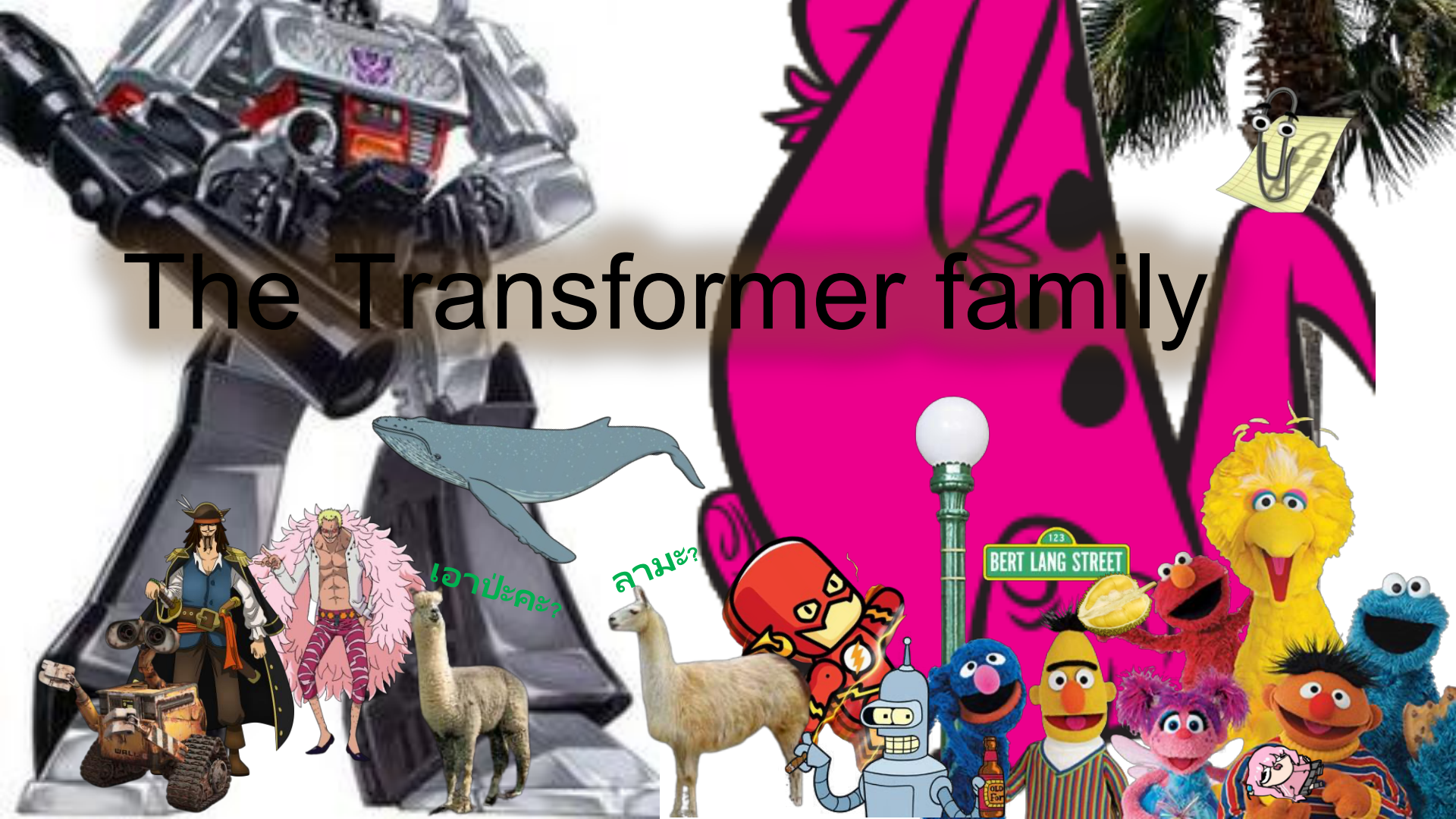
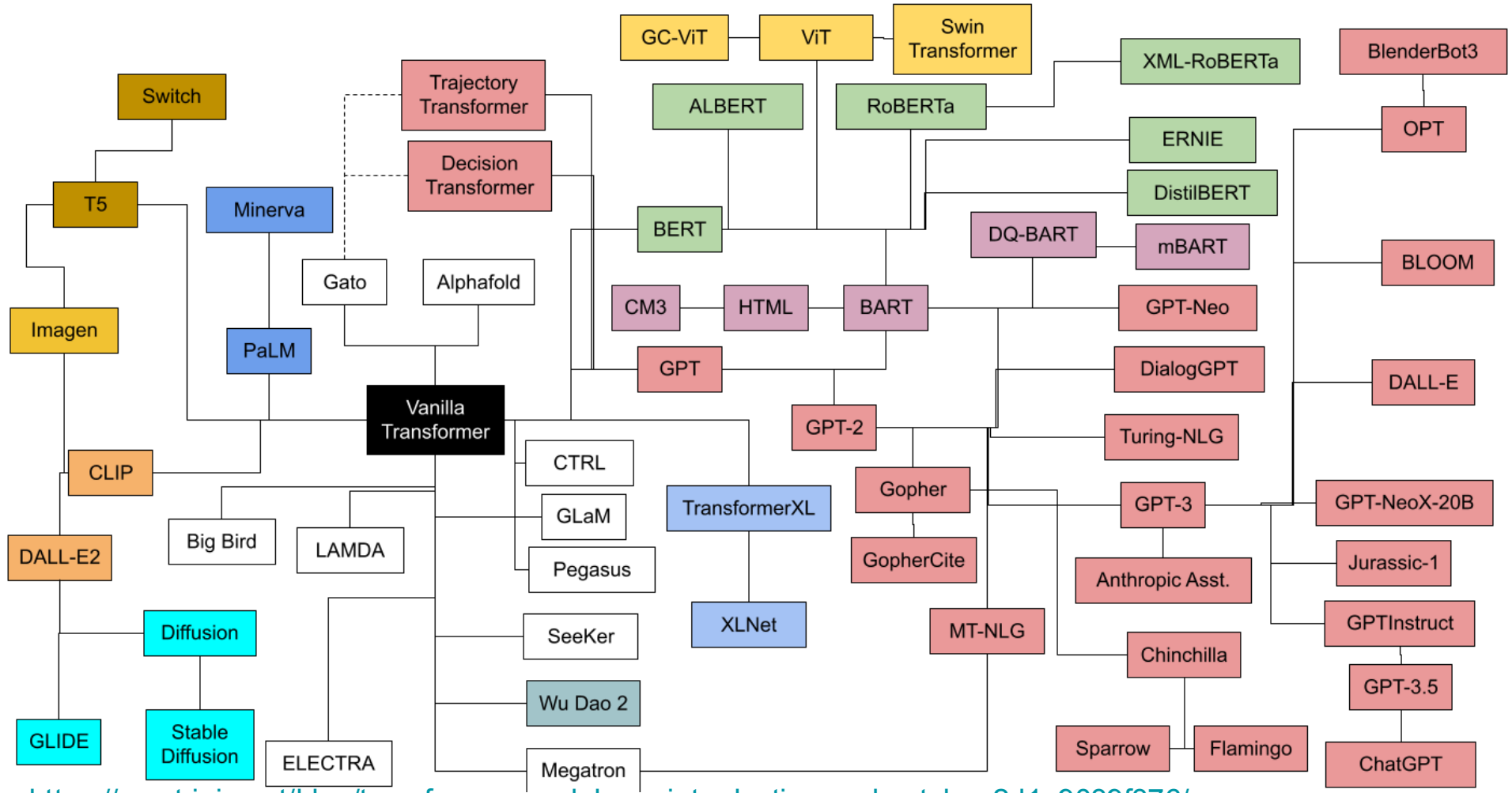
The Transformer family





https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

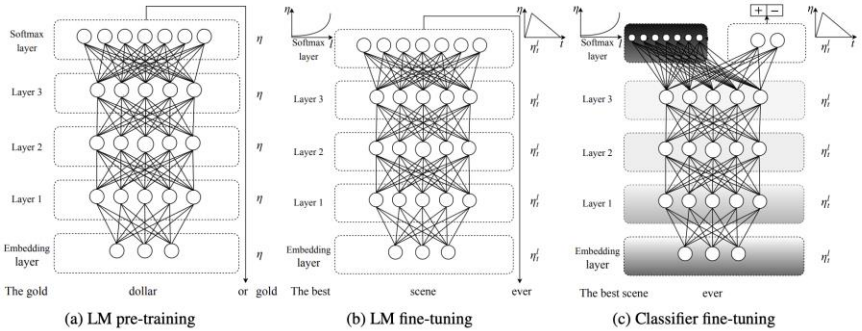
Pretrained models

- With word2vec (2013), you can pretrained models based on large data
- The type of pretraining gets more complicated overtime
    - Word level - Glove (2014)
    - Sentence level - ULMFit (2018)

Predicting words (Language modeling)

This means we can just use sentences with no supervised labels.

How to pre-train?



Pre-trained transformer types (by training method)

Encoder only (autoencoder)

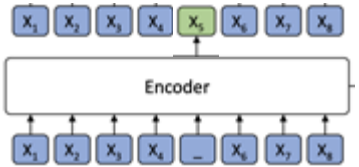
- BERT, ALBERT, RoBERTa
- Seq classification, token classification
- Masked words and predict

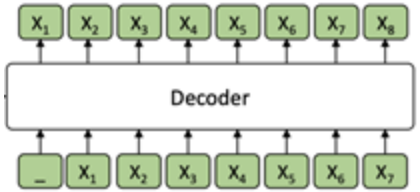
Encoder-decoder (seq2seq)

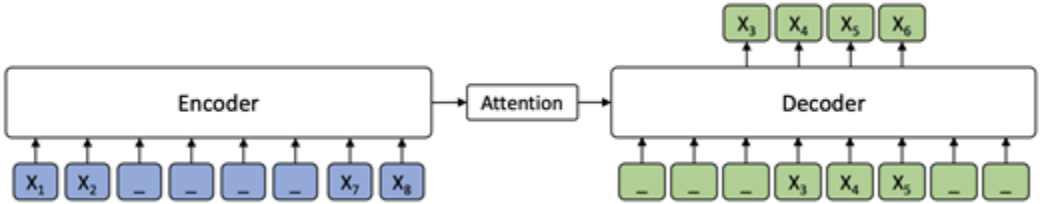
- MASS, BART, T5
- Machine translation, text summary
- Mask phrases

Decoder only (autoregressive)

- GPT, Llama
- Text generation
- Predict next word

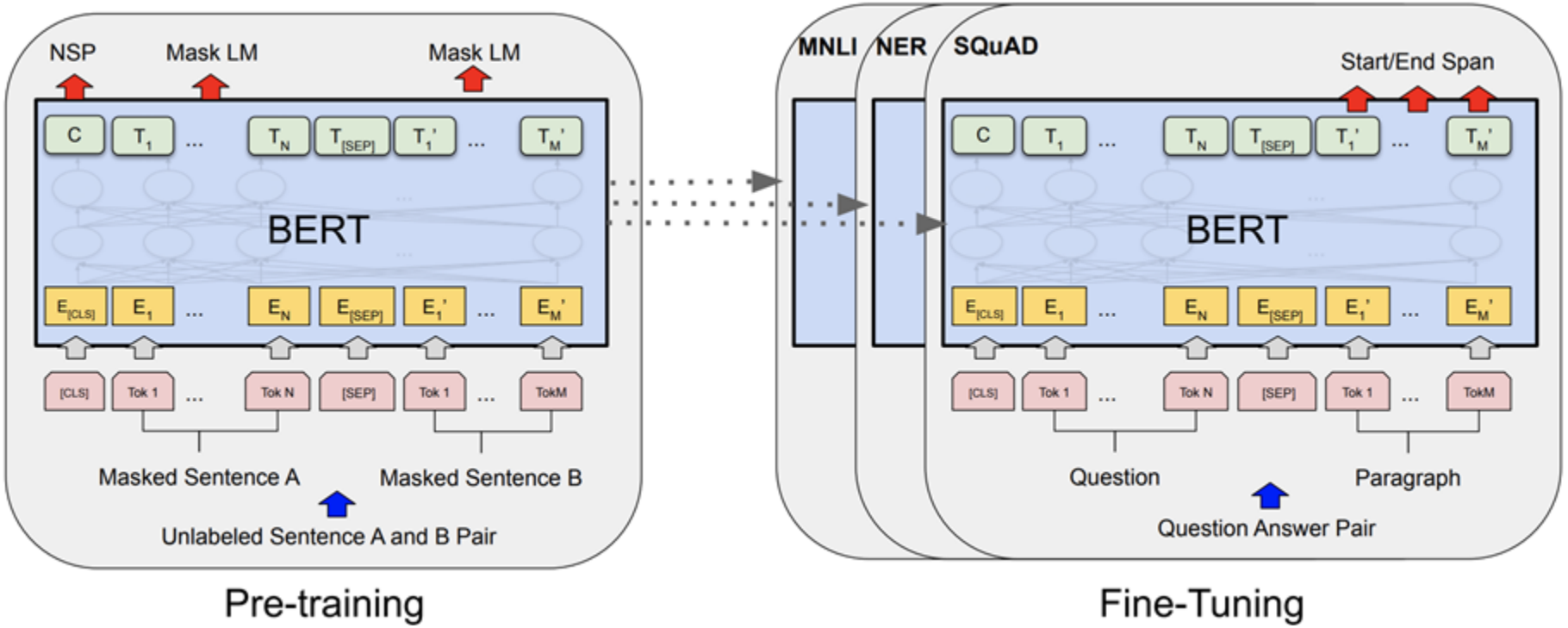






https://huggingface.co/transformers/summary.html https://arxiv.org/pdf/1905.02450.pdf

BERT - Bidirectional Encoder Representations from Transformers



https://arxiv.org/abs/1810.04805

BERT Pretraining

- Full transformer built on wordpiece tokens.
- Masked LM to learn bi-directional LM (has access to future words)
- Next sentence prediction to learn discourse

Masking

- 15% of the token in the training data is selected.
  - 80% becomes [MASK] token
  - 10% becomes a random token
  - 10% left as is

Prevents training/inferencing mismatch.
(No [MASK] in regular sentences).

This is a MASK that I draw with.

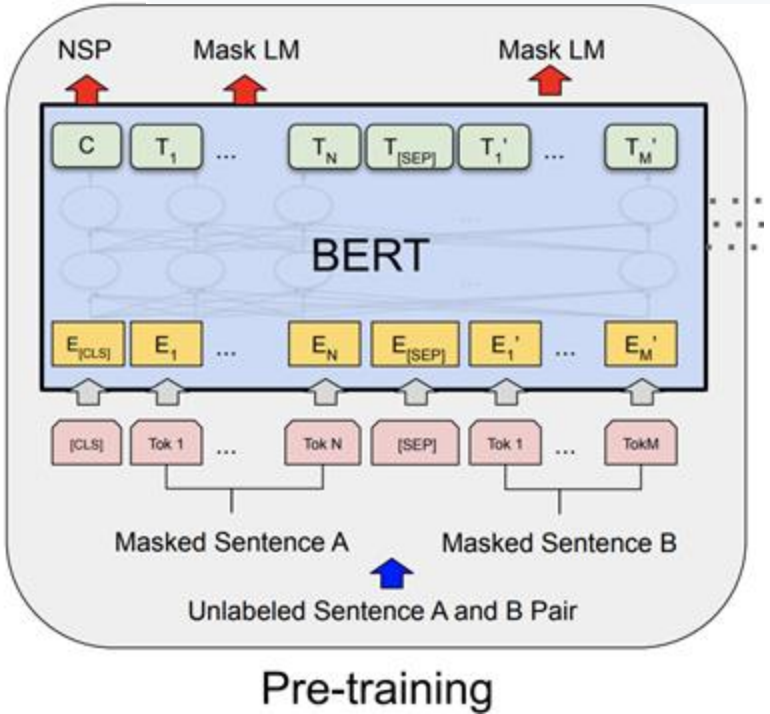
Next Sentence Prediction

```
Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence
```

```
Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence
```

- To help the model learn discourse
- Predict whether
  - two sentences from a paragraph
  - two sentences randomly selected

- The [CLS] token is the position responsible for this prediction.
- [CLS] token is often used as a summary embedding for the sentence.



Pre-training

BERT use cases

- Being an encoder BERT does classification tasks very well
  - Sequence prediction tasks: sentiment analysis, topic classification, spam detection, etc.
  - Token prediction tasks: Name entity prediction, Part of speech tagging, spelling correction
  - Natural language inference (NLI)

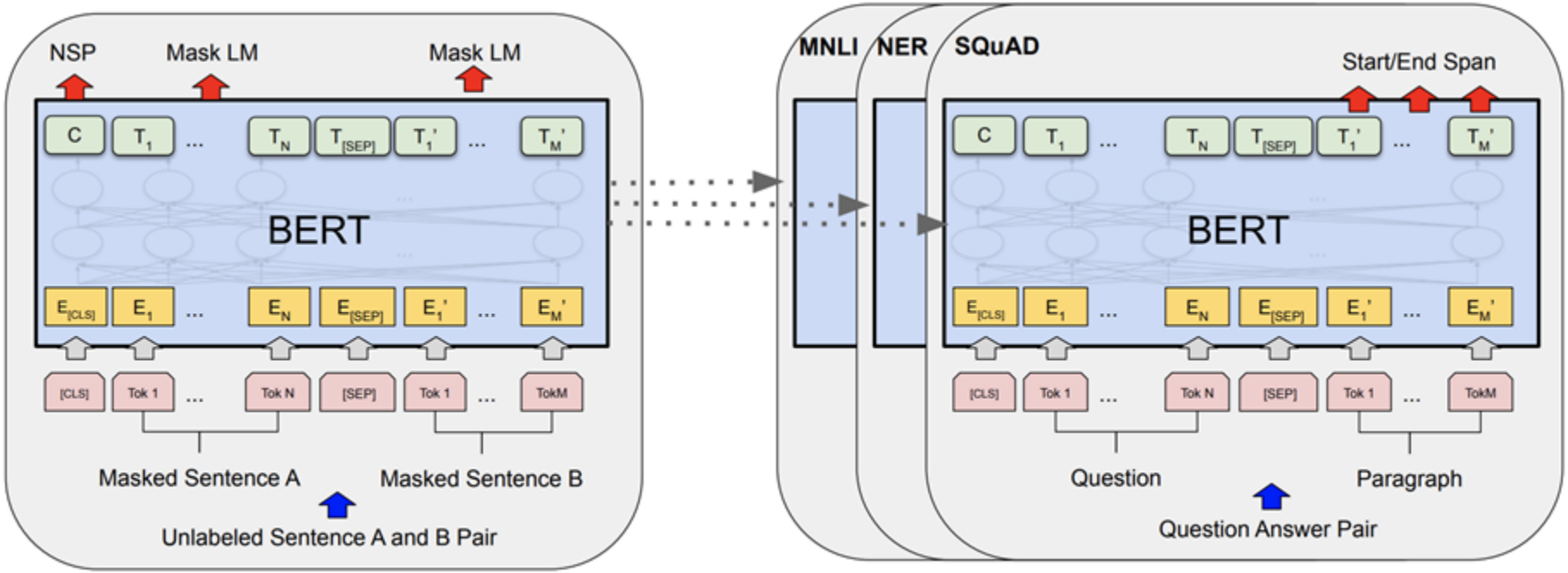
A: The boy is running through the grassy area

B: The boy is in his room.
Ans: Contradiction

B: The boy is outside.
Ans: Entailment

B: The boy is in a park.
Ans: Neutral

BERT use cases



Pre-training

Unsupervised!

Fine-Tuning

Supervised!

Roberta (Robustly optimized BERT approach)

A trick and tuning study based on BERT

Dynamic masking > static

Next sentence prediction is not optimal

Larger batch + higher learning rate

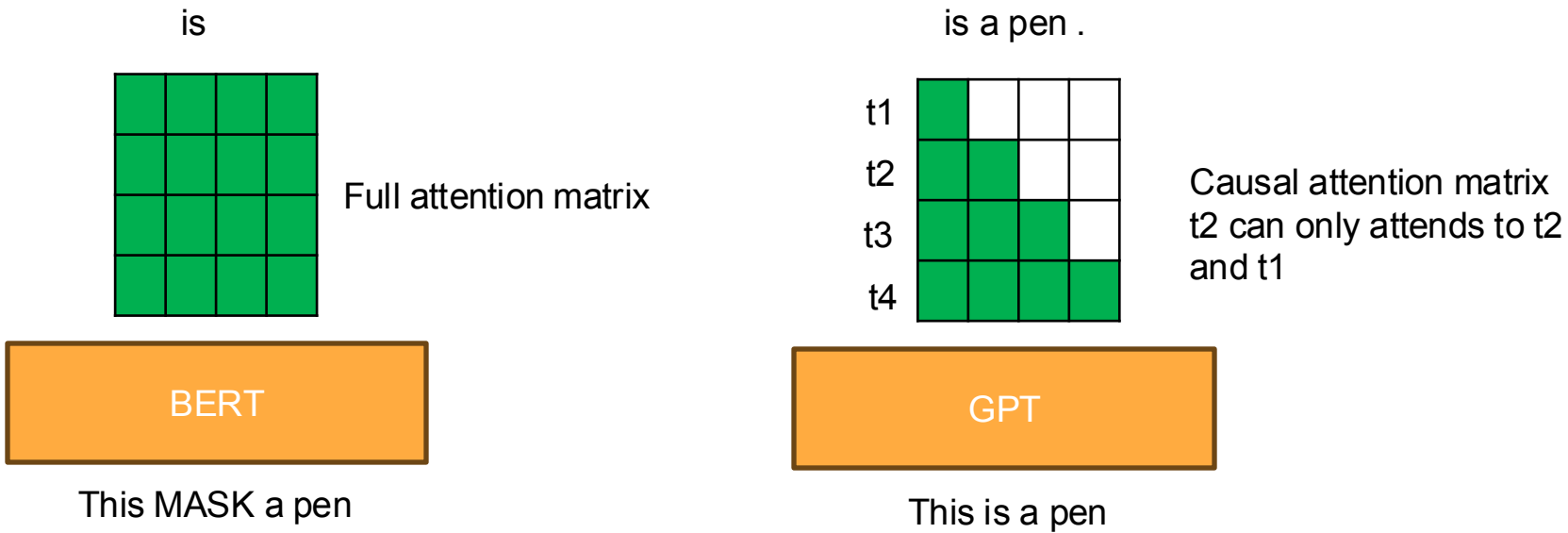
Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

RoBERTa: A Robustly Optimized BERT Pretraining Approach
Used in WangchanBERTa

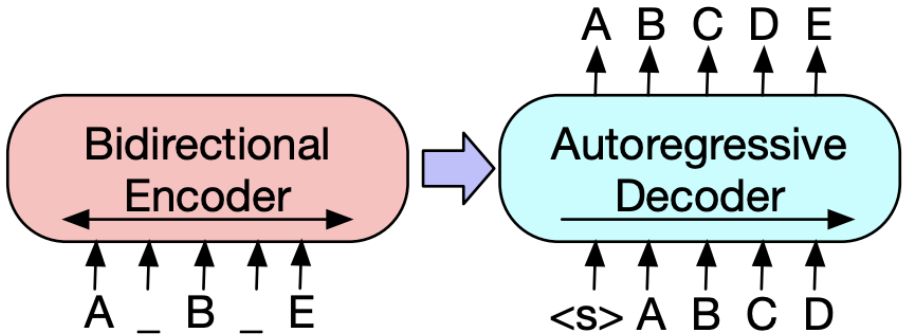
Decoder only pre-training

- GPT (Generative Pre-trained Transformer) pre-trains by predicting next word
- Enforces causal attention (mask attention values on future tokens
- Trained using teacher forcing. Inference in an auto-regressive manner – **can generate a sentence!**



Encoder-Decoder

- BART (Bidirectional and Auto-Regressive Transformers) combines BERT masking and GPT autoregressive characteristics
- Mask spans of words into a single [MASK].
    - The model has to expand the mask into different amount of words in an autoregressive manner
- Not used much after people figuring out that GPT-like models is good enough for generation while simpler



https://arxiv.org/abs/1910.13461

Transformers and scaling

Scaling law in language models

More data more params more compute leads to better models

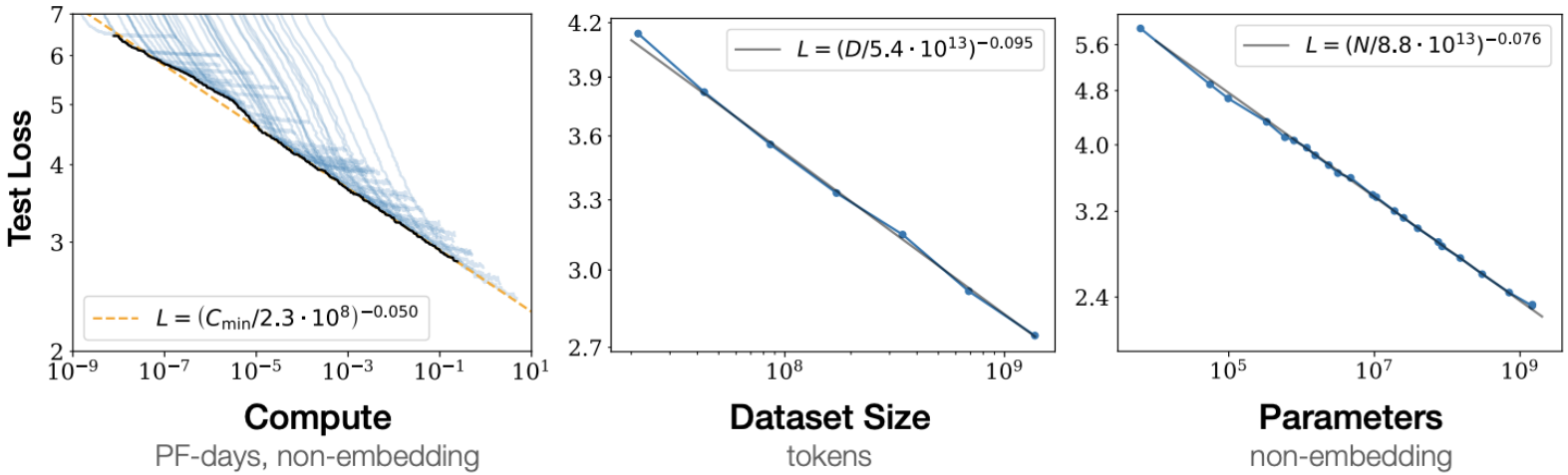


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Scaling Laws for Neural Language Models https://arxiv.org/pdf/2001.08361.pdf

Scaling law in language models

Training usually stops early for efficiency reasons

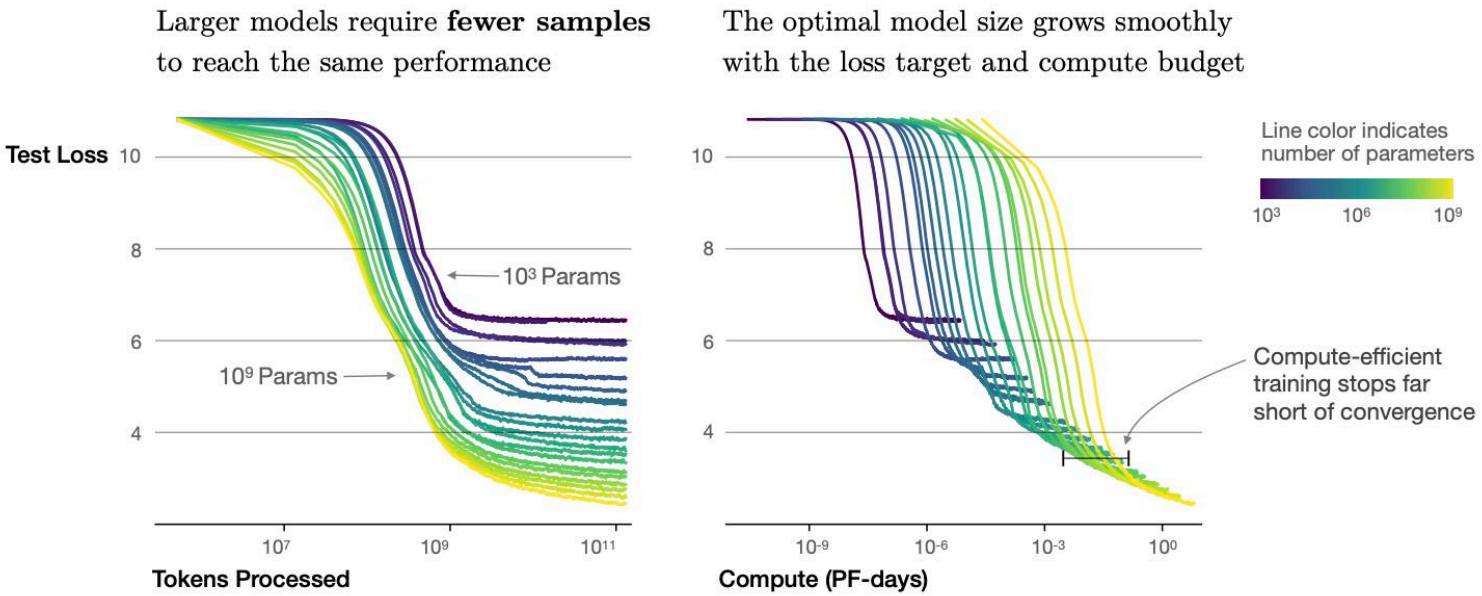


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling Laws for Neural Language Models https://arxiv.org/pdf/2001.08361.pdf

Scaling law in language models

Longer training can be beneficial

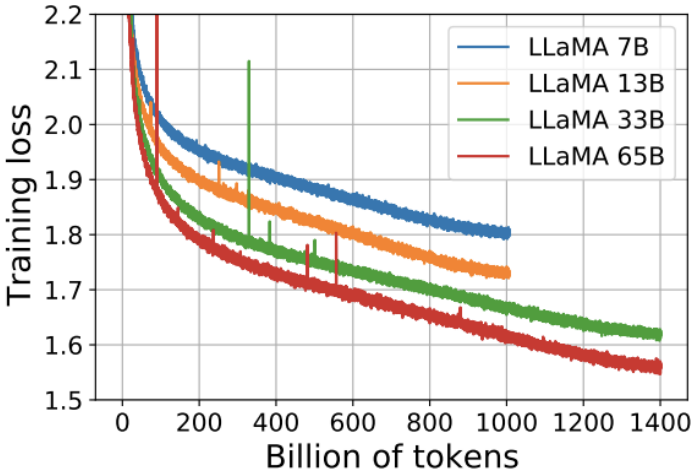


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

LLaMA: Open and Efficient Foundation Language Models https://arxiv.org/abs/2302.13971

Scaling law in language models

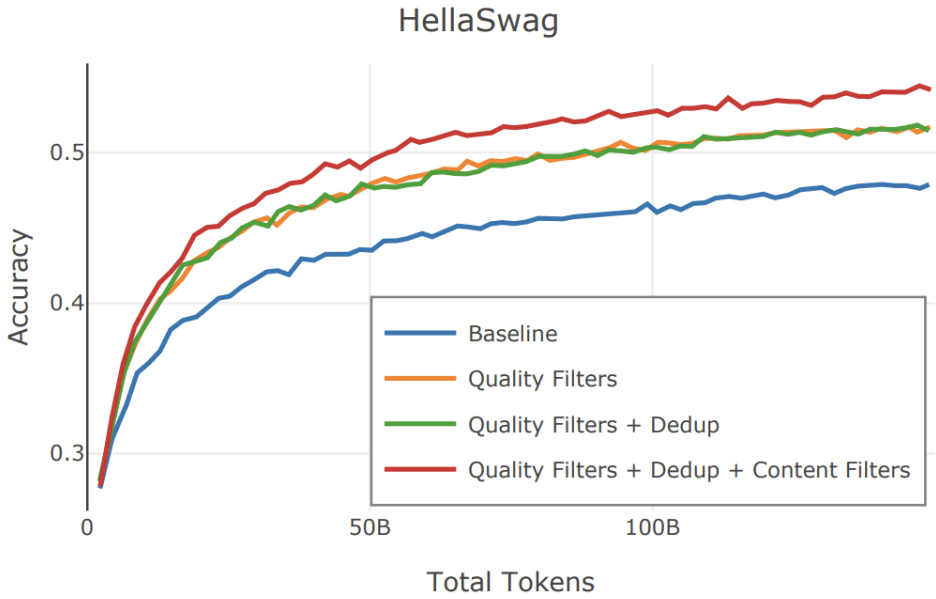
Besides data size, data quality matters

	Dataset Size	Pile (val) (BPB)	Pile (test) (BPB)	WikiText (PPL)	LAMBADA (PPL)	LAMBADA (ACC)
The Pile	825 GiB	**0.9281**	**0.9433**	**5.59**	12.78	**50.1**
CC-100 (en)	300 GiB	1.3143	1.3293	8.27	**11.78**	49.7
Raw CC	45927 GiB†	1.1180	1.1275	11.75	19.84	43.8

Table 3: Size-controlled evaluation results. Each dataset is deduplicated against all evaluation metrics and subsampled to approximately 40GB to control for the effects of dataset size. For LAMBADA, we use the variant of the data introduced in Radford et al. (2019) and only evaluate the perplexity on the final token rather than the final word. For WikiText, we report the perplexity per GPT-2 token. † indicates that the size is an estimate.

The Pile: An 800GB Dataset of Diverse Text for Language Modeling https://arxiv.org/pdf/2101.00027.pdf

Scaling law in language models



Besides data size, data quality matters

Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research
https://arxiv.org/abs/2402.00159

WangchanBERTa:

Pretraining Transformer-based Thai Language Models

Slides courtesy of Lalita Lowphansirikul and Charin Polpanumas

Hardware; 4 K80s vs 8 V100s vs 1,024 v100s

Better hardware means you have more room for iterations. We can iterate with smaller datasets but that sometimes defeat the purpose of training a LARGE language model.

Model	# of GPUS	Dataset Size	Effective Batch Size	Steps	Days Spent
ThaiKeras BERT-th	4 K80s	0.5GB	32	1M	20
WangchanBERTa	8 V100s	78GB	4,092	500k	134
RoBERTa	1,024 V100s	160GB	8,000	500k	1
XLM-RoBERTa	500 V100s	2.5TB	8,192	1.5M	NA

Data Volume and Diversity

- Thai Wikipedia is over 100x smaller than Thai texts used to train XLM-RoBERTa (71.7GB) and over 300x smaller than texts used to train the original RoBERTa.
- Wikipedia also only include formal texts from encyclopedia.

Model	# of GPUS	Dataset Size	Effective Batch Size	Steps	Days Spent
ThaiKeras BERT-th	4 K80s	0.5GB	32	1M	20
WangchanBERTa	8 V100s	78GB	4,092	500k	134
RoBERTa	1,024 V100s	160GB	8,000	500k	1
XLM-RoBERTa	500 V100s	2.5TB	8,192	1.5M	NA

Short Sequence Length

Text; Sequence length=128; WangchanBERTa SentencePiece tokenizer

<s> hamtaro หรือ แฮมทาโร่ แก๊งจิ๋วผจญภัย การ์ตูนญี่ปุ่นที่มีเหล่าแฮมสเตอร์เป็นตัวละครหลัก เป็นผลงานของอาจารย์ kawai ritsuko เดิมที แฮมทาโร่ นั้นเป็นนิทานสำหรับเด็กมาก่อนถูกตีพิมพ์ที่ญี่ปุ่นในปี ค.ศ. 1997 เพราะกองบรรณาธิการของนิตยสารการ์ตูนอยากได้การ์ตูนที่มีตัวเอกเป็นแฮมสเตอร์ และอาจารย์ก็กำลังเลี้ยงแฮมสเตอร์อยู่พอดี ไม่แปลกใจเลย ทำไมอาจารย์ถึงวาดการ์ตูนและถึงเล่าถึงกิจวัตรประจำวันของเหล่าแฮมสเตอร์ได้ออกมาสมจริงและน่ารักสุดๆ หนังสือนิทาน hamtaro ได้รับความนิยมมากจนกลายมาเป็น ทีวี อนิเมะ ในหน้าร้อนของปี ค.ศ. 2000 เป็นที่นิยมทั้งเด็กทุกวัยไป

[illegible] ทั้งประเทศไทย

Model	# of GPUS	Dataset Size	Effective Batch Size	Sequence Length
ThaiKeras BERT-th	4 K80s	0.5GB	32	128
WangchanBERTa	8 V100s	78GB	4,092	416
RoBERTa	1,024 V100s	160GB	8,000	512
XLM-RoBERTa	500 V100s	2.5TB	8,192	512

Tokenization; Most Subword Tokenizers Are Domain Dependent

Even same SentencePiece tokenizers might get different results with different training set. Moreover, WordPiece tokenizer tokenizes too small subwords; we will see in later sections how this leads to a challenge in question answering task.

Text: ศิลปะไม่เป็นเจ้านายใคร และไม่เป็นขี้ข้าใคร

WangchanBERTa (spm): ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้านาย', 'ใคร', 'และ', 'ไม่เป็น', 'ขี้ข้า', 'ใคร', '</s>']

WangchanBERTa-processed (spm): ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้านาย', 'ใคร', '<_>', 'และ', 'ไม่เป็น', 'ขี้ข้า', 'ใคร', '</s>']

XLM-RoBERTa (spm): ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้า', 'นาย', 'ใคร', 'และ', 'ไม่เป็น', 'ขี้', 'ข้า', 'ใคร', '</s>']

MBERT (WordPiece): ['[C L S]', 'ศ', '# # ิ ล', '# # ป', '# # ะ', '# # ไ', '# # ม', '# # ่', '# # เ ป ็ น', '# # เ', '# # จ', '# # ้ า', '# # น า', '# # ย', '# # ใ', '# # ค ร', 'แ ล ะ', '# # ไ', '# # ม', '# # ่', '# # เ ป ็ น', '# # ข', '# # ี', '# # ้', '# # ข', '# # ้ า', '# # ใ', '# # ค ร', '[S E P]']

Space Tokens as Important Boundaries

SentencePiece will create tokens where a space token is merged another non-space token.

Text: ศิลปะไม่เป็นเจ้านายใคร และไม่เป็นขี้ข้าใคร

WangchanBERTa: ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้านาย', 'ใคร', 'และ', 'ไม่เป็น', 'ขี้ข้า', 'ใคร', '</s>']

WangchanBERTa-processed: ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้านาย', 'ใคร', '<_>', 'และ', 'ไม่เป็น', 'ขี้ข้า', 'ใคร', '</s>']

XLM-RoBERTa: ['<s>', '', 'ศิลปะ', 'ไม่เป็น', 'เจ้า', 'นาย', 'ใคร', 'และ', 'ไม่เป็น', 'ขี้', 'ข้า', 'ใคร', '</s>']

mBERT: ['[C L S]', 'ศ', '# # ิ ล', '# # ป', '# # ะ', '# # ไ', '# # ม', '# # ่', '# # เ ป ็ น', '# # เ', '# # จ', '# # ้ า', '# # น า', '# # ย', '# # ใ', '# # ค ร', 'แ ล ะ', '# # ไ', '# # ม', '# # ่', '# # เ ป ็ น', '# # ข', '# # ี', '# # ้', '# # ข', '# # ้ า', '# # ใ', '# # ค ร', '[S E P]']

Token size

Eyeballing for vocab size of SentencePiece

5k

วิชา|ที่|อาจารย์|อร|รถ|พล|สอน|คือ| |ศาสตร์|ที่|นำ|ทฤษฎี|ทาง|ภาษา|ศาสตร์|มา|รวม|กับ|เทคโนโลยี|ต่างๆ| |เป็น|การศึกษา|ที่ใช้|การ|ผสม|ผ|สาน|ระหว่าง|วิทยา|การ|ค|อมพิวเตอร์|และ|ทฤษฎี|ทาง|ภาษา|ศาสตร์

25k

วิชา|ที่|อาจารย์|อรรถ|พล|สอน|คือ| |ศาสตร์|ที่นำ|ทฤษฎี|ทาง|ภาษา|ศาสตร์|มา|รวมกับ|เทคโนโลยี|ต่างๆ| |เป็น|การศึกษา|ที่ใช้|การผสมผสาน|ระหว่าง|วิทยาการ|ค|อมพิวเตอร์|และ|ทฤษฎี|ทาง|ภาษา|ศาสตร์

32k

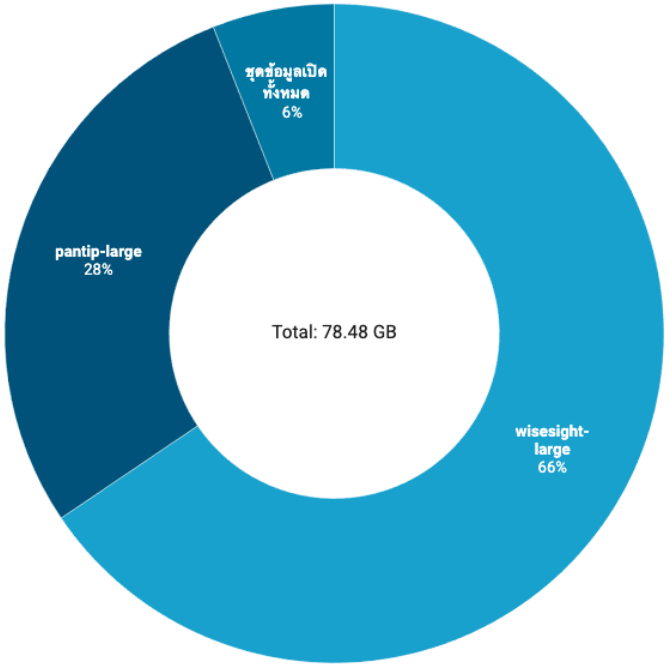
วิชา|ที่|อาจารย์|อรรถพล|สอน|คือ| |ศาสตร์|ที่นำ|ทฤษฎี|ทาง|ภาษาศาสตร์|มา|รวมกับ|เทคโนโลยี|ต่างๆ| |เป็น|การศึกษา|ที่ใช้|การผสมผสาน|ระหว่าง|วิทยาการ|ค|อมพิวเตอร์|และ|ทฤษฎี|ทาง|ภาษาศาสตร์

Assorted Thai Texts

Wisesight, Chaos Theory and Pantip to the rescue

Wisesight contributed 51.44GB of data from Twitter, Facebok, Pantip, Instagram, and YouTube in 2019.

Chaos Theory/Pantip contributed 22.35GB of Pantip data from 2015-2019.



More information https://www.youtube.com/watch?v=kXPMLX0vfYU&ab_channel=EkapolC

Transformer problems

- Embedding layer size.
- Attention matrix is NxN.
    - Problems in terms of compute and memory
    - Cannot scale to long sequences (limited context size)
- Tokens are fixed.

ALBERT 2019

Want higher hidden units without growing the model. Factorized embedding matrix

$$O(V \times H) \text{ to } O(V \times E + E \times H)$$

Share attention layer parameters across layers. More stable training as a side effect.

V = Vocab, H = Hidden, E = Lower Dimensional Embedding space

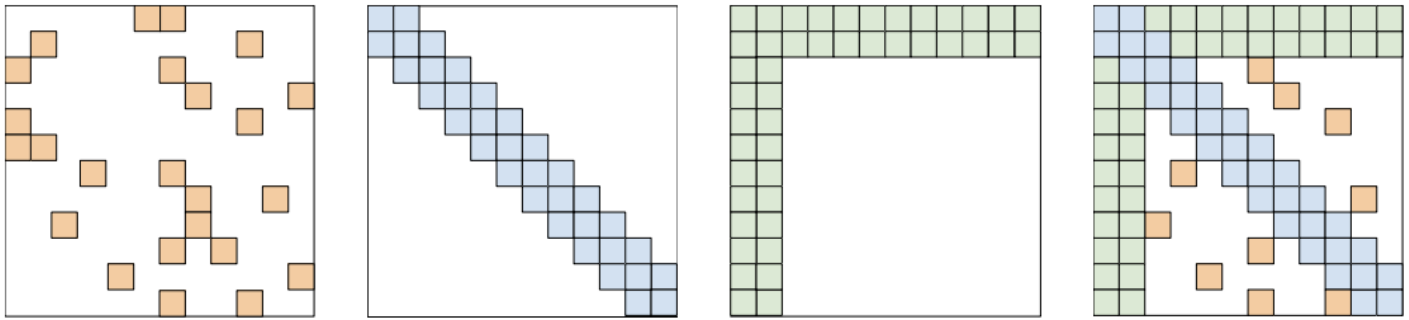
Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing	Avg	Speedup
BERT	base	108M	12	768	768	False	82.1	17.7x
	large	334M	24	1024	1024	False	85.1	3.8x
	xlarge	1270M	24	2048	2048	False	76.7	1.0
ALBERT	base	12M	12	768	128	True	80.1	21.1x
	large	18M	24	1024	128	True	82.4	6.5x
	xlarge	59M	24	2048	128	True	85.5	2.4x
	xxlarge	233M	12	4096	128	True	**88.7**	1.2x

Some experiments show dropout hurt performance

https://arxiv.org/abs/1909.11942

Sparse attentions

- Instead of computing the full attention matrix, people have found that many parts can be dropped



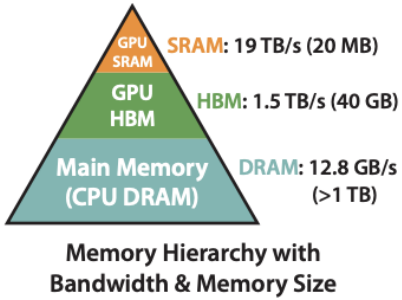
(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

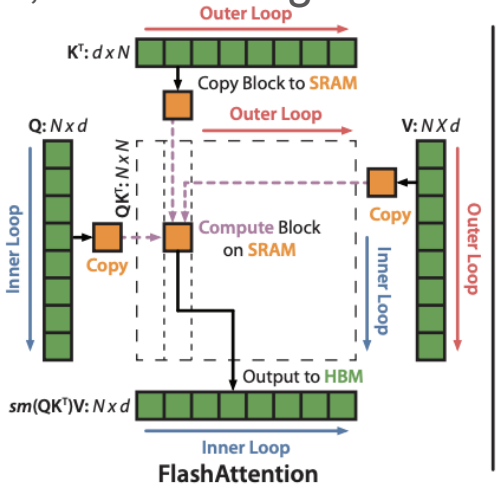
Big Bird: Transformers for Longer Sequences 2020 https://arxiv.org/abs/2007.14062

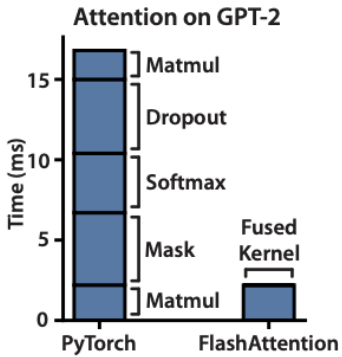
FlashAttention

- Low-level code optimization by taking advantage of the memory heirachy of the GPU
- Higher FLOP but lower memory transfer = W in speed (~3x)
- Use less memory by using recompute rather than saving in memory (linear memory growth, ~20x savings at 4k sequence length)



Memory Hierarchy with Bandwidth & Memory Size





https://arxiv.org/abs/2205.14135 2022

https://github.com/HazyResearch/flash-attention

FlashAttention

FlashAttention adoption

We've been very happy to see FlashAttention being adopted by many organizations and research labs to speed up their training / inference (within 6 months after FlashAttention's release, at the time of writing). This page contains a partial list of places where FlashAttention is being used. If you'd like to add links to your organization / product / codebase, please open a PR or email us. We'd very much like to hear from you!

Integrated into machine learning frameworks

- Pytorch: integrated into core Pytorch in nn.Transformer.
- Huggingface's transformers library. On-going, blogpost coming soon.
- Microsoft's DeepSpeed: FlashAttention is integrated into DeepSpeed's inference engine.
- Nvidia's Megatron-LM. This library is a popular framework on training large transformer language models at scale.
- MosaicML Composer library. Composer is a library for efficient neural network training.
- EleutherAI's GPT-NeoX. This is a research library for training large language transformer models at scale based on NVIDIA's Megatron-LM and Microsoft's DeepSpeed.
- PaddlePaddle: integrated into the framework with API `paddle.nn.functional.flash_attention`.

Flash attention 2 2023

FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning

Tri Dao[1,2]

[1]Department of Computer Science, Princeton University
[2]Department of Computer Science, Stanford University
trid@cs.stanford.edu

Abstract

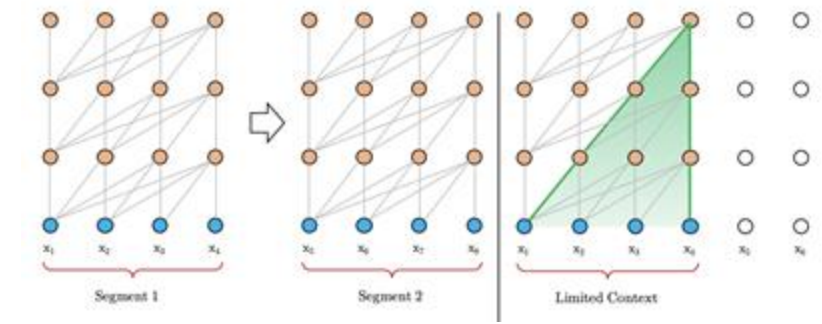
Scaling Transformers to longer sequence lengths has been a major problem in the last several years, promising to improve performance in language modeling and high-resolution image understanding, as well as to unlock new applications in code, audio, and video generation. The attention layer is the main bottleneck in scaling to longer sequences, as its runtime and memory increase quadratically in the sequence length. FlashAttention [5] exploits the asymmetric GPU memory hierarchy to bring significant memory saving (linear instead of quadratic) and runtime speedup (2-4× compared to optimized baselines), with no approximation. However, FlashAttention is still not nearly as fast as optimized matrix-multiply (GEMM) operations, reaching only 25-40% of the theoretical maximum FLOPs/s. We observe that the inefficiency is due to suboptimal work partitioning between different thread blocks and warps on the GPU, causing either low-occupancy or unnecessary shared memory reads/writes. We propose FlashAttention-2, with better work partitioning to address these issues. In particular, we (1) tweak the algorithm to reduce the number of non-matmul FLOPs (2) parallelize the attention computation, even for a single head, across different thread blocks to increase occupancy, and (3) within each thread block, distribute the work between warps to reduce communication through shared memory. These yield around 2× speedup compared to FlashAttention, reaching 50-73% of the theoretical maximum FLOPs/s on A100 and getting close to the efficiency of GEMM operations. We empirically validate that when used end-to-end to train GPT-style models, FlashAttention-2 reaches training speed of up to 225 TFLOPs/s per A100 GPU (72% model FLOPs utilization).[1]

https://arxiv.org/abs/2307.08691

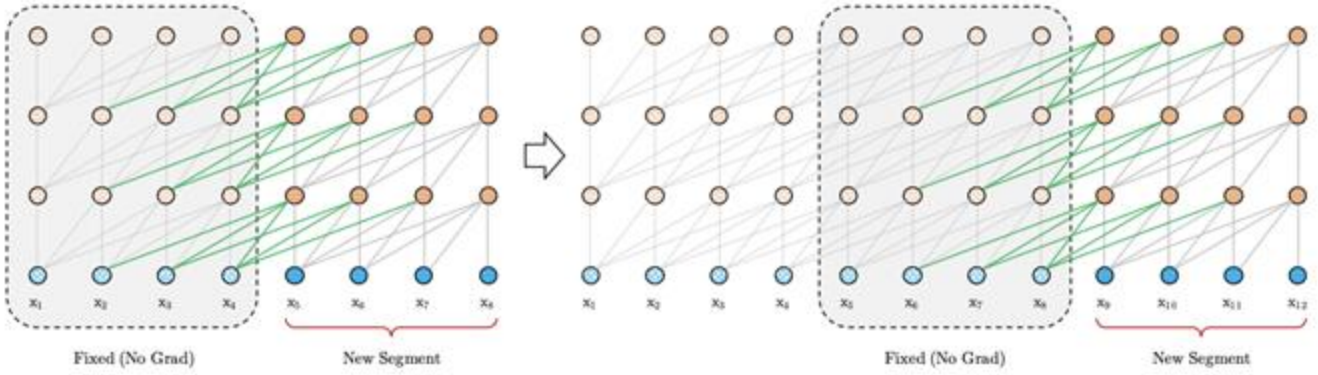
Fixing limited context

- If we use a pre-trained GPT model with token size of 1024, can we use it on a token size of 2048?
  - Not really
  - Positional embeddings were fixed at training (up to 1024)
  - Model only saw up to 1024 length, might not be able to generate something longer

XL-transformer



Chunking limits the length of context



(a) Training phase.

Let next chunk attention has access to previous chunk

This is quite similar to RNNs

Used in XL-Net and others

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context 2019 https://arxiv.org/abs/1901.02860

RoPE (Rotary Position Embedding)

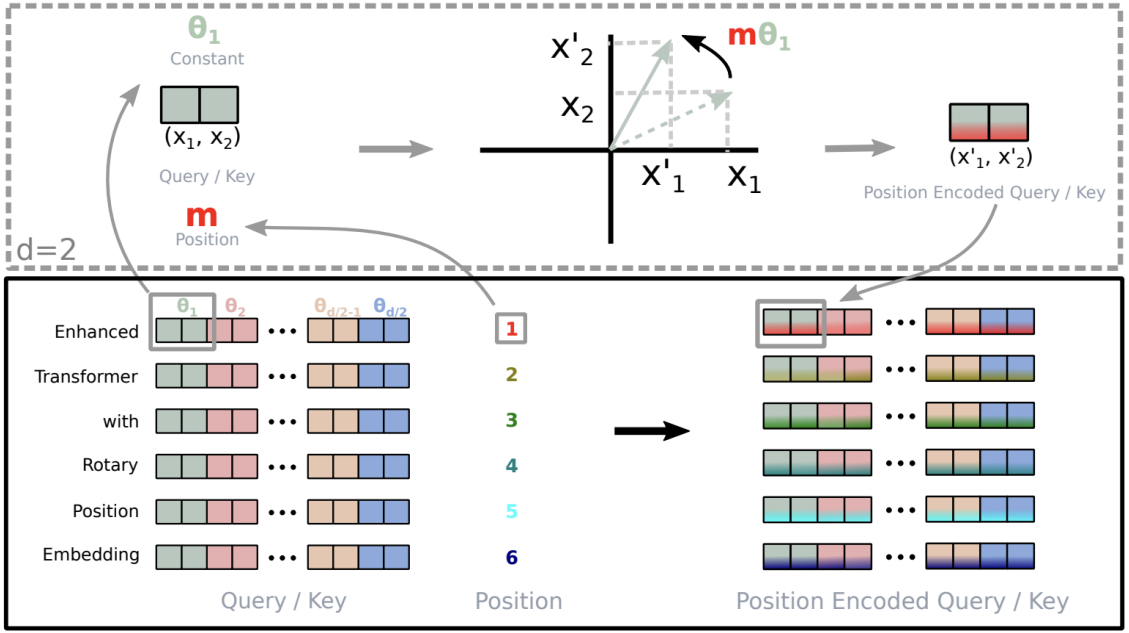
- We want the transformer to generalize beyond the trained context length.
- There are three cars. vs Three cars are there. - Three and cars should have similar attention properties
  3 4 1 2

https://arxiv.org/abs/2104.09864 2021

RoPE

q k v representations can be written as
x – input embedding, p – position embedding
W – weight matrix for q, k, v

$$f_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i, i) := \boldsymbol{W}_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i + \boldsymbol{p}_i)$$



Sets of rotations for each 2D dimension

Figure 1: Implementation of Rotary Position Embedding(RoPE).

RoPE reformulates

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q\boldsymbol{x}_m)e^{im\theta}$$
$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k\boldsymbol{x}_n)e^{in\theta}$$
$$g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) = \mathrm{Re}[(\boldsymbol{W}_q\boldsymbol{x}_m)(\boldsymbol{W}_k\boldsymbol{x}_n)^* e^{i(m-n)\theta}]$$

Dot product of q and k yields a relative difference in position term (m-n)

RoPE is used in Llama 2 and 3, and Qwen 2

LongRoPE, 2024

- Even with RoPE extending a transformer beyond the trained context size requires fine-tuning on longer text
- Downscale RoPE position non-uniformly

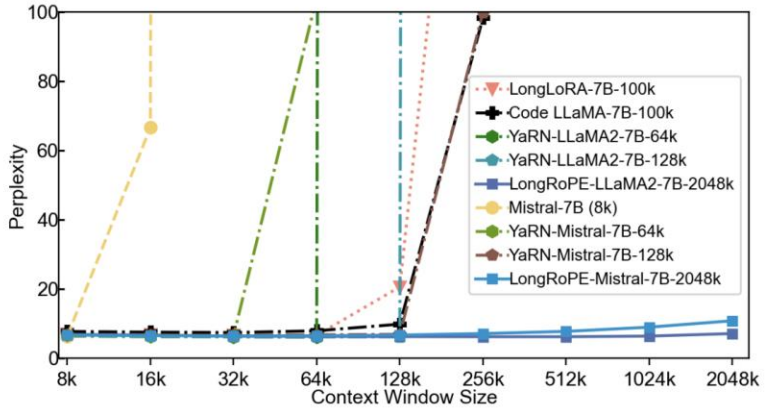


Figure 1. Books3 perplexity comparison between LongRoPE and state-of-the-art long-context LLMs using other extension methods.

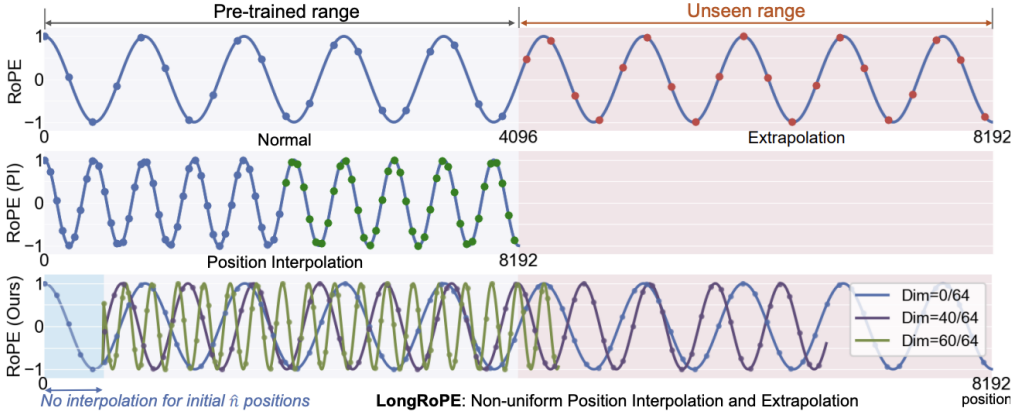


Figure 2. An illustrative example to show RoPE embedding under different interpolation methods. *Upper*: RoPE under direct extrapolation. *Middle*: Rescaled RoPE under linear positional interpolation. *Down*: LongRoPE fully exploits the identified two non-uniformities, leading to varied interpolation and extrapolation across RoPE dimensions at different token positions.

https://arxiv.org/abs/2402.13753

Group Query Attention (GQA), 2023

- Group multiple K and V from multiple head into groups
- Smaller memory footprint, better GPU utilization (KV-cache)
- Better modeling splitting in multi-GPU settings.

https://arxiv.org/abs/2305.13245

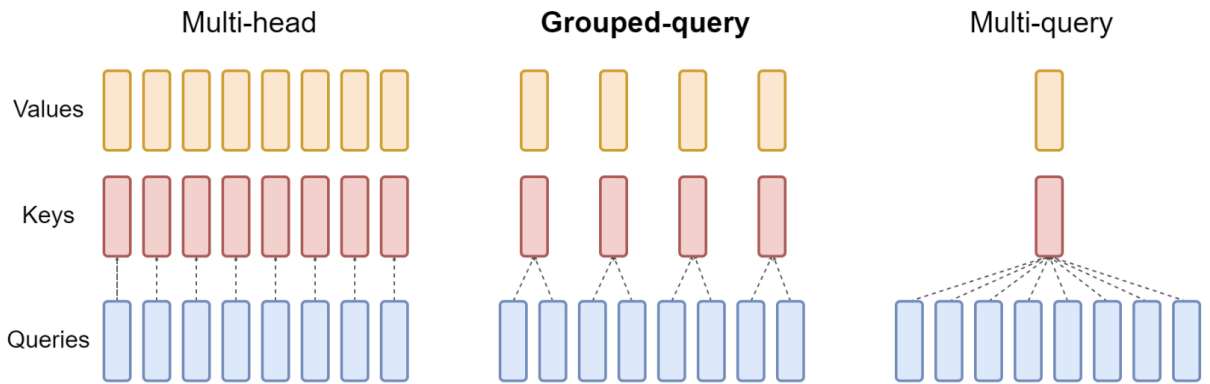


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

KV-cache https://medium.com/@joaolages/kv-caching-explained-276520203249

Pre-LN vs Post-LN transformers

- The original transformer puts Layer norm between the residual blocks (Post-LN)
- Pre-LN puts the layer norm inside.
  - Helps make the training more stable
  - Help eliminates the warm-up step in network training.

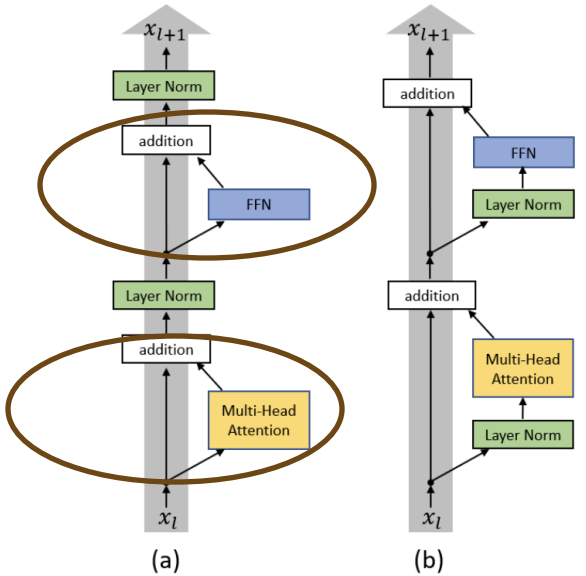


Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

On Layer Normalization in the Transformer Architecture 2020
https://arxiv.org/abs/2002.04745

RMS Layer Norm (Root Mean Square Layer Norm)

- Like Layer Norm, but does not rescale (just normalizes). More compute efficient than layer norm.

$$\bar{a}_i = \frac{a_i - \mu}{\sigma} g_i, \quad y_i = f\left(\bar{a}_i + b_i\right),$$

Layer norm

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

RMS Layer norm

$$\bar{a}_i = \frac{a_i}{\mathbf{RMS}(\mathbf{a})} g_i, \quad \text{where } \mathbf{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}.$$

- Used in LLMs to stabilize training and improve convergence.
  - “Pre-RMSNorm and Pre-CRMSNorm Transformers: Equivalent and Efficient Pre-LN Transformers” discusses how these are equivalent.

https://arxiv.org/abs/1910.07467 Root Mean Square Layer Normalization 2019

https://arxiv.org/abs/2412.15115

QWEN-2.5 Tech Report, Dec 2024

- Let's see how much we understand QWEN-2.5 architecture by this point

For dense models, we maintain the Transformer-based decoder architecture (Vaswani et al., 2017; Radford et al., 2018) as Qwen2 (Yang et al., 2024a). The architecture incorporates several key components: Grouped Query Attention (GQA, Ainslie et al., 2023) for efficient KV cache utilization, SwiGLU activation function (Dauphin et al., 2017) for non-linear activation, Rotary Positional Embeddings (RoPE, Su et al., 2024) for encoding position information, QKV bias (Su, 2023) in the attention mechanism and RMSNorm (Jiang et al., 2023b) with pre-normalization to ensure stable training.

Building upon the dense model architectures, we extend it to MoE model architectures. This is achieved by replacing standard feed-forward network (FFN) layers with specialized MoE layers, where each layer comprises multiple FFN experts and a routing mechanism that dispatches tokens to the top-K experts. Following the approaches demonstrated in Qwen1.5-MoE (Yang et al., 2024a), we implement fine-grained expert segmentation (Dai et al., 2024) and shared experts routing (Rajbhandari et al., 2022; Dai et al., 2024). These architectural innovations have yielded substantial improvements in model performance across downstream tasks.

For tokenization, we utilize Qwen's tokenizer (Bai et al., 2023), which implements byte-level byte-pair encoding (BBPE, Brown et al., 2020; Wang et al., 2020; Sennrich et al., 2016) with a vocabulary of 151,643 regular tokens. We have expanded the set of control tokens from 3 to 22 compared to previous Qwen versions, adding two new tokens for tool functionality and allocating the remainder for other model capabilities. This expansion establishes a unified vocabulary across all Qwen2.5 models, enhancing consistency and reducing potential compatibility issues.

DeepSeek V3, 27 Dec 2024

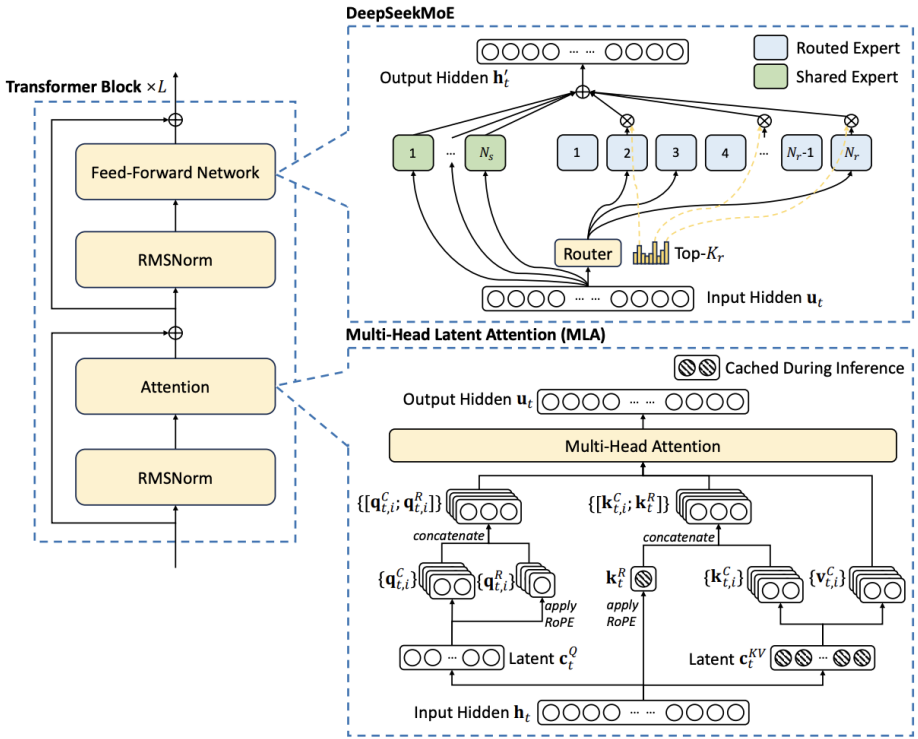


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

https://arxiv.org/abs/2412.19437

On tokenization

- Tokens remains a challenge
- Bad tokens leads to
    - OOVs
    - Longer than expected context length

GPT4o ~1500
unique tokens for Thai

```
' ถ่ายทอดสด',
' ถ่ายทอดสดฟุตบอล',
' ท',
' ทดลอง',
' ทดลองใช้ฟรี',
' ทั้ง',
' ทาง',
' ทางเข้า',
' ทำ',
' ที',
' ทีม',
' ทีเด็ด',
' ที่',
' ทุก',
' ธ',
' ธันวาคม',
' น',
' นัก',
' นักลงทุน',
' นักลงทุนสัมพันธ์',
' นัด',
' นาที',
' นาย',
' นิ',
' นี้',
' น้ำ',
' บ',
' บริษัท',
' บอล',
' บอลสด',
' บา',
' บาคาร่',
' บาคาร่า',
' บาท',
```

Case study: PhayaThaiBERT, 2023

- Adds English loanword tokens to WangchanBERTa (Sentencepiece Unigram)
  - Vocab size increases from 25,005 to 249,262 (randomly initialized new vocabs)
  - Model params from 106M parameters to 278M parameters.

https://arxiv.org/abs/2311.12475

Dataset	Proportion of Samples with Unassimilated Loanwords	Performance Gain Compared to WangchanBERTa
1. wisesight_sentiment	27.59%	+1.8 / +1.57
2. wongnai_reviews	37.20%	+0.16 / -0.52
3. yelp_review_full	Entirely English	+6.72 / +9.86
4. generated_reviews_enth	36.68%	+0.1 / -0.47
5. prachathai67k	8.54%	+1.6 / +2.05
6. thainer	10.90%	+1.78 / +6.58
7. lst20 (pos)	2.58%	+0.05 / -0.33
8. lst20 (ner)	2.58%	+0.12 / -0.17
9. thai_nner	38.54%	+4.95 / +2.85

Table 6: Proportion of samples with unassimilated English words compared with performance gain for each dataset.

Beyond Transformer?

- Some people say we are hitting a wall in transformer architecture.
- There are researches on newer architecture choices. Most are tackling the length/memory issue.

- S4, MAMBA, Jamba – State-space model inspired. Has RNN flavors.
- Titans

https://www.ai21.com/blog/announcing-jamba
https://arxiv.org/abs/2501.00663

Summary

- 3 main architectural choices: Encoder only, encoder-decoder, decoder only.
- Several techniques trying to improve the attention layer bottleneck: hardware, algorithmic, systems.
- Tokenization still plays a big role.