修改方法

- **令每一层初始输入为$\vec{x}$，Multi-head Attention输出(未进行norm)为$\vec{x}'$，norm后结果/MLP输入为$\vec{y}$，MLP输出(未进行norm)为$\vec{y}'$**

方法0：原始模型

方法1：Multi-head Attention残差修改

- 修改$\vec{x}'$处的add方式：第m层由对该层的输入$\vec{x_m}$进行相加，变为加上$\sum_{i=1}^{m-1}\vec{x}_i'$

方法2：考虑方法1的message除以规模

- 在方法1的基础上，第m层Attention的残差除以(m-1)

方法3：MLP残差

- 只在MLP的$\vec{y}'$处进行相加，相加量改为$\sum_{i=1}^{m-1}\vec{y}_i'$

方法4：考虑方法3的message除以规模

- 在方法3的基础上，第m层MLP的残差除以(m-1)

方法5：

- 方法1的变体。每层的$\vec{x}'$处和$\vec{y}'$处都进行更新，相加量都改为$\sum_{i=1}^{m-1}\vec{x}_i'$。与方法1唯一的不同之处在于$\vec{y}'$处也加了。
- 方法2、4的实验结果证明是否除以(m-1)对结果影响不大，所以残差连接没有除以(m-1)。方法6、7同理。

方法6：

- 方法3的变体。每层的$\vec{x}'$处和$\vec{y}'$处都进行更新，相加量都改为$\sum_{i=1}^{m-1}\vec{y}_i'$。与方法2唯一的不同之处在于$\vec{x}'$处也加了。

方法7：分别更新

- 结合方法1和方法3：$\vec{x}'$处和$\vec{y}'$处分别更新

第一批训练

同时训练了方法0、1、2、3、4的模型，将方法1、2、3、4与方法0作对比

方法0：原始模型

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     2.9421
  train_runtime            =  2:25:12.59
  train_samples            =      68009
  train_samples_per_second =     39.029
  train_steps_per_second   =       2.44
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4916
  eval_loss               =     2.6217
  eval_perplexity         =    13.7592
  eval_runtime            = 0:00:02.32
  eval_samples            =        143
  eval_samples_per_second =     61.511
  eval_steps_per_second   =      7.743
```

方法1：Multi-head Attention残差修改

- 修改$\vec{x}'$处的add方式：第m层由对该层的输入$\vec{x_m}$进行相加，变为加上$\sum_{i=1}^{m-1}\vec{x}'_i$

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     3.1353
  train_runtime            = 2:29:13.43
  train_samples            =      68009
  train_samples_per_second =     37.979
  train_steps_per_second   =      2.374
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4732
  eval_loss               =     2.7617
  eval_perplexity         =    15.8263
  eval_runtime            = 0:00:02.34
  eval_samples            =        143
  eval_samples_per_second =     60.888
  eval_steps_per_second   =      7.664
---
```

方法2：考虑方法1的message除以规模

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     3.1159
  train_runtime            = 2:29:25.17
  train_samples            =      68009
  train_samples_per_second =      37.93
  train_steps_per_second   =      2.371
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4754
  eval_loss               =      2.739
  eval_perplexity         =    15.4721
  eval_runtime            = 0:00:02.35
  eval_samples            =        143
  eval_samples_per_second =     60.793
  eval_steps_per_second   =      7.652
```

方法3：MLP残差

- 只在MLP的$\vec{y}'$处进行相加，相加量改为$\sum_{i=1}^{m-1}\vec{y}_i'$

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137248486GF
  train_loss               =     2.9407
  train_runtime            =  2:45:41.57
  train_samples            =      68009
  train_samples_per_second =     34.204
  train_steps_per_second   =      4.276
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4914
  eval_loss               =     2.6273
  eval_perplexity         =    13.8362
  eval_runtime            = 0:00:02.42
  eval_samples            =        143
  eval_samples_per_second =     59.072
  eval_steps_per_second   =      7.436
```

方法4：考虑方法3的message除以规模

训练结果：

```
 ***** train metrics *****
   epoch                    =        5.0
   total_flos               = 137248486GF
   train_loss               =     2.9732
   train_runtime            =  2:45:35.46
   train_samples            =      68009
   train_samples_per_second =     34.225
   train_steps_per_second   =      4.279
 ***** eval metrics *****
   epoch                   =        5.0
   eval_accuracy           =     0.4901
   eval_loss               =     2.6482
   eval_perplexity         =    14.1279
   eval_runtime            = 0:00:02.40
   eval_samples            =        143
   eval_samples_per_second =     59.466
   eval_steps_per_second   =      7.485
```

训练结果比较：相比于方法0，方法1、2性能略有降低，方法3、4基本不变

第二批训练

关注方法5、6、7的效果。由第一批训练可知方法1、2效果较差，所以只训练方法3、4和方法0，与5、6、7对比。

同时训练了方法0、5、6、7、3、4的模型

方法0：

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     2.9424
  train_runtime            =  2:25:54.63
  train_samples            =      68009
  train_samples_per_second =     38.842
  train_steps_per_second   =      2.428
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4911
  eval_loss               =     2.6234
  eval_perplexity         =    13.7826
  eval_runtime            = 0:00:02.33
  eval_samples            =        143
  eval_samples_per_second =     61.169
  eval_steps_per_second   =        7.7`
```

方法5：

- 方法1的变体。每层的$\vec{x}'$处和$\vec{y}'$处都进行更新，相加量都改为$\sum_{i=1}^{m-1} \vec{x}'_i$。与方法1唯一的不同之处在于$\vec{y}'$处也加了。

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     3.1459
  train_runtime            =  2:31:19.49
  train_samples            =      68009
  train_samples_per_second =     37.452
  train_steps_per_second   =      2.341
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4725
  eval_loss               =     2.7748
  eval_perplexity         =    16.0348
  eval_runtime            = 0:00:02.38
  eval_samples            =        143
  eval_samples_per_second =     59.879
  eval_steps_per_second   =      7.537
```

方法6：

- 方法3的变体。每层的$\vec{x}'$处和$\vec{y}'$处都进行更新，相加量都改为$\sum_{i=1}^{m-1}\vec{y}_i'$。与方法2唯一的不同之处在于$\vec{x}'$处也加了。

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137248486GF
  train_loss               =     3.0318
  train_runtime            =  2:47:51.35
  train_samples            =      68009
  train_samples_per_second =     33.764
  train_steps_per_second   =      4.221
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4831
  eval_loss               =     2.6901
  eval_perplexity         =     14.733
  eval_runtime            = 0:00:02.46
  eval_samples            =        143
  eval_samples_per_second =     58.074
  eval_steps_per_second   =       7.31
```

方法7：分别更新

- 结合方法1和方法3：$\vec{x}'$处和$\vec{y}'$处分别更新

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137242029GF
  train_loss               =     3.1158
  train_runtime            =  2:31:20.61
  train_samples            =      68009
  train_samples_per_second =     37.447
  train_steps_per_second   =      2.341
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4745
  eval_loss               =     2.7592
  eval_perplexity         =    15.7878
  eval_runtime            = 0:00:02.39
  eval_samples            =        143
  eval_samples_per_second =     59.602
  eval_steps_per_second   =      7.502
```

方法3：只更新MLP残差

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137248486GF
  train_loss               =     2.9421
  train_runtime            =  2:47:16.45
  train_samples            =      68009
  train_samples_per_second =     33.881
  train_steps_per_second   =      4.236
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4905
  eval_loss               =     2.6307
  eval_perplexity         =    13.8836
  eval_runtime            = 0:00:02.41
  eval_samples            =        143
  eval_samples_per_second =     59.211
  eval_steps_per_second   =      7.453
```

方法4：只更新MLP残差并除以(当前层数-1)

训练结果：

```
***** train metrics *****
  epoch                    =        5.0
  total_flos               = 137248486GF
  train_loss               =     2.9745
  train_runtime            =  2:47:01.57
  train_samples            =      68009
  train_samples_per_second =     33.931
  train_steps_per_second   =      4.242
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =     0.4901
  eval_loss               =     2.6485
  eval_perplexity         =    14.1323
  eval_runtime            = 0:00:02.40
  eval_samples            =        143
  eval_samples_per_second =     59.373
  eval_steps_per_second   =      7.473
```

训练结果比较：

性能从好到坏排序：方法0≈方法3、4＞方法6＞方法5、7

结论总结

- 方法1、2与方法3、4两组方法内部差异很小，说明残差是否除以累加规模影响不大
- 方法3、4为MLP单模块的残差修改，方法6为MLP的修改残差连接到Attention和MLP两个模块，三个方法（都是对MLP的残差进行了修改）性能优于对Attention残差进行修改的方法（方法1、2、5），也优于Attention、MLP全部修改的方法（方法7）
- 修改后的方法最好情况也很难超过原始模型