Large Language Models (LLMs)

References

- Recurrent neural networks [GBC16; Chapter 10]
- Transformers [VSP+17]
- Neural Networks: Zero to Hero by Andrej Karpathy

[GBC16] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning* (MIT Press, 2016). Accessed on Aug 28, 2024.
[VSP+17] A. Vaswani, N. Shazeer, N. Parmar *et al.* *Attention Is All You Need*. In: *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017). Accessed on Oct 11, 2024.

Autoregressive Models

Given a sequence of n_{ctx} past vectors $x_{-1}, \ldots, x_{-n_{\text{ctx}}} \in \mathbb{R}^n$, "predict" the next ones. Key idea : *receding horizon*:

$$\begin{aligned} & p(x_0, x_1 | x_{-1}, \ldots, x_{-n_{\text{ctx}}}) \\ & = p(x_0 | x_{-1}, \ldots, x_{-n_{\text{ctx}}}) p(x_1 | x_0, x_{-1}, \ldots, x_{-n_{\text{ctx}}+1}, x_{-n_{\text{ctx}}}) \\ & \approx p(x_0 | x_{-1}, \ldots, x_{-n_{\text{ctx}}}) p(x_1 | x_0, x_{-1}, \ldots, x_{-n_{\text{ctx}}+1}) \end{aligned}$$

- **Model** : Probability of next vector $\hat{p}(x_0|X)$ where X concatenates $x_{-1}, \ldots, x_{-n_{\text{ctx}}}$.
- **Loss** : Cross-entropy : $\mathcal{L}_{\hat{p}}(X) \triangleq H(p, \hat{p}) = -\mathbf{E}_p[\log(\hat{p})] = -\sum_{x_0} p(x_0|X) \log(\hat{p}(x_0|X))$
- Particular case for $\hat{p}(x_0|X) = \delta_y$: $\mathcal{L}_{\hat{p}}(X) = -\log(\hat{p}(y|X))$

What about Language Models ?

Given "past text", predict the "following text". How to turn text into vectors of $\mathbb{R}^n$?

Text to vectors : step 1 → tokenization

Why not encode each letter ?

- **Idea** : Turn each letter into its one-hot encoding in $\mathbb{R}^{26}$.
- **Issue** : The "past text" only has n_{ctx} characters so n_{ctx} must be **large** but transformers have a complexity **quadratic** in n_{ctx}!
- **Practical details** : Text is encoded with UTF-8 so each character is encoded into 1 to 4 bytes. We encode each byte to a vector in $\mathbb{R}^{256}$ but care must be taken not to generate invalid UTF-8.

Why not encode each word ?

- **Idea** : Turn each word into its one-hot encoding in $\mathbb{R}^n$. The value of n is the number of words. Depending on the language (source):

Language	French	English	Dutch	German
n	408,078	350,000	350,000	200,000

- **Issue** : The value of n is **too large**. We cannot trust the words of languages to be a tokenization that optimally compresses text for our dataset.

Byte Pair Encoding

Byte Pair Encoding algorithm [SHB16] greedily merges the most frequent pair of tokens over the dataset into a new token. Most used implementations are `SentencePiece` [KR18] and `tiktoken` (play with it here). For instance, on this example, the pair `('a', 'a')` is the most frequent so we substitute it by a new token, say `'Z'`:

```
▸Dict(('b', 'd') ⇒ 1, ('a', 'b') ⇒ 2, ('d', 'a') ⇒ 1, ('b', 'a') ⇒ 1, ('a', 'c') ⇒ 1, ('
pair_stats("aaabdaaabac")
```

```
iter_1 = ▸BPE("ZabdZabac", Dict(('a', 'a') ⇒ 'Z'))
iter_1 = new_token("aaabdaaabac")
```

```
iter_2 = ▸BPE("ZYdZYac", Dict(('a', 'b') ⇒ 'Y', ('a', 'a') ⇒ 'Z'))
iter_2 = new_token(iter_1)
```

Note that the new tokens can also be part of the most frequence pair!

```
iter_3 = ▸BPE("XdXac", Dict(('a', 'b') ⇒ 'Y', ('Z', 'Y') ⇒ 'X', ('a', 'a') ⇒ 'Z'))
iter_3 = new_token(iter_2)
```

[SHB16] R. Sennrich, B. Haddow and A. Birch. *Neural Machine Translation of Rare Words with Subword Units* (Jun 2016), arXiv:1508.07909. Accessed on Oct 23, 2024.
[KR18] T. Kudo and J. Richardson. *SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing* (Aug 2018), arXiv:1808.06226. Accessed on Oct 23, 2024.

Increasing length of "past text"

Challenging tradeoff: Encode text to **increase** length of "past text" while keeping n_{ctx} and n **small** enough.

Length of "past text" increases with vocabulary size n_{voc} and context window n_{ctx}.

Name	Ref	n_{voc}	n_{ctx}	Tokenizer
GPT-2	[RWCL19]	50k	1024	tiktoken
GPT-3	[BMRS20]	50k	2048	tiktoken
GPT-3.5		100k	4096	tiktoken
GPT-4		100k	32k	tiktoken
GPT-4o		200k	128k	tiktoken
Gemini-1	[TABA24]	256k	10M	SentencePiece
Gemini-1.5	[TGLB24]	256k	10M	SentencePiece
Gemma	[TMHD24]	256k	8192	SentencePiece
Gemma-2	[TRPS24]	256k	8192	SentencePiece
Llama-2	[TMSA23]	32k	4k	SentencePiece
Llama-3		128k	8k	tiktoken
Llama-3.1		128k	128k	tiktoken
Llama-3.2		128k	128k	tiktoken
MegaByte	[YSFA23]	256	8192	Bytes

[TGL+24] G. Team, P. Georgiev, V. I. Lei *et al.* *Gemini 1.5: Unlocking Multimodal Understanding across Millions of Tokens of Context* (Aug 2024), arXiv:2403.05530. Accessed on Nov 11, 2024.
[TAB+24] G. Team, R. Anil, S. Borgeaud *et al.* *Gemini: A Family of Highly Capable Multimodal Models* (Jun 2024), arXiv:2312.11805. Accessed on Nov 11, 2024.
[TMH+24] G. Team, T. Mesnard, C. Hardin *et al.* *Gemma: Open Models Based on Gemini Research and Technology* (Apr 2024), arXiv:2403.08295. Accessed on Nov 11, 2024.
[TRP+24] G. Team, M. Riviere, S. Pathak *et al.* *Gemma 2: Improving Open Language Models at a Practical Size* (Oct 2024), arXiv:2408.00118. Accessed on Nov 11, 2024.
[SHB16] R. Sennrich, B. Haddow and A. Birch. *Neural Machine Translation of Rare Words with Subword Units* (Jun 2016), arXiv:1508.07909. Accessed on Oct 23, 2024.
[RWC+19] A. Radford, J. Wu, R. Child *et al.* *Language Models Are Unsupervised Multitask Learners* (2019). Accessed on Oct 23, 2024.
[BMR+20] T. B. Brown, B. Mann, N. Ryder *et al.* *Language Models Are Few-Shot Learners* (Jul 2020), arXiv:2005.14165. Accessed on Nov 11, 2024.
[TMS+23] H. Touvron, L. Martin, K. Stone *et al.* *Llama 2: Open Foundation and Fine-Tuned Chat Models* (Jul 2023),

arXiv:2307.09288. Accessed on Oct 23, 2024.
[YSF+23] L. Yu, D. Simig, C. Flaherty *et al.* *MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers* (May 2023), arXiv:2305.07185. Accessed on Oct 23, 2024.

Text to vectors : step 2 → embedding

Consider one-hot encoding with vocabulary size n_{voc} and a *bigram model*

$$\hat{p}(x_0|x_{-1}) = \mathrm{softmax}(W_d \tanh(\cdots \tanh(W_1 x_{-1})\cdots))$$

The matrix W_d has n_{voc} rows and W_1 has n_{voc} columns → issue if n_{voc} is large

Embedding : Use vectors $c_1, \ldots, c_{n_{\text{voc}}} \in \mathbb{R}^{d_{\text{emb}}}$ with *embedding size* (aka *hidden size*) $d_{\text{emb}} \ll n_{\text{voc}}$.

Equivalently, we still use one-hot encoding but we add an encoder $C \in \mathbb{R}^{d_{\text{emb}} \times n_{\text{voc}}}$ and decoder $D \in \mathbb{R}^{n_{\text{voc}} \times d_{\text{emb}}}$

$$\hat{p}(x_0|x_{-1}) = \mathrm{softmax}(DW_d \tanh(\cdots \tanh(W_1 C x_{-1})\cdots))$$

▶ **What difference do you expect with respect to the previous model ?**

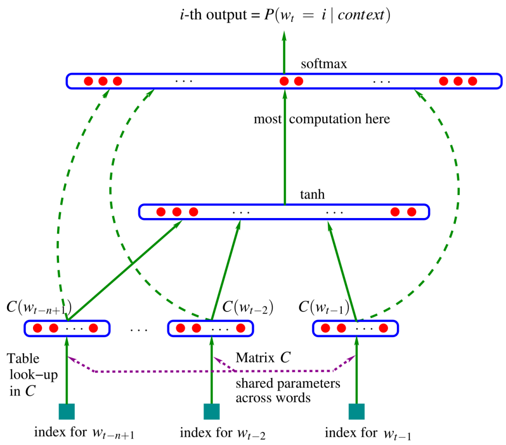
Forcing $D = C^\top$ appears to work well in practice [PW17], this is what is used in [VSP+17].

[PW17] O. Press and L. Wolf. *Using the Output Embedding to Improve Language Models* (Feb 2017), arXiv:1608.05859. Accessed on Nov 11, 2024.
[VSP+17] A. Vaswani, N. Shazeer, N. Parmar *et al.* *Attention Is All You Need*. In: *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017). Accessed on Oct 11, 2024.

Shared embedding

With $n_{\text{ctx}} > 1$, the encoder C is shared by all tokens: See for instance the network below taken from [BDV00; Figure 1], the first popular application of neural nets for languages:

$$\hat{p}(x_0|x_{-1}, \ldots, x_{-n_{\text{ctx}}}) = \text{softmax}(W_2 \tanh(W_1 \begin{bmatrix} Cx_{-1} \\ \vdots \\ Cx_{-n_{\text{ctx}}} \end{bmatrix}))$$



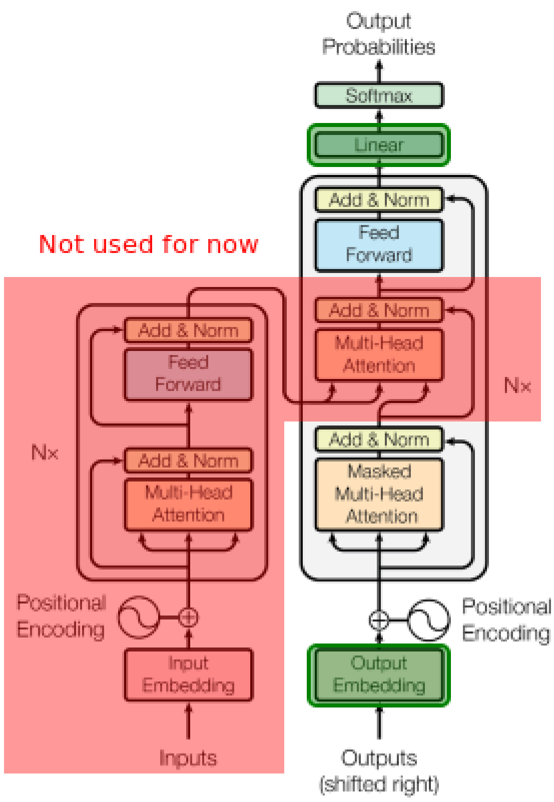
▶ **What are the number of columns of W_1 and number of rows of W_2 now ?**

[BDV00] Y. Bengio, R. Ducharme and P. Vincent. *A Neural Probabilistic Language Model*. In: *Advances in Neural Information Processing Systems*, Vol. 13 (MIT Press, 2000). Accessed on Oct 11, 2024.

Embedding sizes in LLMs

See the table below for the size of embeddings of large language models:

Name	Num params	Ref	n_{voc}	d_{emb}
GPT-2	1.5B	[RWCL19]	50k	768
Gemma	2B	[TMHD24]	256k	2048
Gemma	7B	[TMHD24]	256k	3072
Gemma-2	27B	[TRPS24]	256k	4608
Gemma-2	2B	[TRPS24]	256k	2304
Gemma-2	9B	[TRPS24]	256k	3584
Llama-2	7B	[TMSA23]	32k	4096
base		[VSPU17]	37k	512
big		[VSPU17]	37k	1024



[TMH+24] G. Team, T. Mesnard, C. Hardin *et al.* *Gemma: Open Models Based on Gemini Research and Technology* (Apr 2024), arXiv:2403.08295. Accessed on Nov 11, 2024.
[TRP+24] G. Team, M. Riviere, S. Pathak *et al.* *Gemma 2: Improving Open Language Models at a Practical Size* (Oct 2024), arXiv:2408.00118. Accessed on Nov 11, 2024.
[RWC+19] A. Radford, J. Wu, R. Child *et al.* *Language Models Are Unsupervised Multitask Learners* (2019). Accessed on Oct 23, 2024.
[TMS+23] H. Touvron, L. Martin, K. Stone *et al.* *Llama 2: Open Foundation and Fine-Tuned Chat Models* (Jul 2023), arXiv:2307.09288. Accessed on Oct 23, 2024.

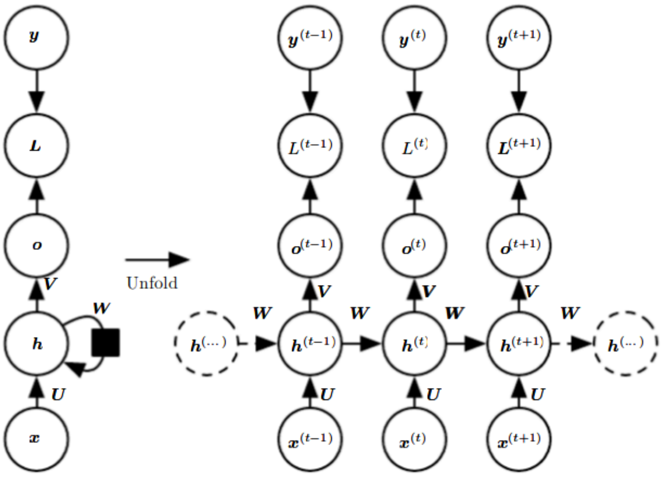
Recurrent neural networks (RNN)

$$
\begin{aligned}
h^{(t+1)} &= \tanh(Wh^{(t)} + Ux^{(t+1)} + b) \\
o^{(t)} &= Vh^{(t)} + c \\
\hat{y}^{(t)} &= \mathrm{softmax}(o^{(t)})
\end{aligned}
$$

Illustrated on the right [GBC16; Figure 10.3].

RNNs as language model showcased in [MKB+10].

Issue: Training time and space complexity is proportional to n_{ctx} and **cannot parallelize** to speed up.



[MKB+10] T. Mikolov, M. Karafiát, L. Burget *et al*. *Recurrent Neural Network Based Language Model*. In: *Proc. Interspeech 2010* (2010); pp. 1045–1048. Accessed on Nov 11, 2024.
[GBC16] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning* (MIT Press, 2016). Accessed on Aug 28, 2024.

Extensions of RNNs

It's difficult to model long-term dependencies as their gradient either vanish or explodes exponentially (think of the power method) [GBC16; Section 10.7]

Gated extensions attempting to solve this issue [GBC16; Section 10.10]:

- Long short-term memory (LSTM) [Gra14]
- Gated recurrent unit (GRU) [CvMBB14]

Recently, Mamba suggests a solution to the complexity issue [GD24]. As it scales better with $n_{\mathbf{ctx}}$, it is even suggested to get rid of the tokenizer : [WGYR24].

[CvMBB14] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches* (Oct 2014), arXiv:1409.1259. Accessed on Nov 11, 2024.
[Gra14] A. Graves. *Generating Sequences With Recurrent Neural Networks* (Jun 2014), arXiv:1308.0850. Accessed on Nov 11, 2024.
[GBC16] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning* (MIT Press, 2016). Accessed on Aug 28, 2024.
[GD24] A. Gu and T. Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces* (May 2024), arXiv:2312.00752. Accessed on Nov 11, 2024.
[WGYR24] J. Wang, T. Gangavarapu, J. N. Yan and A. M. Rush. *MambaByte: Token-free Selective State Space Model* (Aug 2024), arXiv:2401.13660. Accessed on Nov 11, 2024.

Numerical dictionary

What would a numerical dictionary look like ? Consider keys $k_i \in \mathbb{R}^{d_k}$ and values $v_i \in \mathbb{R}^{d_v}$. Given a query $q \in \mathbb{R}^{d_k}$,

```
dict = ▸Dict([1, 0] ⇒ [1, 1], [0, 1] ⇒ [-1, 1])
```

```julia
dict = Dict([1, 0] => [1, 1], [0, 1] => [-1, 1])
```

```
▸[1, 1]
```

```julia
dict[[1, 0]]
```

```
numerical_lookup (generic function with 1 method)
```

```julia
function numerical_lookup(dict, query)
    _, i = findmax([dot(query, key) for key in keys(dict)])
    return collect(values(dict))[i]
end
```

```
▸[1, 1]
```

```julia
numerical_lookup(dict, [0.8, 0.2])
```

Attention head

Attention head provides a differentiable numerical dictionary [BCB16]

$$\alpha = \mathrm{softmax}(\langle q, k_1\rangle, \ldots, \langle q, k_{n_{\mathrm{ctx}}}\rangle) \qquad \mathrm{Attention}(q, k, v) = \sum_{i=1}^{n_{\mathrm{ctx}}} \alpha_i v_i$$

softmax (generic function with 1 method)

```
function softmax(x)
    y = exp.(x)
    return y / sum(y)
end
```

softmax_lookup (generic function with 1 method)

```
function softmax_lookup(dict, query)
    ks = keys(dict)
    α = softmax([dot(query, key) for key in keys(dict)])
    @show α
    return sum(α * value for (α, value) in zip(α, values(dict)))
end
```

[0.291313, 1.0]

```
softmax_lookup(dict, [0.8, 0.2])
```

```
α = [0.6456563062257954, 0.3543436937742045]
```

[BCB16] D. Bahdanau, K. Cho and Y. Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate* (May 2016), arXiv:1409.0473. Accessed on Oct 23, 2024.

Matrix form of attention

$$Q = [q_1 \quad \cdots \quad q_{n_{\text{ctx}}}] \qquad K = [k_1 \quad \cdots \quad k_{n_{\text{ctx}}}] \qquad K^\top Q = \begin{bmatrix} \langle k_1, q_1 \rangle & \cdots & \langle k_1, q_{n_{\text{ctx}}} \rangle \\ \vdots & \ddots & \vdots \\ \langle k_{n_{\text{ctx}}}, q_1 \rangle & \cdots & \langle k_{n_{\text{ctx}}}, q_{n_{\text{ctx}}} \rangle \end{bmatrix}$$

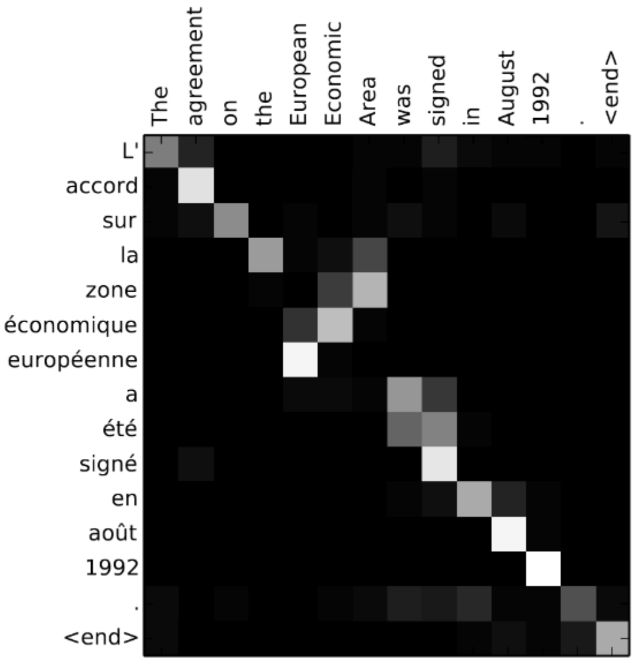
`softmax` is then applied to each **column**:

$$\text{softmax}(K^\top Q / \sqrt{d_k})$$

Division by $\sqrt{d_k}$ scales the input of softmax to preferable regions [VSP+17; Secton 3.2.1].

Illustrated on the right from [BCB16; Figure 3(a)].

$$\text{Attention}(V, K, Q) = V\text{softmax}(K^\top Q / \sqrt{d_k})$$



[BCB16] D. Bahdanau, K. Cho and Y. Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate* (May 2016), arXiv:1409.0473. Accessed on Oct 23, 2024.
[VSP+17] A. Vaswani, N. Shazeer, N. Parmar *et al.* *Attention Is All You Need*. In: *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017). Accessed on Oct 11, 2024.

Masked Attention

💡 **Key idea** In the model for $\hat{p}(x_0|x_{-1},\dots,x_{-n_{\text{ctx}}})$, incorporate sub-models

$$\begin{gathered}\bar{p}(x_0|x_{-1},\dots,x_{-n_{\text{ctx}}})\\ \bar{p}(x_{-1}|x_{-2},\dots,x_{-n_{\text{ctx}}})\\ \vdots\\ \bar{p}(x_{-n_{\text{ctx}}+1}|x_{-n_{\text{ctx}}}).\end{gathered}$$

Mask prevent $\hat{p}$ to look input the future:

$$M=\begin{bmatrix}0 & 0 & \cdots & 0\\ -\infty & 0 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ -\infty & \cdots & -\infty & 0\end{bmatrix}$$

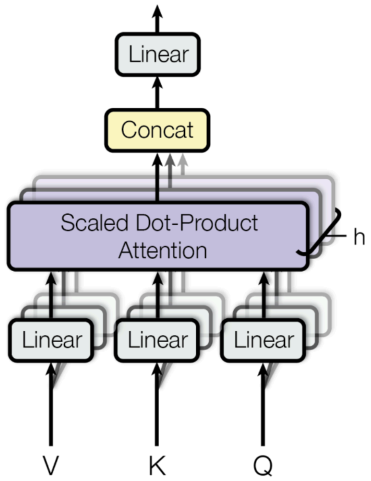
$$\text{Masked-Attention}(V,K,Q)\ = V\text{softmax}(M+K^\top Q/\sqrt{d_k})$$

Multi-Head Attention

Heads focus on different aspects. Their outputs are **combined** with $W^O \in \mathbb{R}^{d_{\text{emb}} \times h d_v}$:

$$\begin{aligned} \text{head}_j &= \text{Attention}(W_j^V V, W_j^K K, W_j^Q Q) \\ \text{MultiHead}(V, K, Q) &= W^O \text{vcat}(\text{head}_1, \ldots, \text{head}_h) \end{aligned}$$

See [VSP+17; Figure 2] on the right.



Similarly, in the masked case:

$$\begin{aligned} \text{head}_j &= \text{Masked-Attention}(W_j^V V, W_j^K K, W_j^Q Q) \\ \text{Masked-MultiHead}(V, K, Q) &= W^O \text{vcat}(\text{head}_1, \ldots, \text{head}_h) \end{aligned}$$

▶ **Is W^O needed if $h = 1$?**

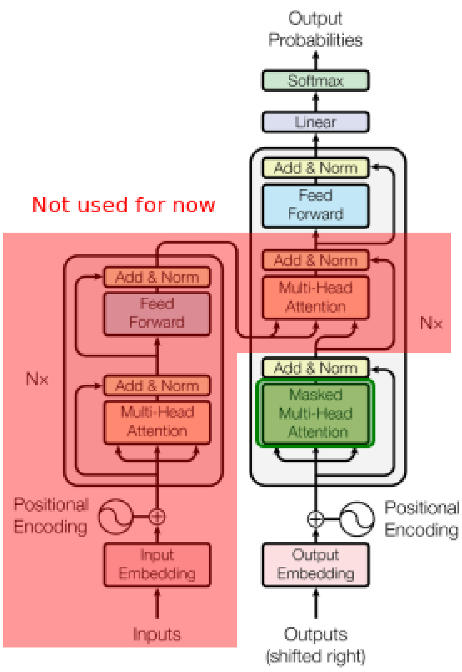
Self-Attention

Self-Attention with embedding C is:

$$\text{Masked-MultiHead}(CX, CX, CX)$$

The embedding vectors CX take then different projections for value, key, query and also for different heads!

$$\text{head}_j = \text{Masked-Attention}(W_j^V CX, W_j^K CX, W_j^Q CX)$$



▶ **Is the order between the tokens taken into account by the model ?**

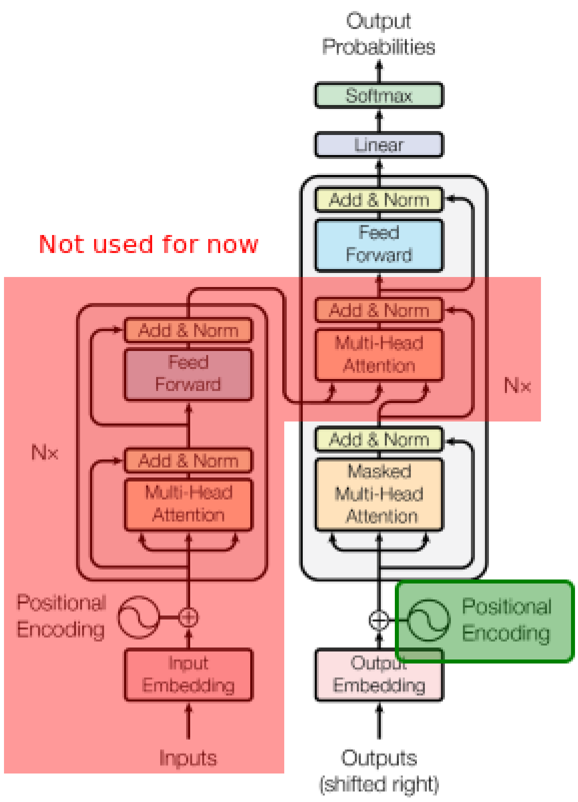
Positional encoding

Cannot sum $Cx_i + e_i$ with one-hot encoding $e_i \in \mathbb{R}^{n_{\text{ctx}}}$ as the dimension of Cx_i is $\mathbb{R}^{d_{\text{emb}}}$.

So we also add a positional embedding P: $Cx_i + Pe_i = Cx_i + p_i$.

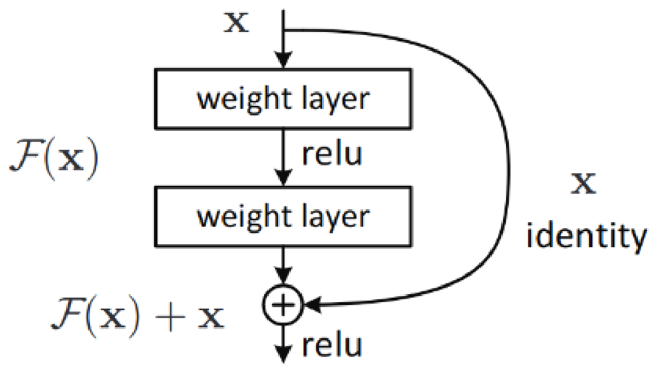
With Self-Attention:

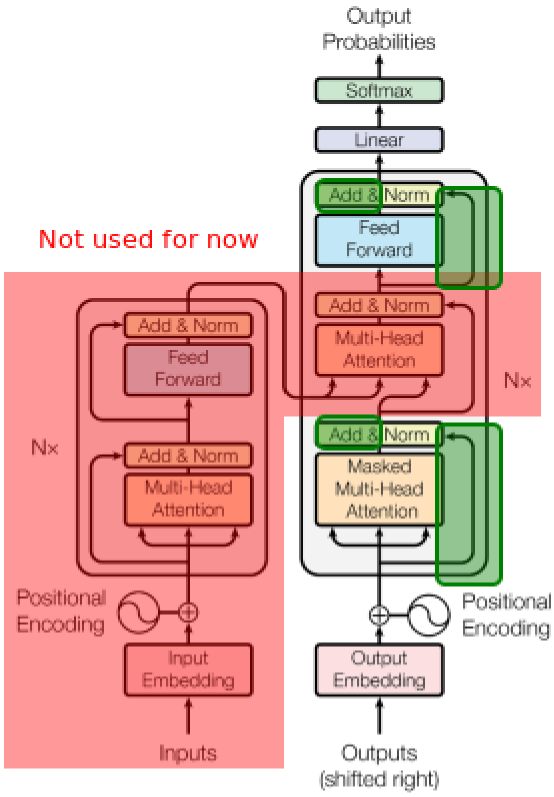
$$\text{Self-MultiHead}(CX + P, CX + P, CX + P)$$



Residual connection

Residual connection [HZRS15]





[HZRS15] K. He, X. Zhang, S. Ren and J. Sun. *Deep Residual Learning for Image Recognition* (Dec 2015), arXiv:1512.03385. Accessed on Nov 12, 2024.

Layer normalization

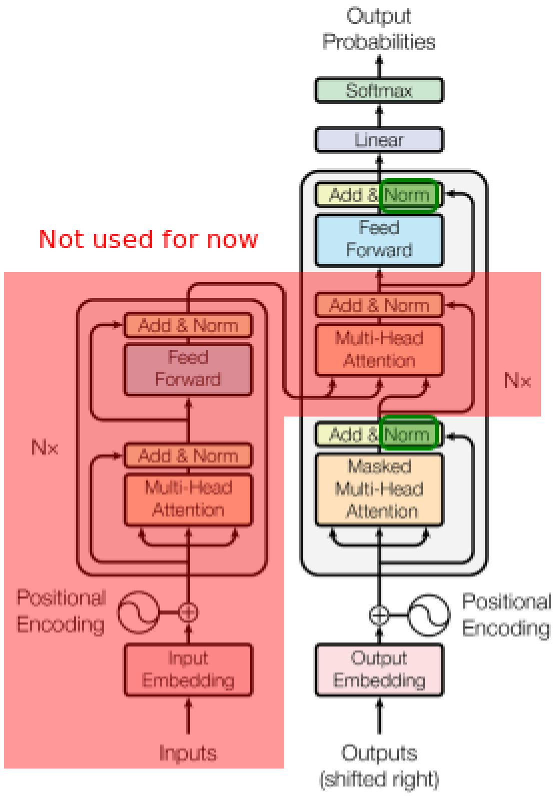
Norm of gradient increases exponentially with depth. Issue for deep neural net. Consider output

$$\begin{bmatrix} y_{1,1} & \cdots & y_{1,d_{\text{emb}}} \\ \vdots & \ddots & \vdots \\ y_{d_{\text{batch}},1} & \cdots & y_{d_{\text{batch}},d_{\text{emb}}} \end{bmatrix}$$

Normalization : $y_{i,j} \mapsto g(y_{i,j} - \mu_{i,j})/\sigma_{i,j}$ for gain g, mean μ and standard deviation σ.

- Batch normalization : $\sigma_{i,j} = \sigma_j$ [IS15]
- Layer normalization : $\sigma_{i,j} = \sigma_i$ [BKH16]

Batch norm depends on the batch hence is tricky to implement. Layer normalization is used in [VSP+17].



[IS15] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (Mar 2015), arXiv:1502.03167. Accessed on Nov 12, 2024.
[BKH16] J. L. Ba, J. R. Kiros and G. E. Hinton. *Layer Normalization* (Jul 2016), arXiv:1607.06450. Accessed on Nov 12, 2024.
[VSP+17] A. Vaswani, N. Shazeer, N. Parmar *et al.* *Attention Is All You Need*. In: *Advances in Neural Information Processing Systems*, Vol. 30 (Curran Associates, Inc., 2017). Accessed on Oct 11, 2024.

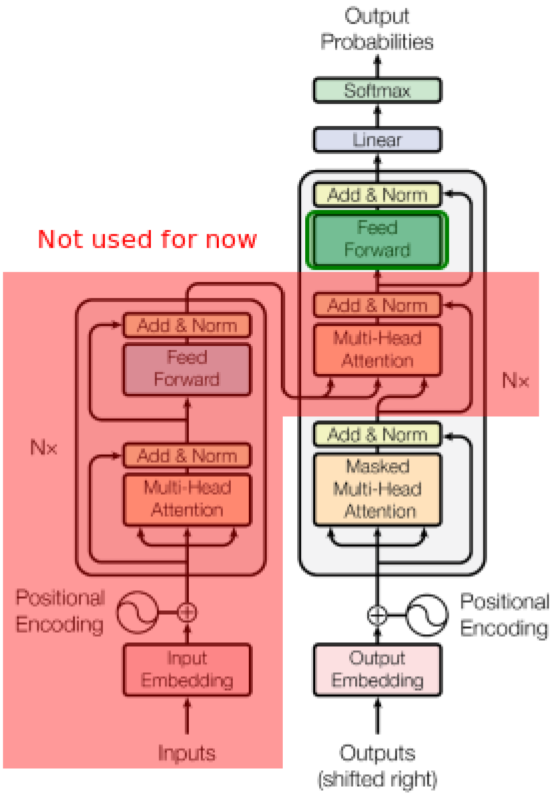
Feed-Forward network

Different weights $W_1 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{emb}}}$, $W_2 \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{ff}}}$ for each layer:

$$x \mapsto W_2 \max(0, W_1 x + b_1) + b_2$$

Expansion factor $d_{\text{ff}}/d_{\text{emb}}$ is typically 4× like suggested in [VSP+17] (but not for Gemma)

Name	Ref	d_{emb}	d_{ff}
GPT-2	[RWCL19]	768	3072
Gemma	[TMHD24]	2048	32768
Gemma	[TMHD24]	3072	49152
Gemma-2	[TRPS24]	2304	18432
Gemma-2	[TRPS24]	3584	28672
Gemma-2	[TRPS24]	4608	73728
Llama-3			4096
base	[VSPU17]	512	2048
big	[VSPU17]	1024	4096

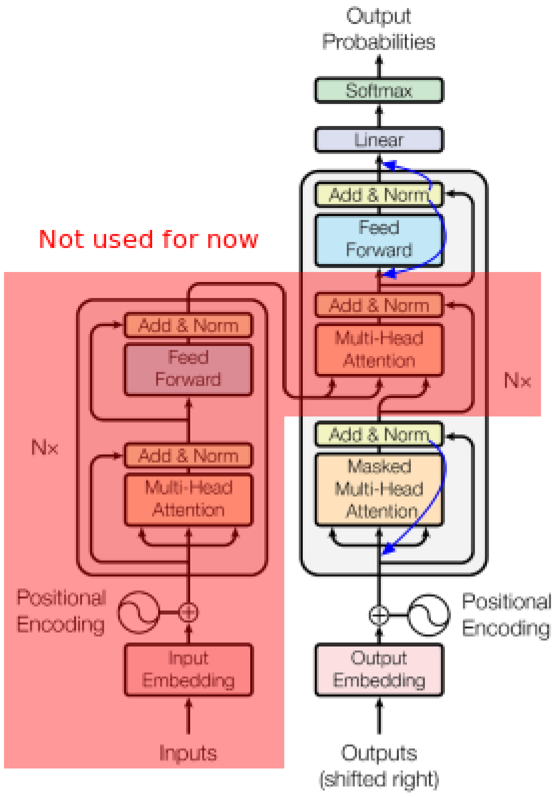


The feed-forward network is implemented **independently** for the output of each query so each query can be processed independently through each **layer**. The next layer allows each queries to then look at the results of the previous layer for **past** (because of the mask) queries.

Transformer variations

Pre-activation for residual neural networks introduced in [HZRS16] and used in GPT-2 [RWC+19]. See figure on the right.

Rotary Positional Encoding [SLP+23] replaces $W^K(Cx_i + p_i)$ and $W^Q(Cx_i + p_i)$ by $R^i W^K C x_i$ and $R^i W^Q C x_i$ where R is a rotation matrix. Advantage : $\langle k_i, q_j \rangle$ contains R^{i-j} → **relative** difference of position.



[HZRS16] K. He, X. Zhang, S. Ren and J. Sun. *Identity Mappings in Deep Residual Networks*. In: *Computer Vision – ECCV 2016*, edited by B. Leibe, J. Matas, N. Sebe and M. Welling (Springer International Publishing, Cham, 2016); pp. 630–645.
[RWC+19] A. Radford, J. Wu, R. Child *et al. Language Models Are Unsupervised Multitask Learners* (2019). Accessed on Oct 23, 2024.
[SLP+23] J. Su, Y. Lu, S. Pan *et al. RoFormer: Enhanced Transformer with Rotary Position Embedding* (Nov 2023), arXiv:2104.09864. Accessed on Nov 12, 2024.

Cost of LLMs

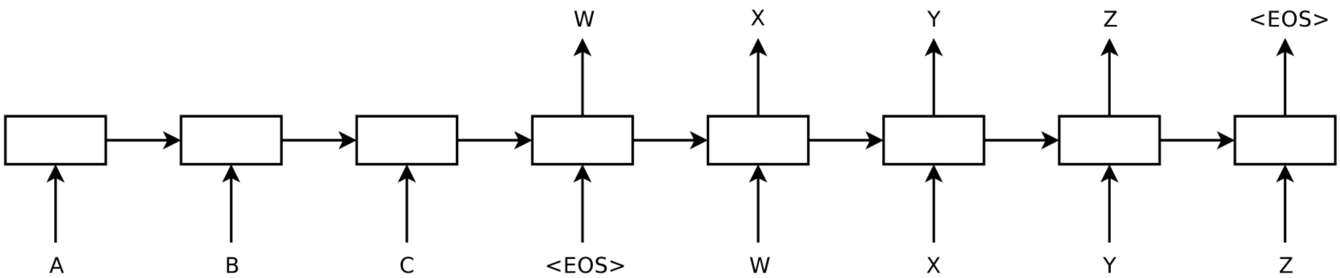
▶ **What is the time complexity of a transformer with respect to d_{emb}, n_{voc}, n_{ctx}, d_{ff}, h and N ?**

▶ **How does the number of parameters of transformers compare with [BDV00] or RNNs for large n_{ctx} ?**

[BDV00] Y. Bengio, R. Ducharme and P. Vincent. *A Neural Probabilistic Language Model*. In: *Advances in Neural Information Processing Systems*, Vol. 13 (MIT Press, 2000). Accessed on Oct 11, 2024.

Machine translation

- LSTM **encoder** → **context** → LSTM **decoder** [SVL14]. See [SVL14; Figure 1] below.
- Issue with *encoder bottleneck*. All information has to be summarized in the **context**.



[SVL14] I. Sutskever, O. Vinyals and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14* (MIT Press, Cambridge, MA, USA, Dec 2014); pp. 3104–3112. Accessed on Oct 23, 2024.

Cross-Attention

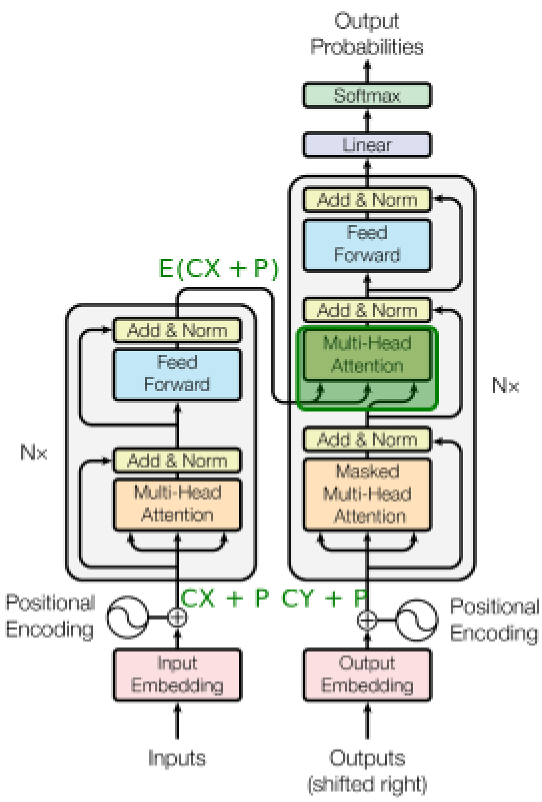
Cross-Attention between

- values and keys $E(CX + P)$ where E is the encoder, and X is the matrix of input tokens
- query Q depending on past output Y and number of layers already applied

$$\mathrm{MultiHead}(E(CX + P), E(CX + P), Q)$$

The embedding vectors CX take then different projections for value, key, query and also for different heads!

$$\mathrm{head}_j = \mathrm{Attention}(W_j^V V, W_j^K K, W_j^Q Q)$$
$$\text{where } V = K = E(CX + P)$$



Utils

```julia
using PlutoUI, DataFrames, PrettyTables, LinearAlgebra, Luxor, LaTeXStrings,
MathTeXEngine
```

```julia
import DocumenterCitations, CSV, Logging
```

```
qa (generic function with 2 methods)
```

```julia
include("utils.jl")
```

```
biblio =
▸CitationBibliography("/home/runner/work/LINMA2472/LINMA2472/Lectures/biblio.bib", AlphaSt
```

```julia
biblio = load_biblio!()
```

```
ⓘ Loading bibliography from `/home/runner/work/LINMA2472/LINMA2472/Lectures/bibli
  o.bib`...

ⓘ Loading completed.
```

```
cite (generic function with 1 method)
```

```julia
cite(args...) = bibcite(biblio, args...)
```

```
bib (generic function with 1 method)
```

```julia
bib(args...) = bibrefs(biblio, args...)
```

```
draw_transformer (generic function with 2 methods)
```

```julia
function draw_transformer(decoder_only = true)
    scale(0.4, 0.4)
    Luxor.placeimage(readpng("images/transformer.png"), centered = true)
    if decoder_only
        sethue("red")
        setopacity(0.4)
        box(Point(-350, -160), Point(320, 20), :fill)
        box(Point(-350, 20), Point(0, 460), :fill)
        translate(Point(-170, -190))
        setopacity(1)
        fontsize(32)
        text("Not used for now", halign = :center)
    end
end
```

```
highlight (generic function with 1 method)
```

```julia
function highlight(a, b, c, d)
    sethue("green")
    setopacity(0.4)
    #box(Point(a, b), Point(c, d), :fill)
    polysmooth(box(Point(a, b), Point(c, d), vertices=true), 10, action = :fill)
    setopacity(1)
    polysmooth(box(Point(a, b), Point(c, d), vertices=true), 10, action = :stroke)
end
```

```julia
struct BPE
    text::String
    pairs::Dict{Tuple{Char,Char},Char}
end
```

add_pair (generic function with 1 method)

```
function add_pair(bpe::BPE, subs)
    pairs = copy(bpe.pairs)
    push!(pairs, subs)
    return BPE(replace(bpe.text, prod(subs.first) => subs.second), pairs)
end
```

pair_stats (generic function with 1 method)

```
function pair_stats(text::String)
    stats = Dict{Tuple{Char,Char},Int}()
    for i in eachindex(text)
        j = nextind(text, i)
        if j > lastindex(text)
            break
        end
        a = text[i]
        b = text[j]
        stats[(a, b)] = get(stats, (a, b), 0) + 1
    end
    return stats
end
```

substitute (generic function with 1 method)

```
function substitute(text::String, pair::Tuple{Char,Char})
    new_char = min('Z' + 1, minimum(text)) - 1
    return replace(text, prod(pair.first) => pair.second)
end
```

new_token (generic function with 1 method)

```
new_token(text::String) = new_token(BPE(text, Dict()))
```

new_token (generic function with 2 methods)

```
function new_token(bpe::BPE)
    stats = pair_stats(bpe.text)
    pair = findmax(stats)[2]
    new_char = min('Z' + 1, minimum(bpe.text)) - 1
    return add_pair(bpe, pair => new_char)
end
```

llms =

	Name	Num params	Ref	``n_\text
1	"Gemini-1.5"	missing	"[TGLB24]"	"256k"
2	"Gemini-1"	"1.[8B/3.25B](https://storage.googleapis.	"[TABA24]"	"256k"
3	"Gemma-2"	"27B"	"[TRPS24]"	"256k"
4	"Gemma-2"	"9B"	"[TRPS24]"	"256k"
5	"Gemma-2"	"2B"	"[TRPS24]"	"256k"
6	"Gemma"	"7B"	"[TMHD24]"	"256k"
7	"Gemma"	"2B"	"[TMHD24]"	"256k"
8	"GPT-2"	"1.5B"	"[RWCL19]"	"[50k](https://github
9	"Llama-2"	"7B"	"[TMSA23]"	"[32k](https://github
10	"GPT-4o"	missing	missing	"[200k](https://githu
⋮	more			
19	"big"	missing	"[VSPU17]"	"37k"

```
llms = load_llms()
```

load_llms (generic function with 1 method)

```
function load_llms()
    llms = DataFrame(CSV.File("llms.csv"))
    rename!(llms, "Embedding dimension" => "``d_\\text{emb}``")
    rename!(llms, "Vocabulary size" => "``n_\\text{voc}``")
    rename!(llms, "Context window" => "``n_\\text{ctx}``")
    rename!(llms, "Feed-Forward hidden dimension" => "``d_\\text{ff}``")
    return llms
end
```

▸ ["Name", "Num params", "Ref", "``n_\\text{voc}``", "``d_\\text{emb}``", "``n_\\text{ctx}`

```
names(llms)
```

```
table (generic function with 1 method)
```

```
function table(df; mandatory_columns = String[], included_columns = nothing)
    for col in mandatory_columns
        df = df[(!ismissing).(df[!, col]), :]
    end
    if !isnothing(included_columns)
        df = unique(df[!, included_columns])
    end
    Markdown.parse(pretty_table(
        String,
        sort(df),
        backend = :markdown,
        column_labels = names(df),
        allow_markdown_in_cells = true,
        formatters = [(v, _, _) -> ismissing(v) ? "" : v],
    ))
end
```

d =

	Name	B
1	"A"	"C"

```
d = DataFrame("Name" => String["A"], "B" => String["C"])
```

	Name	B
1	"A"	"C"
2	"a"	"d"

```
push!(d, ["a", "d"])
```