Version 4

模拟PT中多组Z node的情况修改Transformer的残差连接：

对于注意力机制计算中的QK^T矩阵，改成$\sum_i QK_i^T$，即存储之前所有层计算得的QK^T矩阵，计算累加结果。所以新的attn_weight矩阵为Softmax($\frac{\sum_i QK_i^T}{d_s}$)。

令之前所有层计算得的QK^T矩阵为$\vec{qk}$，维数为当前transformer层数m。令scaling矩阵为$\vec{s}$，维数为当前transformer层数m。所以attn_weight矩阵可统一改写为：Softmax$(\vec{qk}\cdot\vec{s})$。对于$\vec{s}$取值有以下7种方法的尝试：

Method1

对于第m层，将$\frac{1}{\sqrt{d_k}}$广播为向量$\vec{s}$

Method2

对于第m层，将$\frac{1}{\sqrt{d_k * m}}$广播为向量$\vec{s}$

Method3

对于第m层，将$\frac{1}{m\sqrt{d_k}}$广播为向量$\vec{s}$

Method4

对于第m层，将$\frac{1}{d_k^{a_m} * m^{b_m}}$广播为向量$\vec{s}$。其中$a_m, b_m$为可学习参数，正负性不做限定，初始值分别为0.5,1

Method5

对于第m层，$\vec{s_i} = \frac{1}{a_{mi}\sqrt{d_k}}$，其中$a_{mi}$为可学习参数，限定为正值，初始值均为1

Method6

对于第m层，$\vec{s_i} = \frac{1}{a_{mi} d_k^{b_{mi}}}$，其中$a_{mi}, b_{mi}$为可学习参数，$a_{mi}$限定为正值，初始值均为1；$b_{mi}$正负性不做限定，初始值为0.5

Method7

对于第m层，$\vec{s_i} = \frac{1}{a_{mi}}$，其中$a_{mi}$为可学习参数，限定为正值，初始值为$\sqrt{d_k}$

其余对照方法

Baseline

Version2_Method1

CLT复现。MLP处，第m层残差连接为$\frac{\sum_{i=1}^{m-1} \vec{y_i'} + \vec{y_m'}}{m} + \vec{y_m}$。Attention处残差不变

Version2_Method3

CLT的微调方案。将每一层权重$\frac{1}{m}$改为可学习参数。具体来说：除第0层外，每层都拥有一个独立的权重列表，第m层权重列表长度为m+1，涵盖了0-m层所有层的可学习权重。对权重做Softmax归一化后得到每层权重分布。

Version3_Method1

修改MLP处的残差连接为累加先前层MLP输出&该层MlpInput。与CLT不同之处在于先前层MLP输出需要重算

Version3_Method3.1

与Method1基本相同，唯一不同之处在于MLP处残差和进行了归一化，且每一层的权重分布为1/m

Version3_Method3.2

与Method1基本相同，唯一不同之处在于MLP处残差和进行了归一化，且每一层的权重分布为可学习分布

训练结果

性能分类（性能从好到坏，按acc从高到低）：

三条参考线为：Baseline，Version2_Method1（CLT性能），Version2_Method3（微调CLT性能）

- 没有性能高于v2m3的新模型
- **Version2_Method3**
- v3m1,v3m3.1,v3m3.2,v4m1
- **Baseline**
- v4m6,v4m7
- **Version2_Method1**
- v4m2,v4m3,v4m4,v4m5

(全部训练完后进行分类排序)

模型训练状态监控系统

2025/07/26 11:33:39

18	18	0
总模型数	已完成训练	正在训练

base

已完成

准确率:	50.21%
损失值:	2.5297
困惑度:	12.55
训练轮次:	5

base_2

已完成

准确率:	50.21%
损失值:	2.5297
困惑度:	12.55
训练轮次:	5

v2m1

已完成

准确率:	50.10%
损失值:	2.5340
困惑度:	12.60
训练轮次:	5

v2m3

已完成

准确率:	50.30%
损失值:	2.5199
困惑度:	12.43
训练轮次:	5

v3m1

已完成

准确率:	50.25%
损失值:	2.5253
困惑度:	12.49
训练轮次:	5

v3m3_1

已完成

准确率:	50.24%
损失值:	2.5259
困惑度:	12.50
训练轮次:	5

v3m3_2

已完成

准确率:	50.22%
损失值:	2.5261
困惑度:	12.50
训练轮次:	5

v4m1

已完成

准确率:	50.25%
损失值:	2.5293
困惑度:	12.54
训练轮次:	5

v4m2

已完成

准确率:	50.01%
损失值:	2.5512
困惑度:	12.82
训练轮次:	5

v4m3

已完成

准确率:	49.93%
损失值:	2.5556
困惑度:	12.88
训练轮次:	5

v4m4

已完成

准确率:	50.06%
损失值:	2.5451
困惑度:	12.74
训练轮次:	5

v4m4_2

已完成

准确率:	50.06%
损失值:	2.5451
困惑度:	12.74
训练轮次:	5

v4m5

已完成

准确率:	50.00%
损失值:	2.5470
困惑度:	12.77
训练轮次:	5

v4m5_2

已完成

准确率:	50.00%
损失值:	2.5470
困惑度:	12.77
训练轮次:	5

v4m6

已完成

准确率:	50.16%
损失值:	2.5420
困惑度:	12.70
训练轮次:	5

v4m6_2

已完成

准确率:	50.16%
损失值:	2.5420

v4m7

已完成

准确率:	50.10%
损失值:	2.5443

v4m7_2

已完成

准确率:	50.15%
损失值:	2.5429

注：v3m3_1和v3m3_2对应的是Version3_Method3.1和Version3_Method3.2两种不同的方法。其他带下划线的模型名称均为同一方法重复训练了2次。

附录：v4m4 v4m5 v4m6 v4m7的习得参数及

分析

v4m4（分母为$d_k^{a_m} * m^{b_m}$）

```
Method4_v4 Learned Parameters
======================================
Total layers: 8
Parameters per layer: a (power for d_k), b (power for m)

Layer 0:
  a = -0.075787 (d_k^-0.076)
  b = 0.078046 (m^0.078)
  Scaling formula: 1/(d_k^-0.076 * m^0.078)
------------------------------
Layer 1:
  a = 0.015322 (d_k^0.015)
  b = 0.054345 (m^0.054)
  Scaling formula: 1/(d_k^0.015 * m^0.054)
------------------------------
Layer 2:
  a = 0.113107 (d_k^0.113)
  b = 0.152144 (m^0.152)
  Scaling formula: 1/(d_k^0.113 * m^0.152)
------------------------------
Layer 3:
  a = 0.190449 (d_k^0.190)
  b = 0.229469 (m^0.229)
  Scaling formula: 1/(d_k^0.190 * m^0.229)
------------------------------
Layer 4:
  a = 0.219766 (d_k^0.220)
  b = 0.258763 (m^0.259)
  Scaling formula: 1/(d_k^0.220 * m^0.259)
------------------------------
Layer 5:
  a = 0.276817 (d_k^0.277)
  b = 0.315849 (m^0.316)
  Scaling formula: 1/(d_k^0.277 * m^0.316)
------------------------------
Layer 6:
  a = 0.330740 (d_k^0.331)
```

```
  b = 0.369803 (m^0.370)
  Scaling formula: 1/(d_k^0.331 * m^0.370)
------------------------------
Layer 7:
  a = 0.393176 (d_k^0.393)
  b = 0.432214 (m^0.432)
  Scaling formula: 1/(d_k^0.393 * m^0.432)
------------------------------
```

总结：

- 习得参数大小随层数上涨呈现递增趋势。也就是说模型认为层数越高，累积的先前层越多，每层QK^T被压缩程度应当越高（d_k的指数越大，分母越大）。符合直觉。
- 习得参数的次数和（a+b），与0.5相比（原模型为$d_k^{0.5}$）没有必然关系。且d_k的习得参数总是小于m，也许是模型认为d_k对于压缩系数（分母）的影响没有m重要（分母中，m次数高，被幂次压缩的程度相对更小）？

v4m5（分母为$a_{mi}\sqrt{d_k}$）

```
Method5_v4 Learned Parameters
========================================
Total layers: 8
Parameters: Vector a_i for scaling 1/(a_i*sqrt(d_k))

Layer 0 (vector length: 1):
  Parameters: [0.59551203]
  Mean: 0.595512
  Std: 0.000000
  Min: 0.595512
  Max: 0.595512
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 1 (vector length: 2):
  Parameters: [0.6737704 0.6316532]
  Mean: 0.652712
  Std: 0.021059
  Min: 0.631653
  Max: 0.673770
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 2 (vector length: 3):
  Parameters: [1.2201475 1.1302449 0.487827 ]
  Mean: 0.946073
  Std: 0.326101
  Min: 0.487827
  Max: 1.220147
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 3 (vector length: 4):
  Parameters: [1.6988657 1.2097582 1.0033813 0.500667 ]
  Mean: 1.103168
  Std: 0.429878
  Min: 0.500667
  Max: 1.698866
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 4 (vector length: 5):
  Parameters: [2.1834269  1.3578908  1.3780731  0.67840976 0.46795648]
  Mean: 1.213151
```

```
  Std: 0.605119
  Min: 0.467956
  Max: 2.183427
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 5 (vector length: 6):
  Parameters: [2.3978093 1.3637199 1.6458268 1.422701  1.1910692 0.5941637]
  Mean: 1.435882
  Std: 0.539009
  Min: 0.594164
  Max: 2.397809
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 6 (vector length: 7):
  Parameters: [3.4134254  1.831575   2.5524118  1.1480625  0.99837166 1.2432308
 0.8380876 ]
  Mean: 1.717881
  Std: 0.880418
  Min: 0.838088
  Max: 3.413425
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
Layer 7 (vector length: 8):
  Parameters: [3.4769256 1.7098973 2.4375205 1.2935399 1.3714167 2.0177674 1.8446631
 1.0907063]
  Mean: 1.905305
  Std: 0.718559
  Min: 1.090706
  Max: 3.476926
  Scaling formula: weighted sum with 1/(a_i*sqrt(d_k))
------------------------------
```

总结：

- a_{mi}实际上是控制第m层的先前层QK^T矩阵权重压缩程度的参数。
- 随层数增加，a_mi整体成增加趋势，压缩系数越大。符合直觉。
- 同一层对应的a_mi序列，先前层的层数越低（距当前层越远），a_mi越大，被压缩程度越大。说明层数越靠前，在加权和中的重要性越低。符合直觉。

v4m6（分母为$a_{mi}d_k^{b_{mi}}$）

```
Method6_v4 Learned Parameters
========================================
Total layers: 8
Parameters: Vectors a_i and b_i for scaling 1/(a_i * d_k^b_i)

Layer 0 (vector length: 1):
  a_params: [1.0261725]
  a_mean: 1.026173, a_std: 0.000000
  b_params: [0.06484602]
  b_mean: 0.064846, b_std: 0.000000
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 1 (vector length: 2):
  a_params: [1.0446548 1.019886 ]
  a_mean: 1.032270, a_std: 0.012384
  b_params: [0.08270097 0.05869877]
  b_mean: 0.070700, b_std: 0.012001
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 2 (vector length: 3):
  a_params: [1.2458147 1.234046  1.0329162]
  a_mean: 1.170926, a_std: 0.097706
  b_params: [0.25877285 0.24926989 0.07140473]
  b_mean: 0.193149, b_std: 0.086174
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 3 (vector length: 4):
  a_params: [1.3618367 1.2592884 1.2171055 1.0180739]
  a_mean: 1.214076, a_std: 0.124803
  b_params: [0.34787902 0.26959047 0.2354987  0.05694263]
  b_mean: 0.227478, b_std: 0.106556
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 4 (vector length: 5):
  a_params: [1.4983184 1.3290555 1.3596028 1.117755  1.0797607]
  a_mean: 1.276899, a_std: 0.156702
  b_params: [0.4433707  0.32352433 0.34622875 0.15035088 0.1157584 ]
  b_mean: 0.275847, b_std: 0.123829
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
```

```
Layer 5 (vector length: 6):
  a_params: [1.4725072 1.2975746 1.3824346 1.3200445 1.3491977 1.1710941]
  a_mean: 1.332142, a_std: 0.091105
  b_params: [0.42594677 0.29956022 0.36288306 0.3166899  0.3385353  0.19692843]
  b_mean: 0.323424, b_std: 0.069417
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 6 (vector length: 7):
  a_params: [1.5996022 1.3738438 1.5512241 1.3033801 1.2886906 1.394163  1.2700971]
  a_mean: 1.397286, a_std: 0.120698
  b_params: [0.50863063 0.35668734 0.478061   0.30408898 0.2927198  0.37131837
 0.2781189 ]
  b_mean: 0.369946, b_std: 0.084374
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
Layer 7 (vector length: 8):
  a_params: [1.7275689 1.4052073 1.6333361 1.2568688 1.3307034 1.5554422 1.5324541
 1.3940474]
  a_mean: 1.479454, a_std: 0.149291
  b_params: [0.58576643 0.3791555  0.52969813 0.2675872  0.32476252 0.48078758
 0.4659001  0.3712876 ]
  b_mean: 0.425618, b_std: 0.100873
  Scaling formula: weighted sum with 1/(a_i * d_k^b_i)
------------------------------
```

总结：

- 关于a_mi
  - 层数越高，a_mi总体越高
  - 同一层中，层数越靠前，a_mi数值总体越高
  - 规律与v4m5相同，但相比之下规律非常不明显，且存在较多例外
- 关于b_mi
  - 规律与a_mi相似，但也不够明显。且层数越高，越容易出现例外（如layer 7 ）
- 与v4m5相似，模型可能认为先前层数累积越多则压缩量应越大；层数越靠前则压缩量越大

v4m7（分母为a_{mi}）

```
Method7_v4 Learned Parameters
========================================
Total layers: 8
Parameters: Vector a_i for scaling 1/a_i

Layer 0 (vector length: 1):
  Parameters: [0.8651804]
  Mean: 0.865180
  Std: 0.000000
  Min: 0.865180
  Max: 0.865180
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 1 (vector length: 2):
  Parameters: [1.1061262  0.85660547]
  Mean: 0.981366
  Std: 0.124760
  Min: 0.856605
  Max: 1.106126
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 2 (vector length: 3):
  Parameters: [2.0853024 1.5279002 0.8036076]
  Mean: 1.472270
  Std: 0.524726
  Min: 0.803608
  Max: 2.085302
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 3 (vector length: 4):
  Parameters: [2.741859  1.6588601 1.5634537 0.8681922]
  Mean: 1.708091
  Std: 0.670345
  Min: 0.868192
  Max: 2.741859
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 4 (vector length: 5):
  Parameters: [3.5736108  1.8778572  2.1221435  1.2149361  0.99082714]
  Mean: 1.955875
```

```
  Std: 0.908967
  Min: 0.990827
  Max: 3.573611
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 5 (vector length: 6):
  Parameters: [4.2221904 1.9464111 2.7159483 2.4464192 2.532675  1.4372126]
  Mean: 2.550143
  Std: 0.859981
  Min: 1.437213
  Max: 4.222190
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 6 (vector length: 7):
  Parameters: [5.6141024 2.5861645 4.016112  2.0443852 2.0940177 2.8696806 2.1900334]
  Mean: 3.059214
  Std: 1.220056
  Min: 2.044385
  Max: 5.614102
  Scaling formula: weighted sum with 1/a_i
------------------------------
Layer 7 (vector length: 8):
  Parameters: [5.941946  2.42075   3.906437  2.1697185 2.7143924 4.3760734 4.2783594
 2.8379207]
  Mean: 3.580699
  Std: 1.195800
  Min: 2.169719
  Max: 5.941946
  Scaling formula: weighted sum with 1/a_i
------------------------------
```

总结

- 规律几乎同v4m6一致。
- 查阅config文件计算得:当前模型的$\sqrt{d_k} = 8$。但当前习得的参数大小（初始值为8）均不到8。这似乎和v4m5的学习结果存在出入：v4m5中$\sqrt{d_k}$的参数几乎都大于1，说明压缩系数均大于8。但这与v4m6有一定的相似之处：v4m6中d_k的习得次数明显小于5，压缩系数也小于8。