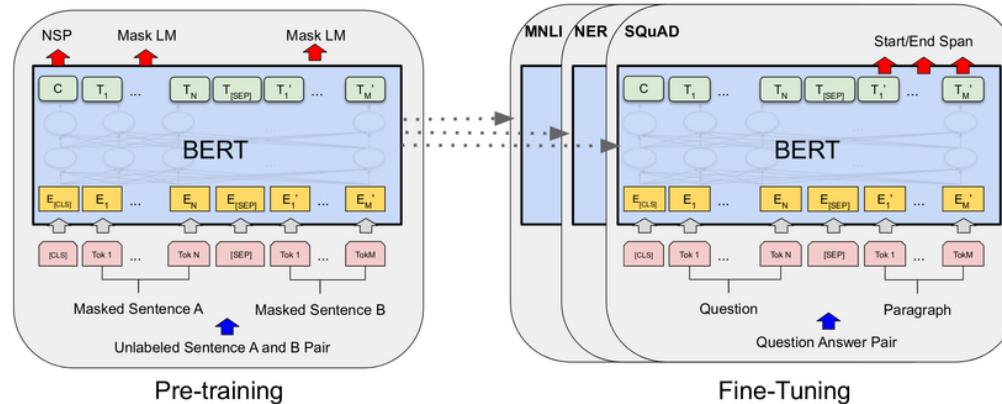
## Auto-regressive language model

$$\max_{\theta} \log p_{\theta}(x) = \sum_{t=1}^T \log p_{\theta}(x_t | x_{<t})$$



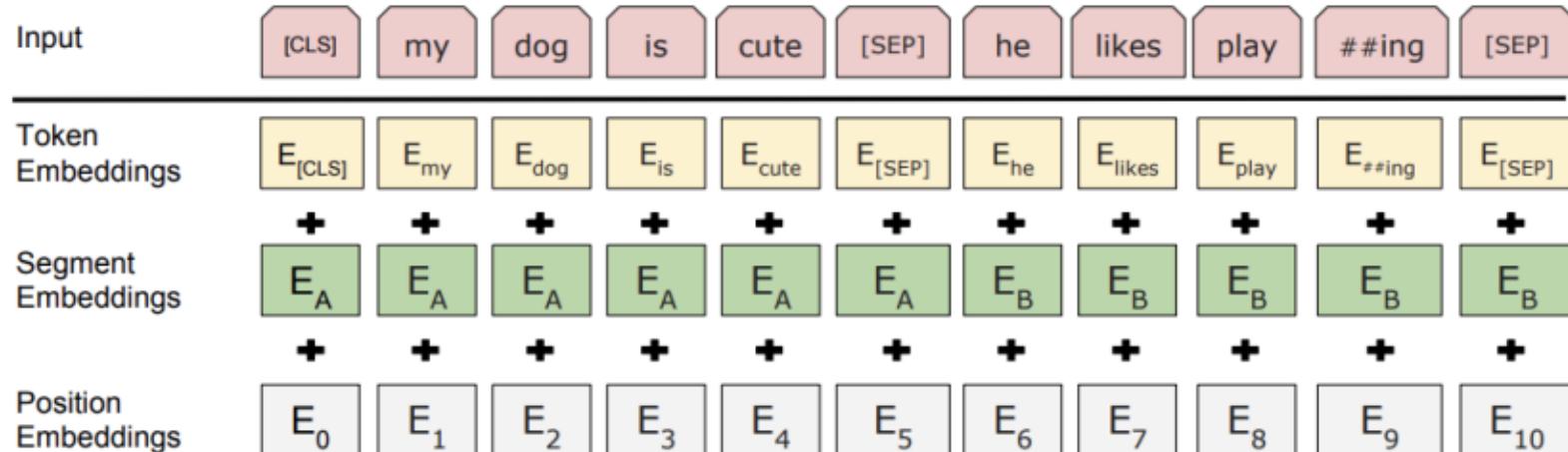
# Bert : Pre-training of Deep Bidirectional Transformers for Language Understanding

- BERT는 양방향 인코더 기반 언어모델
  - 입력 시퀀스를 양쪽에서 동시에 처리하고, 각 단어를 예측할 때 왼쪽, 오른쪽 문맥 모두를 고려해 임베딩.
    - ✓ BERT\_base: 12 blocks, 768 hidden space dimension, 12 heads, in total 110M parameters
    - ✓ BERT\_large: 24 blocks, 1024 hidden space dimension, 16 heads, in total 340M parameters
- Pre-training과 Fine-tuning의 조합
  - 큰 규모의 일반 텍스트 데이터셋에서 사전 학습(pre-training)된 후, 특정 NLP 작업에 맞게 미세 조정(fine-tuning)
  - 대규모의 언어 이해를 위해 지도학습 대신 문장의 일부 단어를 스스로 예측해 학습



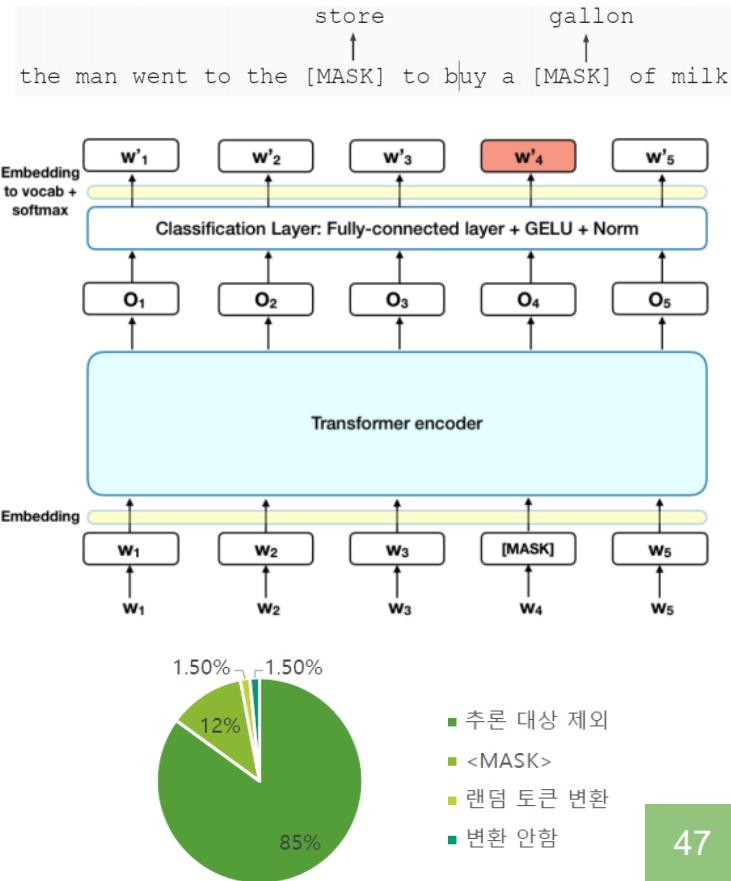
# Bert의 입력구조

- Bert는 입력 시퀀스를 효과적으로 처리하기 위한 special token을 가지고 있음.
  - [CLS]: 입력 문장의 embedding을 생성하기 위한 토큰으로 문장의 내재적 구조를 이해
  - [SEP]: 두 개의 문장을 구분하거나, 입력 시퀀스의 끝을 나타내는 용도의 token
  - [MASK]: Masked Language Model(MLM) 학습 과정에서 사용되는 토큰
  - [PAD]: 모델에 입력되는 시퀀스의 길이를 동일하게 맞추기 위해 사용되는 토큰



# Pre-Training Tasks

- Task #1 : Masked Language Model(MLM)
  - 랜덤하게 전체 token의 15%를 추론의 대상으로 사용.
    - ✓ 80%의 경우: [MASK]  
e.g. my dog is hairy → my dog is [MASK]
    - ✓ 10%의 경우: random word e.g. my dog is hariy → my dog is apple
    - ✓ 10%의 경우: 원래의 단어
  - 단어가 문장 안에서 가지는 “맥락”을 파악하여 단어의 중의성 문제를 해결
    - ✓ e.g. 밤을 먹다. vs 밤이 되었다
    - ✓ 같은 단어, 같은 위치에 있더라도 주변 문맥을 통해 서로 다른 embedding 벡터를 가지게 됨



# Pre-Training Tasks

- Task #2 : Next Sentence Prediction (NSP)
  - 두 문장이 주어졌을 때 이어지는 문장인지 여부를 예측
    - 50%: sentence A, B가 실제 next sentence
    - 50%: sentence A, B가 corpus에서 random으로 뽑힌(관계가 없는 두 문장)

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT <sub>BASE</sub>	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9

\*No NSP : MLM은 사용하지만, NSP를 없앤 모델

MNLI-m : Textual Entailment

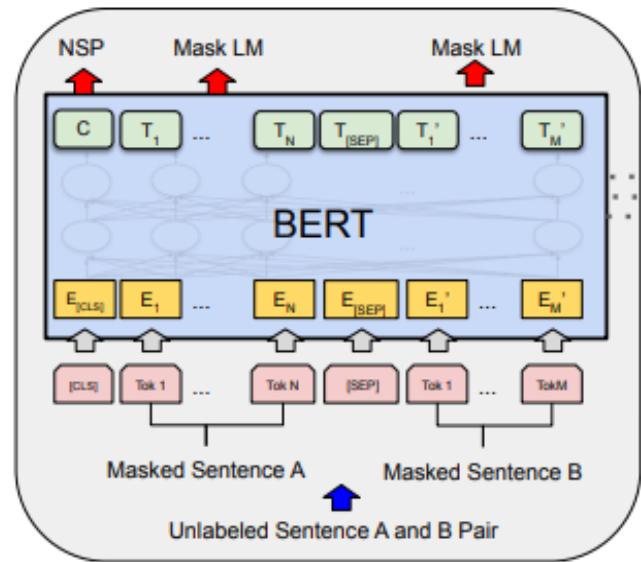
QNLI : Question Answering

MRPC, SST-2 : Text Classification

SQuAD : Question Answering

Input = [CLS] The man want to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]  
Label = IsNext

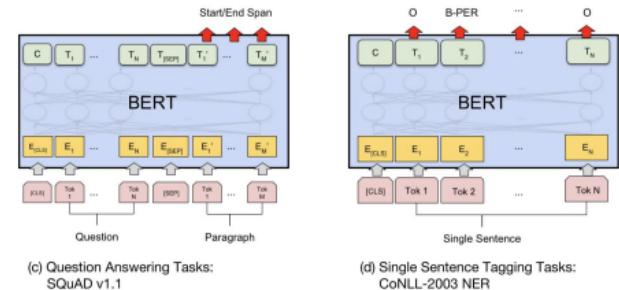
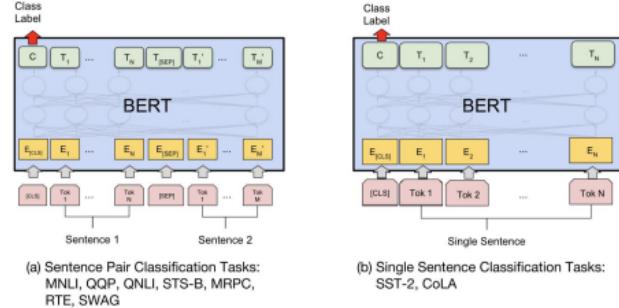
Input = [CLS] The man want to [MASK] store [SEP] penguin [MASK] are flight less birds [SEP]  
Label = NotNext



# Fine-tuning

- Fine-tuning은 batch size, learning rate, training epochs를 제외하고 pre-training 시의 hyper parameter와 동일하게 세팅
  - 사전 학습한 모델의 구조를 유지한 채 다양한 downstream 태스크에 대해 좋은 성능을 보임
    - ✓ Sentence Pair Classification : 두 문장의 관계를 분류
    - ✓ Single Sentence Classification : 하나의 문장이 어떤 카테고리에 속하는지를 판별
    - ✓ Question Answering : 문단과 질문을 함께 입력으로 받고, 문단 내에서 답변의 시작과 끝 위치를 예측
    - ✓ Single Sentence Tagging : 각 토큰(단어나 구)에 특정 태그를 할당
  - 모든 task에 대해 SOTA 달성

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>



# BERT 개선 모델

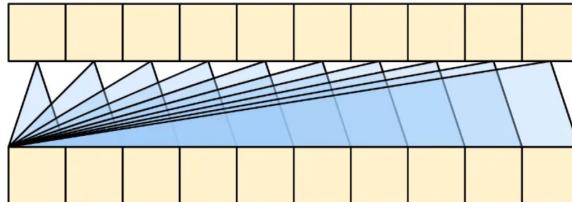
- RoBERTa (Robustly optimized BERT pretraining approach)
  - BERT의 학습 방식을 개선하여 Robustness를 향상
  - NSP가 성능 향상에 크게 기여하지 않는다고 보고, 이를 제외
  - 동일한 데이터를 여러 번 사용하여 학습하는 대신, 동적마스킹 방법 사용
- DistilBERT (Distilled BERT)
  - BERT를 학습된 모델 (Teacher)로 사용하여, 더 작고 빠른 모델 (Student)을 학습
  - 학습 과정에서 Knowledge Distillation 기술 사용
  - BERT보다 40% 작고 60% 빠르면서도 97% 수준의 성능 유지
- ALBERT (A Lite BERT)
  - BERT 모델의 파라미터 수를 줄여 속도를 향상시키면서 성능 유지
  - 모델 구조 및 학습 방식 개선
  - Next Sentence Prediction (NSP) 대신 Sentence Order Prediction (SOP) 사용

# BERT 개선 모델

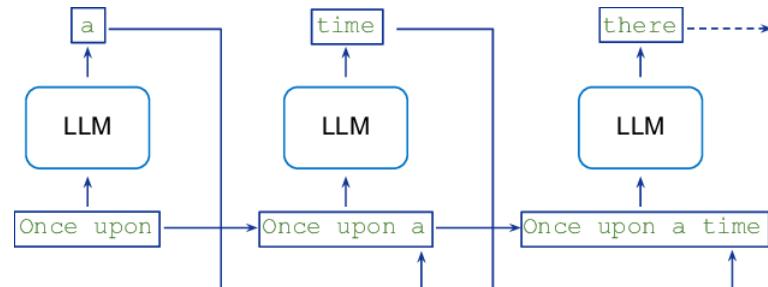
Comparison	BERT October 11, 2018	RoBERTa July 26, 2019	DistilBERT October 2, 2019	ALBERT September 26, 2019
Parameters	<b>Base:</b> 110M <b>Large:</b> 340M	<b>Base:</b> 125 <b>Large:</b> 355	<b>Base:</b> 66	<b>Base:</b> 12M <b>Large:</b> 18M
Layers / Hidden Dimensions / Self-Attention Heads	<b>Base:</b> 12 / 768 / 12 <b>Large:</b> 24 / 1024 / 16	<b>Base:</b> 12 / 768 / 12 <b>Large:</b> 24 / 1024 / 16	<b>Base:</b> 6 / 768 / 12	<b>Base:</b> 12 / 768 / 12 <b>Large:</b> 24 / 1024 / 16
Training Time	<b>Base:</b> 8 x V100 x 12d <b>Large:</b> 280 x V100 x 1d	1024 x V100 x 1 day (4-5x more than BERT)	<b>Base:</b> 8 x V100 x 3.5d (4 times less than BERT)	[not given] <b>Large:</b> 1.7x faster
Performance	Outperforming SOTA in Oct 2018	88.5 on GLUE	97% of BERT-base's performance on GLUE	89.4 on GLUE
Pre-Training Data	BooksCorpus + English Wikipedia = 16 GB	BERT + CCNews + OpenWebText + Stories = 160 GB	BooksCorpus + English Wikipedia = 16 GB	BooksCorpus + English Wikipedia = 16 GB
Method	Bidirectional Transformer, MLM & NSP	BERT without NSP, Using Dynamic Masking	BERT Distillation	BERT with reduced parameters & SOP (not NSP)

# GPT : Generative Pre-training Transformer

- OpenAI의 GPT는 방대한 데이터셋에서 사전 학습한 Decoder 기반 언어 모델
  - 사전 학습된 모델을 다양한 NLP 작업에 맞게 추가 학습하여 적용할 수 있는 범용성
  - GPT는 챗봇, 번역, 텍스트 생성 등에 활용
- BERT의 양방향 인코딩 방식과 달리 GPT의 각 토큰은 단방향 컨텍스트에만 집중
  - NLG(Natural Language Generation)에는 좋은 특성

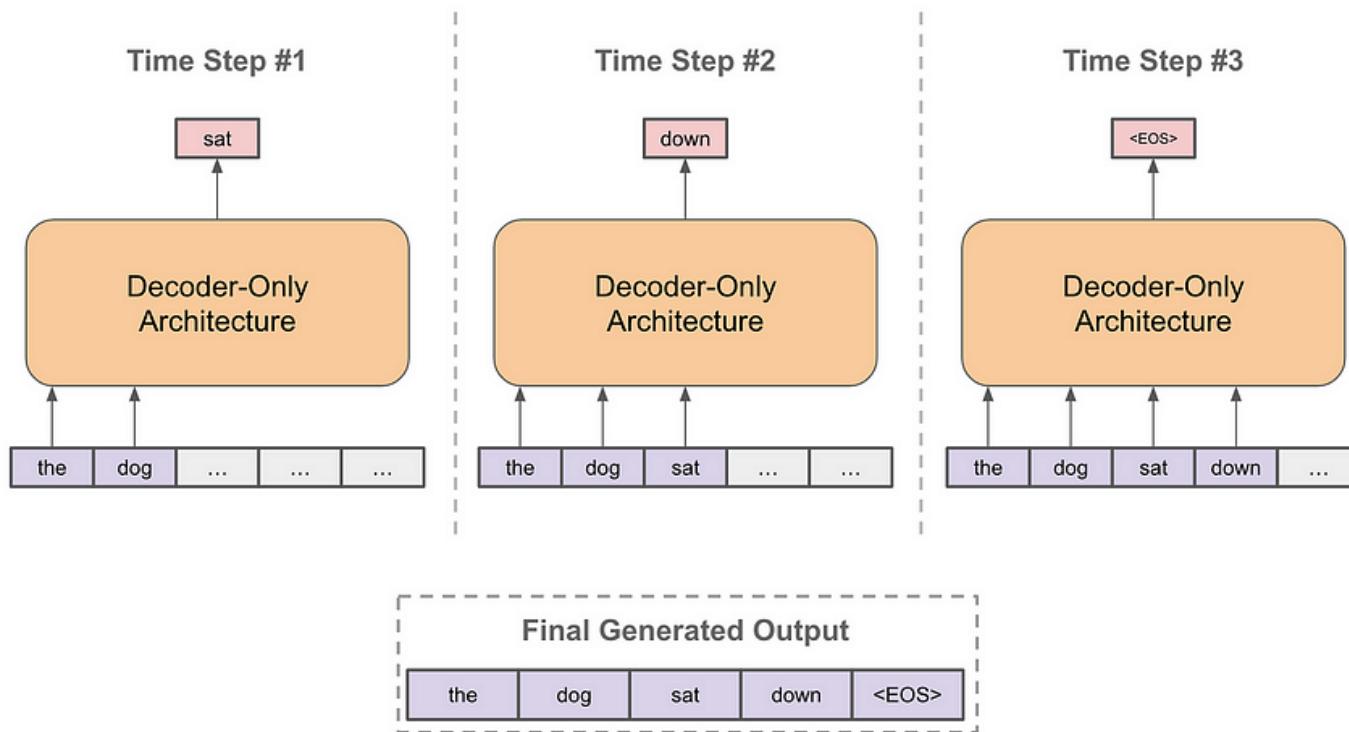


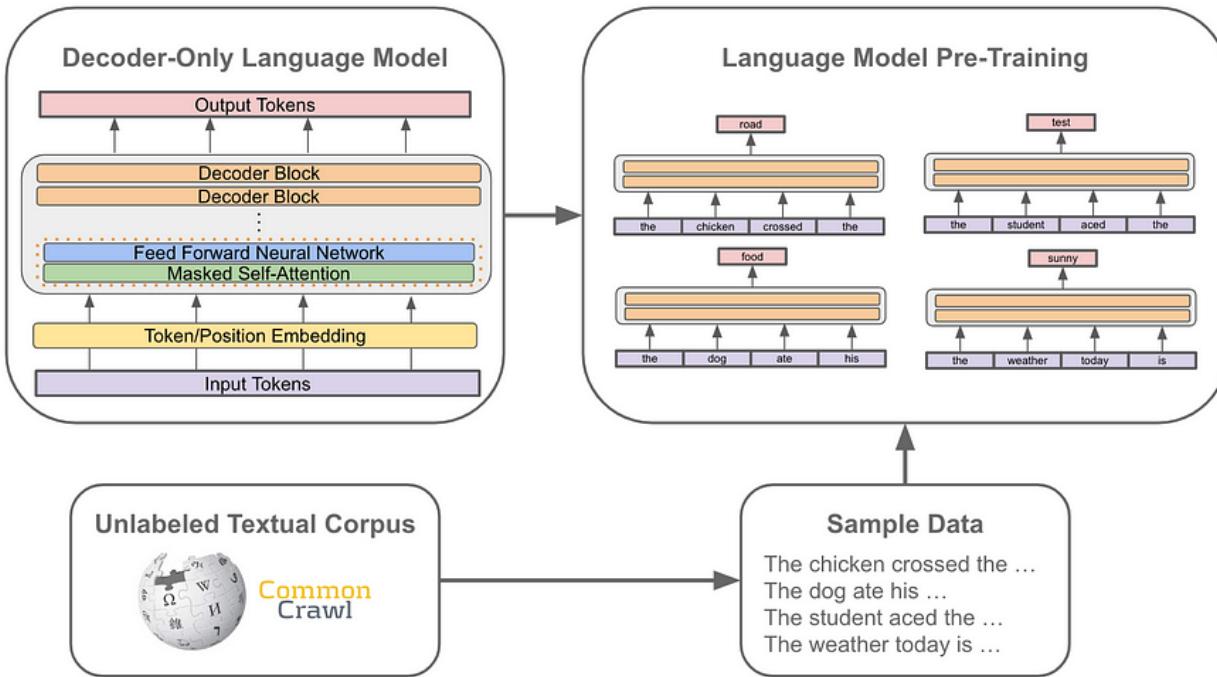
<Causal self attention : 단방향 attention>



<자기회귀적(auto-regressive) 문장 생성>

# Autoregressive Language Modeling

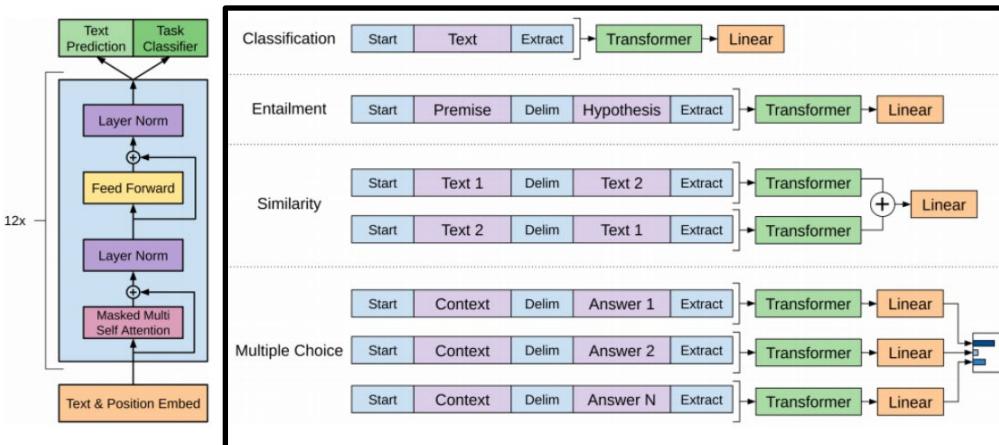




**Autoregressive** 방식의 학습은 레이블이 없는 방대한 코퍼스 데이터셋에서 잘 작동함.  
이는 데이터 부족 문제를 크게 완화시킴!

# Fine-tuning

- 자연어 이해(Natural Language Understanding, NLU), 기계 독해(Reading Comprehension), 문장 완성(Sentence Completion), 번역(Translation), 요약(Summarization)에 대해 평가
  - GPT-1은 12개 작업 중 9개 작업에서 지도학습으로 훈련된 모델보다 더 나은 성능을 보임.



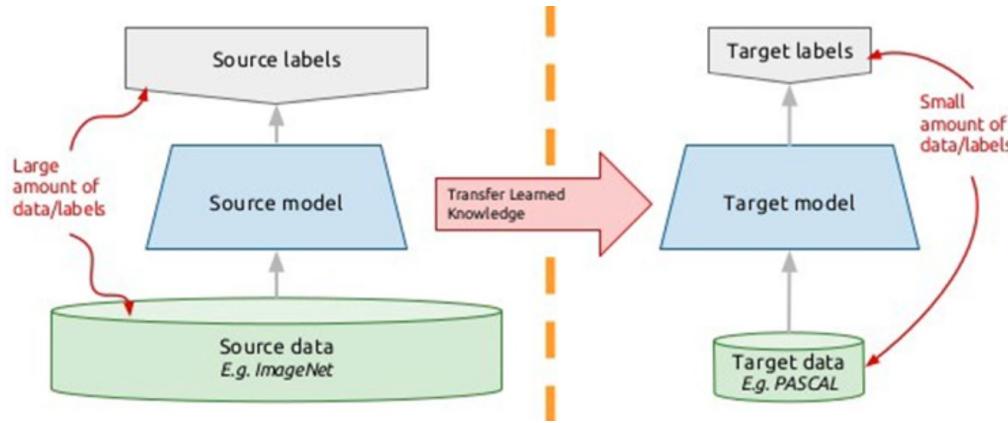
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>	-	-
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	<b>61.7</b>
Finetuned Transformer LM (ours)	<b>82.1</b>	<b>81.4</b>	<b>89.9</b>	<b>88.3</b>	<b>88.1</b>	56.0

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	<u>77.6</u>	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	<u>60.2</u>	<u>50.3</u>	<u>53.3</u>
Finetuned Transformer LM (ours)	<b>86.5</b>	<b>62.9</b>	<b>57.4</b>	<b>59.0</b>

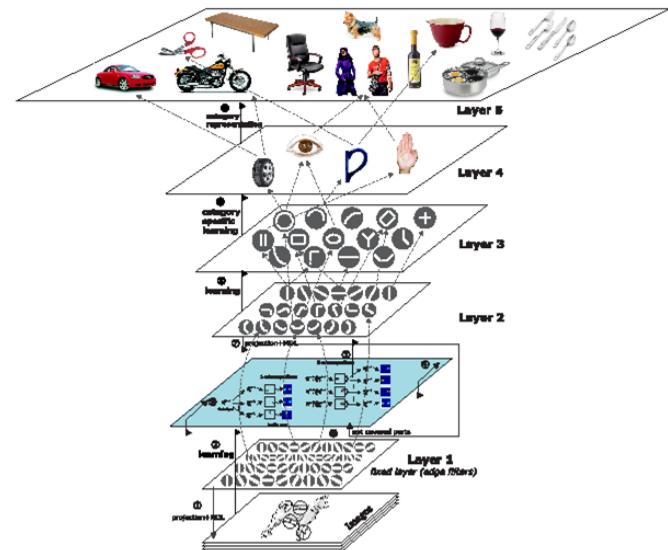
# BERT, GPT and PLM

- PLM(pretrained language model)의 시대를 연 BERT, GPT
  - 대규모 데이터의 사전학습 언어모델 pretrained language model을 다른 문제에 대해 재학습 fine-tuning이 가능해짐.
  - Pretrained model을 Downsteam task에 적용하는 것이 전이학습 transfer learning
    - ✓ CNN 기반 비전 모델들에서 이미 널리 사용되었음.



# BERT, GPT and PLM

- 비전모델의 전이학습이 이미지의 여러 층위의 특징을 학습하는 것과 유사하게 텍스트의 전이학습도 언어의 여러 층위의 특징을 전달
  - 이미지 특징 vs 텍스트 특징
    - ✓ Edge, blob, color vs Lexical(어휘적)
    - ✓ Object, texture vs Syntactic(구문적), Semantic(의미적)
    - ✓ Scene understanding vs Contextual understanding
- 지도학습을 위한 학습용 corpus 구성은 매우 비쌈
  - 방대한 unlabeled corpus를 활용한 방법을 구상
    - ✓ SSL(Self-supervised learning)으로 사전학습 모델 개발
    - ✓ Transfer learning으로 task-specific 모델 개발



이미지 특징계층의 시각화

# GPT-2 GPT의 후속 연구

- GPT-2는 모델과 데이터의 확장을 통해 성능을 개선하였음.
- 모델과 데이터를 10배 이상 증가시켜 학습
  - 48 Decoder blocks, 1600 hidden space dimension, 25 heads, in total 1.5B parameters
  - Web에서 수집한 40GByte의 텍스트
- 과제조건화(Task Conditioning) 현상
  - 언어 모델의 학습 목표는  $P(\text{출력}|\text{입력})$ 으로 표현
    - ✓ 그러나 웹크롤링된 텍스트 내에는  $P(\text{출력}|\text{입력}, \text{과제})$  학습 가능한 데이터들이 다수 존재
  - 모델에게 주어진 입력에 대해 과제에 따라 여러 출력을 학습하는 meta learning이 가능
  - 새로운 개념을 이해하고 활용하는 **Few-shot Learning 능력**을 보임
  - 처음 수행하는 새로운 작업을 이해하고 해결하는 **Few-shot task transfer 능력**을 보임

# GPT-2

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	IBW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	<b>21.8</b>
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50	75.20
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37	55.72
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05	44.575
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>	42.16

- 요약, 번역, Q&A 같은 일반적인 task에서 SOTA를 달성 못 했지만 지도학습 없이 높은 성능을 낸 것에 매우 큰 의미가 있음.
- 언어모델의 Meta learning에 대한 새로운 가능성
  - 더 발전시키기 위한 방법론은?

# New Normal for overfitting

- Devansh Arpit(2017)은 적정 모델의 복잡도의 결정은 데이터의 수량 뿐만 아니라 패턴의 난이도로 연관되어 있음을 주장.
  - 뉴럴넷은 학습 초기에 파악하기 쉬운 패턴을 학습하고 진행될수록 어려운 패턴을 학습함.
    - ✓ 따라서 마지막에 데이터를 암기하는 것처럼 보이는 것.
  - 모델의 용량이 높을 수록 노이즈 샘플에 의한 비정상적인 학습에 오히려 강건해짐.



Devansh Arpit  
(Université de Montréal)

제가 자세히 들어보니 시간에 따라 쉬운 패턴에서 복잡한 패턴을 학습합니다.  
그래서 모델의 적정용량은 데이터  
수량이 아니라 패턴의 복잡도에 따라  
정해야 될 듯 합니다.

근데 재밌는게 모델의 용량이 높으면  
노이즈를 섞어 학습해도 헛깔리지 않고  
정상적으로 잘 추론하는 능력이  
좋아지더라고요.

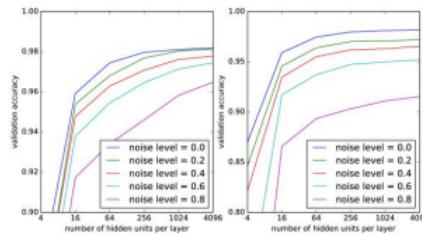


Figure 5. Performance as a function of capacity in 2-layer MLPs trained on (noisy versions of) MNIST. For real data, performance is already very close to maximal with 4096 hidden units, but when there is noise in the dataset, higher capacity is needed.

# New Normal for overfitting

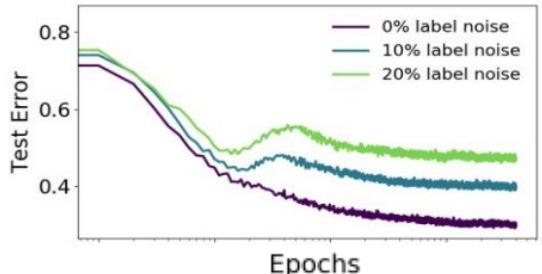
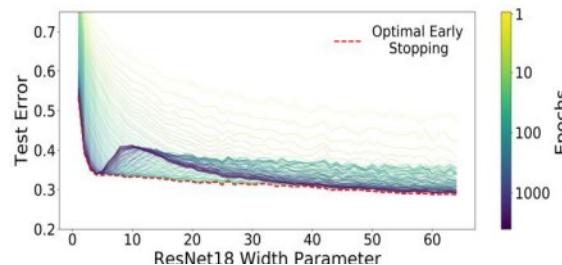
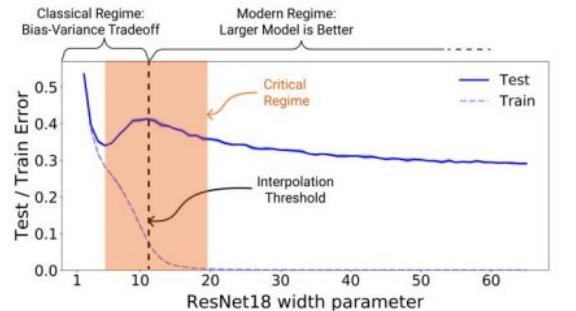
- Preetum Nakkiran(2019)은 double descent 현상을 발견함.
  - 모델의 복잡도를 계속 올리다보면 validation error가 다시 감소하는 구간이 존재함.
  - 시간(Epoch)에서도 학습을 오래하면 다시 성능이 개선되는 double descent 구간이 존재.



validation error가 감소하다가 갑자기  
증가하면 오버피팅이라고 했잖아요. 근데  
모델 사이즈랑 학습시간이 커질수록 두 번째  
감소구간(double descent)이 있더라고요?

오히려 모델을 더 복잡하게 해서 더 오래  
학습하면 성능이 계속 증가할 수도 있지  
않을까요?

Preetum Nakkiran(Harvard univ.)



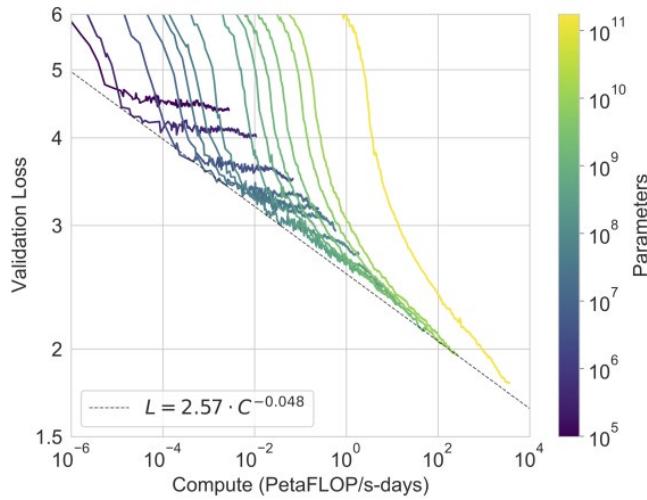
# New Normal for overfitting

- Tom B. Brown(2020)의 **Scaling Laws** : GPT-3 모델의 few-shot learner 연구를 진행하면서 파라미터가 많은 수록 더 높은 성능과 연산량 대비 더 빠른 학습 속도를 보임.



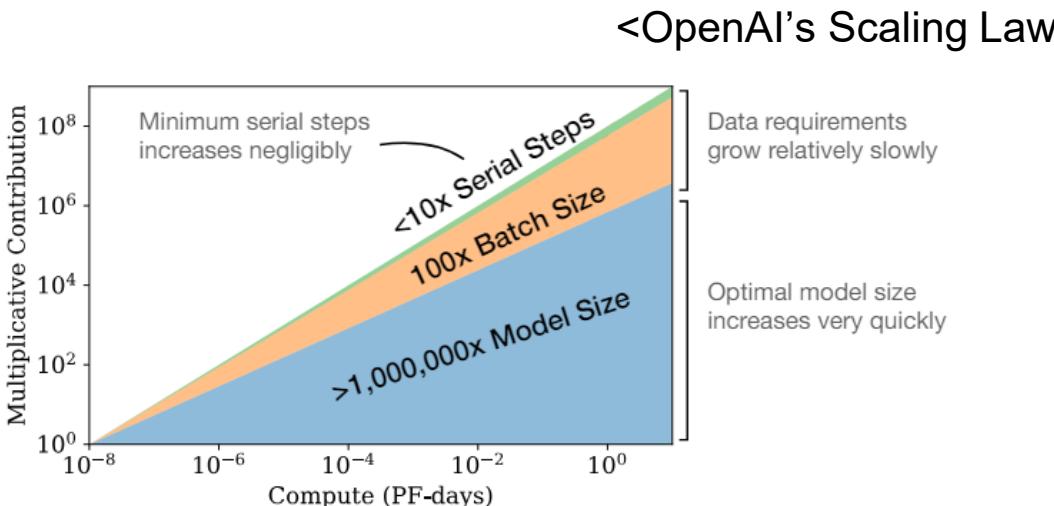
Tom B. Brown(OpenAI)

MS에서 투자 받은 돈으로 모델 파라미터를 몇십 억개가 아니라 한 1750억 개로 GPT 만들어서 테스트해보니까 모델이 복잡하면 복잡할수록 더 좋을뿐만 아니라 학습 속도도 더 빠른데요? 그러니까 모델은 크면 클수록 좋다는 결론!



# Scaling law drives LLM era

- 2020년, OpenAI은 GPT 기반 대규모 언어모델을 개발하며 성능에 모델 크기와 데이터 규모가 가장 직접적인 영향을 준다는 Scaling law를 발표
  - 모델 성능에 있어 특히 모델 규모를 키우는 것이 **가장 효율적임**.
  - 생성, 번역, 추론 등 다양한 성능을 개선하기 위해 거대언어모델의 시대가 시작

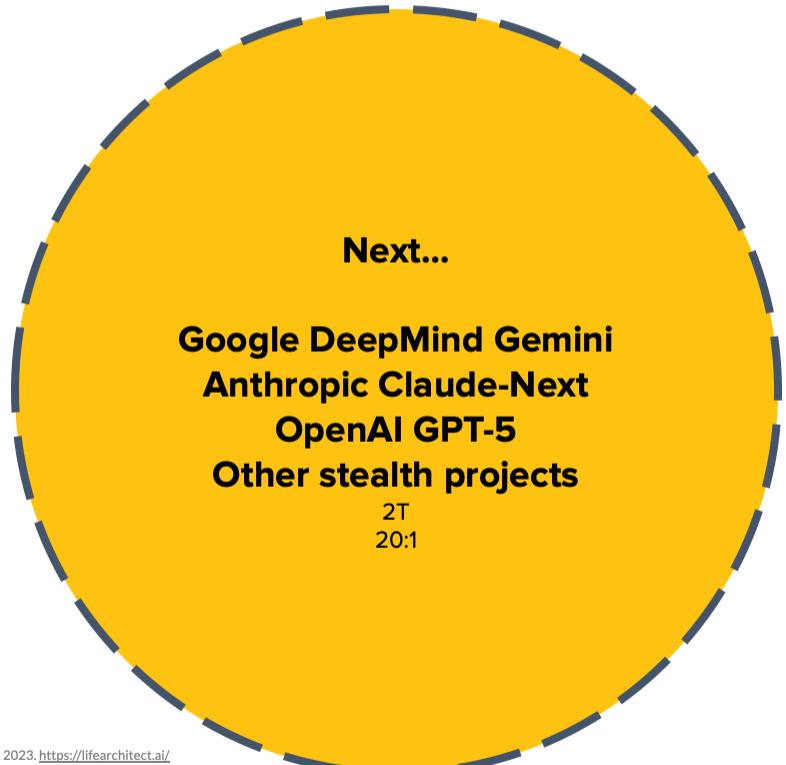
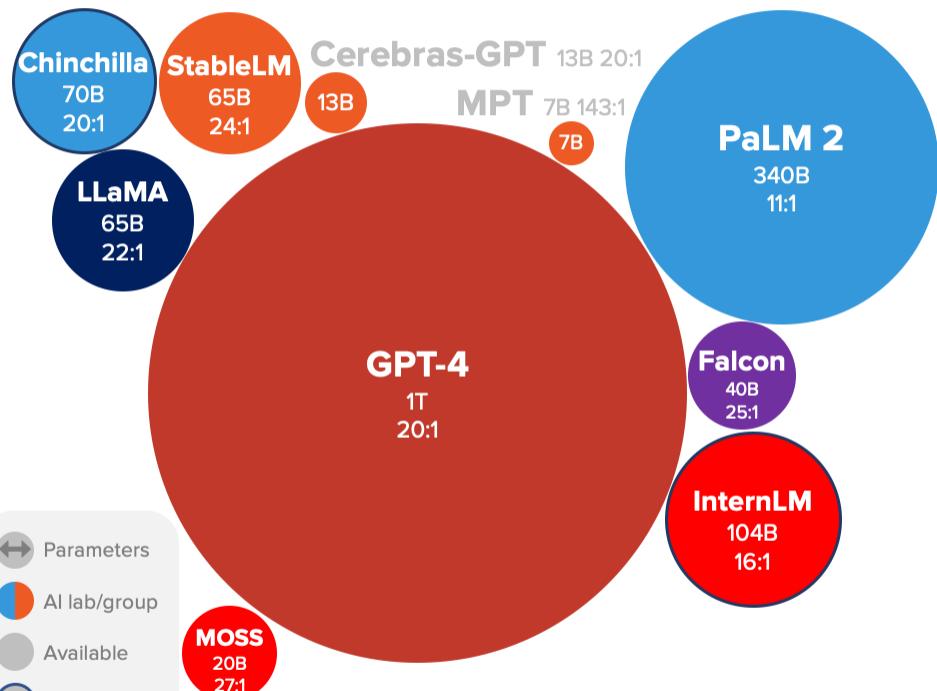


$$L(N, D) = \left[ \left( \frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

- L : cross entropy loss
- N : 모델 매개변수 크기
- D : 데이터셋의 token 수
- $N_c, D_c$ 는 상수,
- $\alpha_D = 0.095, \alpha_N = 0.076$

# 2023-2024 OPTIMAL LANGUAGE MODELS

JUN/  
2023



Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. \*Chinchilla scale means tokens:parameters ratio  $\geq 11:1$ . <https://lifearchitect.ai/chinchilla/> Alan D. Thompson. June 2023. <https://lifearchitect.ai/>



LifeArchitect.ai/models

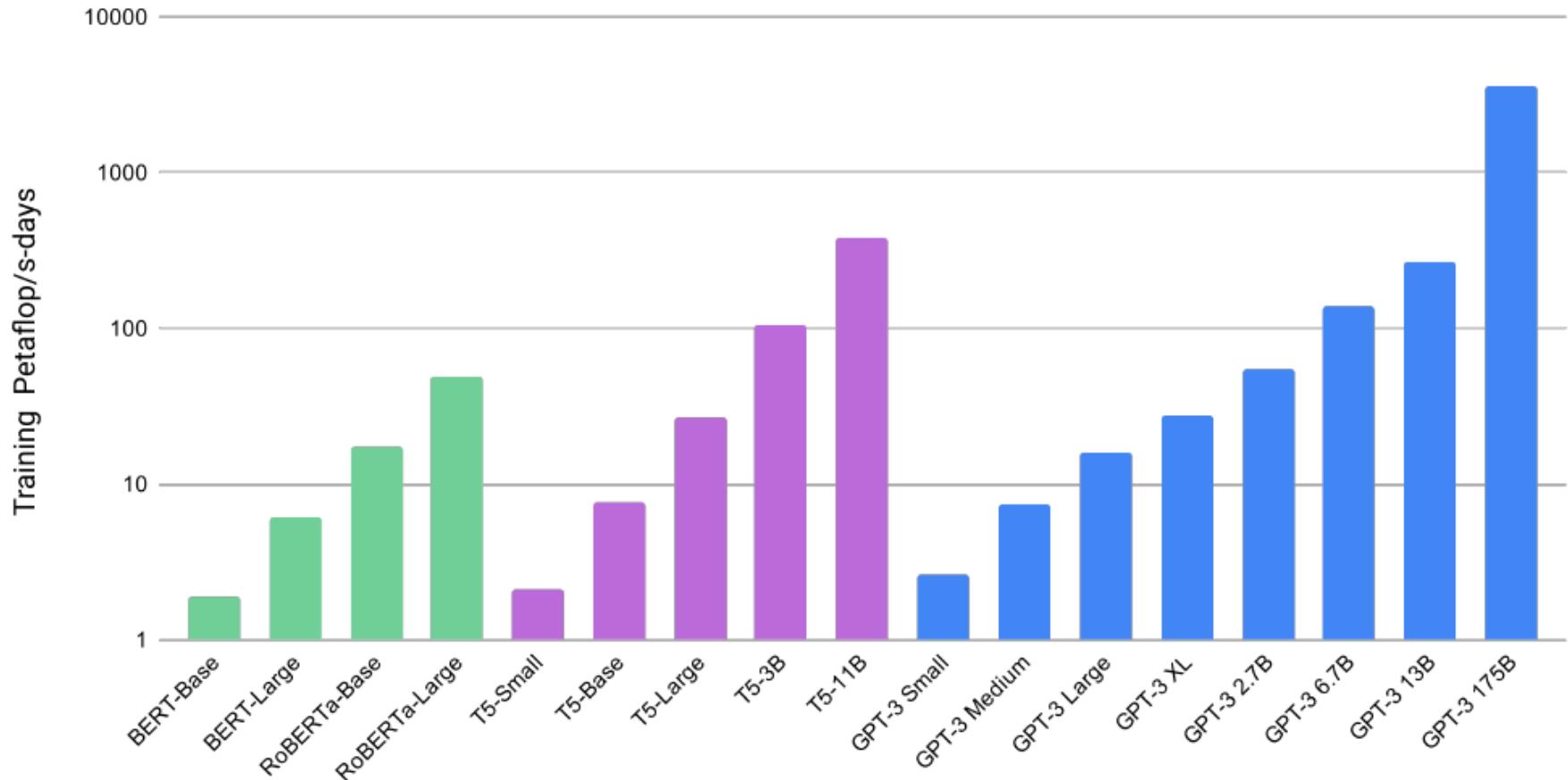
# GPT-3

- 이전 모델인 GPT-2보다 100배 더 크고 1750억 개의 매개변수 사용해 GPT-3 모델링
  - 규모 외 세부 구조는 GPT-2와 크게 변경되지 않음.
- 대규모 학습을 위해 5000억 단어의 'Common Crawl' 데이터 컬렉션 학습

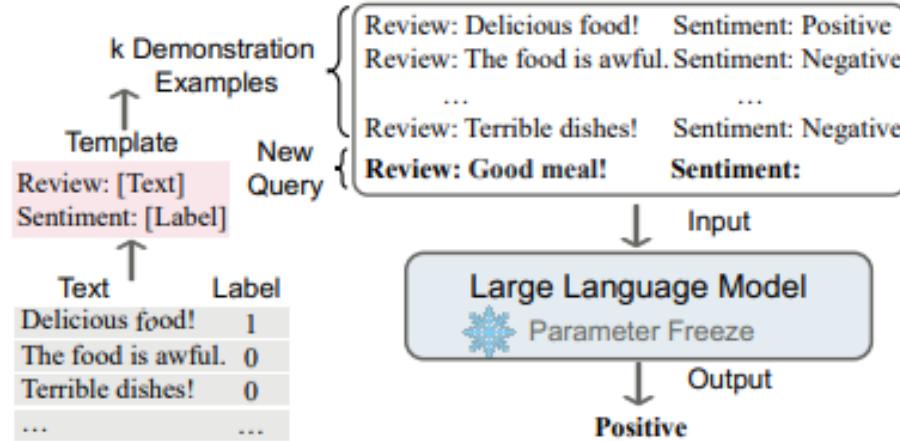
Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

- GPT-3은 학습 결과로 고도로 발전된 비지도학습 성능 관찰
  - 인간이 작성한 것처럼 보이는 정교한 문장과 문단을 생성 가능
  - SQL 쿼리와 같은 코드 스니펫을 작성하는 추가적인 지능적인 작업을 수행 가능
  - GPT-3는 더이상 세밀한 조정 없이도 다양한 지능적인 작업을 수행 가능

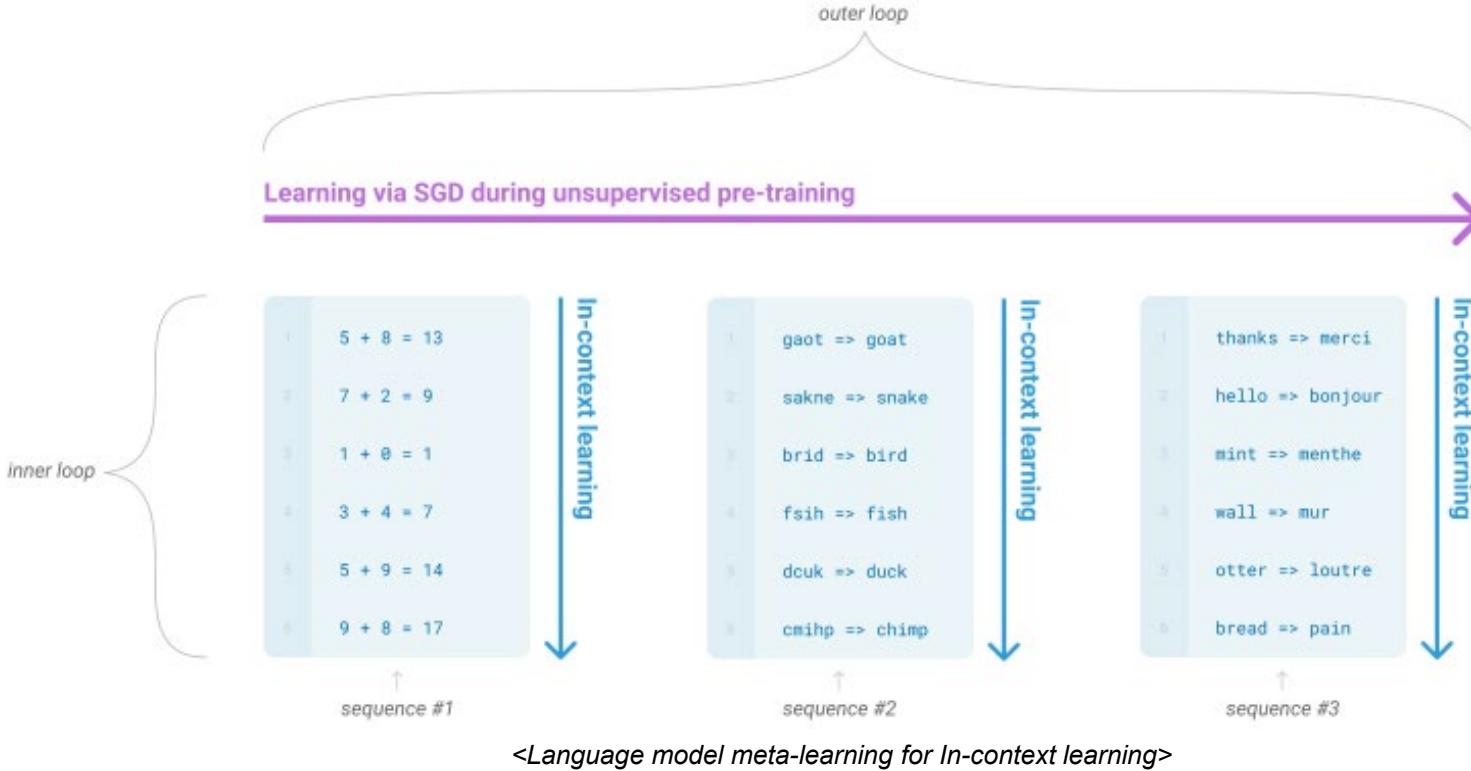
## Total Compute Used During Training



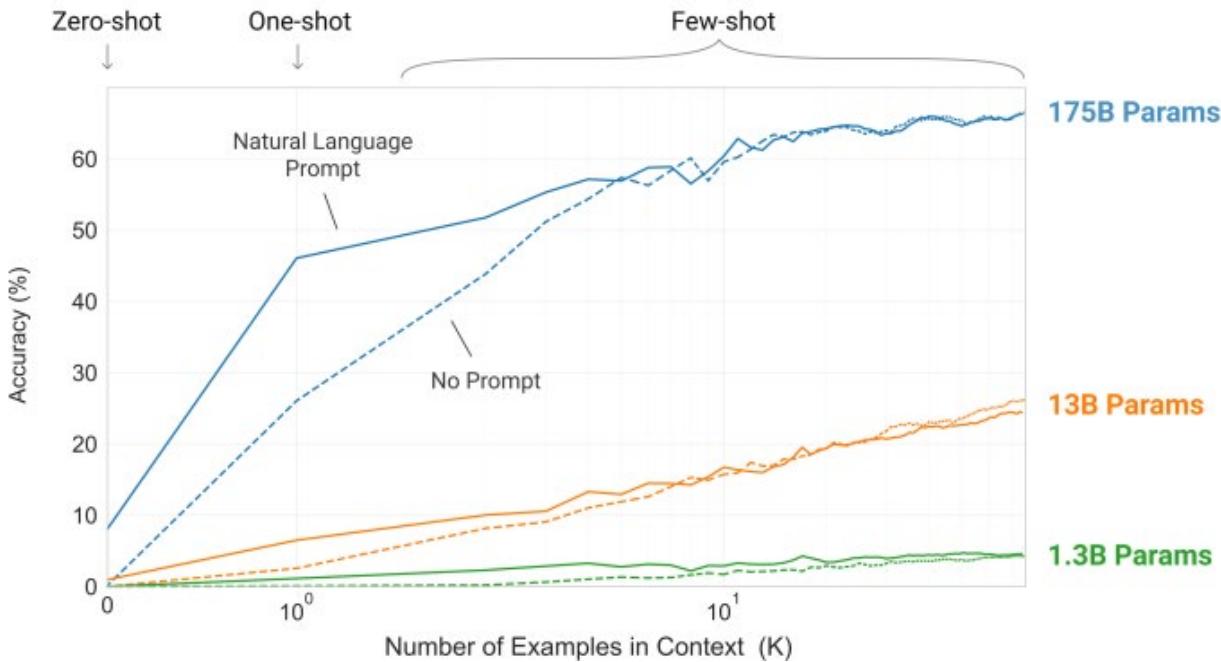
# In-context approach



- ICL은 질의에 도움이 될 몇 가지 학습 샘플을 함께 제공하는 방식임.
  - ICL Prompt : Task description + query text + few-shot examples+
  - 예시 샘플로 데이터를 텍스트 template 형태로 작성(상단 그림 참고)
- Prompt engineering의 근간이 되는 아이디어

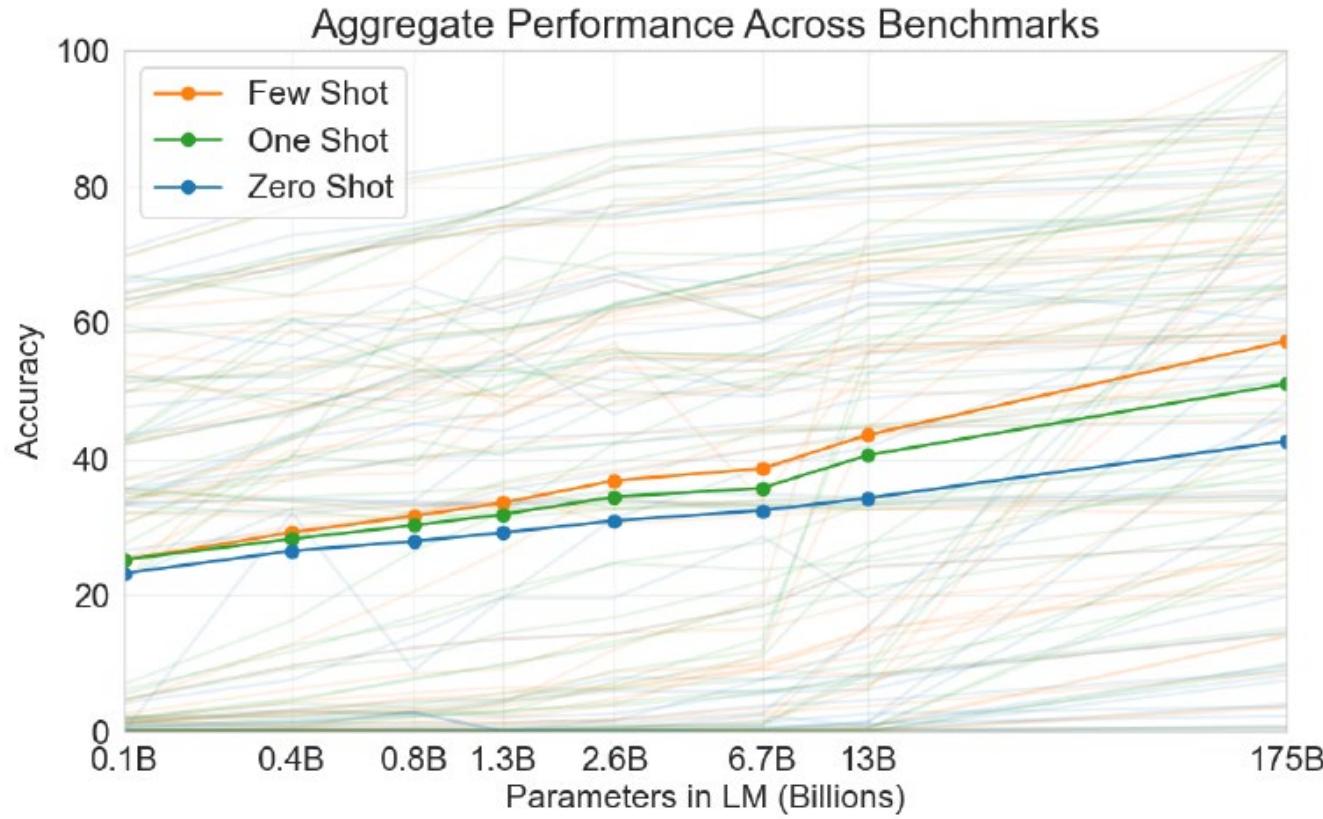


- 사전학습 중 방대한 Task 레벨의 학습 경험은 GPT가 Task와 샘플을 기반으로 문제를 해결하는 능력 부여



*<Larger models make increasingly efficient use of in-context information.>*

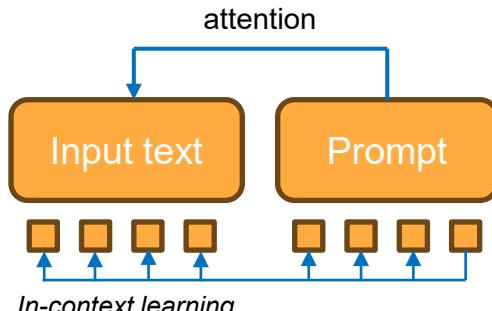
- Zero-shot : 모델은 특정 작업에 대해 명시적으로 훈련 받지 않았더라도, 학습 과정에서 습득한 지식을 다른 작업에 적용
- One-shot, Few-shot : 주어진 예제들을 통해 모델이 새로운 작업을 이해하고 수행하는 방식
- No Prompt는 Task description 없이도 예제를 통한 맥락 추론을 통해 해결하는 능력



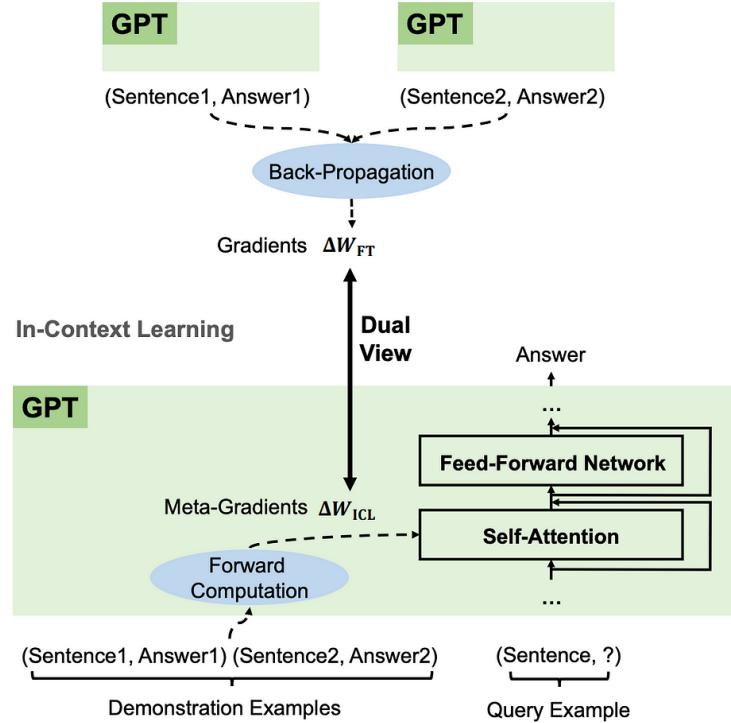
- Zero-shot 성능은 모델 크기에 따라 꾸준히 향상되지만, few-shot 성능은 더 빠르게 증가하여 더 큰 모델이 상황 내 학습에 더 능숙함을 보임.

# Summary

- 대규모 사전학습 LLM은 언어 이해 및 생성 뿐만 아니라 In-context learning과 추론 능력까지 향상
- CNN, RNN, FCN 등과 달리 Transformer는 입력 데이터에 대한 동적 가중치 계산이 가능
  - 방대한 데이터에 의한 meta learning과 함께 동적 가중치 계산이 In-context learning을 설명한다는 연구 존재.



Dai, Damai, et al. "Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers." arXiv preprint arXiv:2212.10559 (2022).



# Discussion

- 거대 인공지능의 시대
  - 모델 구조 또는 알고리즘 개선이 아닌 모델 스케일업에 따른 성능 개선
  - 탄소 배출과 에너지 사용에 따른 환경 파괴
  - 딥러닝 커뮤니티에 의한 투명한 AI 발전의 종말?
    - ✓ AI 기술에 대한 폐쇄성의 심화
- 딥러닝 킬러앱의 탄생
  - AI 기술 활용의 대중화
  - 개발 진입장벽 하향에 의한 AI응용 개발자의 시간
- 인공지능 윤리문제
  - 웹크롤링 데이터의 개인정보 이슈
    - ✓ 현실적으로 학습 데이터 내 민감 정보 제거는 어려움.
  - 학습에 의해 저작권 침해를 인정할 것인가?

# Prompt engineering

# Foundation 모델의 시대

- Foundation 모델은 기계학습(ML) 분야에서 대규모 사전 학습된 모델들이 다양한 다운스트림 작업에 범용적으로 적용될 수 있는 기술적 발전을 의미
  - 광범위한 데이터로부터 언어, 이미지, 오디오 등의 복잡한 패턴과 지식을 학습하여, 이를 다양한 특정 작업에 적용할 수 있는 능력을 갖추고 있음.
    - ✓ GPT, BERT,
    - ✓ Vision Transformers(ViT)
- Foundation 모델은 범용성, 데이터 효율성, 성능 향상 측면에서 엄청난 기회임과 동시에, 리소스 집약적, 오버피팅의 위험, 대규모 망각 문제에 대응해야 함.
  - Parameter-Efficient Fine-Tuning(PEFT) 기법은 모델의 매개변수 일부만을 조정
  - 리소스 사용을 최소화하고 모델의 범용성을 유지하며 오버피팅과 대규모 망각의 위험을 최소화

# The Rise of Prompt Engineering: Insights from In-Context Learning

- In-Context Learning은 입력으로 추가적인 정보가 주어졌을 때 언어모델이 학습을 통해 더 나은 답변을 하는 능력.
  - 모델이 주어진 몇 개의 예제를 "학습 컨텍스트"로 사용해 올바른 응답을 생성하도록 배움.



## The three settings we explore for in-context learning

### Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



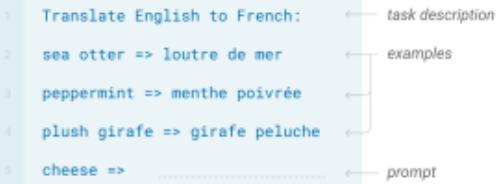
### One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



### Few-shot

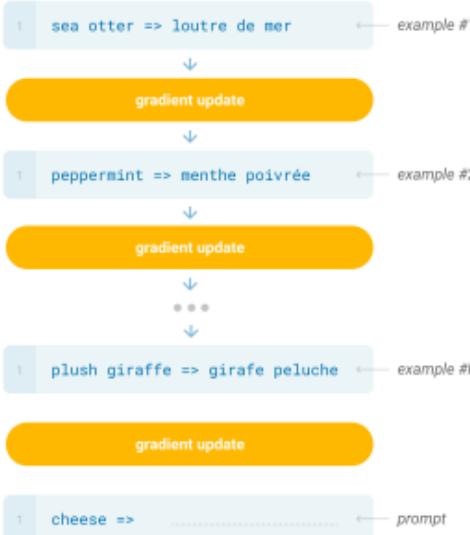
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



## Traditional fine-tuning (not used for GPT-3)

### Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

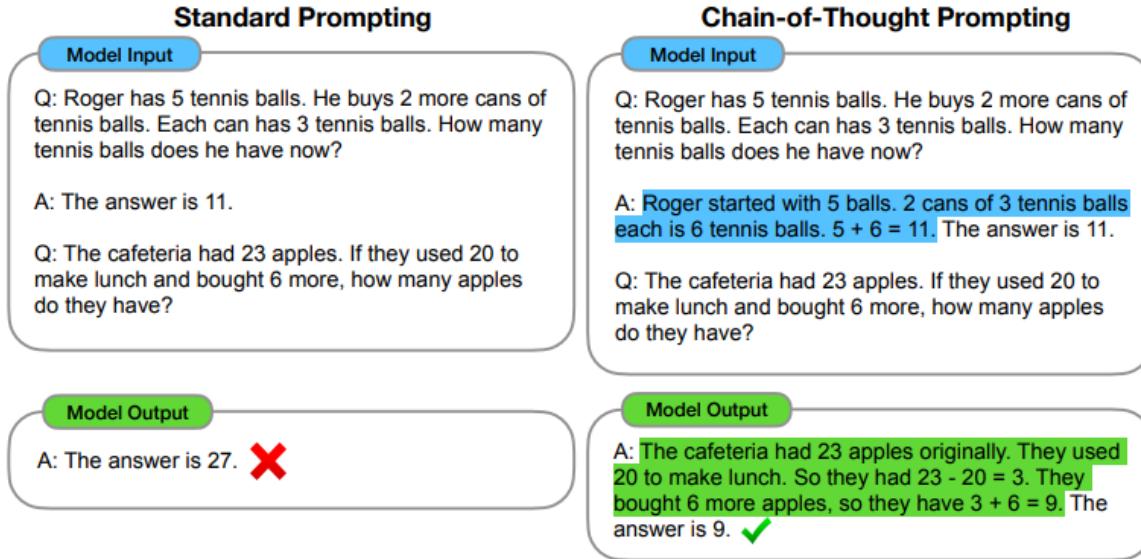


## • Zero/Few-shot In-context learning

- **Task**에 맞는 올바른 응답을 내도록 입력 예시로 모델이 학습하는 방식
- Meta learning으로 일반화된 GPT-3는 실제로 zero-shot 수준에서도 뛰어난 성능을 나타냄
- 모델의 가중치를 직접 조정하는 Fine-tuning과 달리, 트랜스포머 내 hidden state에 좋은 컨텍스트를 형성해 성능을 개선

## Zero/Few-shot learning vs. Fine-tuning

# Chain-of-thought(COT)



- LLM이 복잡한 문제를 해결하기 위해 주어진 문제를 단계별로 분해하고, 추론을 연쇄적으로 이어가며 최종 결론에 도달하도록 하는 prompt-based approach

### (a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. **X**

### (b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. **✓**

### (c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

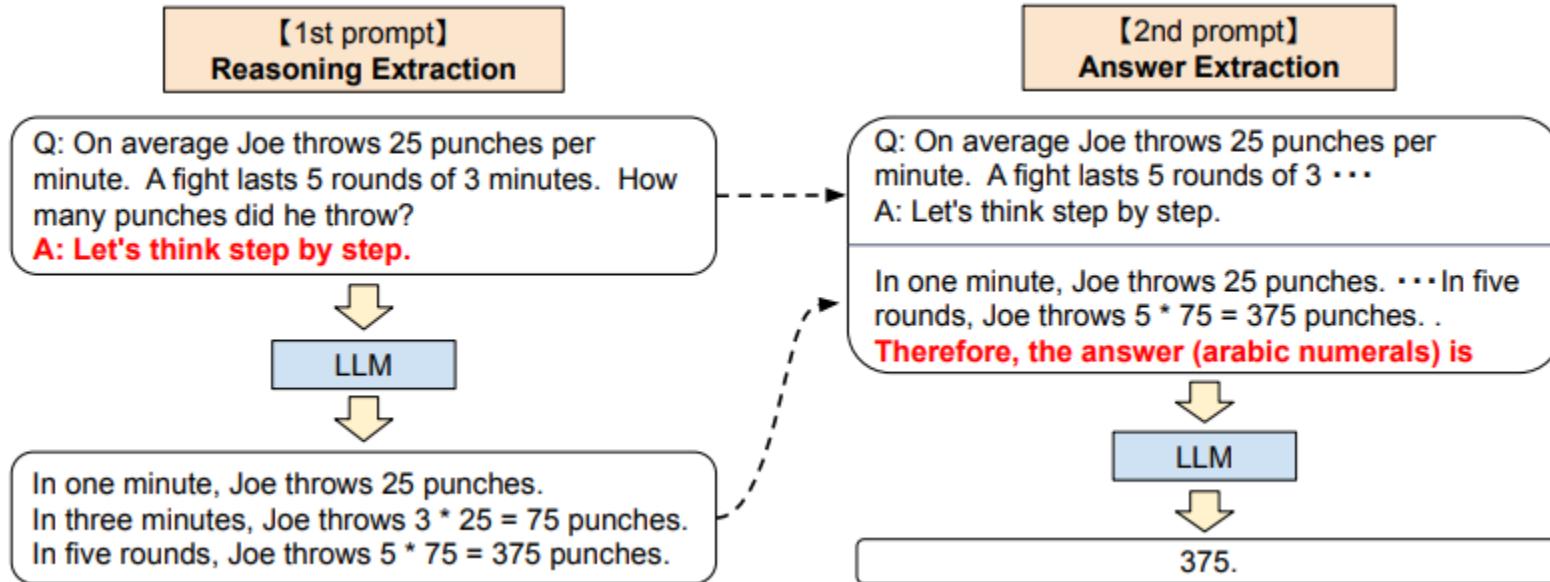
(Output) 8 **X**

### (d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

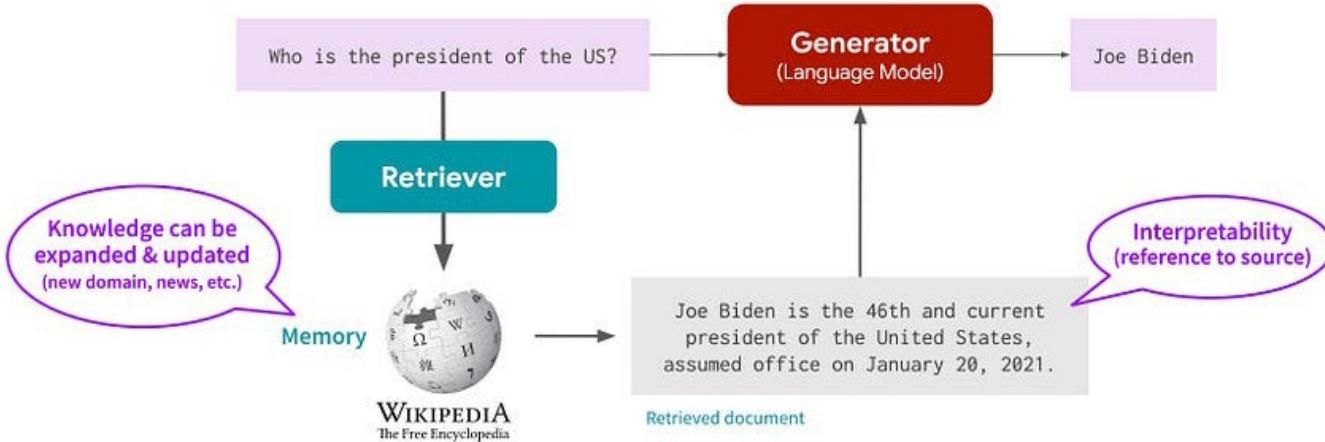
A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. **✓**

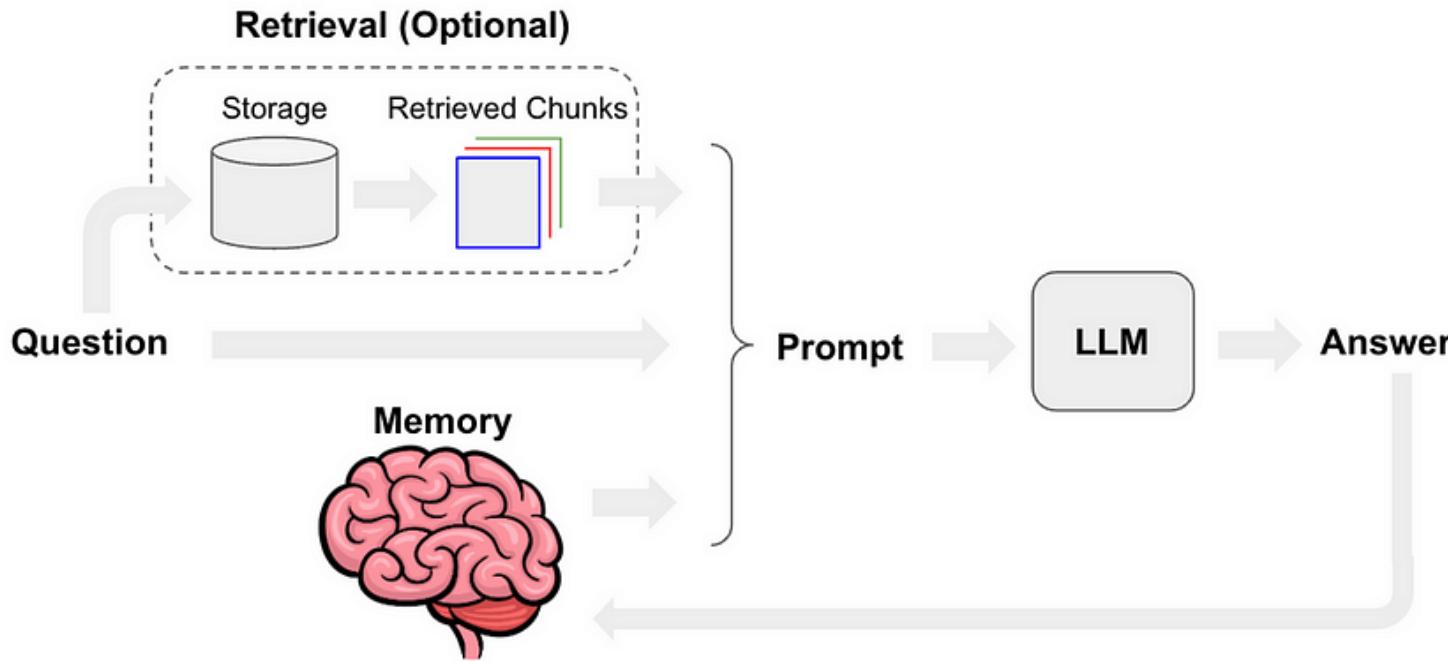


- Zero-shot COT는 두 단계로 구성
  - 첫 번째는 언어 모델이 전체 추론 경로를 추출하는 “추론 프롬프트 추출”
  - 두 번째는 추론 텍스트에서 올바른 형식의 답변을 추출하는 “답변 프롬프트 추출”

# RAG(Retrieval-Augmented Generation)

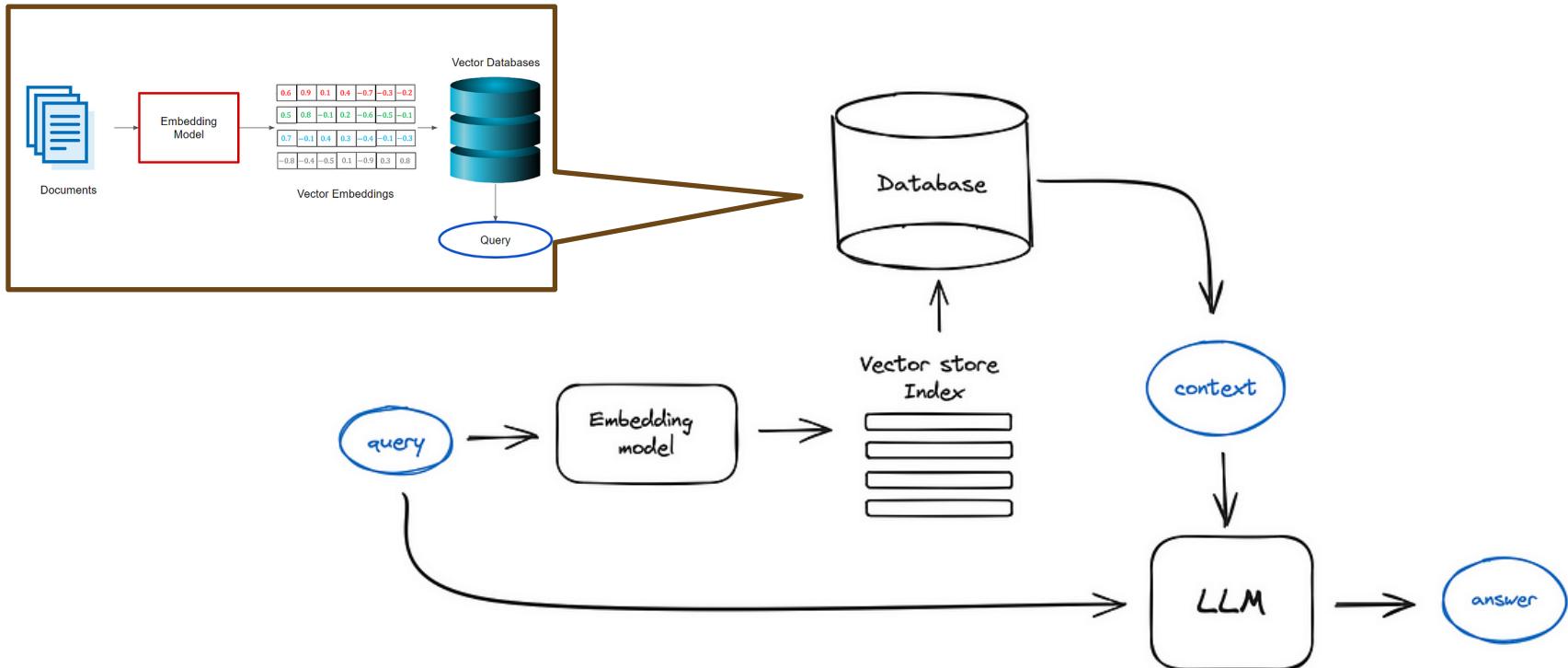


- RAG(Retrieval-Augmented Generation)은 색 기반의 정보 증강과 생성 모델을 결합한 방식
  - 검색과 생성의 결합: RAG는 주어진 질문이나 프롬프트에 대응하는 정보를 데이터셋에서 검색
  - 지식 기반 확장: 검색을 통해 모델은 학습 데이터에 포함되지 않은 최신 또는 특정한 지식을 활용
  - 생성 정확도 향상: 검색을 통해 얻은 구체적인 정보를 사용하여 생성된 답변의 정확성을 높임

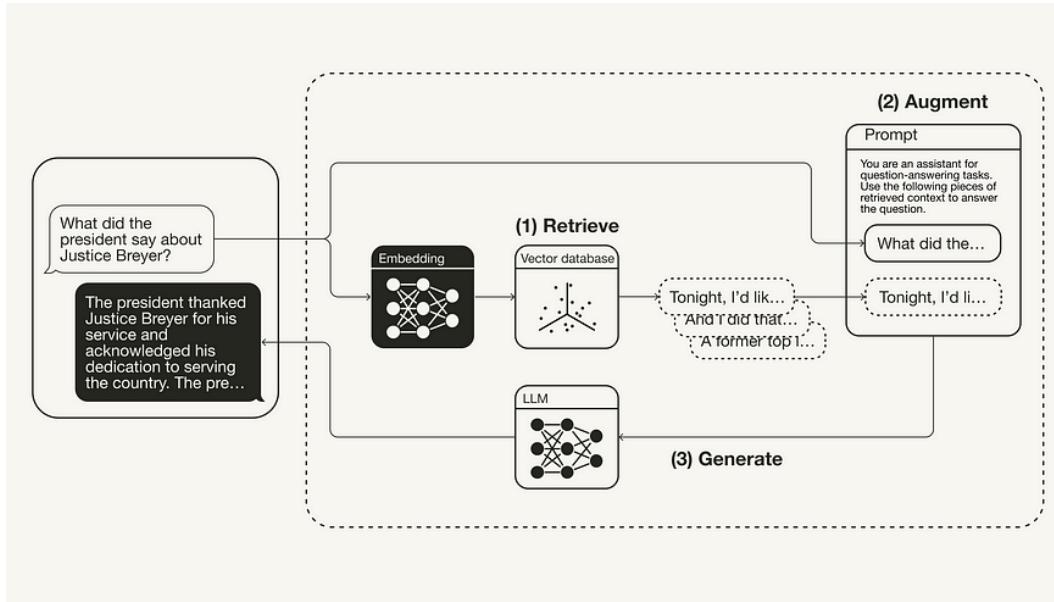


- RAG는 외부 데이터를 통합해, 질문에 대한 컨텍스트 이해를 통해 LLM의 지식 한계를 해결

# Efficient indexing of documents for quick retrieval



# RAG(Retrieval-Augmented Generation)



- (Augment) 프롬프트를 추가적인 외부 텍스트로 증강시킬 수 있음
  - 프롬프트 템플릿을 준비해서, 프롬프트와 문서를 합칠 수 있음
  - 또는, LLM 모델을 통해 프롬프트와 외부 텍스트를 자연스럽게 병합할 수 있음(모델 학습 필요)
- (Generate) 마지막으로 프롬프트 템플릿을 LLM에 입력해 답변을 생성

# **LLM Fine-tuning**

# Instruction tuning

- Pretrained model의 단순한 다음 토큰 예측 모델을 답변생성기 전환 (feat. Chatgpt)

Completion (Base model without fine-tuning):

User Prompt:

A large language model is a type of artificial intelligence model that

LLM Response:

is trained on a huge amount of text data and can generate human-like language in response to queries.

Instruction-Tuned Model:

User Prompt:

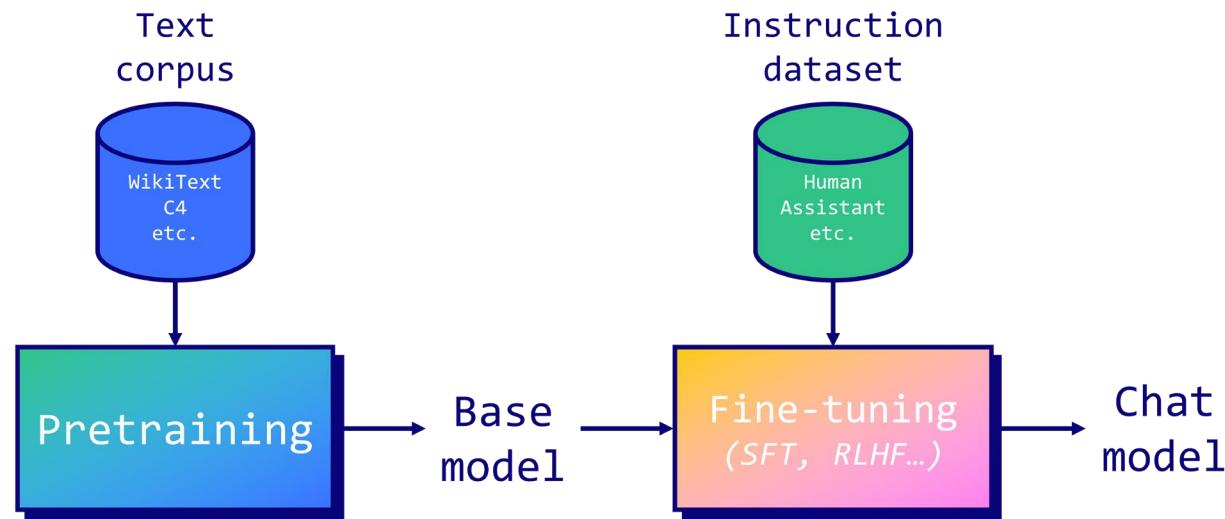
What is a large language model?

LLM Response:

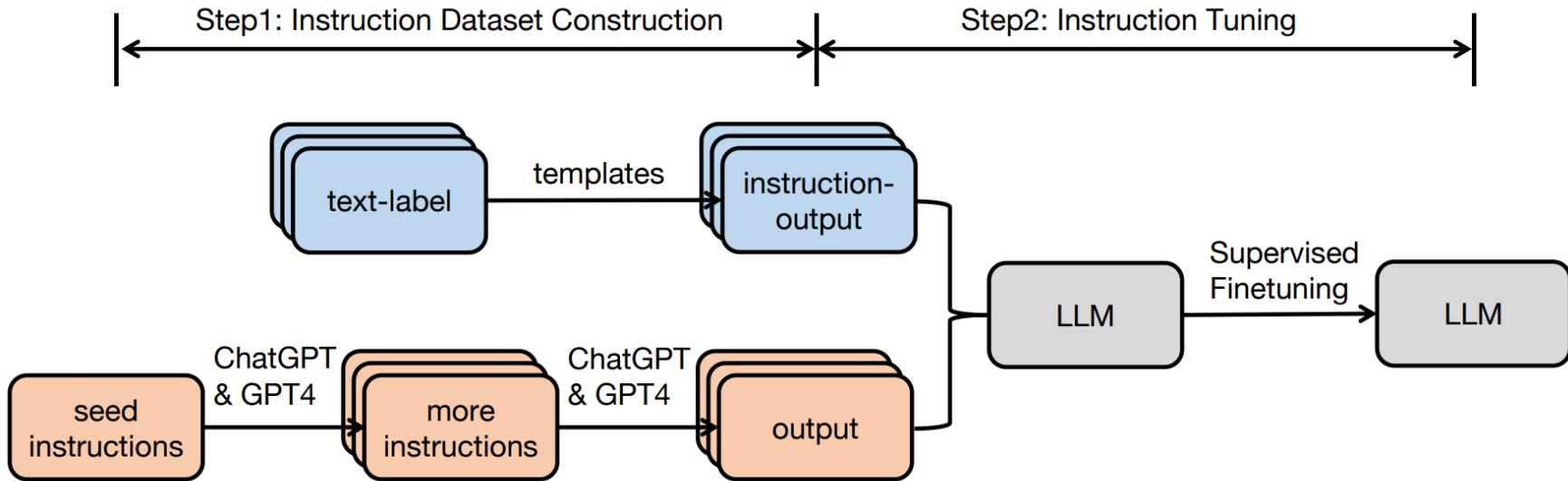
A large language model is a type of artificial intelligence model that is trained on a huge amount of text data and can generate human-like language in response to queries.

<Instruction tuning은 모델이 질문과 지시에 적절한 형식으로 응답>

- Instruction Tuning은 모델이 다양한 작업에 대해 자연어 지시를 수행하도록 학습
  - 데이터 규모가 작거나, 연산자원에 따라 SFT + LoRA와 같은 PEFT를 활용할 수도 있음.
  - 모델의 편향성, 위험성, 답변품질을 향상하기 위해 RLHF 수행할 수도 있음



# Instruction Tuning workflow



# RLHF : Reinforcement Learning from Human Feedback

Step 1

Collect demonstration data  
and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



We give treats and punishments to teach...



This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2

Collect comparison data and train a reward model.

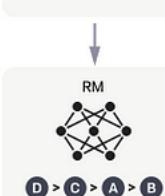
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



D > C > A > B

Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



Write a story about otters.



The PPO model is initialized from the supervised policy.



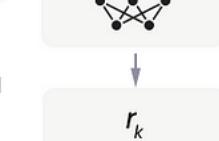
The policy generates an output.



The reward model calculates a reward for the output.

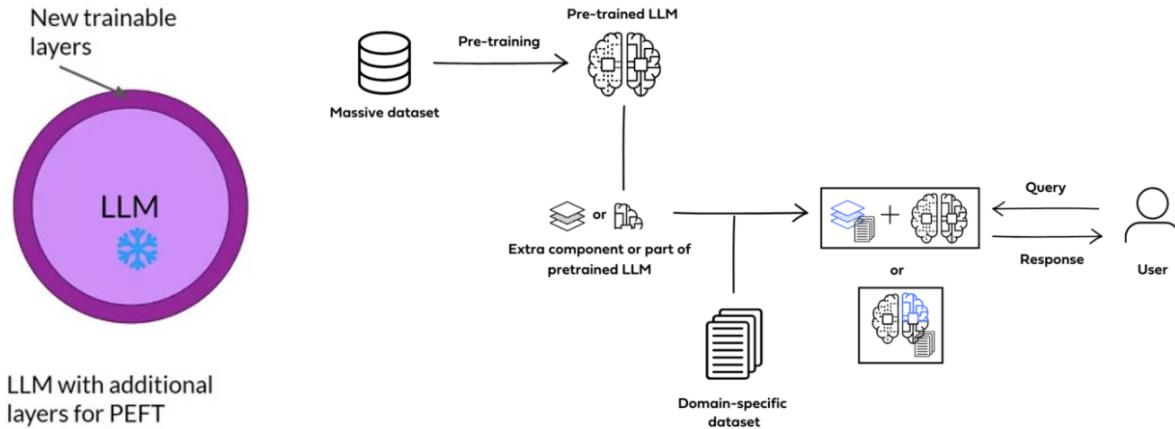


The reward is used to update the policy using PPO.



# PEFT : Parameter-Efficient Fine-Tuning

- Parameter-Efficient Fine-Tuning (PEFT) 방법론은 대규모 Foundation 모델을 더 적은 매개변수 변경으로 특정 작업에 효과적으로 적용할 수 있게 하는 다양한 전략
  - Prompt-Based Methods: 모델 입력에 특정 프롬프트나 작은 조정 가능한 매개변수 집합을 추가하여, 기존 모델이 특정 작업을 더 잘 수행할 수 있도록 유도
  - Adapter Modules: 모델의 기존 아키텍처 사이에 작은 신경망 모듈(Adapters)을 삽입하여, 전체 모델을 재학습하는 대신 이러한 모듈만을 특정 작업에 맞게 학습.



# LoRA: Low-Rank Adaptation of Large Language Models

- 핵심 아이디어는 모델을 동결한 채, 학습 정보를 분리 가능한 어댑터에 저장
- 어댑터는 모델 파라미터 대신 업데이트되는 학습 가능한 저차원 연산 모듈
  - 학습할 행렬을 두 개의 low-rank matrix로 분리
  - 두 matrix는 곱하여 기존 행렬과 동일 크기 갖음.

Finetuned Weights

$$\overbrace{W_{\text{ft}}}^{\text{Finetuned Weights}} = \underbrace{W_{\text{pt}}}_{\text{Pretrained Weights}} + \overbrace{\Delta W}^{\text{Weight Update}}$$

Pretrained Weights

Weight Update

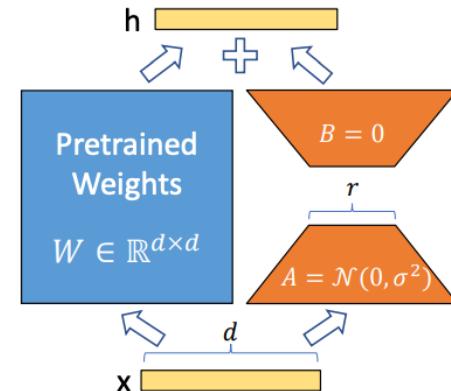
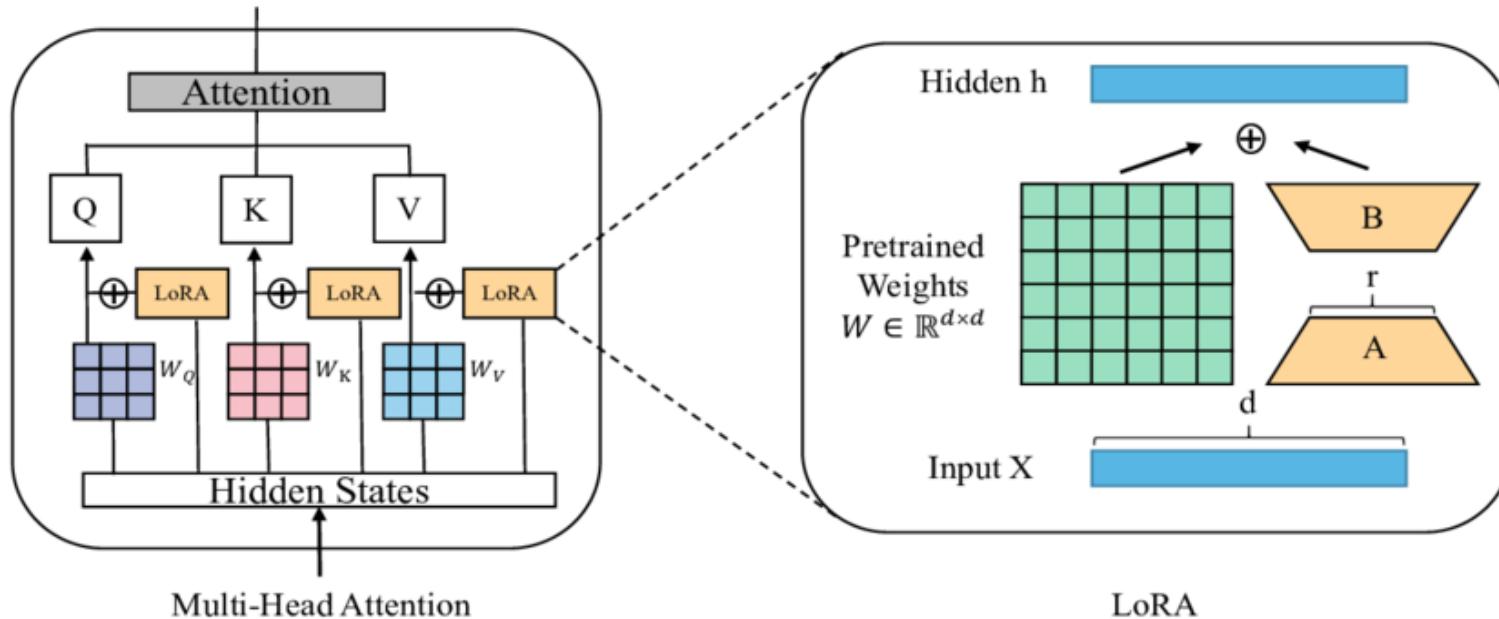


Figure 1: Our reparametrization. We only train  $A$  and  $B$ .

# LoRA in Transformers



# Small Language Model

# SLM(Small Language Models)

- LLM이 자연어 분야에서 혁신적인 성공을 이어가는 한편 저용량, 저전력 조건을 달성하기 위한 SLM에 관련한 새로운 요구가 발생함.
- SLM은 LLM(Large Language Model)의 작은 버전으로, 일반적으로 수백만에서 수십억 개의 매개변수를 가짐.



# SLM or sLLM

- 다양한 문서에서 SLM과 sLLM로 혼용되고 있음.
  - 국내의 경우 sLLM으로 호칭되는 경우가 많음.
- SLM은 종종 sLLM으로 호칭되며, 둘 다 엄밀한 학술 용어는 아님.
  - SLM은 Small Language Model의 약자
  - sLLM은 Smaller Large Language Model의 약자.
- sLLM은 사실 ‘작은 언어모델’ 보다는 ‘경량화 언어모델’이라는 번역이 더 적합함.
  - sLLM은 ‘작기도 하고 거대하기도 한 언어모델’이 아니라 ‘경량화된 대형언어모델’이라고 이해 가능
  - 즉 **SLM 또는 sLLM은 처음부터 작게 만든 것이 아니라 LLM을 경량화 과정을 통해 작게 만든 것.**
    - ✓ 모델 압축(Model Compression)은 LLM을 SLM으로 경량화하는 과정



# Introduction to Model Compression

- Model Compression은 LLMs(e.g BERT, GPT, T5...)의 크기와 복잡도를 줄이면서 그들의 기능성을 보존하는 기술
  - 필요한 파라미터, 연산, 비트 수를 줄임으로써 계산 자원을 절약
  - 모델의 훈련 및 실행에 필요한 전력과 시간을 낮춤
  - 모바일 폰, 태블릿, 엣지 디바이스와 같은 자원이 제한된 기기에서도 모델을 실행
- LLM을 SLM으로 만드는 과정은 모델 압축(Model compression) 또는 모델 경량화(Model distillation)이라고 불림.

# Model Compression methods

- Model Compression의 주요 4가지 방법

- 양자화(Quantization)

- ✓ 모델의 가중치와 활성함수를 더 낮은 비트로 표현함으로써 모델의 메모리 사용량을 줄이고, Inference 속도를 향상
    - ✓ 적용 범위 大, 효과성 大, 구현 복잡성 中

- 지식 증류(Knowledge Distillation)

- ✓ 큰 모델(선생님)의 지식을 소형 모델(학생)에 전달하는 방식으로 학습. 학생 모델이 선생님 모델의 성능을 모방
    - ✓ 적용 범위 大, 효과성 大, 구현 복잡성 大

- 가지치기(Pruning)

- ✓ 모델에서 상대적으로 중요도가 낮은 가중치나 뉴런을 제거해 모델의 크기를 줄이고 계산 효율성을 높이는 기법.
    - ✓ 적용 범위 中, 효과성 中, 구현 복잡성 中

- 희소화(Sparsification)

- ✓ 모델의 파라미터 중 일부를 제거하여 모델 전반적으로 희소한(즉, 많은 가중치가 0인) 구조를 만드는 방법
    - ✓ 적용 범위 小, 효과성 中, 구현 복잡성 大

# LLM serving framework

# Why do we need serving frameworks?

- LLM을 활용한 서비스 구현 방식은 실행 환경과 목표에 따라 달라질 수 있음.
- 개인용 컴퓨터나 소형 장치에서 LLM을 최적화하여 사용하려고 할 경우, llama.cpp와 같은 소프트웨어를 사용하며, 모델 크기가 메모리에 비해 클 때 GPU 계층 오프로딩과 모델 압축을 통해 컴퓨팅 자원을 효과적으로 관리합니다.
- 대규모 사용자 기반에 추론 서비스를 제공하고자 할 때는 고성능 서빙 프레임워크를 사용하는 것이 필요함. 이런 프레임워크는 대용량의 동시 요청을 효율적으로 처리하기 위한 기능들을 제공함.

# Metrics for LLM Serving

메트릭	설명
최초 토큰까지의 시간 (TTFT)	사용자 질의 후 모델이 첫 번째 출력을 보여주기까지의 시간. 실시간 상호작용에 중요.
출력 토큰 당 시간 (TPOT)	각 사용자의 쿼리에 대해 하나의 출력 토큰을 생성하는 데 소요되는 시간. 사용자가 느끼는 모델의 속도를 나타냄.
대기 시간(Latency)	모델이 사용자의 전체 응답을 생성하는 데 걸리는 총 시간. TTFT와 TPOT을 사용해 계산 가능.
처리량(Throughput)	모든 사용자와 요청에 걸쳐 추론 서버가 초당 생성할 수 있는 출력 토큰의 수.

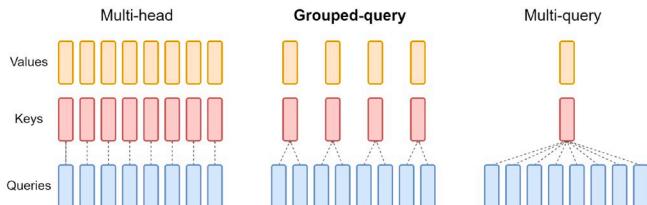
# Comparison of frameworks for LLMs inference

Framework	Tokens Per Second	Query per second	Latency	Adapters	Quantisation	Variable precision	Batching	Distributed Inference	Custom models	Token streaming	Prometheus metrics
vLLM	115	0.94	4.8 s	✓	✓	✓	✓	✓	✓	✓	✓
Text generation inference	50	0.26	4.8 s	✓	✓	✓	✓	✓	✓	✓	✓
CTranslate2	93	0.55	4.5 s	✗	✓	✓	✓	✓	✓	✓	✗
DeepSpeed-MII	80	0.47	2.5 s	✗	✗	✓	✓	✓	✗	✗	✗
OpenLLM	30	0.15	6.6 s	✓	✓	✓	✗	✗	✓	✓	✓
Ray Serve	28	0.15	7.1 s	✗	✓	✓	✓	✓	✓	✗	✓
MLC LLM	25	0.13	7.2 s	✗	✓	✗	✗	✗	✓	✓	✗

# LLM trends

# Gemma 2 주요 특징

- Gemma 2는 소형화된 트랜스포머 아키텍처로, 특정 태스크에 맞춘 경량화 모델로 파라미터 수를 줄이는 대신, 특정 도메인이나 태스크에서 높은 성능을 유지하려고 설계
  - 8192 토큰의 컨텍스트 길이
  - Rotary Position Embeddings (RoPE), GeGLU activation 사용
  - 각 레이어에서 슬라이딩 윈도우 어텐션과 글로벌 어텐션을 번갈아 가며 사용.
  - RMSNorm: 트랜스포머 서브 레이어(어텐션 레이어와 피드포워드 레이어 포함)의 입력과 출력을 정규화하기 위해 RMSNorm을 사용하여 학습의 안정성 향상
  - Group Query Attention(GQA) 사용



Parameters	2.6B	9B	27B
$d_{\text{model}}$	2304	3584	4608
Layers	26	42	46
pre-norm	yes	yes	yes
post-norm	yes	yes	yes
Non-linearity	GeGLU	GeGLU	GeGLU
Feedforward dim	18432	28672	73728
Head type	GQA	GQA	GQA
Num heads	8	16	32
Num KV heads	4	8	16
Head size	256	256	128
global att. span	8192	8192	8192
sliding window	4096	4096	4096
Vocab size	256128	256128	256128
Tied embedding	yes	yes	yes

- **사전 학습(Pre-Training)**
  - 데이터 소스: 주로 영어로 된 다양한 데이터 소스를 사용하여 사전 학습이 이루어집니다. 여기에는 웹 문서, 코드, 과학 논문 등이 포함됩니다.
  - 토큰 처리: 모델 크기에 따라 처리하는 토큰의 수가 다릅니다. 예를 들어, 27B 모델은 13T 토큰으로 학습되며, 9B 모델은 8T 토큰, 2.6B 모델은 2T 토큰으로 학습됩니다.
  - 토크나이저: Gemma 1 및 Gemini에서 사용된 SentencePiece 토크나이저를 사용합니다. 이 토크나이저는 256k의 단어 사전을 가지고 있습니다.
  - 데이터 필터링: 불필요하거나 안전하지 않은 발언의 위험을 줄이기 위해 사전 학습 데이터셋은 엄격하게 필터링됩니다. 개인 정보나 민감한 데이터는 필터링되고, 평가 데이터셋은 사전 학습에서 제외됩니다.
  - 지식 증류(Knowledge Distillation): 27B 모델을 교사 모델로 사용하여 9B 및 2.6B 모델을 학습시킵니다. 이는 모델의 품질을 향상시키는 데 도움이 됩니다.
- **사후 학습(Post-Training)**
  - 지도 파인튜닝(Supervised Fine-Tuning, SFT): 사전 학습된 모델을 텍스트 기반의 프롬프트-응답 페어를 사용하여 미세 조정합니다. 이 과정에서 더 큰 교사 모델로부터의 지식 증류도 포함됩니다.
  - 사용자 피드백을 통한 강화 학습(RLHF): 미세 조정된 모델을 기반으로 RLHF 알고리즘을 사용하여, 사용자 피드백 데이터로 보상을 학습하고 모델의 정책을 더욱 정교하게 다듬습니다.
  - 모델 병합(Model Merging): 여러 단계에서 얻은 모델을 결합하여 전반적인 성능을 향상시킵니다.

# Llama 3 주요 특징

- Llama 3는 Meta에서 개발한 모델로, 트랜스포머 아키텍처의 발전된 버전. Llama 시리즈는 기존의 트랜스포머 아키텍처를 더욱 최적화하고, 파라미터 효율성을 개선하는 방향으로 발전
  - 8192 토큰의 컨텍스트 길이, 128K의 토큰을 가진 새 토크나이저가 사용. 언어를 효율적으로 인코딩하며, 모델 성능을 크게 향상
  - Rotary Position Embeddings (RoPE), Group Query Attention(GQA) 사용

	Finetuned	Multilingual	Long context	Tool use	Release
Llama 3 8B	✗	✗ <sup>1</sup>	✗	✗	April 2024
Llama 3 8B Instruct	✓	✗	✗	✗	April 2024
Llama 3 70B	✗	✗ <sup>1</sup>	✗	✗	April 2024
Llama 3 70B Instruct	✓	✗	✗	✗	April 2024
Llama 3.1 8B	✗	✓	✓	✗	July 2024
Llama 3.1 8B Instruct	✓	✓	✓	✓	July 2024
Llama 3.1 70B	✗	✓	✓	✗	July 2024
Llama 3.1 70B Instruct	✓	✓	✓	✓	July 2024
Llama 3.1 405B	✗	✓	✓	✗	July 2024
Llama 3.1 405B Instruct	✓	✓	✓	✓	July 2024



	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	6,144	12,288	20,480
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	$3 \times 10^{-4}$	$1.5 \times 10^{-4}$	$8 \times 10^{-5}$
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ( $\theta = 500,000$ )		

- **사전 학습(Pre-Training)**
  - 초기 사전 학습: 학습 초기 단계에서는 안정성을 높이기 위해 배치 크기를 조정하고, 이후 단계에서는 효율성을 높이기 위해 배치 크기를 늘립니다. 비영어 데이터와 수학적 데이터를 포함한 특정 작업의 성능을 향상시키기 위해 사전 학습 데이터의 구성도 조정됩니다. 또한, 최신 웹 데이터를 추가하여 모델의 지식이 최신 상태를 유지하도록 합니다.
  - 긴 문맥 사전 학습: 최종 사전 학습 단계에서는 Llama 3 405B 모델이 최대 128K 토큰의 컨텍스트 윈도우를 지원할 수 있도록 긴 시퀀스를 사용해 학습합니다. 이 과정에서 지원되는 컨텍스트 길이를 점진적으로 늘려가며, 모델이 짧은 컨텍스트 평가에서 성능을 회복시킵니다.
  - Annealing 단계: 마지막 사전 학습 단계에서는 Llama 3 405B 모델이 남은 40M 토큰으로 학습되며, 이때 학습률은 선형적으로 0까지 감소합니다. 이 단계에서도 컨텍스트 길이는 128K 토큰으로 유지됩니다. 또한, 고품질 데이터 소스를 더 많이 활용하도록 데이터 구성이 조정됩니다. 최종 사전 학습 모델은 annealing 과정에서 여러 체크포인트의 평균값을 계산하여 생성됩니다.

- **사후 학습(Post-Training)**
  - 지도 파인튜닝(Supervised Fine-Tuning, SFT): 사전 학습된 모델을 텍스트 기반의 프롬프트-응답 페어를 사용하여 미세 조정합니다. 이 과정에서 더 큰 교사 모델로부터의 지식 종류도 포함됩니다.
  - 사용자 피드백을 통한 강화 학습(RLHF): 미세 조정된 모델을 기반으로 RLHF 알고리즘을 사용하여, 사용자 피드백 데이터로 보상을 학습하고 모델의 정책을 더욱 정교하게 다듬습니다.
  - 모델 병합(Model Merging): 여러 단계에서 얻은 모델을 결합하여 전반적인 성능을 향상시킵니다.

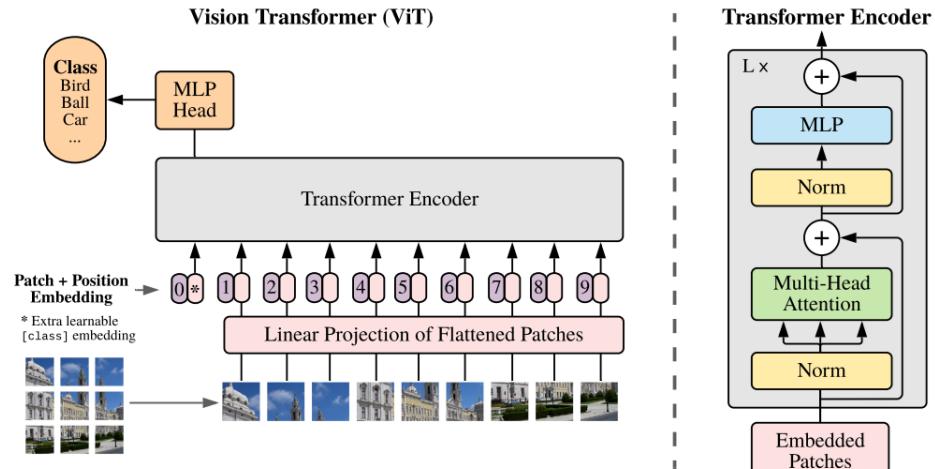
# LLM trends

- 모듈성 강화: LLM은 특정 태스크에 맞춤화된 모듈을 유연하게 추가하거나 제거할 수 있는 모듈화된 구조를 채택해, 다양한 작업에 대해 최적화된 성능을 발휘. 예를 들어, Mixture of Experts(MoE) 구조와 같이 코드 생성, 논리적 추론, 감정 분석 등 특정 작업에 필요한 모듈만 활성화 가능.
- 학습 과정의 개선: Attention Mechanism의 변형, Dynamic Sparsity 등의 기법이 적용되어 계산 효율성, 메모리 사용량, 훈련 속도가 개선.
- 멀티모달 학습: 텍스트뿐만 아니라 이미지, 오디오, 동영상 등 다양한 형태의 데이터를 동시에 학습할 수 있는 멀티모달 기능이 강화.
- 파인튜닝 및 커스터마이제이션: Instruction Tuning과 RLHF를 통해 특정 도메인이나 사용자 요구에 맞춤화된 모델을 개발할 수 있는 파인튜닝 기법이 발전
- 장기 문맥 처리: LLM이 매우 긴 문맥(최대 128K 토큰)을 처리할 수 있도록 설계되어, 긴 문서나 복잡한 대화에서의 성능이 향상.
- 데이터 품질과 다양성: 고품질 데이터의 사용과 다양한 데이터 필터링 기법이 적용되어, 보다 신뢰성 있고 정교한 모델이 개발되고 있으며, 다국어 데이터의 활용도 중요해지고 있음.

# Vision Transformer

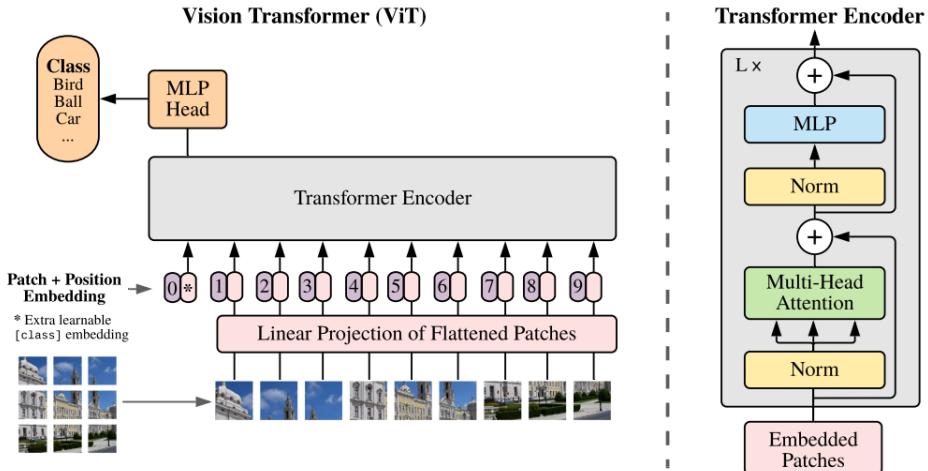
# Vision Transformer(ViT)

- Vision Transformer (ViT)는 Transformer 아키텍처를 이미지 인식 작업에 적용
  - 언어모델인 Transformer의 기원 원리를 시각데이터에 적용해 큰 성공 사례
- ViT는 이미지를 여러 개의 작은 패치로 분할하고, 이러한 패치들을 Transformer 모델의 입력으로 사용하여 이미지를 처리
  - 언어 토큰을 이미지 패치로 대체
  - ViT는 입력 이미지를 고정된 크기의 패치로 분할(e.g 16x16 패치) 후 벡터로 Flatten
  - 각 패치에 위치 정보를 추가하는 Position Embeddings가 적용

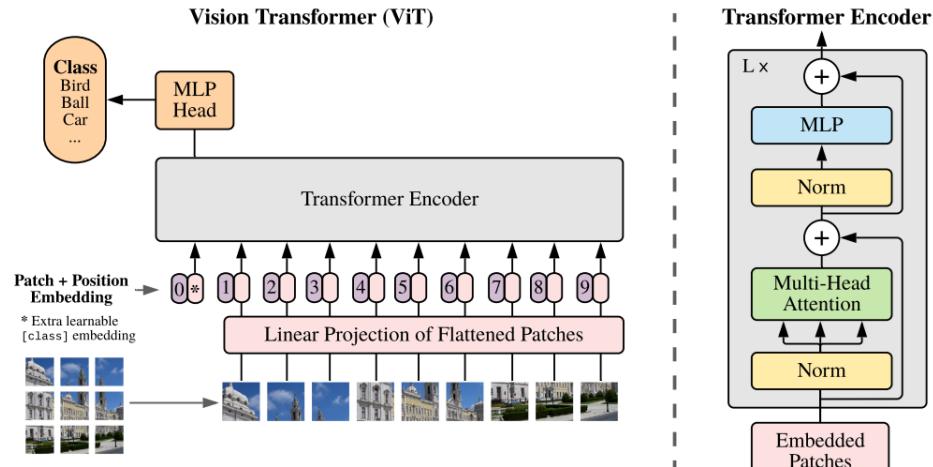


Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

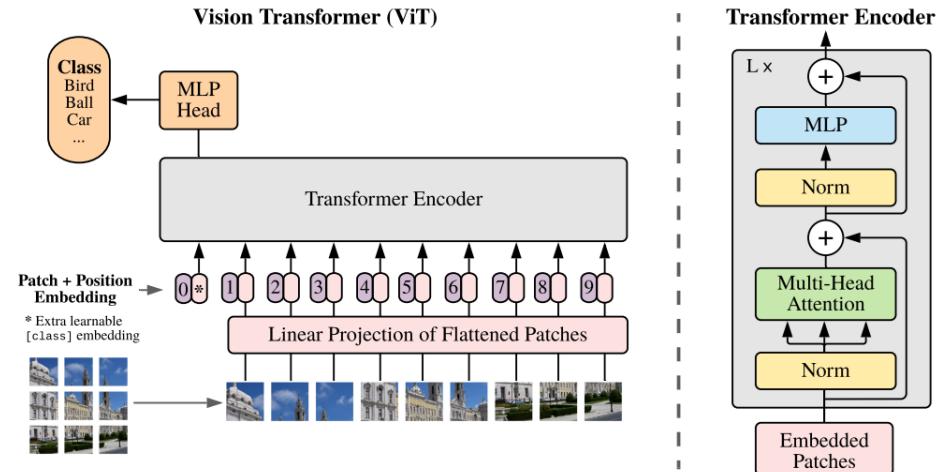
- 입력 사진을  $16 \times 16$  패치로 분할 후 평탄화 후 각 패치를 토큰처럼 다룸.
  - $(16, 16, 3) \Rightarrow (768)$
- Flatten된 패치는 linear embedding 수행
  - $(768) \Rightarrow (128)$
- 언어모델과 동일하게 patch signal에 Positional embedding 더함.
  - $(128) \Rightarrow (128)$



- Transformer Encoder 사용
  - 데이터는 Transformer의 Encoder 구조를 통해 처리
  - Encoder는 Multi-head Self-Attention 메커니즘과 Feed Forward 네트워크로 구성
- ViT는 분류 작업을 위해, 각 입력 시퀀스의 시작에 "Class Token" 토큰을 추가
  - 이 토큰은 이미지 분류를 위한 최종 출력 벡터로 사용되는 특별한 토큰
- Vision Transformer의 장점은 확장성이 좋으며, 대규모 스케일 학습에서 CNN 보다 한계 성능이 더 우월함.



- Inductive bias의 부족으로 학습을 위해 CNN 보다 더 많은 데이터가 필요하다는 단점 있음.
- 논문에서는 303M개 이미지와 18K 클래스가 포함된 Google의 비공개 데이터 JFT-300M을 사용.
- ImageNet으로 pre-training을 했을 때는 그리 좋은 성능을 보이지 못했음.  
Transformer는 CNN과 달리 Inductive bias가 없기에 더 많은 데이터가 필요

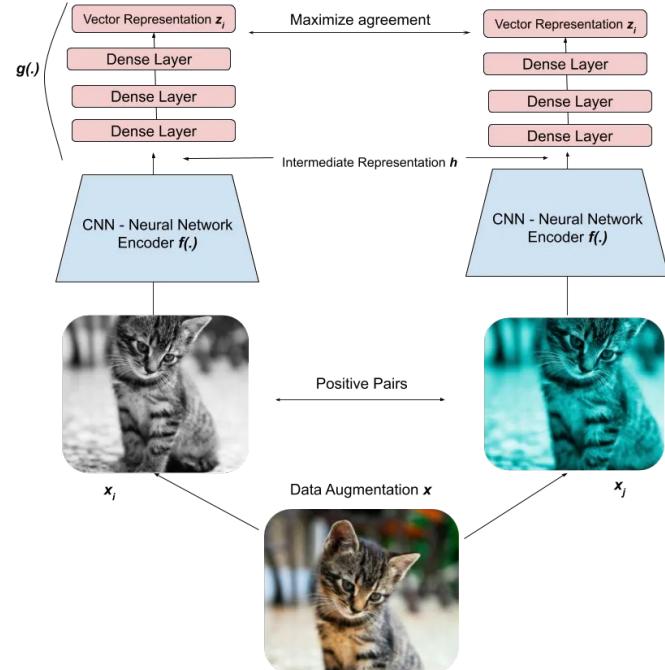


# Self-supervised learning for vision tasks

- 최근 비지도학습(unsupervised learning)보다 자기주도학습(self-supervised learning)이라는 용어가 더 잘 받아들여지고 있음. 자기주도학습은 정답 생성이 가능한 문제를 스스로 풀어보는 과정에서 자연스러운 데이터 분포가 무엇인지 학습.
- 가장 간단한 방식은 모델 학습을 위한 새로운 형태의 문제인 pretext task를 학습
  - 이미지를 회전시키고 회전 방향과 각도를 맞추도록 학습
  - 이미지를 잘라 zigsaw 퍼즐을 만들고 모델이 퍼즐을 풀도록 학습
- 그러나 pretext task를 통해 학습하는 방식은 pretext task를 잘 풀게끔 학습될 뿐 이미지의 일반적인 시각 특징을 포착하는데 실패해왔음.

# SimCLR

- SimCLR은 Google Research에서 발표한 contrastive learning을 위한 학습 구조
- 하나의 이미지에서 무작위적으로 증강한 두 개의 이미지는 서로 positive pair로 정의



Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.

# *Image augmentation in SimCLR*



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate  $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise



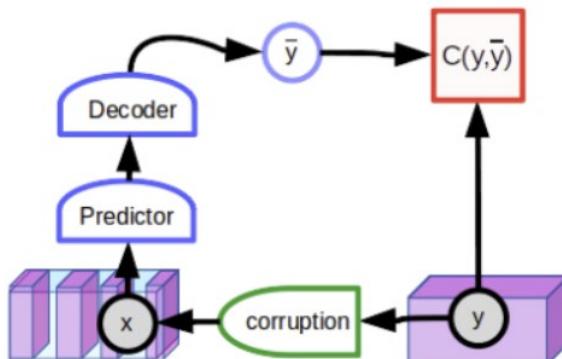
(i) Gaussian blur



(j) Sobel filtering

# Masked language learning for vision tasks

- 또 다른 유망한 접근방법은 데이터에서 드러난 파트에서 숨겨진 파트를 추론하도록 학습하는 것.
  - 예를 들어 자연어처리에서는 전체 문장 중 일부 단어를 숨기고, 나머지 단어로 숨겨진 단어들을 추론해 복원하도록 학습해 임팩트 있는 성과를 얻음(BERT, RoBERTa, XLM-R)

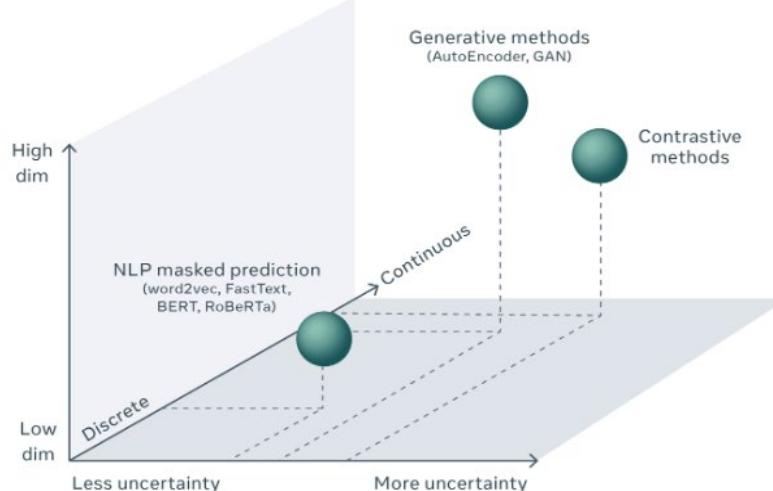


Decoder는 주변 단어를 바탕으로 빈칸의 단어(단어사전의 60k~100k 중 1) 예측. 방대한 문장이 주어진다면 유의어 관계를 이해할 수 있을 정도의 확률이 주어짐.

This is a [...] of text extracted  
[...] a large set of [...] articles

This is a piece of text extracted  
from a large set of news articles

- 반면 컴퓨터비전 분야에서는 같은 수준의 진보를 가져오지 못 했는데 주된 이유는 단어들보다 이미지들에서 예측의 불확실성이 높기 때문
  - 자연어처리에서 손실된 단어의 예측은 거대한 단어사전 중 해당 빈칸에 들어갈 수 있는 단어에 대한 예측이므로 해당 문제에 대한 불확실성은 단어사전 내 모든 가능한 출력 상의 확률 분포가 됨.
- 비디오에서 손실된 프레임들의 예측, 이미지에서 손실된 패치들의 예측, 음성신호에서 손실된 세그먼트의 예측은 고차원 연속값 객체에 대한 예측으로, 가능한 모든 예측들을 특정하는 것이 불가능

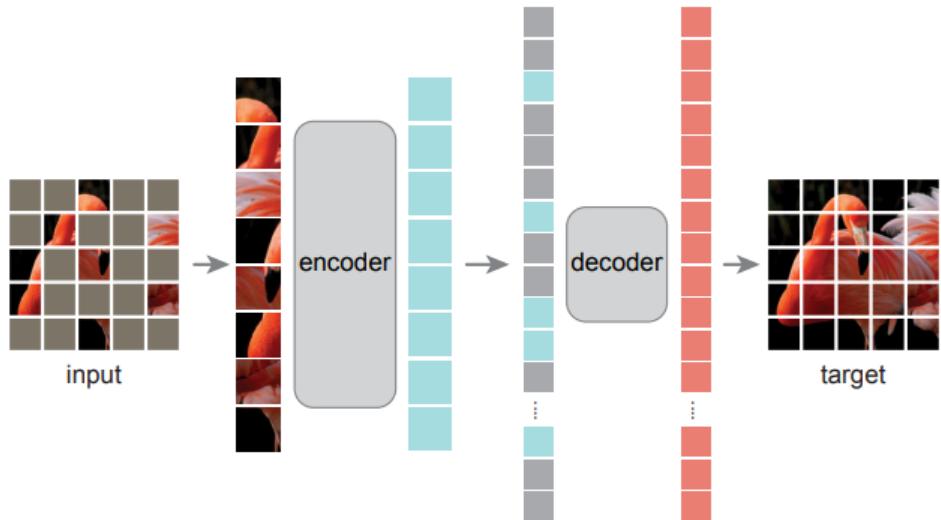


# MAE(Masked AutoEncoder)

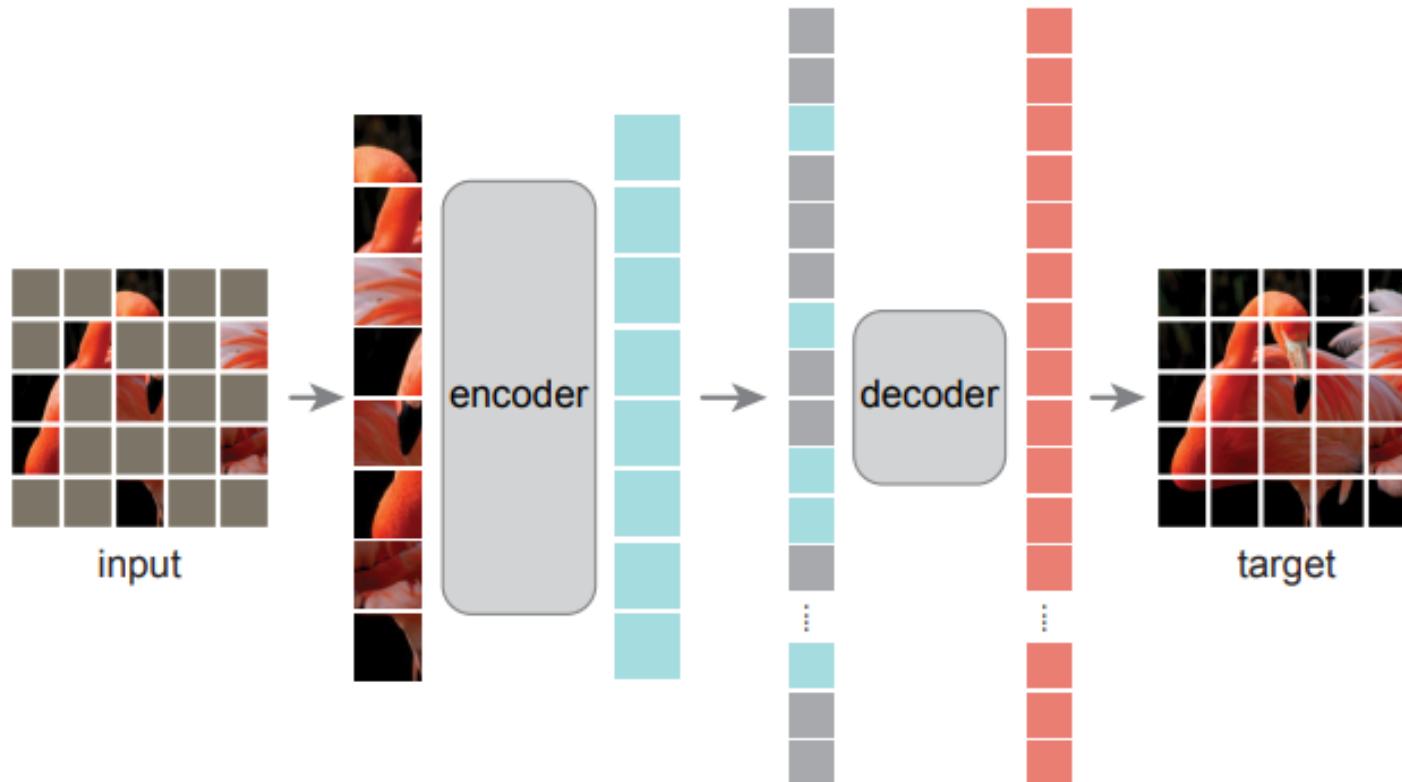
- 저자들은 비전 모델에 masked modeling이 적용되지 않았던 이유를 다음과 같이 설명함.
  - 지금까지 시각모델로 CNN이 지배적이었기 때문에 구조적 차이가 있었음.
    - ✓ ViT가 개발됐기 때문에 해결되었음.
  - 자연어와 시각은 정보 밀도가 굉장히 다르다. 자연어는 인간에 의해 만들어졌기 때문에 훨씬 의미론적이고 정보 밀도가 높다.
  - 제거된 단어를 예측하는 행위는 의미론적 정보를 학습하도록 유도하지만, 제거된 시각정보를 복원하는 행위는 인지적 정보인 주변 영상 패치들에 더 의존할 수 있다.
- 이를 극복하기 위해 전체 넓이의 75%라는 매우 높은 비율로 영상의 픽셀을 제거하였음. 이것은 놀랍게도 다음 두 가지 이점이 있음.
  - 영상의 매우 많은 정보가 제거됐기 때문에 모델은 사진에서 의미론적 정보를 해석하도록 강제
  - 동일한 이유로 학습 시 연산량은 매우 감소시킴(x3).

# MAE(Masked AutoEncoder)

- 2022년 Facebook AI Research 팀은 masked language modeling 기법을 vision task에 적용 성공
- 언어 모델링과 마찬가지로 데이터(이 경우 사진)의 일정 부분을 제거하고, 모델이 제거된 부분을 예측하도록 학습함.



# MAE architecture



# MAE performance

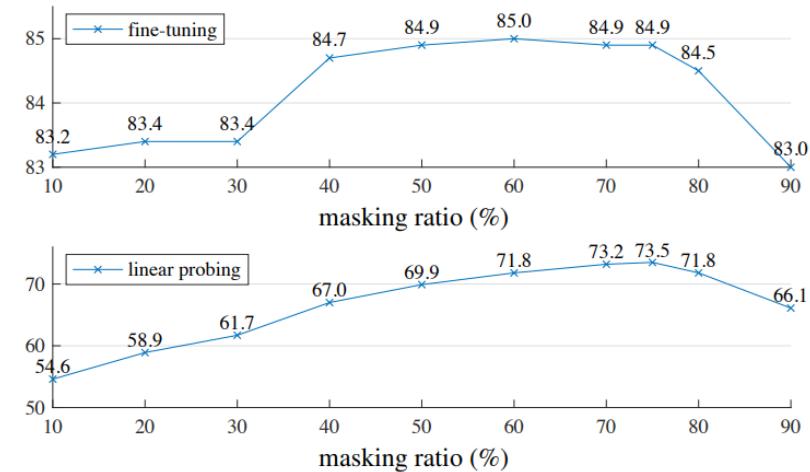
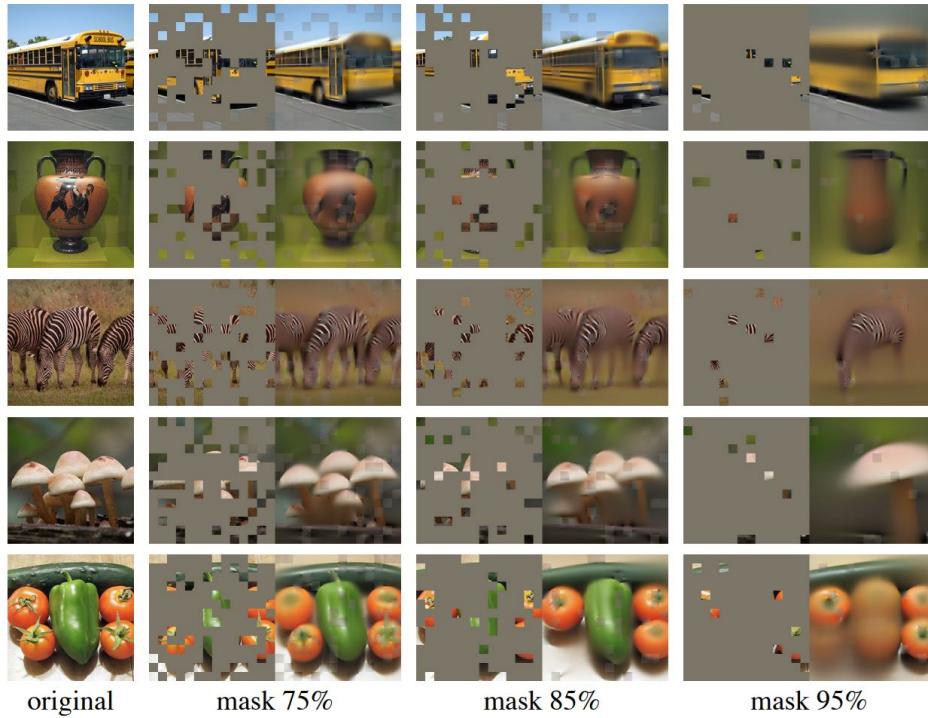
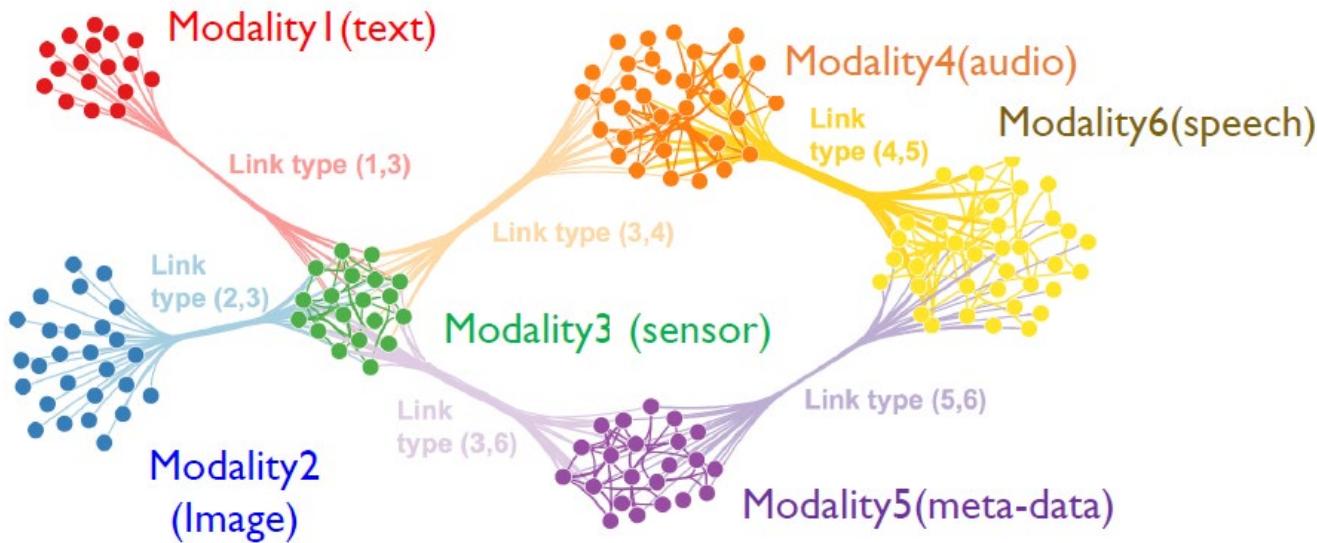


Figure 5. **Masking ratio.** A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

# Multimodal models : CLIP

# Multimodal Deep Learning

- Multimodal Deep Learning은 컴퓨터 비전, 자연어 처리, 음성 인식 등 다양한 모달리티(modality)에서 온 데이터를 처리하고 학습하는 딥러닝 기술

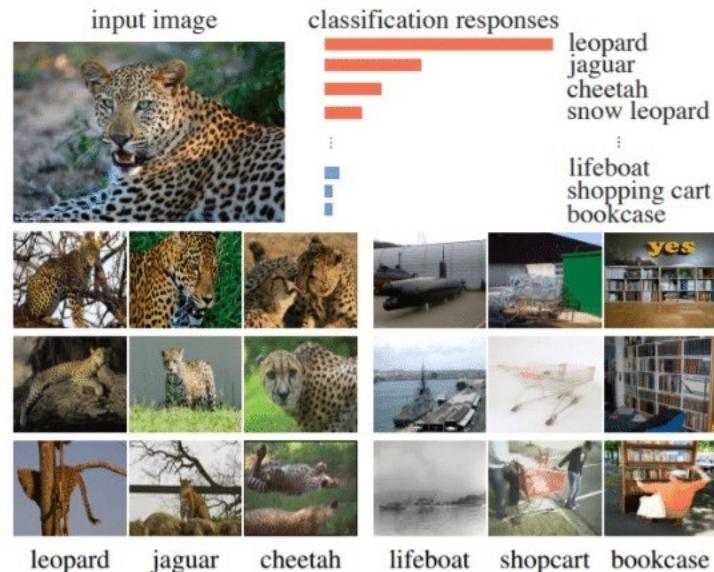


# Task-agnostic learning

- What is the motivation behind CLIP?
  - 왜 비전모델은 언어모델에 비해 재사용성이 낮은가?
- GPT-3와 같은 최신 text-to-text 언어모델들은 특정 작업이나 데이터셋에 특화되지 않고 다양한 다운스트림 작업들에서 잘 작동함.
  - Autoregressive modeling, masked language modeling과 같은 Task-agnostic learning 사용
- 대규모 웹 텍스트를 학습한 Task-agnostic pre-trained model을 구축하는 전략이 작업자가 생성한 높은 품질의 NLP 데이터셋을 구축하는 것보다 유리할 수 있음을 시사.
  - 새로운 작업에 대한 지도학습 데이터가 대규모로 필요하지 않음.
  - Zero-shot transfer, Transfer learning
  - 반면 비전모델은 언어모델에 비해 일반화(generality)와 사용성(reusability)이 낮음.

# 지도학습 방법론의 한계

- 인기있는 비전데이터 ImageNet의 케이스
  - 구축을 위해 노동집약적, 고비용의 문제
  - 22,000 객체의 1400만 여장의 사진을 레이블링하기 위해 25,000명의 작업자 참여
  - 하위데이터셋 ImageNet-1k는 1000 카테고리의 1,281,167장 학습이미지로 구성.
- 제한된 카테고리의 문제
  - 사전정의된 1-of-N 카테고리 레이블 학습
  - 좁은 시각적 개념만을 학습



# 대규모 image-caption 데이터셋의 구축 (1)

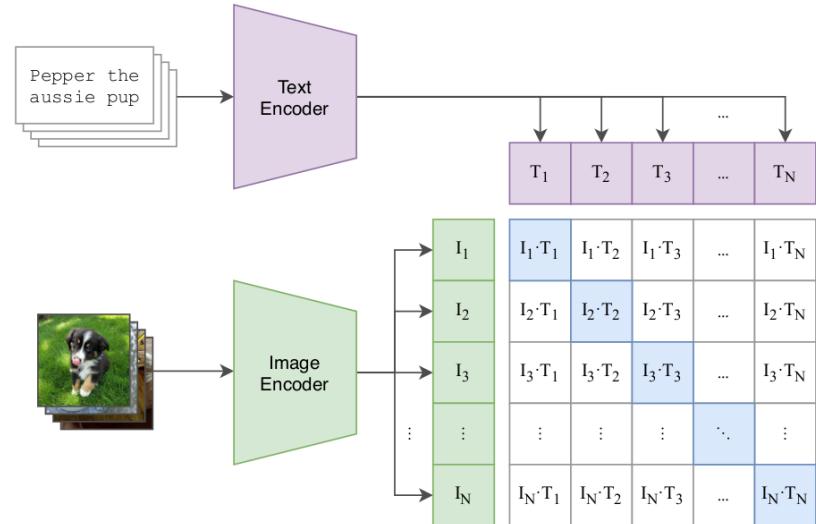
- 자연어 지도학습은 개념적으로 훌륭하지만 낮은 성능 때문에 실제로 사용하는 경우는 매우 드묾
- 최신 성능의 시각모델들은 지도학습 방식보다 데이터 확장성이 큰 semi-supervised, self-supervised 방법 등을 사용
  - 지도학습은 수백~수천만 규모의 데이터를 사용하는 반면 semi-supervised, self-supervised 방법은 수억~수십억 규모의 데이터를 사용해 사전학습 모델을 구축
- 자연어 지도학습을 위한 데이터셋 구축 필요
  - 기존의 자연어 지도학습이 테스트한 MS-COCO(Lin et al., 2014), Visual Genome (Krishna et al., 2017), YFCC100M (Thomee et al., 2016) 등은 십만 장 규모로 작고, YFCC100M은 1억장 수준이지만 자연어 레이블의 낮은 수준이 문제임.

# 대규모 *image-caption* 데이터셋의 구축 (2)

- OpenAI 연구팀은 50만 개의 쿼리 리스트를 구성하고, 인터넷에서 각 쿼리와 관련한 4억 개의 데이터쌍 (이미지, 캡션)을 수집함.
  - 다양한 단어로 수집한 이유는 최대한 다양한 시각적 개념을 학습하기 위함
  - 구축된 데이터셋은 GPT-2를 학습하는데 사용된 WebText 데이터셋과 비슷한 규모
- 쿼리 리스트는 영문 위키피디아에서 100회 이상 등장한 단어 50만 개로 구성
  - 기본 쿼리들은 다시 bi-gram을 사용해 이를 증강함. bi-gram은 자연어에서 두 단어 조합에 대한 확률모델로 증강 시 한 단어보다 더 특정한 의미를 포착할 수 있음.
  - 예를 들어, “고양이”와 “개”가 각각 기본 쿼리 리스트에 있다면 bi-gram에 의해 “고양이 개”가 새로운 쿼리가 추가될 수 있음.
  - 최대한 클래스 균형을 맞추기 위해 각 쿼리마다 2만 여개의 데이터쌍 수집

# CLIP : Contrastive Language-Image Pretraining

- CLIP(Contrastive Language-Image Pretraining)은 사진과 캡션 사이 연관성을 학습하는 자연어 지도학습
  - CLIP의 임베딩 공간에서 사진은 연관된 캡션과 가까워지고 무작위적인 캡션과 멀어짐
- OpenAI 연구팀은 4억 쌍(사진-캡션)의 웹데이터를 수집해 CLIP 모델의 사전학습에 사용



# *CLIP : Contrastive Language-Image Pretraining*

---

- Easy to scale
  - 전통적인 지도학습의 막대한 노동이 필요한 1-of-N 레이블링 방식보다 자연어 지도학습은 웹수집을 통한 데이터 규모의 확장이 상대적으로 쉬움.
- Language-image connection
  - 자율학습(self-supervised learning) 또는 비지도학습(unsupervised learning)과 달리 단순히 시각표현만 학습하는 것이 아닌, 시각표현과 언어를 연결한다는 강점 존재
- Zero-shot transferable
  - CLIP은 자연어를 시각적 패턴과 연결할 수 있기 때문에 언어로 문제를 정의해 추가적 학습 없이 문제를 푸는 zero shot transfer가 높은 정확도로 가능.

# 학습 방법론 개발(1/2)

- 연구초기 접근방법은 Virtex처럼 CNN과 text transformer를 결합해 초기값(scratch) 학습 채택
  - 간단한 Bag-of-words 모델을 쓰는 것보다도 효과적으로 확장하는데 어려움
  - 캡션의 단어들을 정확히 예측하려고 하는 방식이 너무 어렵기 때문
- Contrastive learning 사용
  - 이미지의 아웃풋과 캡션이 동일하도록 예측하는 대신 contrastive learning이 더 효과적

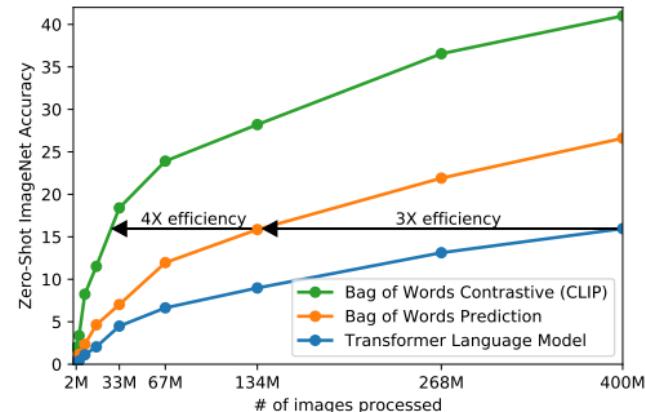
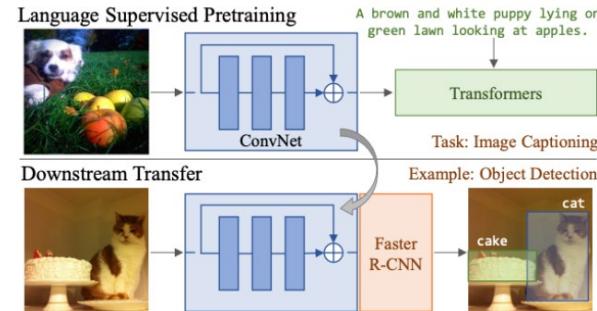


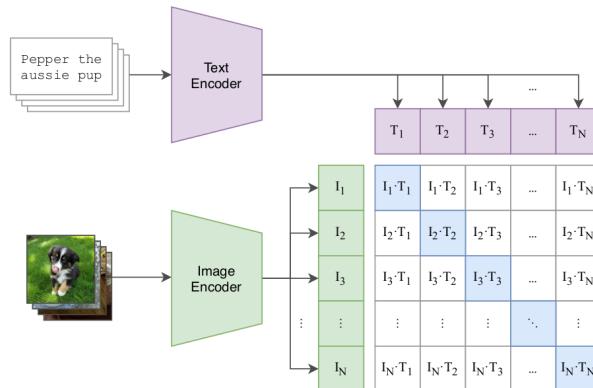
그림. Transformer Language Model(Virtex) 의 낮은 확장성

# 학습 방법론 개발(2/2)

- 연구초기 접근방법은 Virtex처럼 CNN과 text transformer를 결합해 초기값(scratch) 학습 채택
  - 간단한 Bag-of-words 모델을 쓰는 것보다도 성능향상이 어려움
  - 이유는 캡션의 단어들을 정확히 예측하려고 하는 방식이 학습을 어렵게 만듦.
- Contrastive learning이 새로운 대안으로 여겨짐
  - Tian et al., 2019에 따르면 캡션을 직접 예측하는 방식보다 캡션이 영상과 연관있는가를 예측하는 것이 더 효과적

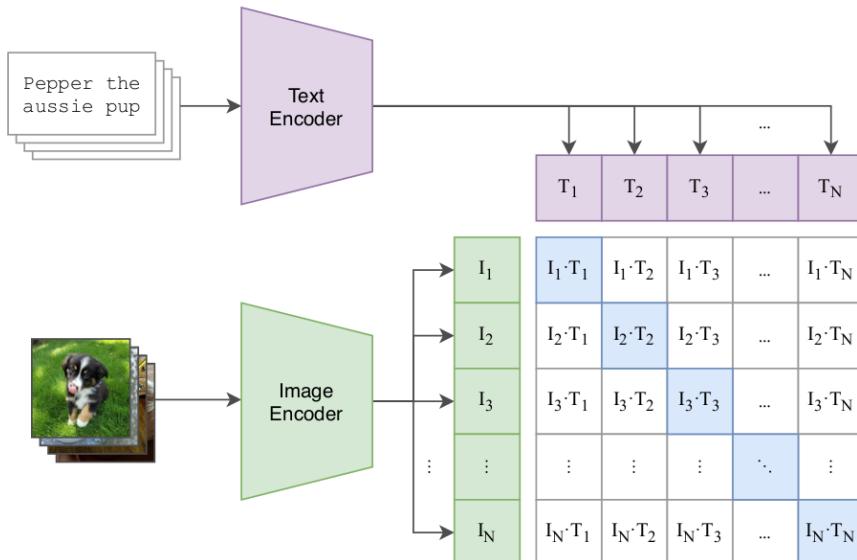


그림A. Equivalent predictive objective method(Virtex)



그림B. Contrastive learning method(CLIP)

# 학습 알고리즘(1/3)



그림A. N batch contrastive learning

```

# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

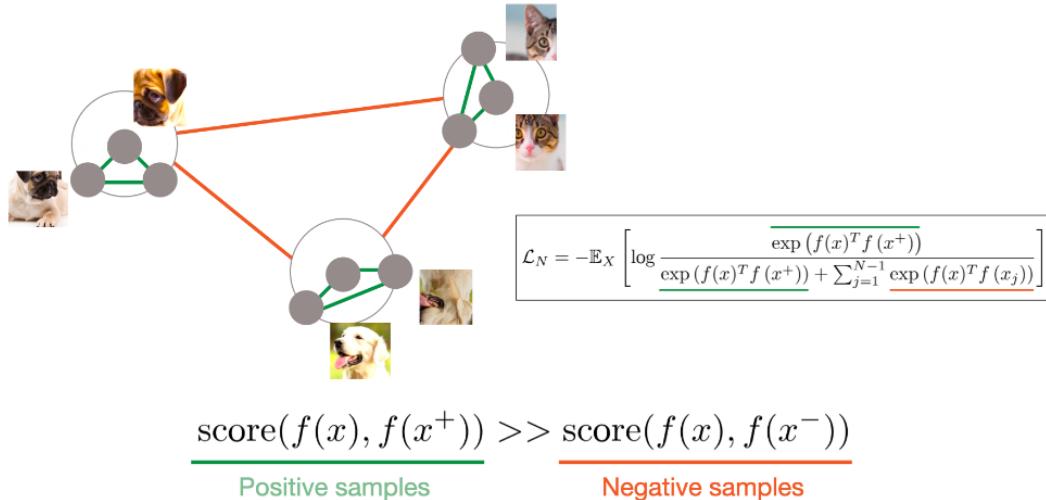
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2

```

그림B. training pseudo code

# 학습 알고리즘(2/3)

- CLIP은 N 개의 배치사이즈에서 N x N의 비교쌍(영상, 캡션)을 학습
  - N x N 비교쌍 중 긍정쌍은 N개 부정쌍은 N\*\*2 - N개 생성
  - 영상 인코더와 텍스트 인코더는 비교쌍 중 긍정쌍의 코사인 유사도를 최대화, 부정쌍은 최소화함
  - 학습의 결과로 영상 인코더와 텍스트 인코더는 하나의 multi-modal 임베딩 공간에서 연결됨

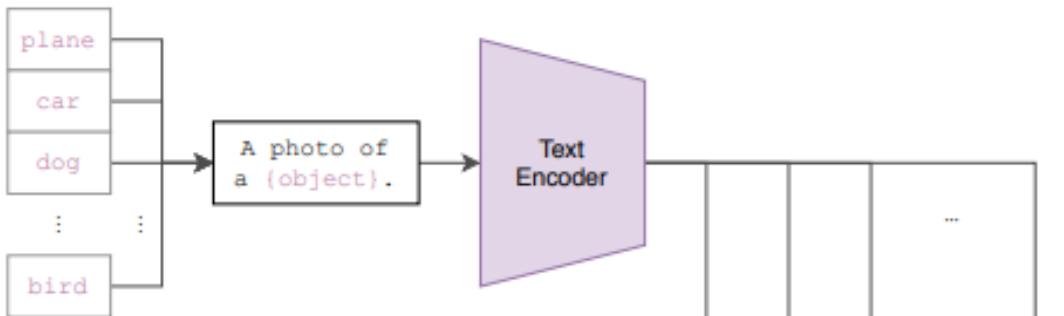


## 학습 알고리즘(3/3)

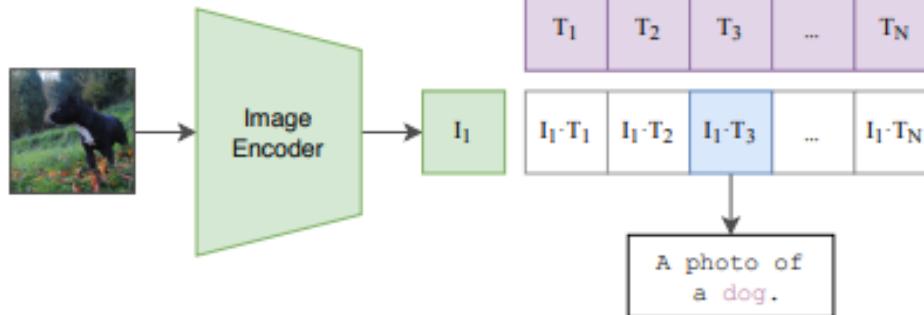
- 영상 인코더는 scratch, 텍스트 인코더는 pre-trained weights 사용
- 영상 인코더는 ResNet과 ViT를 사용
  - ResNet-50의 global average pooling은 attention pooling으로 변경
  - 3가지 타입 : ResNet-50, ResNet-101, EfficientNet-style scaling ResNet 테스트
- 텍스트 인코더는 Transformer 사용
  - 63M-parameter 12-layer 512-wide model with 8 attention heads.
  - 49,152 vocab size
  - 3가지 타입 : ViT-B/32, ViT-B/16, ViT-L/14
- during 32 epochs, train with Adam optimizer, decoupled weight decay, cosine learning rate scheduler, mini-batch 32,768

# Zero-shot transferable

Create dataset classifier from label text



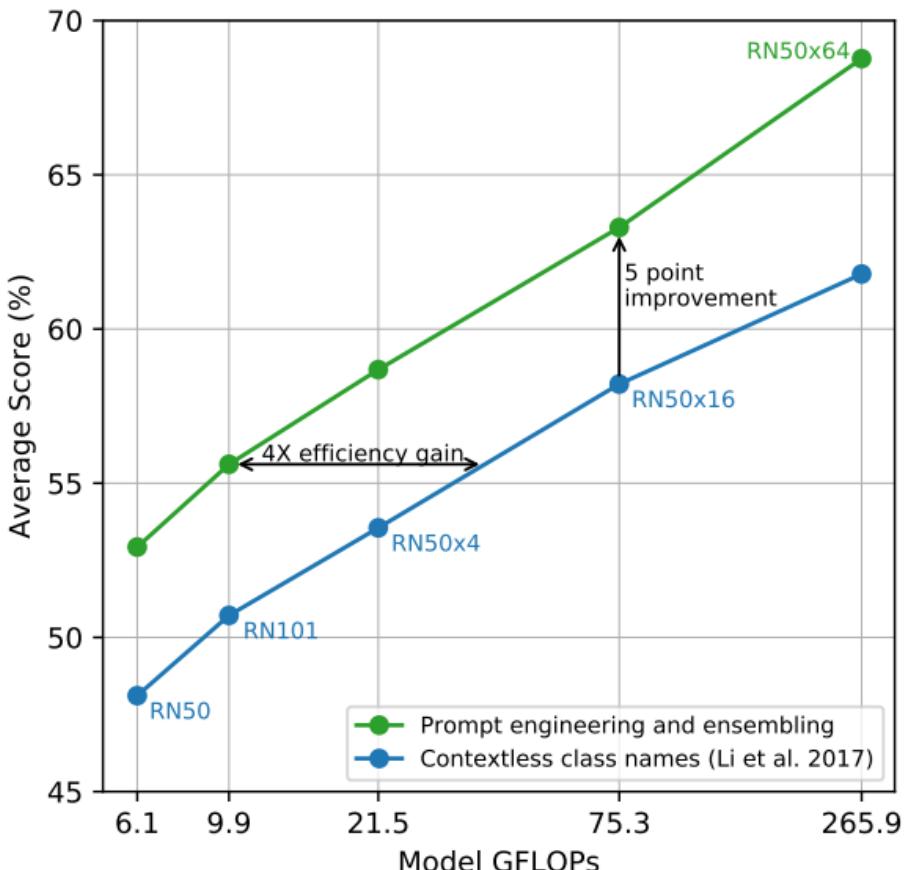
Use for zero-shot prediction



	aYahoo	ImageNet	SUN
Visual N-Grams	72.4	11.5	23.0
CLIP	<b>98.4</b>	<b>76.2</b>	<b>58.5</b>

Table 1. Comparing CLIP to prior zero-shot transfer image classification results. CLIP improves performance on all three datasets by a large amount. This improvement reflects many differences in the 4 years since the development of Visual N-Grams (Li et al., 2017).

# Zero-shot transferable

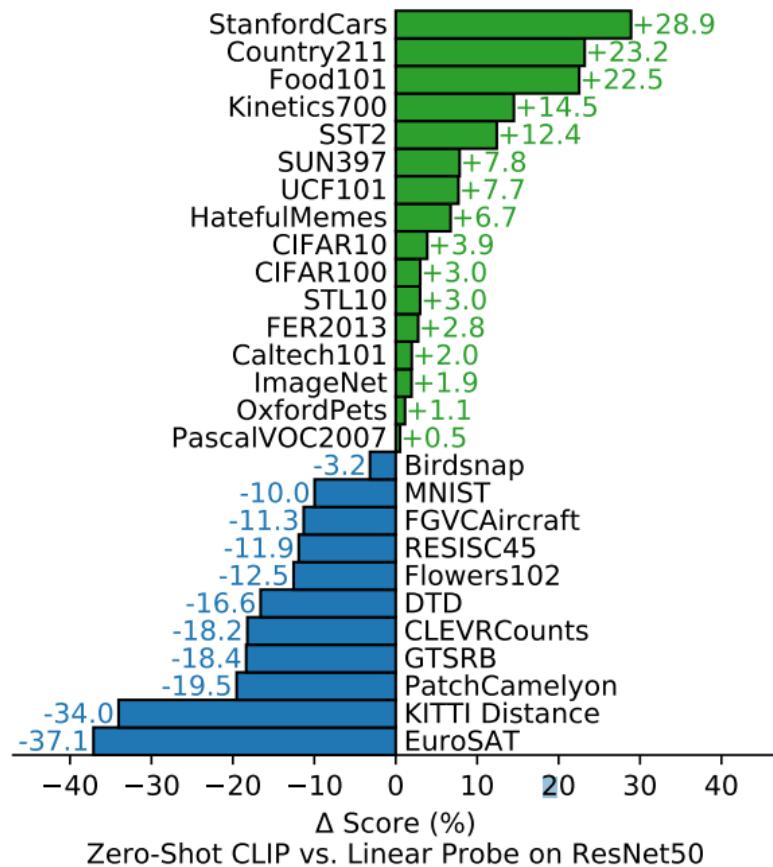


**Figure 4. Prompt engineering and ensembling improve zero-shot performance.** Compared to the baseline of using contextless class names, prompt engineering and ensembling boost zero-shot classification performance by almost 5 points on average across 36 datasets. This improvement is similar to the gain from using 4 times more compute with the baseline zero-shot method but is “free” when amortized over many predictions.

다의성 문제(건설 기계 Crane과 날아다니는 Crane), 캡션의 형식에 따라 달라지는 성능...

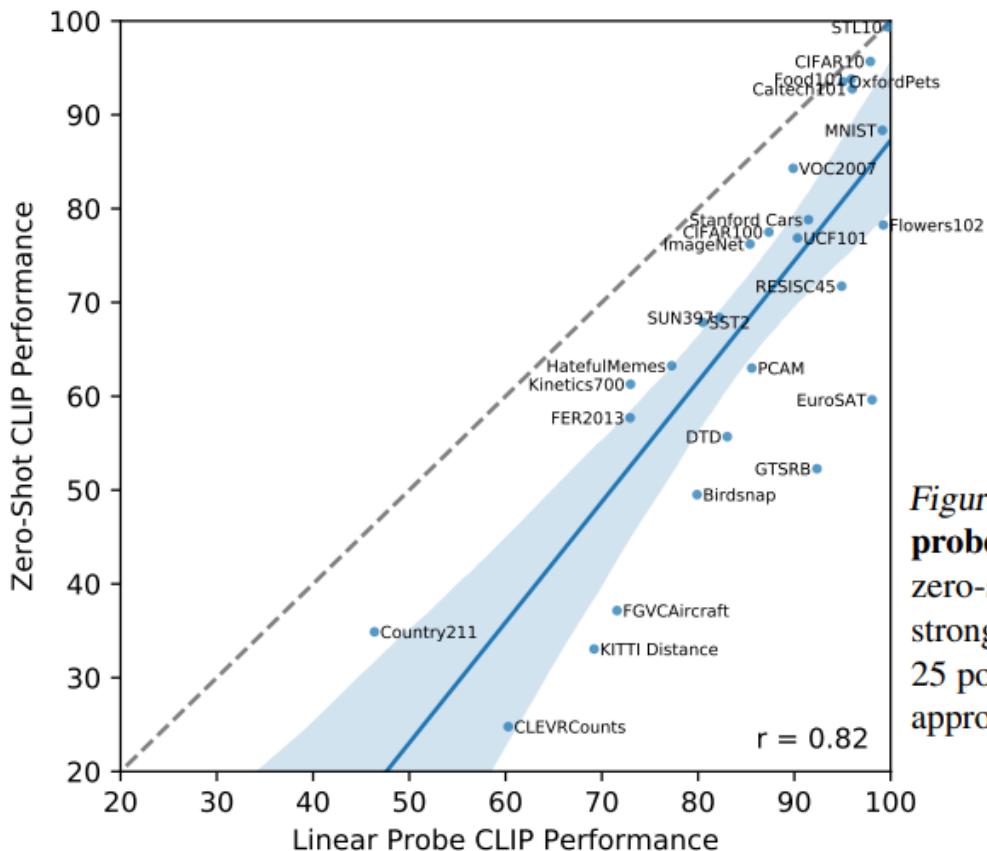
프롬프트 엔지니어링이 상당히 중요

# Zero-shot transferable



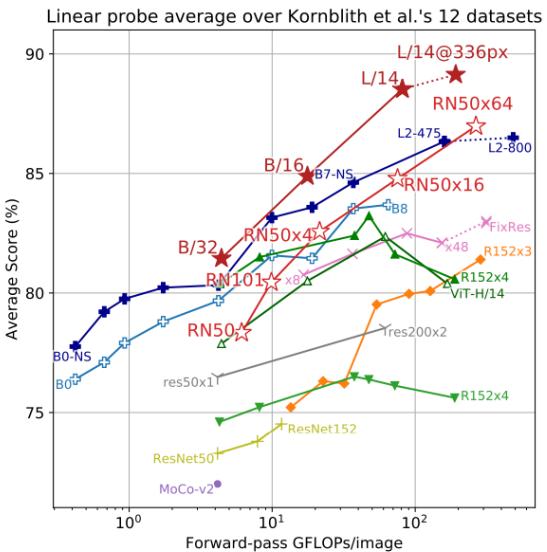
**Figure 5. Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

# Zero-shot transferable



**Figure 8. Zero-shot performance is correlated with linear probe performance but still mostly sub-optimal.** Comparing zero-shot and linear probe performance across datasets shows a strong correlation with zero-shot performance mostly shifted 10 to 25 points lower. On only 5 datasets does zero-shot performance approach linear probe performance ( $\leq 3$  point difference).

# Performance



★ CLIP-ViT	✗ Instagram-pretrained	△ ViT (ImageNet-21k)
★ CLIP-ResNet	◆ SimCLRv2	▲ BiT-M
■ EfficientNet-NoisyStudent	× BYOL	▼ BiT-S
□ EfficientNet	○ MoCo	— ResNet

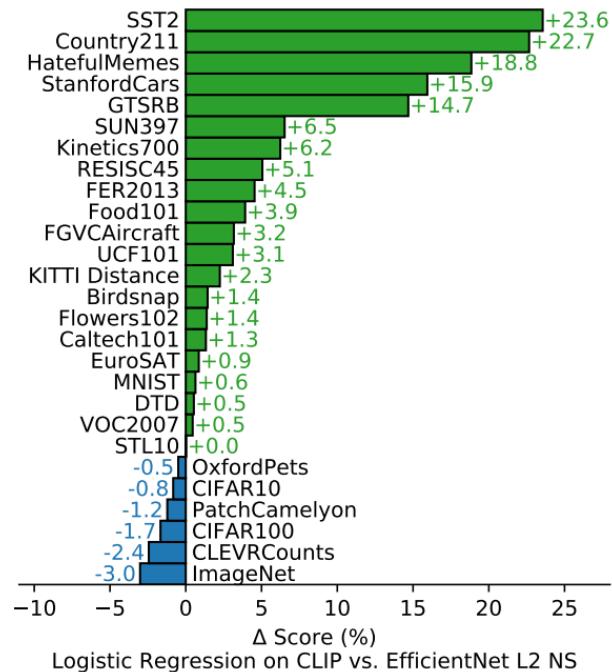
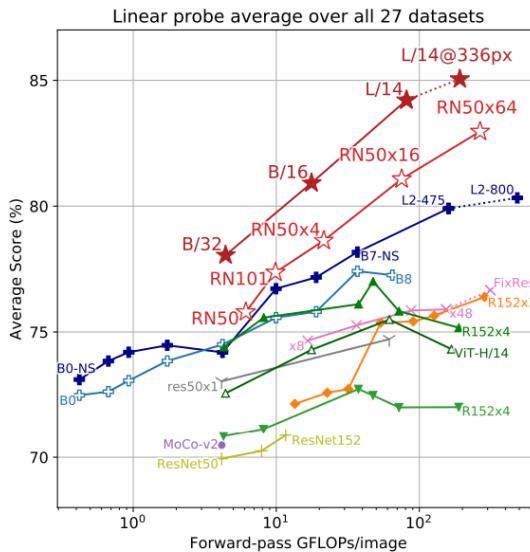
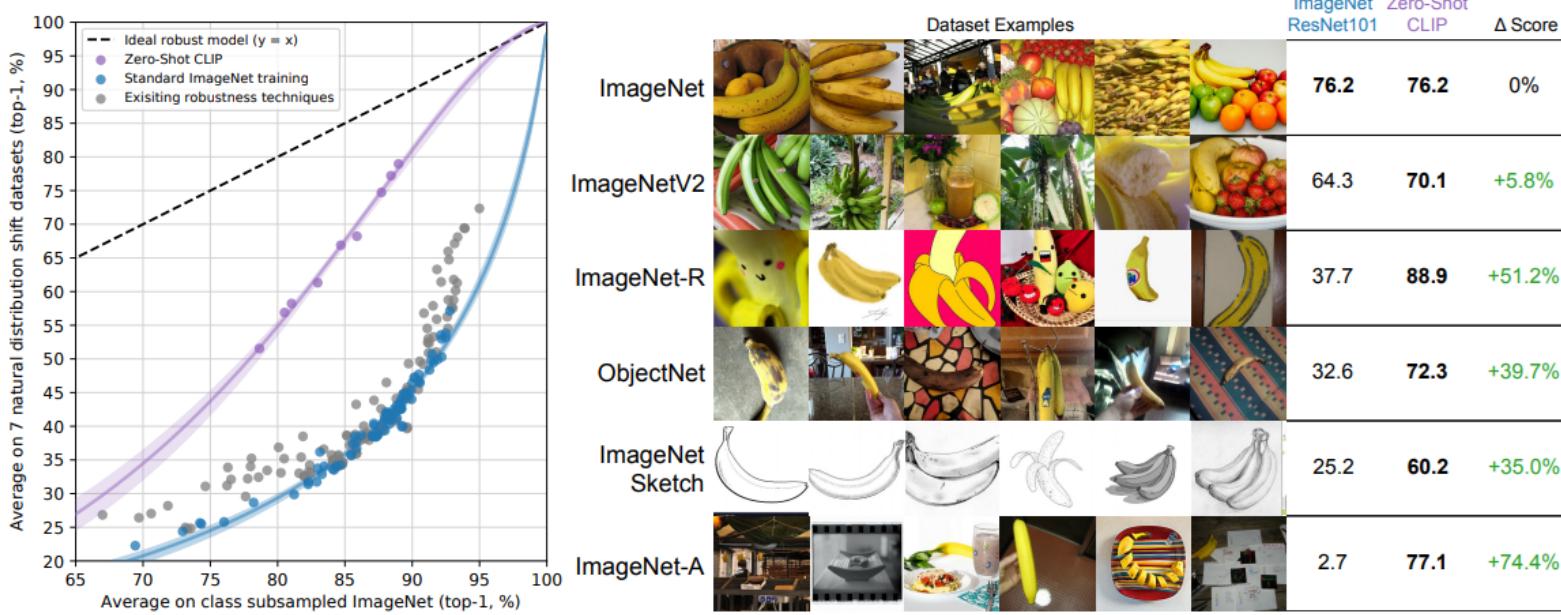


Figure 10. Linear probe performance of CLIP models in comparison with state-of-the-art computer vision models, including EfficientNet (Tan & Le, 2019; Xie et al., 2020), MoCo (Chen et al., 2020d), Instagram-pretrained ResNeXt models (Mahajan et al., 2018; Touvron et al., 2019), BiT (Kolesnikov et al., 2019), ViT (Dosovitskiy et al., 2020), SimCLRv2 (Chen et al., 2020c), BYOL (Grill et al., 2020), and the original ResNet models (He et al., 2016b). (Left) Scores are averaged over 12 datasets studied by Kornblith et al. (2019). (Right) Scores are averaged over 27 datasets that contain a wider variety of distributions. Dotted lines indicate models fine-tuned or evaluated on images at a higher-resolution than pre-training. See Table 10 for individual scores and Figure 20 for plots for each dataset.

Figure 11. CLIP's features outperform the features of the best ImageNet model on a wide variety of datasets. Fitting a linear classifier on CLIP's features outperforms using the Noisy Student EfficientNet-L2 on 21 out of 27 datasets.

# Performance



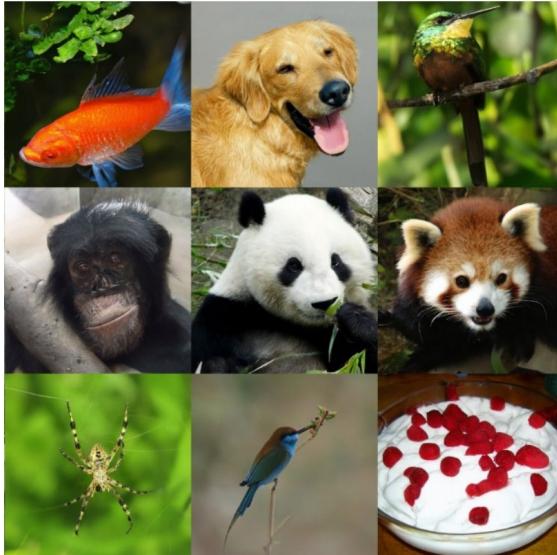
**Figure 13. Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models.** (Left) An ideal robust model (dashed line) performs equally well on the ImageNet distribution and on other natural image distributions. Zero-shot CLIP models shrink this “robustness gap” by up to 75%. Linear fits on logit transformed values are shown with bootstrap estimated 95% confidence intervals. (Right) Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets. The performance of the best zero-shot CLIP model, ViT-L/14@336px, is compared with a model that has the same performance on the ImageNet validation set, ResNet-101.

# Multimodal models : LDM

# LDM fine-tuning

# High-quality image generated by Diffusion Models

확산 모델(Diffusion Models, DMs)은 생성된 이미지의 품질과 다양성이 놀랍도록 뛰어나 GANs, VAEs, Autoregressive models 등 다른 생성 모델들을 능가함



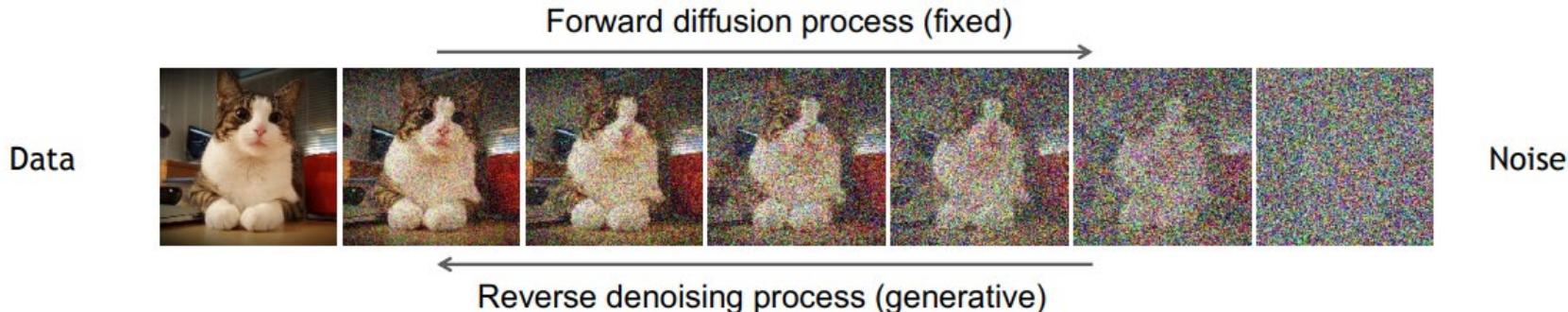
"Diffusion Models Beat GANs on Image Synthesis"  
Dhariwal & Nichol, OpenAI, 2021



"Cascaded Diffusion Models for High Fidelity Image Generation"  
Ho et al., Google, 2021

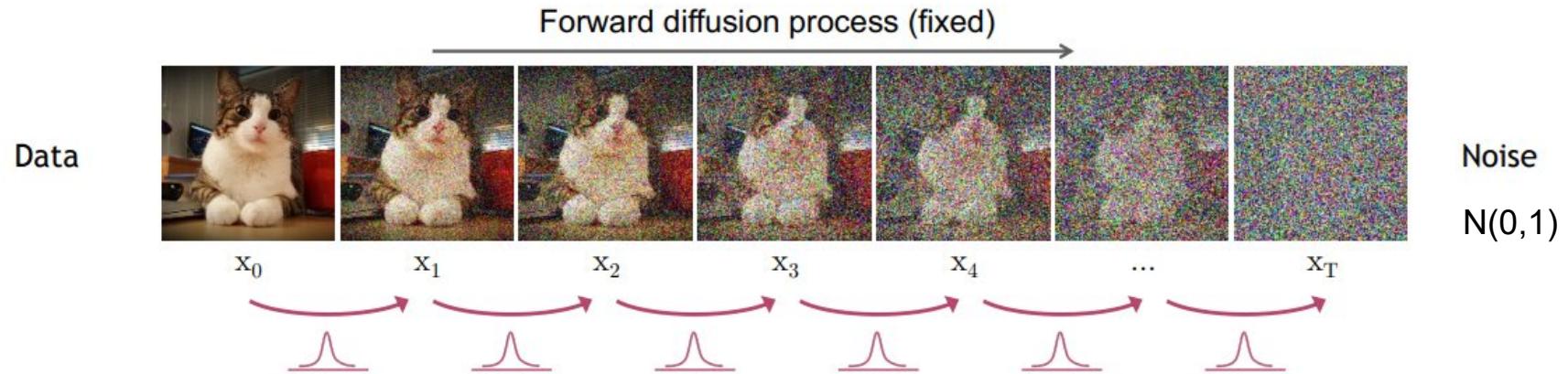
# Denoising diffusion models

- Diffusion models은 두 가지 프로세스로 구성
  - Forward diffusion process : 노이즈를 데이터에 반복적으로 주입해 서서히 파괴하는 과정
  - Reverse denoising process : 데이터의 노이즈를 제거해 서서히 역으로 복원하는 과정
- 실제 이미지에 노이즈를 주입해 만든 학습용 이미지를 다시 복원하도록 학습하면, 모델은 입력된 무의미한 노이즈를 그럴듯한 실제 이미지로 만들 수 있는 생성형 모델이 된다.



# Forward diffusion process

T step 동안 점진적으로 입력에 노이즈를 더한다.



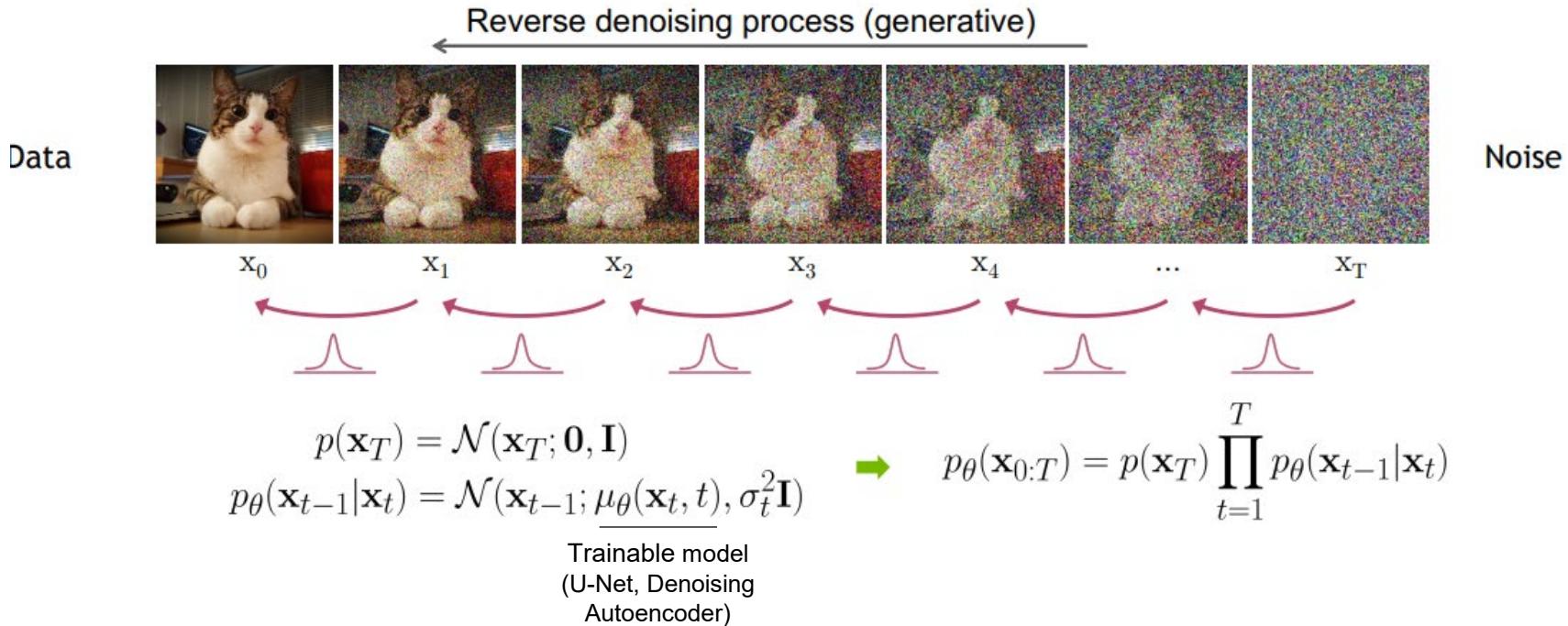
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T} | \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}) \quad (\text{joint})$$

$$0 \leq \beta_t \leq 1 \quad \alpha_t = 1 - \beta_t \quad q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}))$$

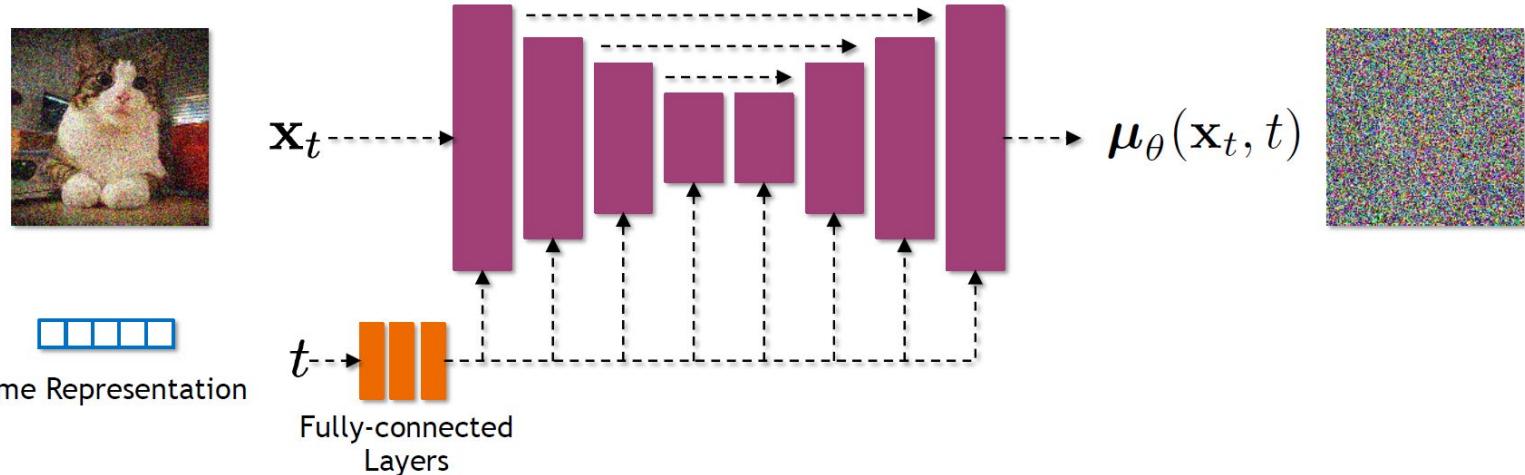
$$\bar{\alpha}_t = \Pi_{s=1}^t (\alpha_s)$$

# Reverse denoising process

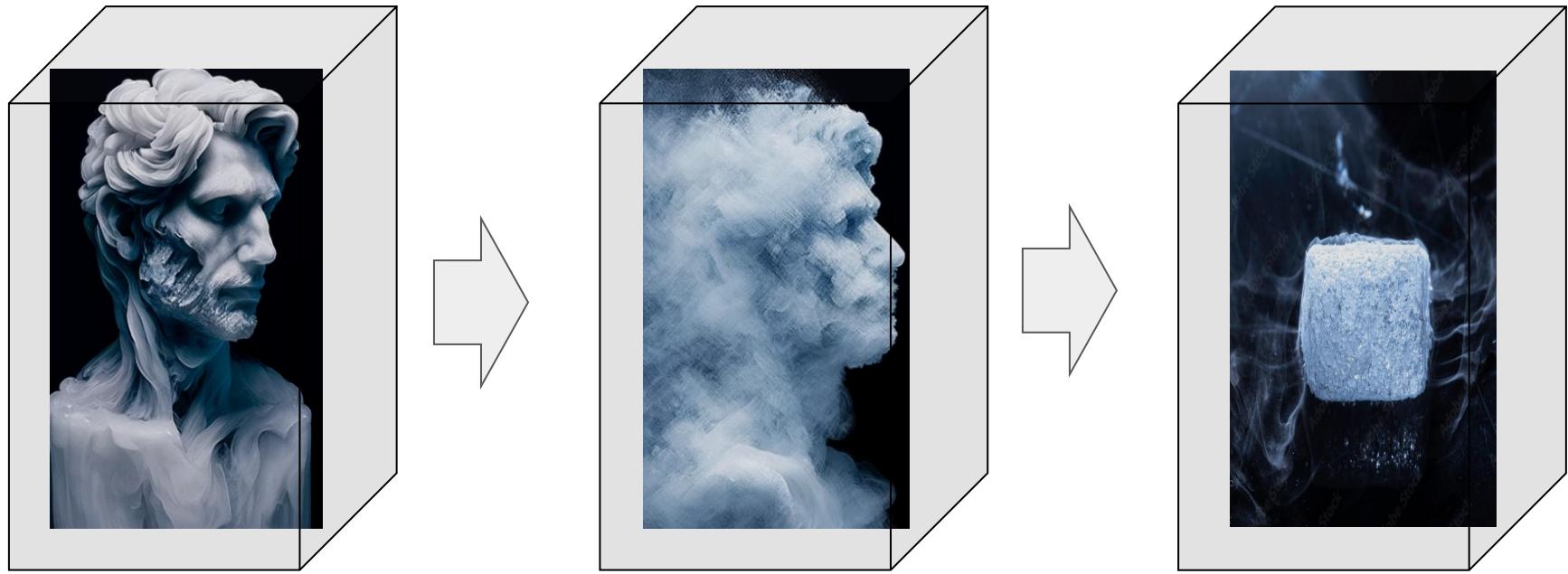
모델  $\mu_\theta$ 은 데이터에서 노이즈를 제거하는 법을 학습한다.



# U-Net architecture

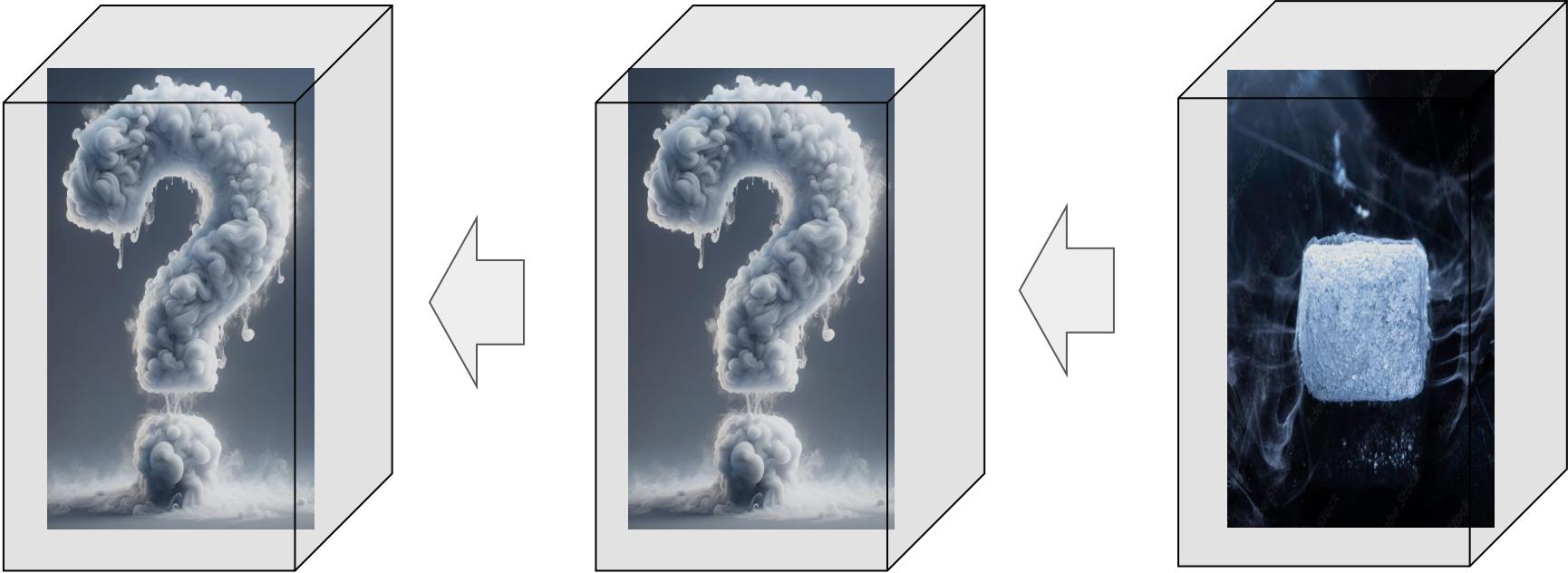


- U-Net은 ResNet blocks, Skip connection, self-attention layers로 구성
- Time representation은 sinusoidal positional embedding 또는 random Fourier features 사용.
  - Time feature는 residual block으로 simple spatial addition으로 입력



드라이아이스 조각상의 입자는 기화되어 밀폐된 공간으로 무작위적으로 확산됨.

이 과정에서 입자들이 이뤘던 조각상의 구조와 패턴은 파괴됨.

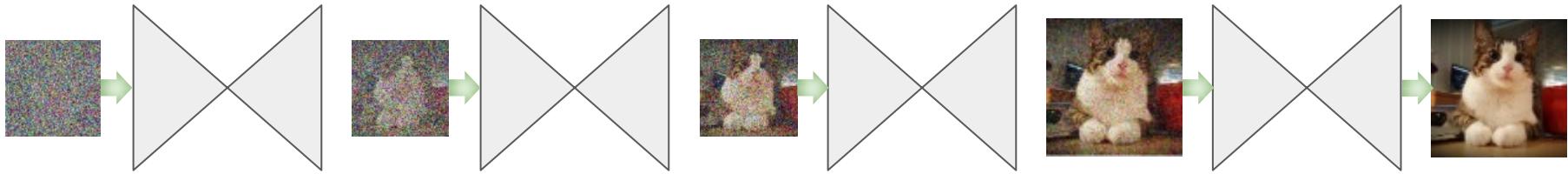


특정 시점에서 모델로 1초전 모든 입자의 위치를 예측해보자.

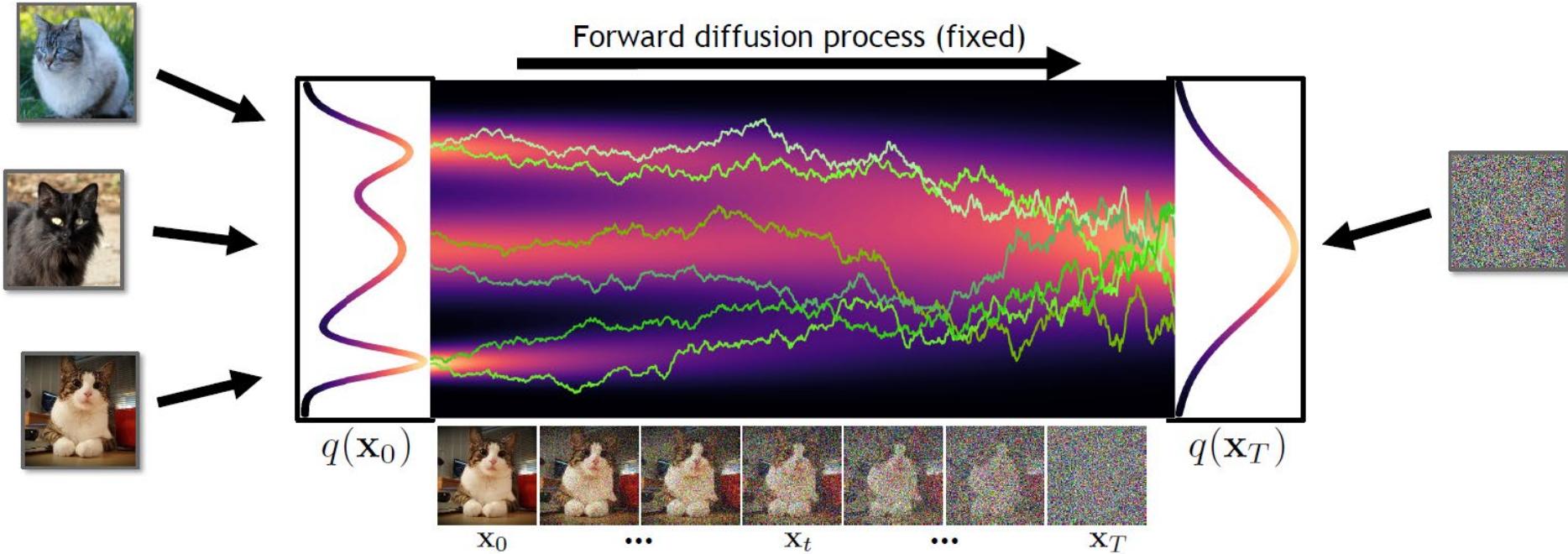
그 과정은 입자들이 이루는 구조와 패턴을 복원하는 과정.

이 과정을 반복하면 처음의 조각상이 다시 나올까?

# U-Net learns the reverse diffusion process



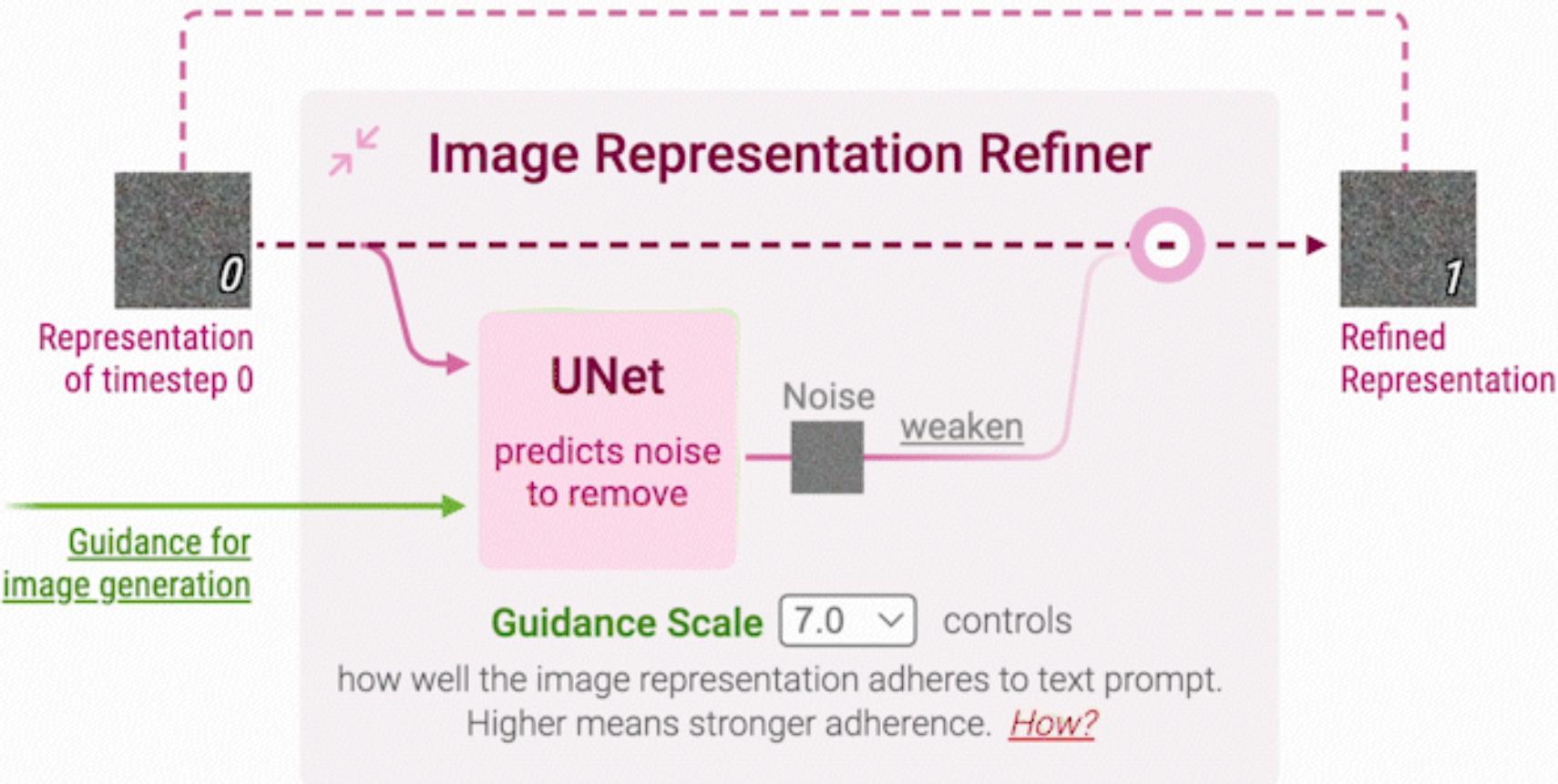
- 모델은 대규모 데이터에서 T시점에서 T-1시점으로의 reverse diffusion process를 학습
  - 학습데이터  $x_t$ 와  $x_{t-1}$ 는 forward process로 생성
- DM 신경망이 대규모 데이터에서 학습됐다면 완전히 무작위적인 노이즈에서 이미지의 구조를 복원하는 방법을 배운다.
  - 이미지 생성 단계에서는 Reverse denoising process만 연산
  - Sohl-Dickstein et al이 즉각적으로 노이즈를 제거하는 방식은 tractable하지 않고, 결과의 품질이 낮다는 것을 보고함. 따라서 한 번에 영상을 복원하는 방법은 효과적이지 않음.



$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}, \tau \sim p_\tau, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\|\mathbf{y} - \mathbf{f}_\theta(\mathbf{x}_\tau; \mathbf{c}, \tau)\|_2^2]$$

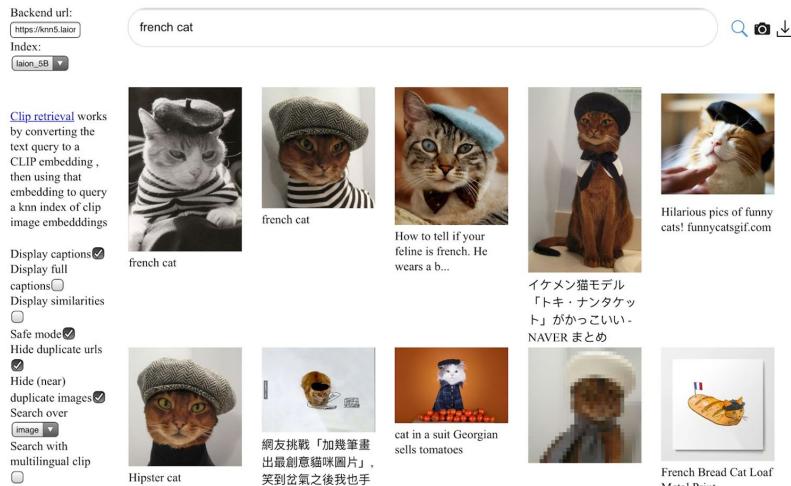
A denoiser model  $f_\theta$  (parameterized with learnable parameters  $\theta$ ) receives the diffused  $x_\tau$  as input

where  $c$  is optional conditioning information, such as a text prompt, and the target vector  $y$

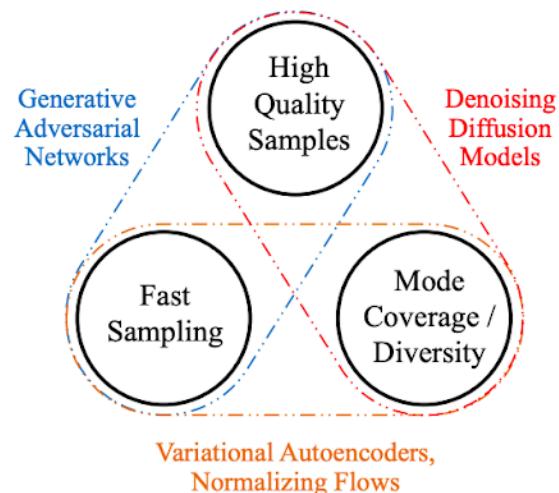


# Diffusion models are too expensive

- 확산 모델의 단점은 훈련을 위해 많은 데이터와 계산이 필요
- 이미지의 모든 픽셀에 대해 작업을 진행하고, 이 픽셀들에 대해 많은 처리 단계를 거치기 때문에, 계산량이 급격히 증가



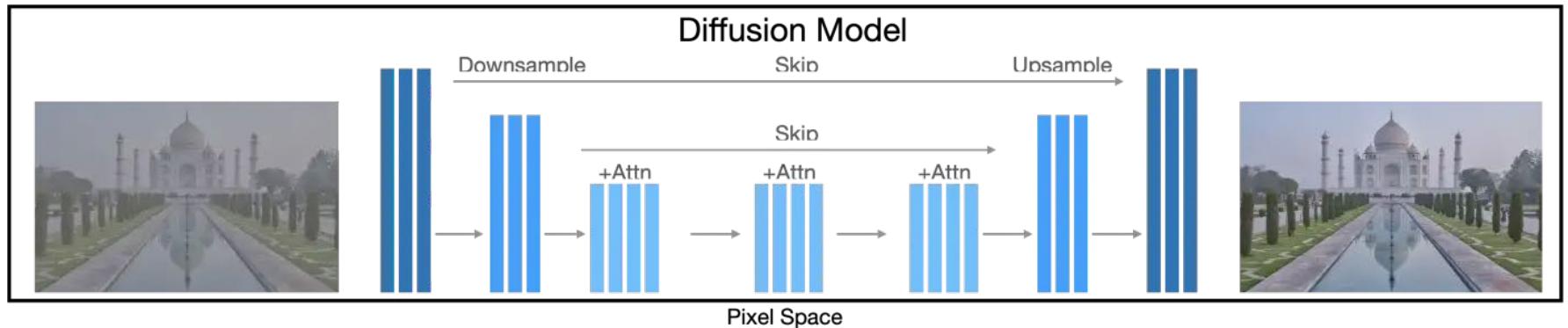
<LAION-5B data : 5.85 billions image-text pairs>



<Generative learning trilemma>

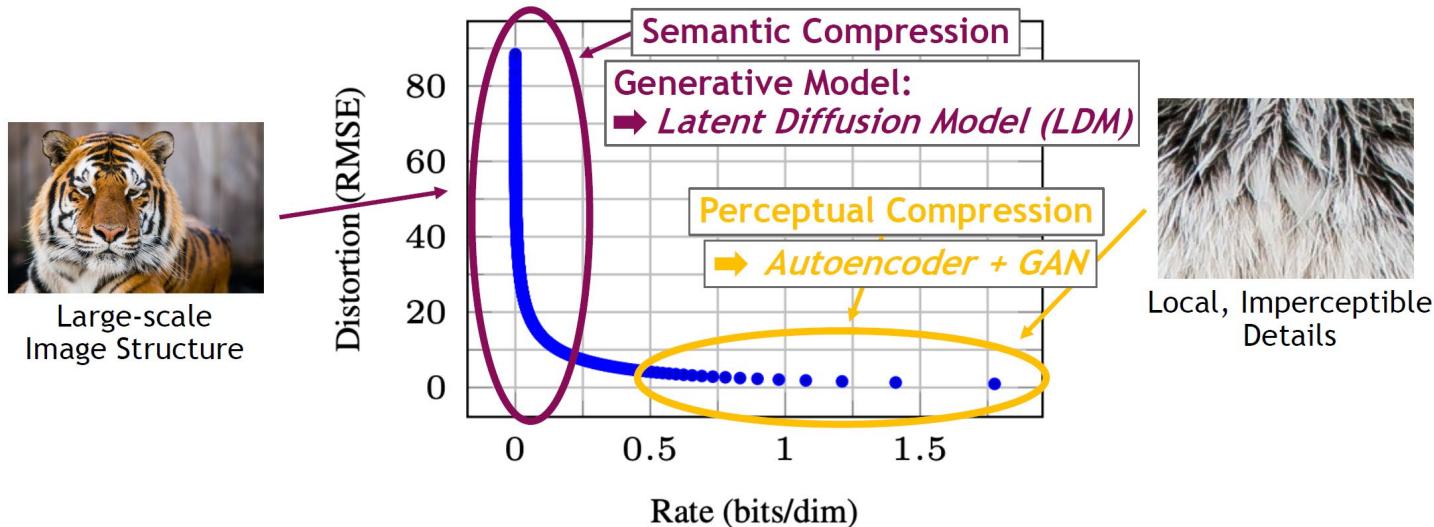
# The diffusion model learns in pixel space

확산 모델에서는 모든 잡음 추정은 이미지의 노이즈 제거에 기여한다. 문제는 확산 모델이 픽셀 공간에서 학습한다는 것이다.



<The conventional Diffusion model architecture>

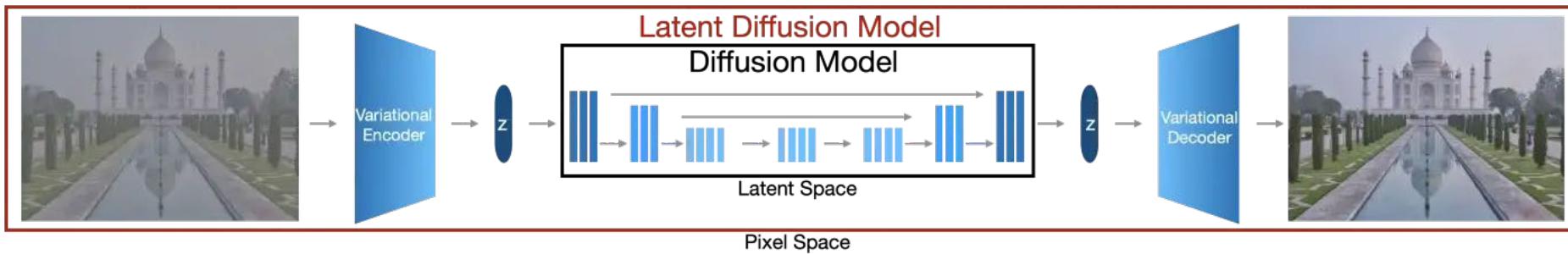
# The diffusion model learns in pixel space



- Diffusion model이 생성하는 디지털 이미지의 대부분의 비트는 인지할 수 없는 세부적 특징표현
- 지각적 정보를 따로 압축한다면, 확산 모델을 효율적으로 제어할 수 있지 않을까?

# The diffusion model learns in latent space

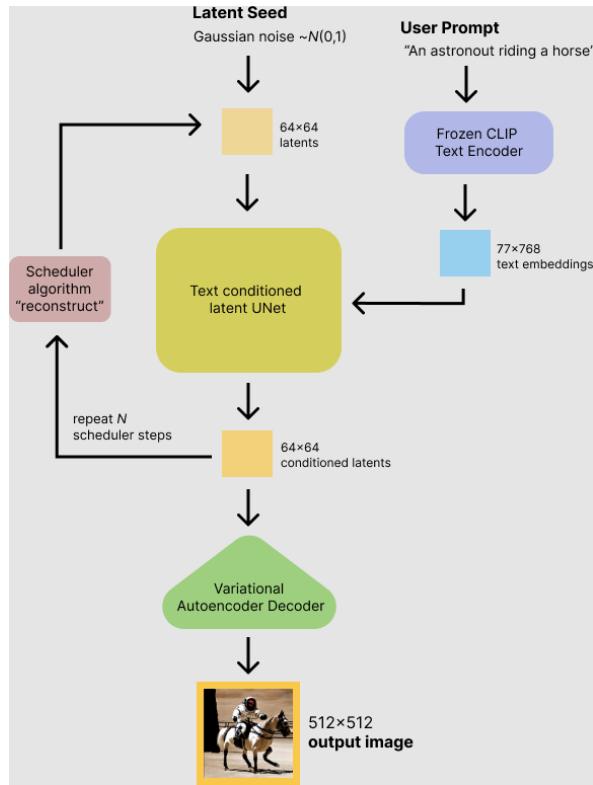
Latent diffusion model은 이미지를 VQ-VAE 모델로 잠재 공간으로 변환한 후, 잠재 변수에 대해 확산을 수행한다. 따라서 U-net은 고품질 픽셀 이미지의 잠재 변수에서 잡음을 제거하는 데 적합한 매개변수를 학습한다.



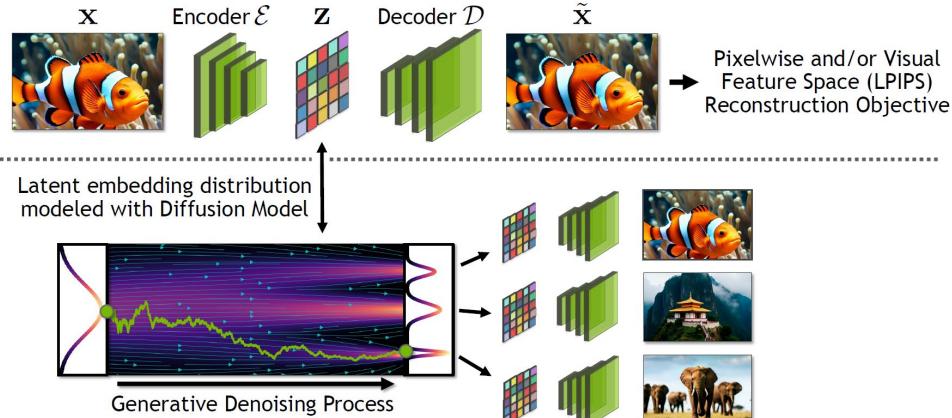
<Latent diffusion model proposed by authors>

$$v_* = \arg \min_v \mathbb{E}_{z \sim \mathcal{E}(x), y, \epsilon \sim \mathcal{N}(0,1), t} \left[ \|\epsilon - \epsilon_\theta(z_t, t, c_\theta(y))\|_2^2 \right]$$

# Latent Diffusion Process

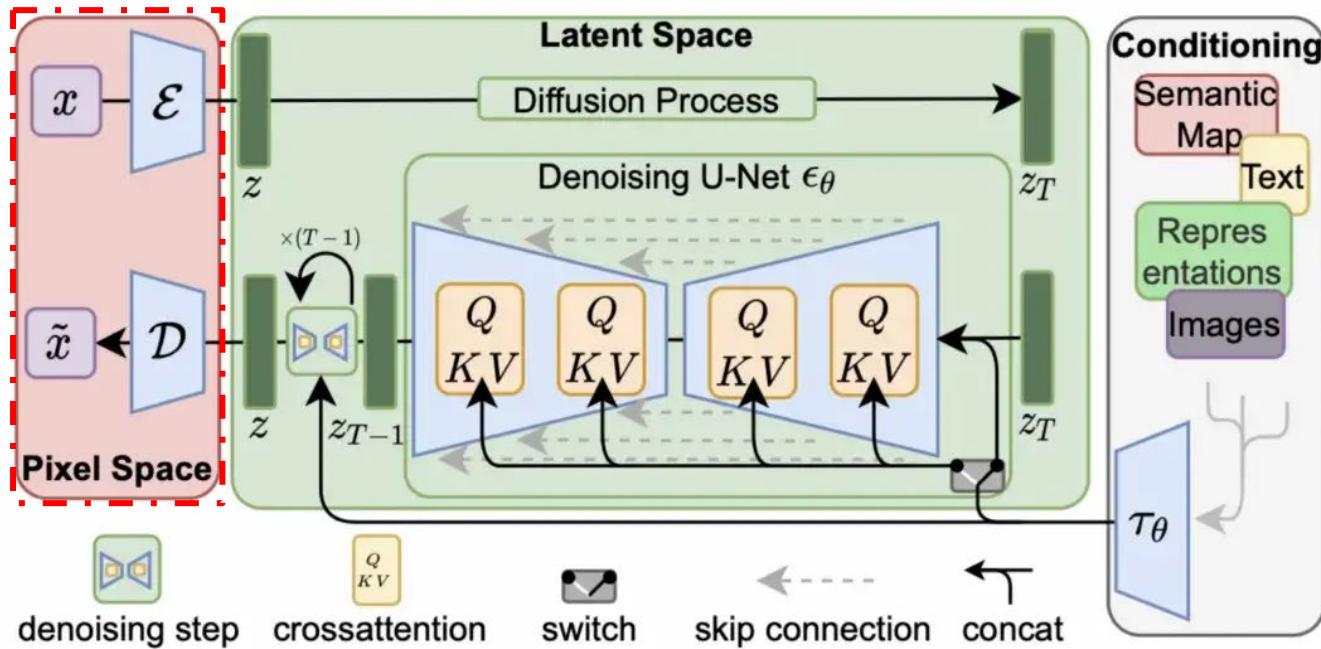


- Stage 1:  
Train Autoencoder  
 $\tilde{\mathbf{x}} = \mathcal{D}(\mathcal{E}(\mathbf{x}))$
- Stage 2:  
Train **Latent** Diffusion Model



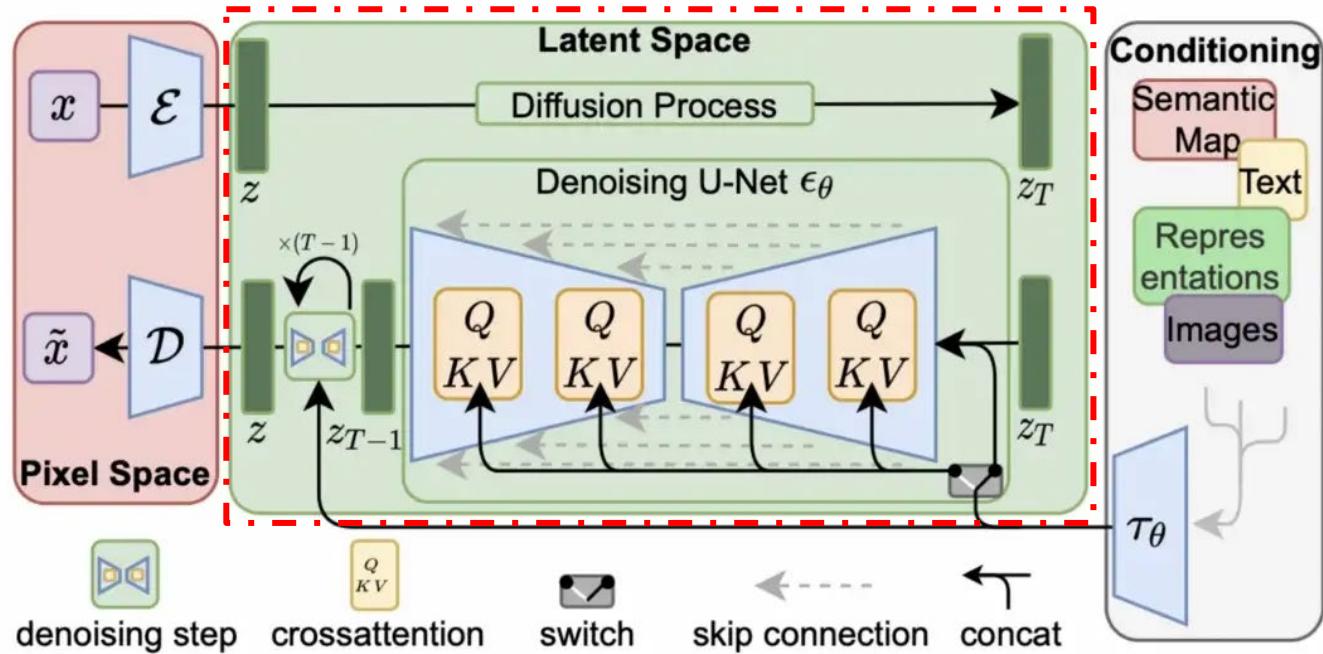
# LDM architecture

첫 번째 단계는 perceptual compression 단계로, 이미지에서 불필요하게 미세한 부분을 줄이면서도, 그 의미를 유지하는 데 중점을 둔다.



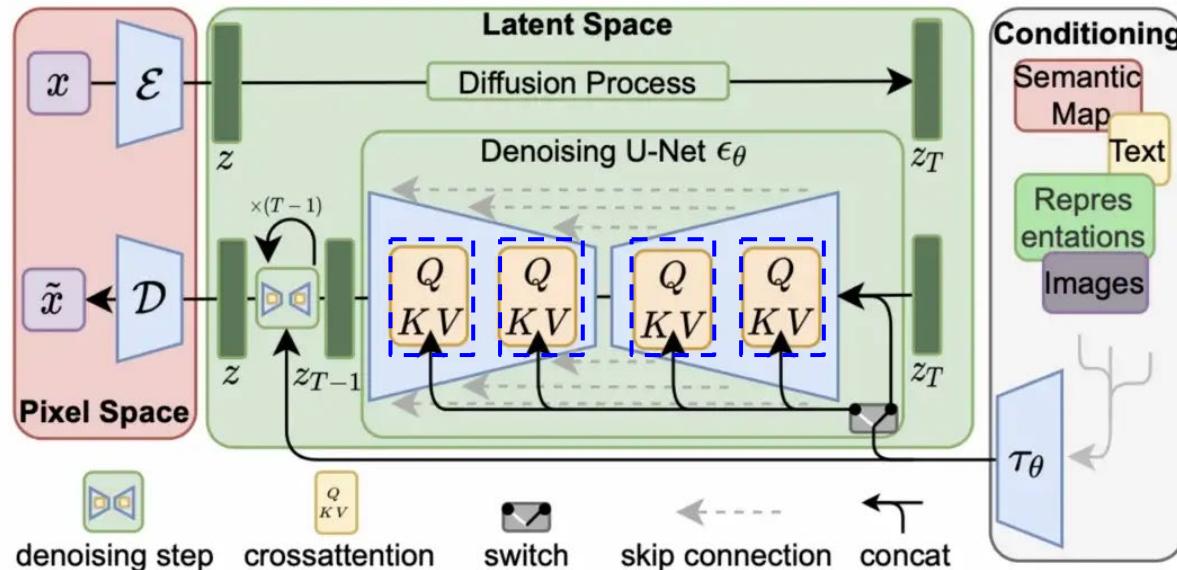
# LDM architecture

두 번째 단계에서는 실제 생성 모델이 의미적 압축을 학습한다. 이 단계에서 이미지의 더 깊은 의미와 개념적 구조에 중점을 두어, 단순한 시각적 특성을 넘는 학습이 이루어진다.



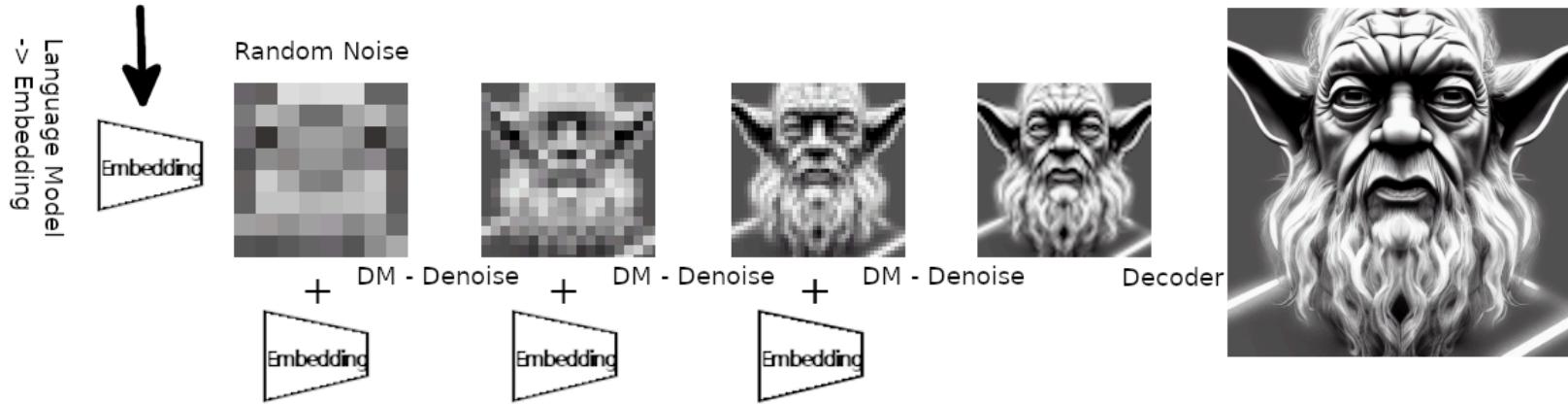
# Cross Attention in U-Net

text-guided latent diffusion model에서 U-Net은 여러 denoising step 동안 픽셀과 텍스트 사이의 상호작용을 통해 이미지의 구조와 내용을 생성



# Text-guided diffusion model

"A person half Yoda half Gandalf"



- Text-guided diffusion model은 생성할 이미지의 종류를 사용자 제공 텍스트로 제어하는 모델
  - 생성할 이미지를 묘사하는 텍스트인 프롬프트로 원하는 이미지를 생성할 수 있음.
- Diffusion model은 denoising process 동안 프롬프트와 연관된 이미지를 점차적으로 생성함.
- 프롬프트는 생성되는 이미지를 의미론적 수준에서 제어하는 언어 수준 인터페이스.







<https://stability.ai/news/stable-diffusion-sdxl-1-announcement>

# Reference

---

## <Comprehensive tutorials for diffusion models>

- CVPR 2022 Tutorial: Denoising Diffusion-based Generative Modeling: Foundations and Applications
- Weng, Lilian. (Jul 2021). What are diffusion models? Lil'Log. <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>.
- Diffusion models with math explained : <https://youtu.be/HoKDTa5jHvg>
- <https://medium.com/mlearning-ai/enter-the-world-of-diffusion-models-4485fb5c5986>
- [Align your latents explained in Youtube](#)

## <Main papers for important diffusion models>

- Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." International Conference on Machine Learning. PMLR, 2015.
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems 33 (2020): 6840-6851.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In NeurIPS, 2020.
- Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022.
- Radford, Alec, et al. "Learning transferable visual models from natural language supervision." International Conference on Machine Learning. PMLR, 2021.
- Razavi, Ali, Aaron van den Oord, and Oriol Vinyals. "Generating diverse high-resolution images with VQ-VAE." (2019).
- Blattmann, Andreas, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. "Align Your Latents: High-Resolution Video Synthesis With Latent Diffusion Models," 22563–75, 2023.
- Blattmann et. al, Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets, 2023

# Thank you!