Model

大多数的模型流程可以简化成
Input -> {preprocess} -> {representation learning} -> {prediction} -> Output

给定一个输入字符串（string），LLM对应的运算可以切分成以下几个模块：

- tokenization：string -> token (id). token 是 LLM的最基本的特征单元，可以类比成图像识别中的pixel.
- embedding：token id -> token vector. 将 token id 向量化，作为 transformer 的输入，来支持计算 token vector 之间的 similarity.
- representation learning：[token vectors] -> [token vectors]. transformer 根据 token vector 的 similarity 层层 (deep) 计算每个 token 的 representation
- next-token prediction: [token vectors] -> [token id probabilities]. 根据 vector 计算其输出的 token id 的概率. 注意这里当前位置 token 的输出对应的是 next token 的概率.

I. Prerequisites

tokenization
目标：给定一个 vocabulary size 是 V 的 tokenizer，找到每个 word 对应的 token id (查字典)
function: text -> [token] -> [token id]
example: 'Here I am' -> ['Here', 'I', 'am'] -> [0, 1, 2]

(positional) embedding
目标：

1. 找到每个 token id 对应的维度为 C 的 embedding (字典注释): [0, 1, 2] -> [E0, E1, E2]
2. 每个 token embedding 加上对应的位置编码: [E0, E1, E2] <- [E0+P0, E1+P1, E2+P2]

II. Transformer (Block)

目标：根据 attention 机制，通过计算 token 之间相似性来学习 next token 对应位置的 embedding.
Input : [E0, E1, E2] # (dim: T x C)
Output: [E0', E1', E2'] # (dim: T x C)

(multi-head) causal attention

- to QKV matrix: [E0, E1, E2] -> { Q [Q0, Q1, Q2], K [K0, K1, K2], V [V0, V1,

V2] }

- matrix multiplications: 注意 causal mask 决定了每个 token 只能“看到”之前位置的 embedding

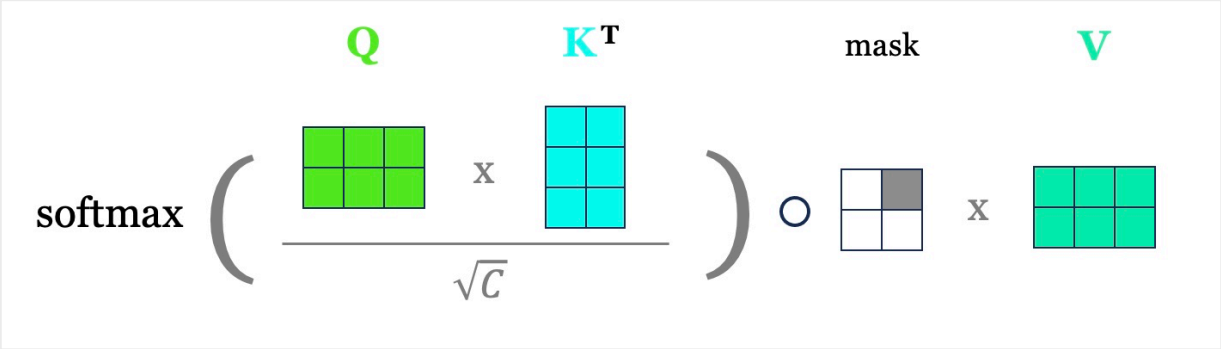


Figure 1: causal attention

position-wise feed-forward
对每个位置的 E 的非线性化投影，E′ <- ff(E).

(layer) normalization
对每个位置的 E 的独立的归一化.

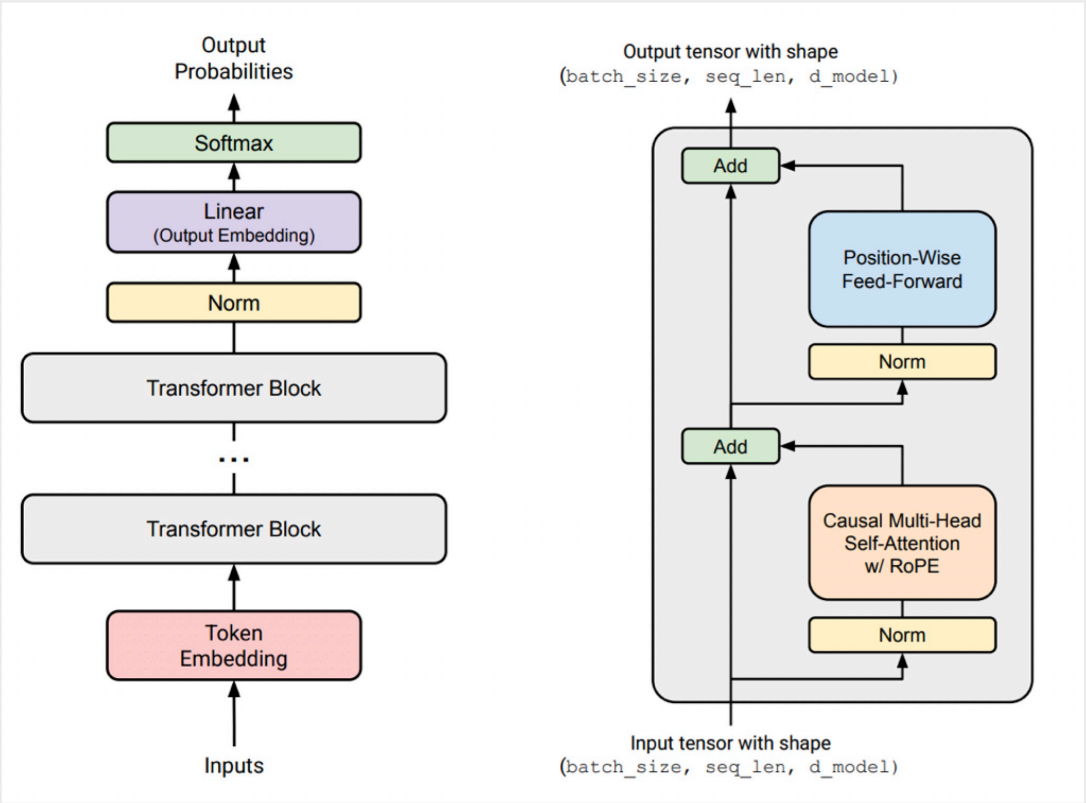
residual connection
same technique as ResNet.

III. Next-token Prediction

目标：根据 embedding 计算其所预测的 token 在 vocabulary 中的概率分布.
Input ：[E0′, E1′, E2′] # (dim: T x C)
Output: [P1′, P2′, P3′] # (dim: T x V)
注意每个位置的 P′ 对应于 next token，这也是 generative 的定义.

* References

model: Stanford cs336, Lecture 3
tokenizer: https://github.com/karpathy/minbpe
code: https://github.com/Lightning-AI/lit-llama



model **transformer**

Figure 2: from Stanford cs336, Lecture 3