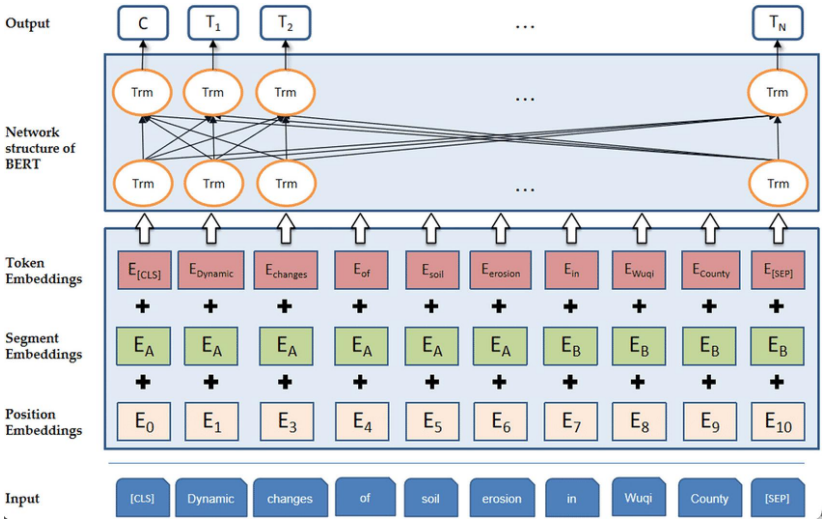
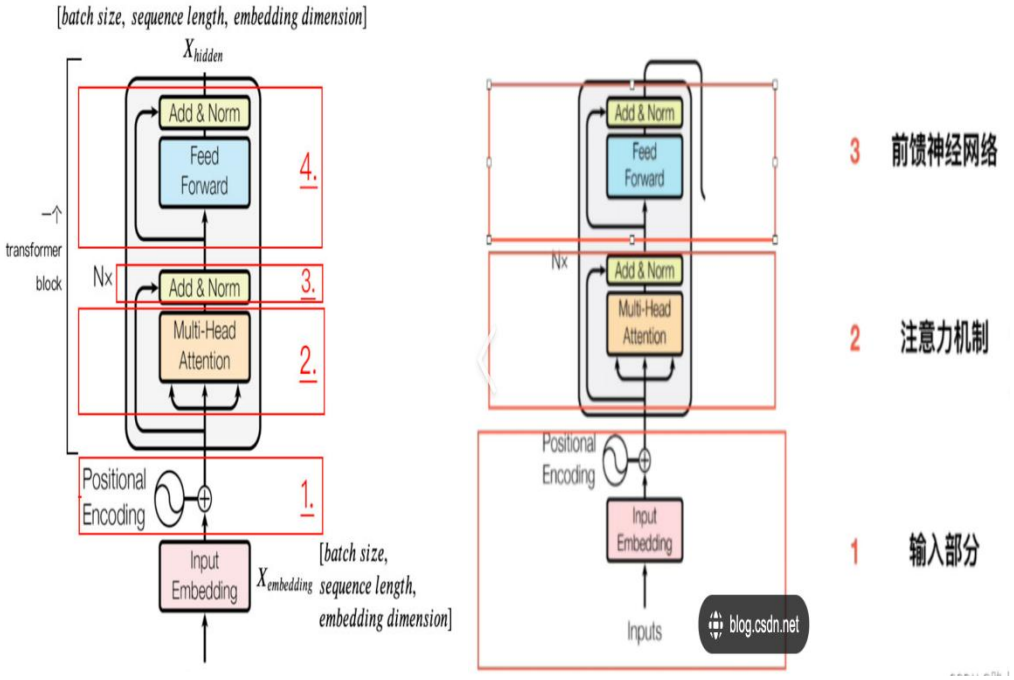
BERT

BERT 网络 Transformer 层数： 12 层堆叠的 Transformer 编码器，结构：



每一层通过自注意力机制和前馈神经网络提取文本的深层语义表示，最终形成对输入文本的双向编码。单层结构：



1. 整体架构

参数	BERT-base
Transformer层数	12层
隐藏层维度（hidden_size）	768
注意力头数（num_heads）	12（每头维度64）
总参数量	~110M

层级	学习内容	示例
底层（1-3层）	捕捉局部语法、词性、短语结构	识别名词、动词，判断主谓关系
中层（4-8层）	学习句法结构、简单语义关系	分析修饰关系（如形容词修饰名词）
高层（9-12层）	建模复杂语义、长距离依赖、任务相关抽象特征	理解段落主题、情感倾向、指代消解

2. 单层Transformer编码器的结构

每一层Transformer编码器包含以下核心组件：

2.1 多头自注意力机制（Multi-Head Self-Attention）

- **目的**：捕捉序列中任意两个token之间的依赖关系（无论距离远近）。
- **操作**：
  1. 将输入分为12个头（每个头维度64），并行计算注意力。
  2. 每个头生成独立的注意力权重矩阵，拼接后映射回768维。
- **公式**：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

2.2 残差连接与层归一化（Add & Norm）

- **残差连接**：将自注意力层的输入与输出相加，缓解梯度消失。
- **层归一化**：对相加后的结果进行归一化，加速训练收敛。

LayerNorm:

BERT 在 每个 Transformer 层中使用 LayerNorm,作用是保证每一层输出的数值稳定；保持训练过程的平滑；在预训练过程中防止梯度爆炸。

- Attention 后的残差连接之后（Residual + LayerNorm）
- Feed-Forward 后的残差连接之后（Residual + LayerNorm）

AE 中推荐使用 LayerNorm,用于稳定特征压缩和解码

对于一个输入向量 $x \in \mathbb{R}^d$（例如隐藏层输出向量），LayerNorm 的计算如下：

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

其中：

- $\mu = \frac{1}{d} \sum_{i=1}^{d} x_i$：均值（沿特征维度）
- $\sigma^2 = \frac{1}{d} \sum_{i=1}^{d} (x_i - \mu)^2$：方差
- ϵ：一个小数，防止除以 0（数值稳定性）
- γ, β：可学习参数（缩放和平移）

2.3 前馈神经网络（Feed-Forward Network, FFN）

- **结构**：两层的全连接网络，中间用激活函数（如GELU）连接。

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2$$

- **参数**：
  输入维度768 → 中间维度3072 → 输出维度768。

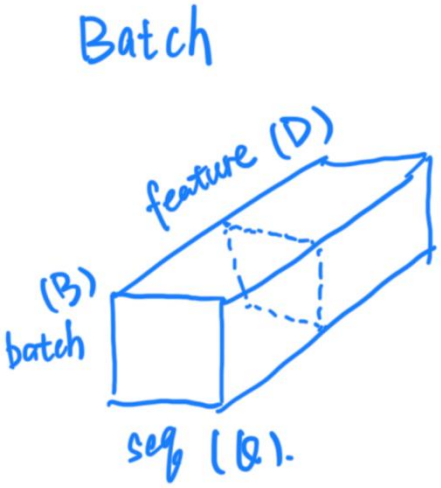
2.4 第二次残差连接与层归一化

- 与前一步类似，对FFN的输出再次进行残差连接和归一化。

BERT 的标准最大 token 长度为 **512**

例如，使用BERT-base时，一个Batch的形状可能为 `[32, 512, 768]`，其中：

- `batch_size=32`：同时处理32个样本。
- `sequence_length=512`：每个样本包含512个token（BERT的最大长度限制）。
- `hidden_size=768`：每个token的向量维度（由BERT-base的架构决定）。



B: Batch size.

Q: sequence Length.

D: Feature Dim (Hideden_size)

eg. Input Tensor: (2, 128, 768)

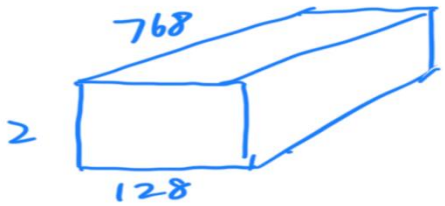
Batch Size: 2.

Seq Length: 128

Hideden Size: 768

Mutti Head: 12.

Head_dim (Feature dim) = $\frac{768}{12} = 64$



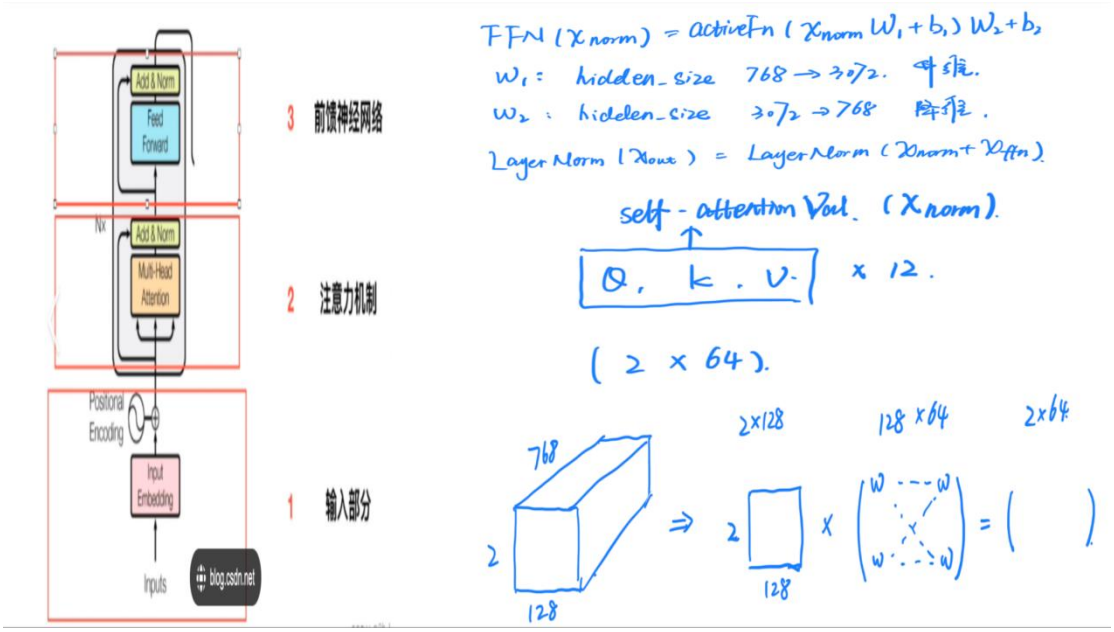
2个样本在序列长度为128

在64维向量空间中

矩阵表示.

- 64维表示为每个样本的64个维度.
  eg. 年龄、身高、体重、性别.
- 12头表示每个维度关注不同方面组合
  eg. Head1: 关注主谓一致关系.
  Head2: 捕捉指代关系.
  Head3: 处理局部修饰关系.

多头的目的是为了并行计算，提高计算速度.



$$h^{fusion} = \alpha \cdot \text{LayerNorm}(h^{sem}) + (1-\alpha) \cdot \text{LayerNorm}(h^{clu})$$

LayerNorm(h^768)和 LayerNorm(h^128)

1. 直接相加的问题

- **维度不匹配**：
  - `LayerNorm(h^768)` 输出形状：`[batch_size, ..., 768]`
  - `LayerNorm(h^128)` 输出形状：`[batch_size, ..., 128]`

  两者的最后一维（特征维度）不同，无法直接逐元素相加。

(1) 线性投影（推荐）

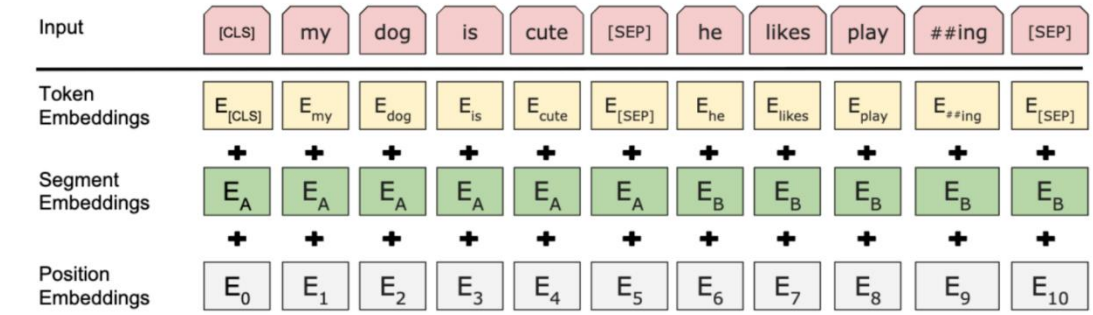
将低维向量（h^128）投影到高维空间（768维）：

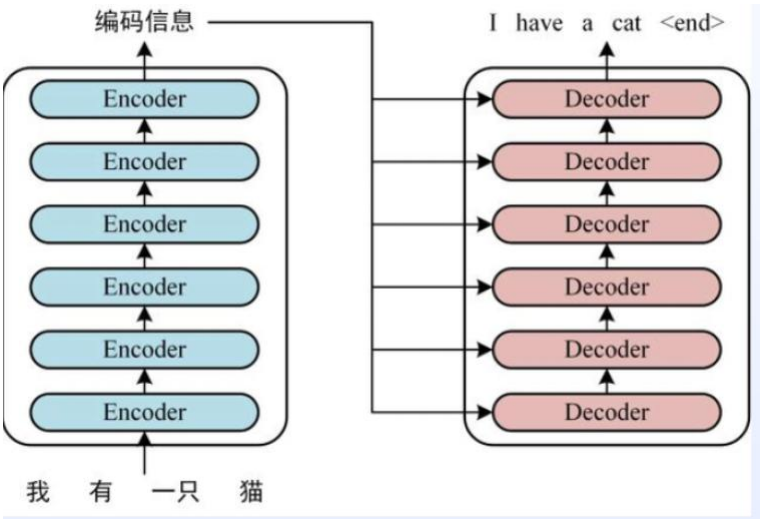
python Copy Download

```python
import torch.nn as nn

# 假设输入
h_768 = torch.randn(batch_size, seq_len, 768)  # LayerNorm(h^768)
h_128 = torch.randn(batch_size, seq_len, 128)  # LayerNorm(h^128)

# 方法1：使用全连接层升维
projection = nn.Linear(128, 768)
h_128_projected = projection(h_128)  # 形状变为 [batch_size, seq_len, 768]

# 相加
combined = h_768 + h_128_projected
```

表征学习：

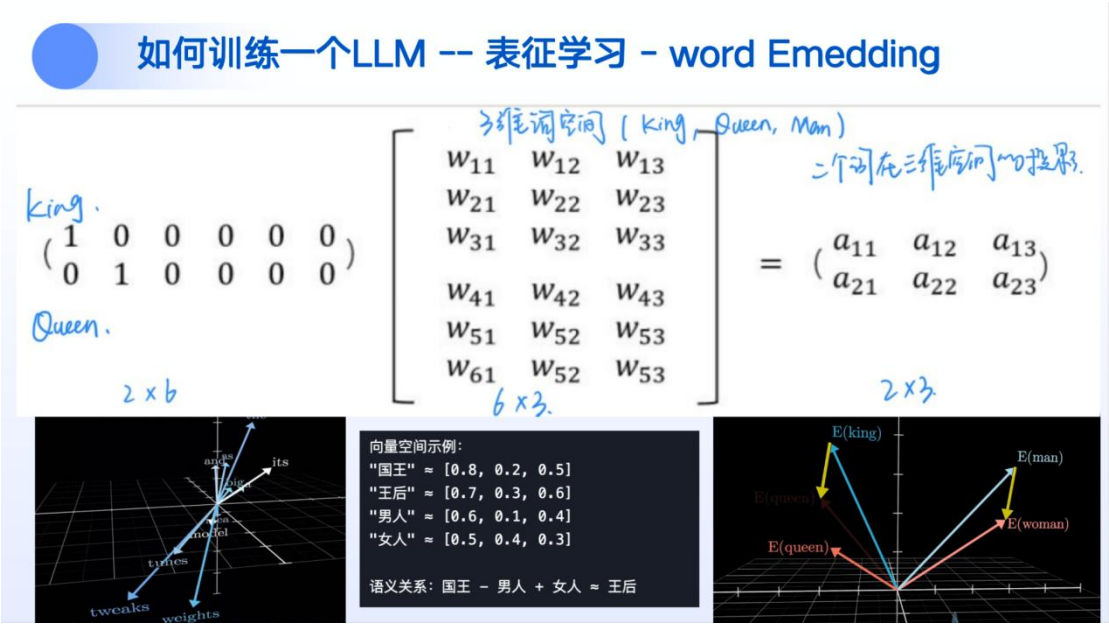
tokenizer -> word embedding -> Batch



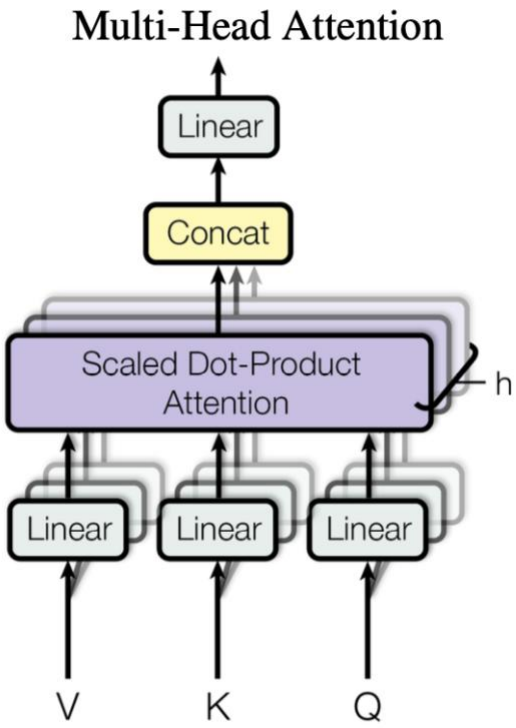


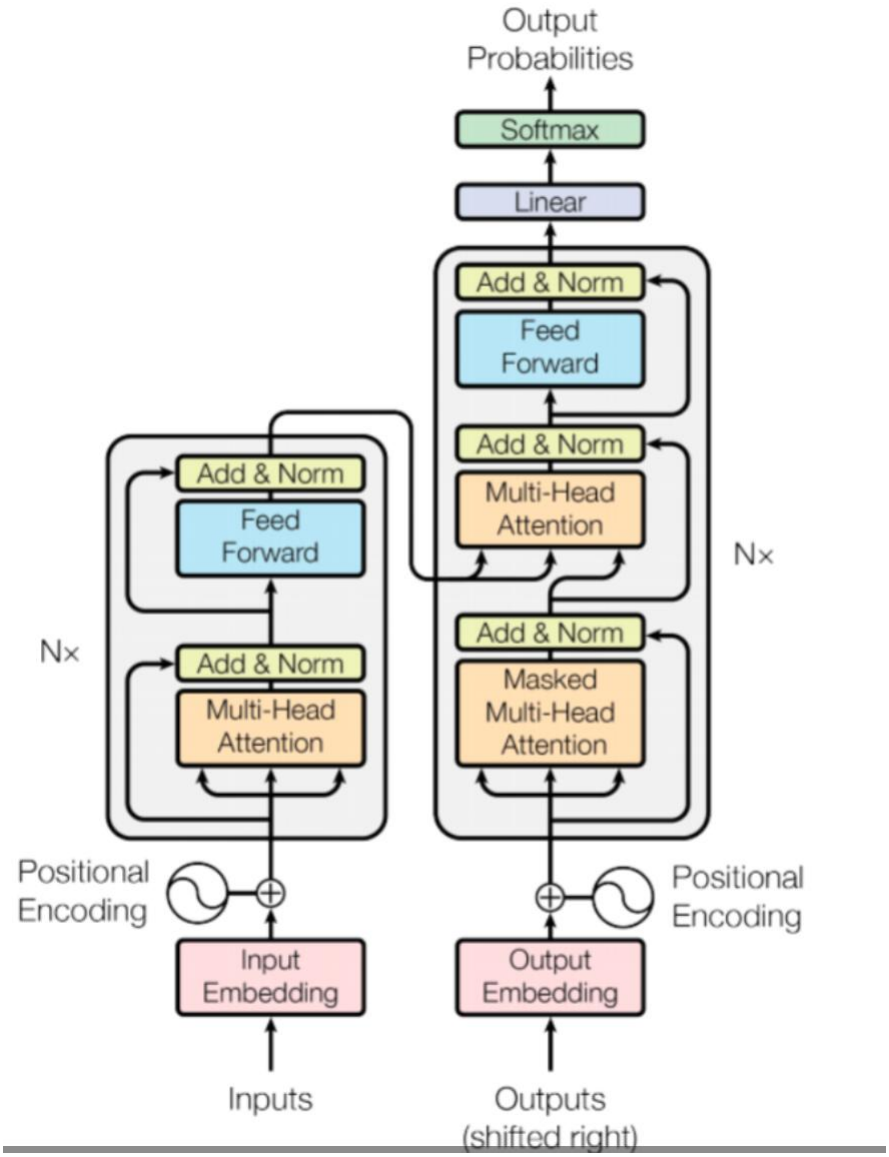
tokenlizer:

word enbedding: 词在词表空间的投影



Multi-Head Attention:




Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

MLM Head：

1. **BERT 预训练阶段**：
    - 输入：[CLS] I love [MASK] food.
    - 输出：[MASK] 的位置应该是 “spicy” → MLM Head 就是用来预测它的。
2. **下游任务中的表征学习**（如伪标签、对比学习）：
    - 有时会复用 MLM Head 的结构，用于生成 token-level 表征或预测词。

💡 **MLM Head** 就是接在 Transformer 主体（比如 BERT encoder）后的那一部分，用来对每个 token 的表示向量进行处理，输出词表大小的 logits（每个词的预测分数）。

一般包括以下几个部分（以 BERT 为例）：

```vbnet
MLM Head =
  LayerNorm(
    Dense(
      Hidden states from Transformer encoder
    )
  ) →
  Linear projection to vocabulary size
```

✅ 可视化对比（BERT 输出）

假设你输入一条评论：

```
css                                                    Copy  Edit

Input: [CLS] 这 款 手机 很 好 用 。 [SEP]
```

BERT 输出隐藏向量（每个 token 一个）：

```
css                                                    Copy  Edit

[768维][768维][768维]...[768维]
  ↑
[CLS] 向量（句子整体语义）
```

🧠 总结一句话：

> **BERT 的 `[CLS]` 向量就是输入句子整体语义的浓缩表示**，广泛应用于分类等句子级别的下游任务。

层次化特征融合机制：

对抗性损失函数（adversarial loss），用于对齐两种特征分布：语义特征（semantic features）和聚类特征（cluster features），以提升它们的融合效果

$$\mathcal{L}_{adv} = \mathbb{E}_{h^{sem}}[\log D(h^{sem})] + \mathbb{E}_{h^{clu}}[\log(1 - D(h^{clu}))]$$

1. h^{sem} 和 h^{clu}：
    - 分别表示语义特征（semantic features）和聚类特征（cluster features）的表示。
    - 这些是通过神经网络编码器提取的不同类型的特征。
2. $D(\cdot)$：
    - 是一个判别器（Discriminator），它试图区分输入特征是来自 h^{sem} 还是 h^{clu}。
    - 输出一个概率值，越接近1表示来自语义特征，越接近0表示来自聚类特征。

第一项：表示希望判别器能正确识别语义特征（判别为 1）。

第二项：表示希望判别器能正确识别聚类特征（判别为 0）

目的与机制：

该损失函数本质上是一个**二分类的对抗训练目标**，与GAN（生成对抗网络）中的判别器损失类似。

通过引入**梯度反转层（GRL）**，在反向传播时：

- 编码器将被训练使得 h^{sem} 和 h^{clu} 无法被判别器区分（即两者的分布变得相似）；
- 判别器则努力区分这两种特征。

总结：

这个公式实现了语义特征与聚类特征的**分布对齐（distribution alignment）**。借助对抗性训练，使得这两类特征在分布空间上更为接近，从而促进它们的融合，提升最终模型的表示能力。