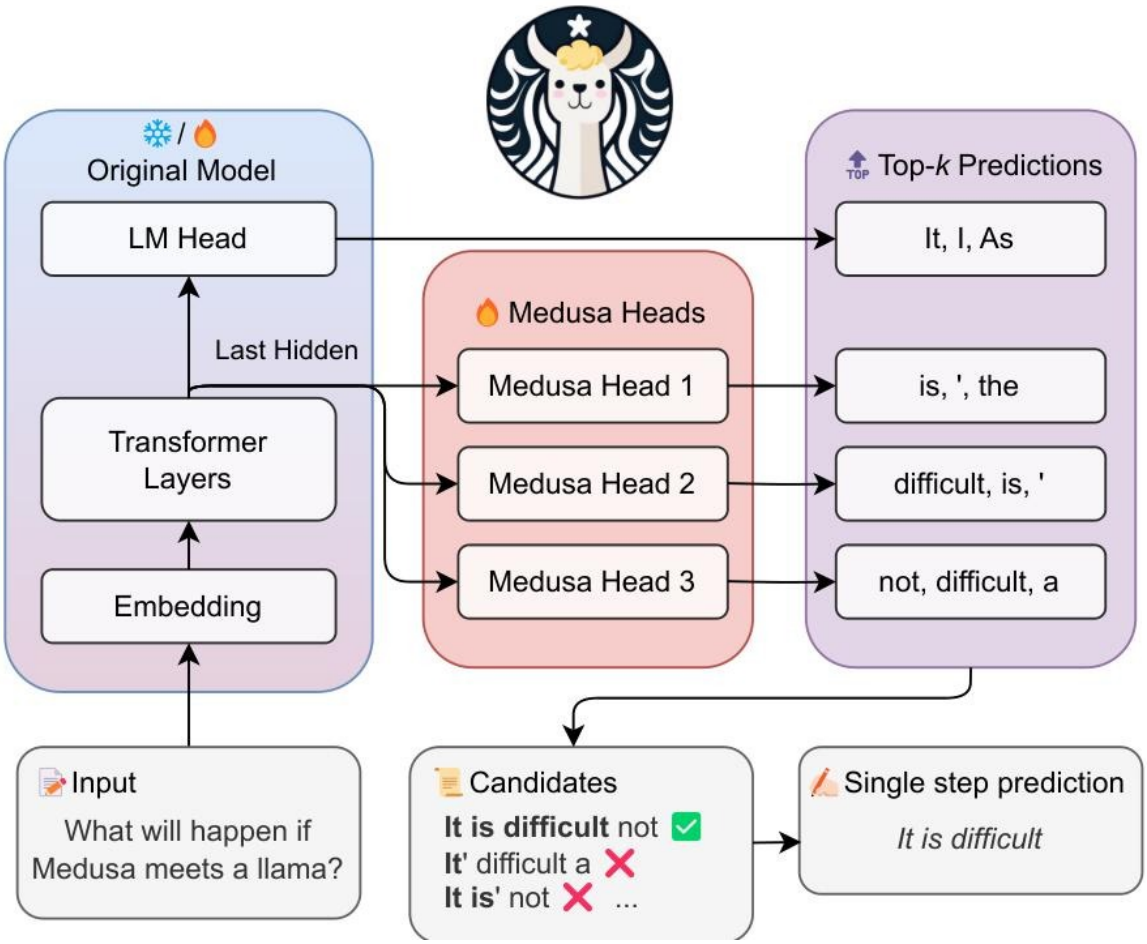
Medusa



> 本文介绍一下Medusa这个方法，通过这篇论文学习树注意力机制。

Medusa遵循和投机解码相同的框架，可以分为三个步骤：(1) 生成候选，通过Medusa 头实现；(2) 处理候选，通过树注意力实现； (3) 接受候选，通过拒绝采样或者典型接受实现。

MEDUSA 头

设原始模型在位置t的最后一层隐藏状态为 h_t ，通过在LLM的最后一层隐藏状态上增加K个解码头，每个头使用带残差的单层FFN，适合分布式部署。其中第k个解码头用于预测第 $t+k+1$ 个位置的词，预测记为 p_t^k 。原始模型的头用于预测第 $t+1$ 个位置，预测记为 p_t^0 。第k个头的定义的预测为：

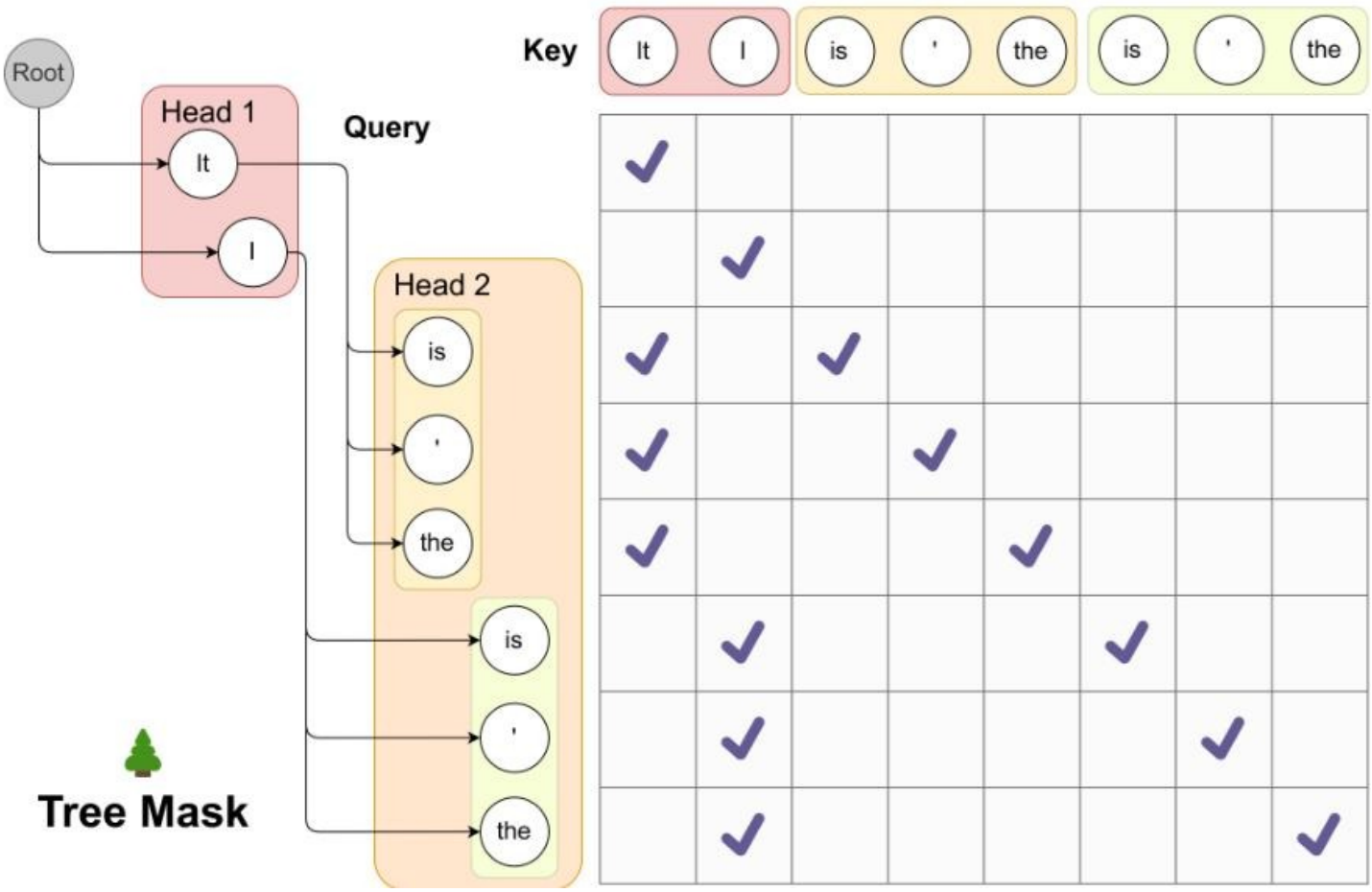
$$p_t^{(k)} = \text{softmax}\left(W_2^{(k)} \cdot \left(\text{SiLU}(W_1^{(k)} \cdot h_t) + h_t\right)\right),$$
$$\text{where } W_2^{(k)} \in \mathbb{R}^{d \times V}, W_1^{(k)} \in \mathbb{R}^{d \times d}.$$

其中 d 表示隐藏层的维度，V是词表的维度。 $W_2^{(k)}$ 初始化为原语言模型head权重， $W_1^{(k)}$ 初始化为0，保证MEDUSA 头的初始预测与原始模型一致。MEDUSA的头部跟主干模型一起训练，能利用到大模型的强大表征能力，并且确保MEDUSA头部的分布与原始模型一致，减轻分布偏移问题。

Tree Attention

通过MEDUSA头获取了后续K+1个预测token，传统的投机解码方法每个头只采样一个候选。MEDUSA为每个头生成多个候选token，以提升预测token的接受率（从top1变成了top5，自然预测正确的概率就高了），为了平衡计算量，采用树注意力来同时处理多个候选。

对于第k个头，预测 $top\text{-}s_k$ 个token，然后根据每个头生成预测的笛卡尔积来构建出多个序列，形成树状候选空间。如左图所示，$s_1 = 2$ 和 $s_2 = 3$，构成了6个候选序列，其对应注意力掩码矩阵如右图。Token 树中的token数为 $\sum_{k=1}^{K}\prod_{i=1}^{k} s_i = 8$，虽然解码阶段的计算量比较低，但也不意味着可以无限增加要生成的token数。



稀疏树

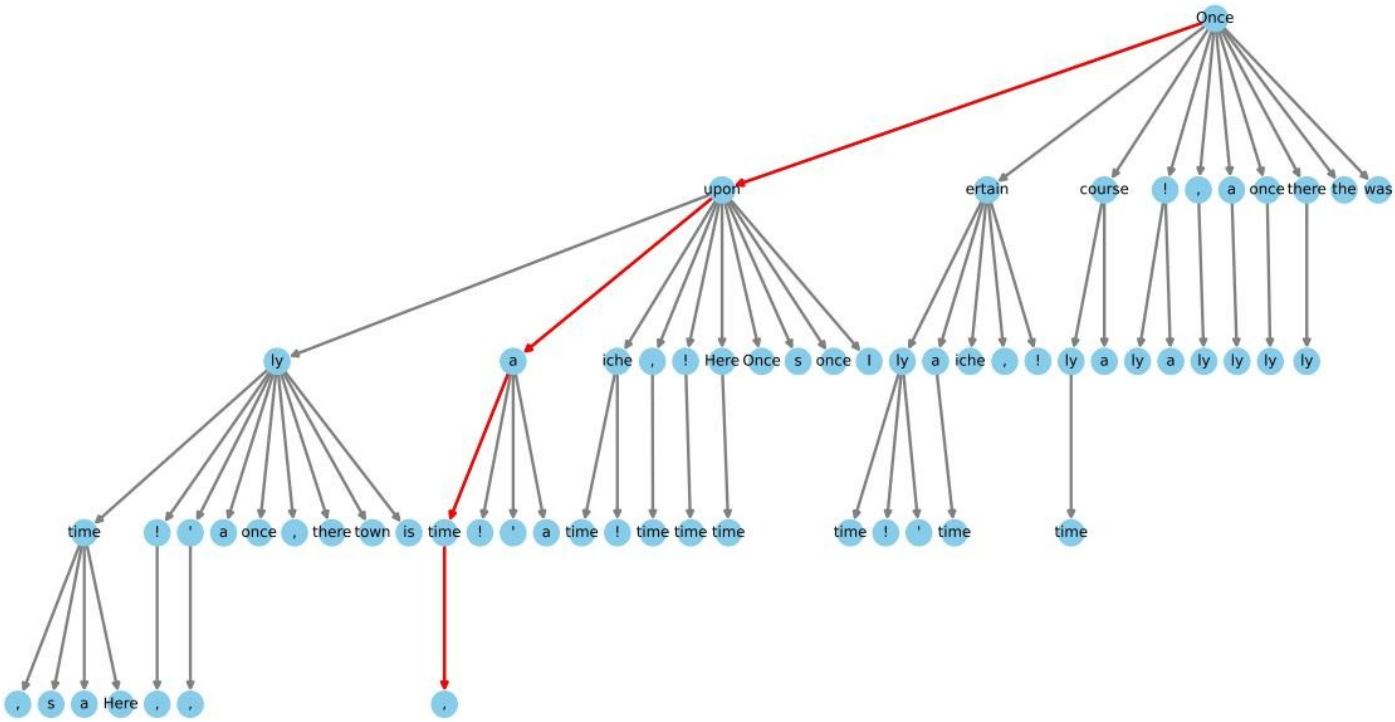
每个Medusa 头都取top-k，然后做笛卡尔积构建树是一种最简单的实现（稠密树），但某些Token准确率很低，计算这些Token会浪费算力，特别是在节点数受限的场景。

因此作者提出在校准集上统计每个Medusa 头的top-i候选的准确率，第k个头的第i名准确率记为 $a_k^{(i)}$。假设各个Medusa 之间独立，一个由 [i1, i2, ..., ik] 组合成的候选token序列，其整体准确率约为 $\prod_{j=1}^{k} a_j^{(i_j)}$。

- **目标：** 给定节点数预算，构造一棵能**最大化候选被接受长度期望**的最优稀疏树。

- 枚举计算所有候选序列的准确率，再贪心地把最高准确率高的分支优先加入树，直到节点数用完。
- 这种方法能够得到一棵“稀疏但高效”的树，显著提升树中被采纳的平均prefix长度，从而提升推理效率和质量。

例如下图所示，有4个Medusa 头，64个节点，每个节点表示一个Medusa 头的前top-k预测token，每个头根据在Alpaca-eval数据集上测量的准确率进行修建，红色线表示正确预测未来token的路径。



训练策略

假设有与原始模型输出分布一致的数据集（一般为SFT用的同源数据，没有分布一致的数据集可以通过以下介绍的自蒸馏方法获取），为了提升MEDUSA头预测未来token的能力，有两种方案：MEDUSA-1（**只训练解码头**）和MEDUSA-2（**主干与解码头一起训练**）。

- **MEDUSA-1：**利用MEDUSA头和真实值之间的交叉熵损失，设在第 $t+k+1$ 处的真实token为 y_{t+k+1} ， $p_t^{(k)}(y)$ 为第k个头预测为y的概率，第k个头预测的损失为 $L_k = \sum_{k=1}^{K} -\log p_t^{(k)}(y_{t+k+1})$ 。k越大预测越不稳定，因此预测的损失越大，因此在损失中添加权重 λ_k 以平衡各个头（实际中用 0.8 的k次幂）：

$$L_{MEDUSA-1} = \sum_{k=1}^{K} -\lambda_k \log p_t^{(k)}(y_{t+k+1})$$

为了进一步加速训练效率和节省显存空间，还可以对主干模型进行QLoRA量化。

- **MEDUSA-2**：为了进一步提升MEDUSA头的预测精度，可以将主干与MEDUSA头一同训练，为了保留主干模型预测下一个token的能力和预测精度，采用了以下策略：
  - **组合损失**：将主干模型的交叉熵损失加到MEDUSA损失中：

$$L_{MEDUSA-2} = L_{LM} + \lambda_0 L_{MEDUSA-1}$$

  - **不同的学习率**：主干模型学习率低而MEDUSA头学习率高，促使MEDUSA 头更快收敛且主干保持原有能力。
  - **Medusa头预热**：由于MEDUSA头中 W_1 初始化为0，训练初期 $L_{MEDUSA-1}$ 比较大，导致较大的梯度并可能破坏主干模型。因此采用两阶段训练，第一阶段只训练MEDUSA头，第二阶段使用warmup策略一起训练。

通过以上策略，**MEDUSA-2可与SFT流程无缝融合，获得“原生支持MEDUSA”的LLM**。

MEDUSA头数：最多5个头。

典型接受

在MEDUSA框架下，每步会并行生成多个候选序列，接下来的问题就是：**如何在众多候选序列中判断哪些可以被采纳?**

- 传统的投机解码论文采用**拒绝采样**，即先生成一批候选token，然后用原始模型“验收”每个候选序列，只有通过验收的才被采纳，从而保障输出不会偏离原始模型本来的分布。拒绝采样的问题在于，一方面引入了额外的开销，尤其是drafter模型和原始模型分布差异大时，另一方面为了保障多样化的输出，不应该要求完全匹配原始模型的分布。
- **典型接受（Typical acceptance）**不要求输出分布与原模型一模一样，而是在保证多样性和高效性的前提下，优先接受“原始模型也认为合理”（生成概率不是极低）的典型序列。具体而言，给定上下文 $x_1, \ldots, x_t$ ，对于候选token序列 $x_{n+1}, \ldots, x_{n+K+1}$ ，依次计算每个token在原始模型下的概率，满足以下条件即可被接受：

$$p_{\text{original}}(x_{n+k}|x_1, \ldots, x_{n+k-1}) > \min\left(\epsilon, \delta \exp(-H(p_{\text{original}}(\cdot|x_1, \ldots, x_{n+k-1})))\right)$$

$H(\cdot)$ 是原始模型的熵函数， ϵ、δ 是阈值参数，这就保证了概率高的token和熵比较高的序列都会被接受。并且为了保证每一步至少生成一个 token，我们对第一个 token 使用贪婪解码并无条件接受它，而在后续 tokens 的处理中采用典型接受。如果某个token不满足条件，则后续序列都被拒绝，继续验证下一个候选，每一步最终的预测由所有候选中接受的最长前缀决定。

> 🦄 **典型接受总结**
>
> 相较严格的拒绝采样，**Typical Acceptance更宽容，速度更快，同时输出质量相似**。只要序列在原始模型下“不是很奇怪”，即可被快速采纳，不会引入无意义/低质量token。并且无需严格复现原始模型分布，能适应分布波动，容忍一定偏差，利于工程部署和推理加速。

自蒸馏

在没有与原始模型输出分布匹配的训练数据集的情况下（例如未开源原始的SFT数据，或者模型经过RLHF，输出的分布与训练集不同），可以通过**自蒸馏**的方式为MEDUSA头训练生成数据集。使用与原始模型领域相似的公开种子数据集里的提示（例如ShareGPT用于聊天模型），喂给原始模型自己生成回答，形成符合原始模型输出分布的QA对。

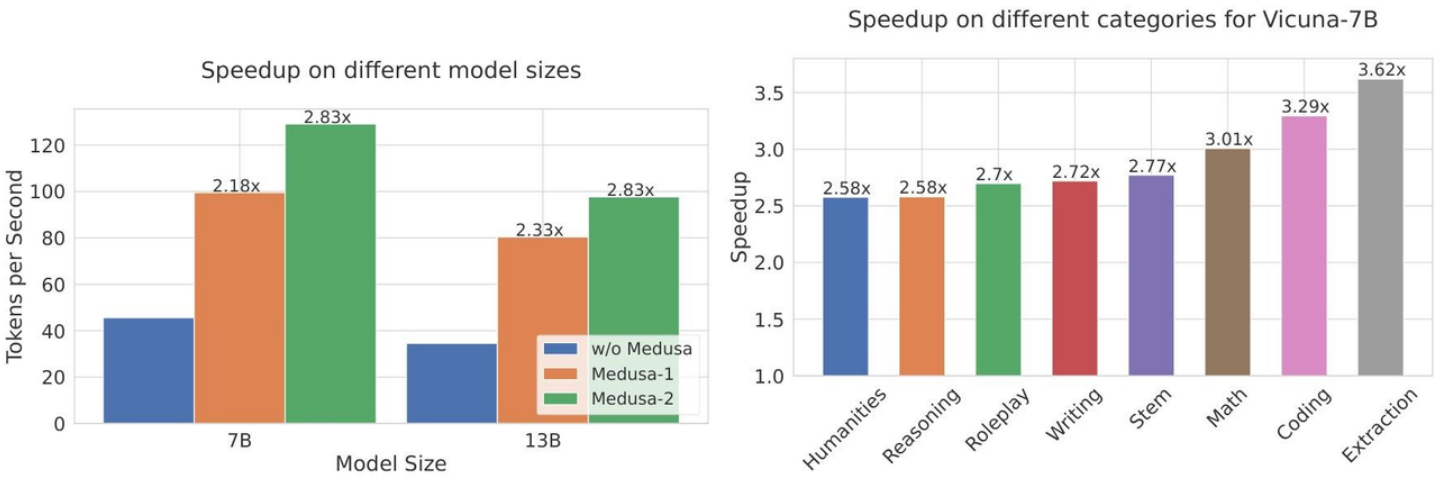
- 对于**MEDUSA-1**来说，直接用该数据集训练MEDUSA 头即可。
- 对于**MEDUSA-2，仅用自蒸馏数据会降低模型生成质量**，为了避免主干模型能力退化，用原始模型的概率分布作为label，而不是one-hot真实token作为label，主干模型的损失函数为：

$$L_{\text{LM-distill}} = KL(p^{(0)}_{\text{original},t} \| p^{(0)}_{t})$$

其中 $p^{(0)}_{\text{original},t}$ 表示原始模型在位置t的预测概率分布。为了缓解内存压力，采用LoRA微调来训练主干模型，原始模型则是关闭LoRA的主干模型，而无需采用量化。

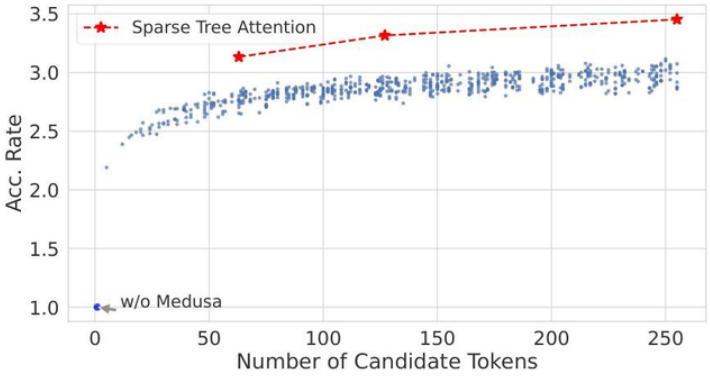
实验结果

实验结果如下图所示，作者在 Vicuna 模型上实验，使用ShareGPT数据集在7B和13B模型上训练MEDUSA头2个epoch，训练可以在一张A100上完成，训练时间只需要几个小时。在7B和13B模型上Medusa-1能取得2.18x 和 2.33x 的加速，Medusa-2能获得2.83x 的加速。右图在不同的任务类型进行了实验，也可以获得 2 倍以上的加速。这是主要因为一方面**投机解码本身就比自回归解码效率更高**，另一方面通过**树注意力生成了更多得候选序列用于验证**，并采用了更宽松的**典型接受方法，提高了每一次解码接受token数的期望**。

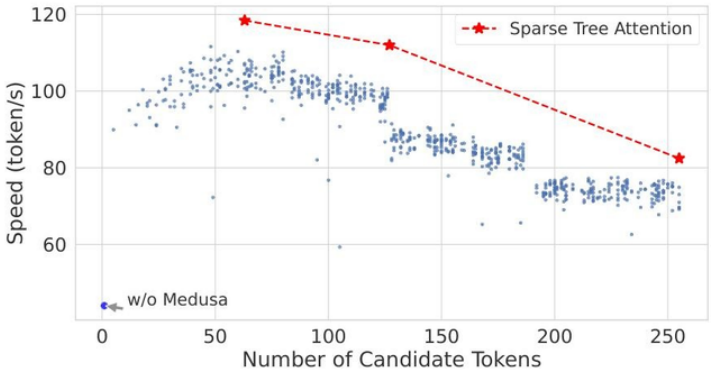


(a) (b)

前文提到Medusa头可以生成多个候选token，然后采用笛卡尔积的方式获取候选序列，更多的候选序列会提升投机解码的加速效果（左图），但节点数越多会导致计算量越大。如右图所示，引入更多节点消耗了大量的计算资源，反而减慢了推理速度，因为Decode阶段虽然是通信密集型任务，但引入过多的计算会导致运算强度超过GPU算力，导致计算时延显著增加。此外，还观察到并且采用稀疏树的效果要比稠密树更佳。



(a)



(b)

参考文章：

- 原论文：《MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads》
- 代码地址：《Medusa: Simple Framework for Accelerating LLM Generation with Multiple Decoding Heads》