语言分析与机器翻译报告

基于参数初始化的深层 Transformer

学 院 名 称 ：计算机科学与工程学院

专 业 名 称 ：计算机科学与技术

学 生 姓 名 ：曹智泉

学　　号 ：2101694

班　　级 ：计硕 2102

1 概述

1.1 项目内容

1）实现 Transformer 的程序编写，并在 Iwslt14 数据集上进行试验，但是自己完成的 Transformer 效果由于缺少很多优化策略，效果未达预期，与成熟的框架下的 Transformer 还有一定差距，实现了 post 和 pre 两种结构的 Transformer，实现代码在 MyTransformer 文件夹中。

2）基于 Fairseq 工具，从深层神经网络模型结构、参数初始化策略两个方面进行了实验。使用和实现 DLCL 网络模型和 T-Fixup 初始化方法，并在 DLCL 上进行初始化方法 T-Fixup 的融合实验，包括浅层和深层实验。最终实现的融合模型提升了 DLCL 的性能。

1.2 项目使用的模型

1.2.1 Transformer 模型结构

项目后续所使用的网络模型和参数初始化策略都是直接或者间接建立在 Transformer 模型上的。

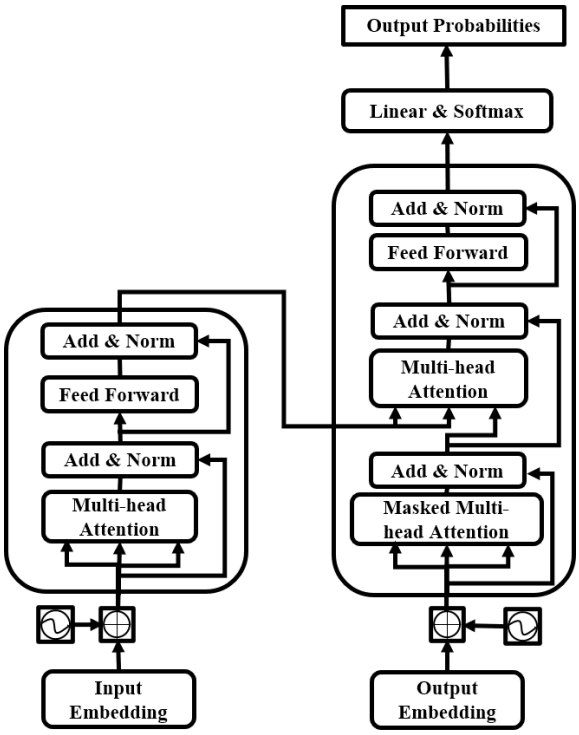

图 1.1 Transformer 整体结构

图 1.1 展示了 Transformer 的整体结构。模型主要分为三大部分：处于最下面的词嵌入层，左侧黑框内的编码端以及右侧黑框内的解码端。其中，黑框所表示的结构为实际模型的一层，标准的 Transformer 模型会堆叠 6 层这样的结构。输入的句子经过词嵌入层的处理得到的表示会与位置编码信息相加得到新的表示，并将新的表示作为编码端或解码端的输入。下面详细介绍编码端和解码端的内部结构：

（1）多头自注意力（Multi-head Attention）：使用多头自注意力机制对输入序列许学习，从而得到新的表示；

（2）前馈神经网络（Feed Forward）：使用两层全连接层的前馈神经网络将输入序列进行变换，得到新的表示；

（3）残差连接（Add）：对于多头自注意力子层和前馈神经网络子层，都拥有一个残差连接，其作用是将输入到自注意力子层和前馈神经网络子层的表示直接跨过该子层，并与该子层的输出相加。在 Transformer 中，残差连接不仅能够使得深层网络的信息传递更加有效，同时还能够保证网络的稳定性。

（4）层标准化（Norm）：该层的作用是将自注意力子层和前馈神经网络子层经过残差连接的输出做标准化，保证输出的向量数据在某一取值范围内。对 Transformer 网络结构的稳定性有很大的作用。

1.2.2 DLCL 网络模型

DLCL 模型是 2019 年由东北大学自然语言处理实验室的王强、李北等人提出的，该模型主要为了提高信息在 Transformer 中的传递效率，保证在网络中较低层的信息可以传递到相对较深的网络中去。DLCL 在网络中加入了一个带有可学习权重的矩阵，该矩阵是 DLCL 算法的核心，它的作用是将先前所有层的计算结果通过权重矩阵加权求和最后得到一个融合信息表示，将该表示再作为输入进行后续网络的计算。通过这种方式，DLCL 可以很好地将相对较低层的信息传递到相对较高的层中去。DLCL 的计算公式如公式（2.3）所示：

$$x_{l+1} = G(y_0, y_1, \dots, y_l) \tag{1.1}$$

公式（2.3）中的$y_0, y_1, \dots, y_l$为前$l+1$层的输出结果表示，$G(\cdot)$是一个线性变换函数，它可以将前$l+1$层的输出结果进行整合，并且得到一个新的融合表示。在 2.1.2 小节中介绍了两种层标准化的方法：Post-Norm 和 Pre-Norm，因此在 DLCL 的网络设

计中对于$G(\cdot)$的计算也拥有两种计算方式。对于 Post-Norm 结构来说，DLCL 的$G(\cdot)$的计算方式为：

$$G(y_0, y_1, \ldots, y_l) = \sum_{k=0}^{l} W_k^{l+1} \mathrm{LN}(y_k) \tag{1.2}$$

在公式（2.4）中W_k^{l+1}是一个可学习的权重矩阵，该权重矩阵对其输入进行线性变换，并且随着层数的不同而不同，在堆叠的层中可以很好的使用。该矩阵即使在处理相同层的输出结构的时候也可以使用不同的权重进行线性变换，即$W_k^l \neq W_k^{l+1}$。相似的，当 DLCL 应用在 Pre-Norm 结构中时其公式如公式（2.5）所示：

$$G(y_0, y_1, \ldots, y_l) = \mathrm{LN}(\sum_{k=0}^{l} W_k^{l+1} y_k) \tag{1.3}$$

该方法是应用在 Transformer 模型上的，经过$G(\cdot)$计算后得到的x_l可以作为后续层的输入，即作为 query、key 或 value 参与计算。

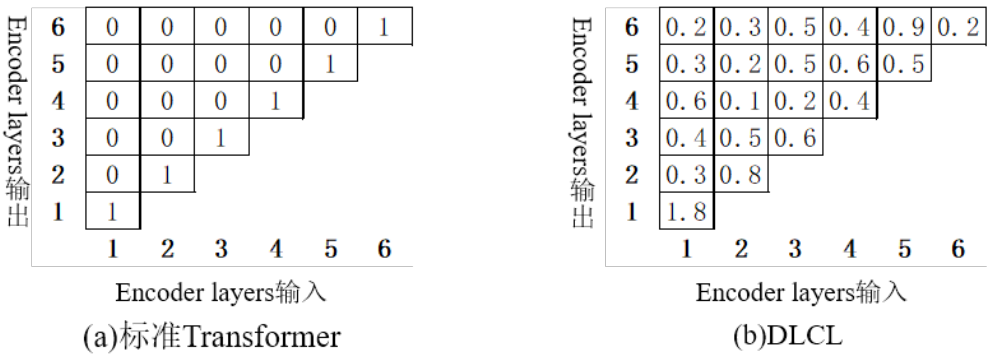

图 1.2 标准 Transformer 和 DLCL 的信息传递比较

图 1.2 中展示了标准 Transformer 的信息传递方式和 DLCL 的信息传递方式的区别表格下方和左方加粗的数字代表编码端的层数。

1.2.3 T-Fixup 参数初始化方法

T-Fixup 参数初始化策略是由 Huang 等人 2020 年提出来的，该参数初始化策略是基于 Transformer 这种网络模型结构设计出来的。具体的策略如下：

（1）移除 Transformer 网络模型中的所有层标准化 LN，并且在训练 Transformer 的时候不再使用 Warmup 这种学习率预热策略，其中层标准化 LN 指的是多头自注意力子层和前馈神经网络后续出现的。

（2）对 Transformer 中所包含的所有参数矩阵除了词嵌入层的权重W_e先进行 Xavier 参数初始化，包括：多头自注意力机制的权重W_q, W_k, W_v, W_o和前馈神经网络的

权重W_1, W_2。并对词嵌入层的W_e进行高斯初始化，分布服从$N(0, d^{-\frac{1}{2}})$，这里的d指的是词嵌入层的维度。

（3）对解码端的两种多头自注意力机制（包括解码-编码多头自注意力和多头自注意力）的权重W_v, W_o、前馈神经网络的权重W_1, W_2以及编码端和解码端的词嵌入层的权重W_e进行一个比例缩放，其公式表述如下：

$$W^{\sim} = (9\mathrm{L})^{-\frac{1}{4}} W \tag{1.3}$$

公式（1.3）中的W代表未处理前的权重矩阵，而$W^{\sim}$代表经过缩放处理后的权重矩阵。

（4）对编码端多头自注意力机制的权重W_v, W_o、前馈神经网络的权重W_1, W_2进行一个比例缩放，其公式表述如下：

$$W^{\sim} = 0.67\mathrm{M}^{-\frac{1}{4}} W \tag{1.4}$$

公式（1.4）中的W代表未处理前的权重矩阵，而$W^{\sim}$代表经过缩放处理后的权重矩阵。

2 项目设计

2.1 数据预处理设计

当前的神经机器翻译模型想要进行训练就必须要有数据的支持，所以进行模型基线搭建，新模型测试等一系列的任务都需要先得到数据，这一部分将详细介绍本文所进行的实验中需要的数据集和对数据集的处理设计，最终得到所需要的训练集、验证集和测试集。具体的数据获取流程如图 2.1 所示：

（1）获得数据。根据翻译课程提供的 Iwslt14 的英德数据。

（2）进行 BPE 处理。在处理 Iwslt14 数据集的时候使用 BPE 处理方式进行处理，最后得到一个共享的词表。

（3）格式转变。使用 Fairseq 的预处理文件对 Iwslt14 已选择的数据集进行处理，将 Iwslt14 的训练数据集、验证数据集以及测试数据集转变成 Fairseq 的对应的二进制文件。

2.2 T-Fixup 应用于 DLCL 设计

前面详细介绍了 T-Fixup 参数初始化方法和 DLCL 网络模型的原理和实现方法，现在将 Tfixup 参数初始化方法应用在 DLCL 网络模型的结构上。T-Fixup 参数初始化方法在 DLCL 网络模型上的应用具体设计为：

（1）移除 DLCL 网络模型中的所有层标准化 LN，并且在训练 DLCL 的时候不再使用 Warmup 这种学习率预热策略。

（2）对 DLCL 中所包含的所有参数矩阵除了词嵌入层的权重W_e先进行 Xavier 参数初始化，包括：多头自注意力机制的权重W_q, W_k, W_v, W_o和前馈神经网络的权重W_1, W_2。并对词嵌入层的W_e进行高斯初始化，分布服从$N(0, d^{-\frac{1}{2}})$，这里的d指的是词嵌入层的维度。

（3）对解码端的两种多头自注意力机制（包括解码-编码多头自注意力和多头自注意力）的权重W_v, W_o、前馈神经网络的权重W_1, W_2以及编码端和解码端的词嵌入层的权重W_e进行一个比例缩放，其公式表述如下：

$$W^{\sim} = (9\mathrm{L})^{-\frac{1}{4}} W \tag{2.1}$$

公式（2.1）中的W代表未处理前的权重矩阵，而$W^{\sim}$代表经过缩放处理后的权重矩阵。

（4）对编码端多头自注意力机制的权重W_v, W_o、前馈神经网络的权重W_1, W_2进行一个比例缩放，其公式表述如下：

$$W^{\sim} = 0.67M^{-\frac{1}{4}}W \tag{2.2}$$

公式（2.2）中的W代表未处理前的权重矩阵，而$W^{\sim}$代表经过缩放处理后的权重矩阵。

根据上述描述可以得到经过 DLCL 网络模型和 T-Fixup 参数初始化策略融合后的模型，其网络结构图如图 2.1 所示：

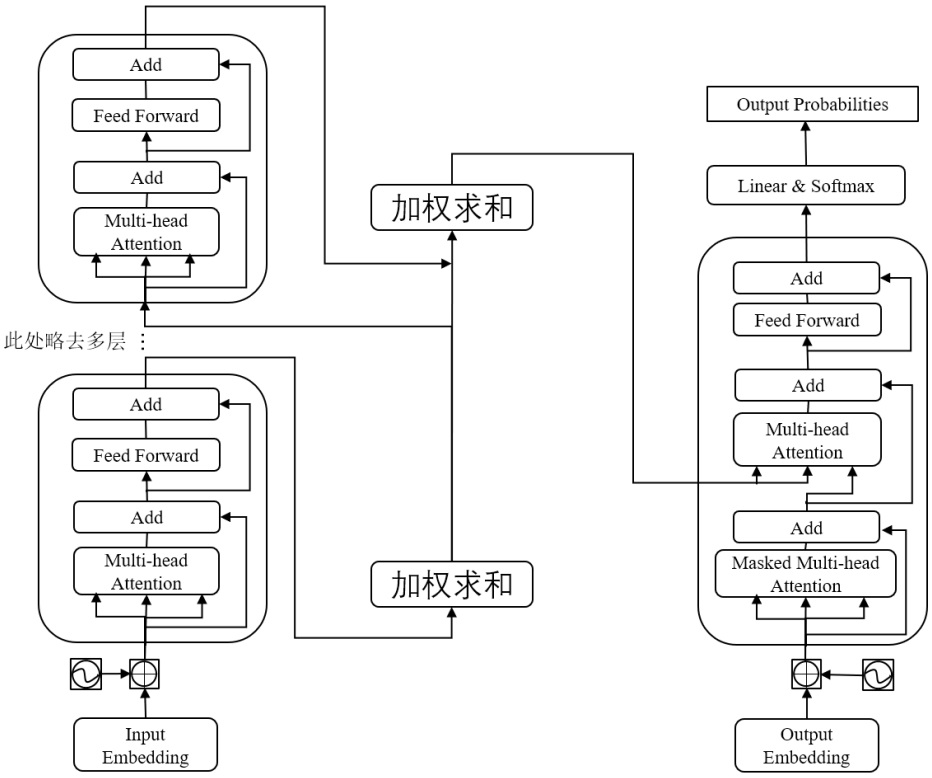

图 2.1 DLCL 网络模型和 T-Fixup 参数初始化的融合模型结构

3 模型测试

3.1 相关超参数介绍

本文在复现各种模型时需要使用到大量的超参数，而这些超参数在各个模型中介绍会导致篇幅过长，为了简化文章，在所有模型的实验进行之前先介绍本文实验中会使用到的超参数。

（1）Encoder：该超参数指的是编码端的层数，其中每一层包括一层多头自注意力机制和一层前馈神经网络；

（2）Decoder：该超参数指的是解码端的层数，其中每一层包括一层多头自注意力机制、编码-解码多头自注意力机制和一层前馈神经网络；

（3）Ln：该超参数代表模型使用的结构，如果 Ln 取值为 Post 则使用 Post-Norm 结构进行实验，如果 Ln 取值为 Pre 则使用 Pre-Norm 结构进行实验；

（4）Warmup：该超参数指的是在神经网络模型中使用 Warmup 这种学习率更新策略时使用的预热更新步数；

（5）Lr：该超参数指的是神经网络训练过程的学习率；

（6）Batch：该超参数指的是神经网络模型每一次更新所使用的训练数据量；

（7）Adam-β_1：该超参数指的是使用 Adam 作为梯度更新策略是，Adam 优化器的β_1值；

（8）Adam-β_2：该超参数指的是使用 Adam 作为梯度更新策略是，Adam 优化器的β_2值；

（9）Lf：该超参数指的是神经网络模型使用的损失函数种类，当 Lf 取值为交叉熵时代表使用交叉熵损失函数，当 Lf 取值为 Label 时代表使用 Label-smoothed 损失函数，且其参数 Label-smoothing 设置为 0.1；

（10）Init：该超参数指的是神经网络模型使用的参数初始化策略种类；

（11）FP16：该超参数指的是训练神经网络模型是是否使用半精度训练，如果 FP16 取值为 1 则代表使用半精度进行训练，反之则不使用；

（12）Dropout：该超参数指的是整个神经网络模型中所设置的 Dropout 率；

（13）Att-Dropout：该超参数指的是多头自注意力机制中所设置的 Dropout 率；

（14）Relu-Dropout：该超参数指的是前馈神经网络中所设置的 Dropout 率；

（15）Rpr：该超参数指的时神经网络模型是否使用相对位置编码以及相对位置编码的相对位置取值，若 Rpr 取值为 0 则代表不适用相对位置编码，若 Rpr 的取值大于 0 则代表使用相对位置编码，且其取值便代表了相对位置编码的相对位置大小。

（16）Share-all-embedding：该超参数指的是将编码端的词嵌入层、解码端的输入词嵌入层、解码端的输出词嵌入层三个层共享，使用同样的表示空间；

（17）Share-decoder-input-output-embed：该超参数指的是将解码端的输入词嵌入层、解码端的输出词嵌入层两个层共享，使用同样的表示空间；

（18）容量：该参数有两种取值情况：Small 和 Base。当容量取值为 Small 则代表此时的网络模型其隐藏维度为 512 维，前馈神经网络中间层为 1024，多头自注意力机制为 4 头；反之，如果容量的取值为 Base 则代表此时的网络模型其隐藏维度为 512 维，前馈神经网络中间层为 2048，多头自注意力机制为 8 头。

3.2 Transformer 实现测试

Transformer 网络模型的超参数有：解码端均设置为 6 层；模型隐藏层维度为 512 维；前馈神经网络中间层为 1024；多头注意力机制设置为 4 头，学习率 Lr 设置为 0.0015；Batch 设置为 4096；梯度更新方法选择 Adam，并且将 Adam-β_1 设置为 0.9，将 Adam-β_2 设置为 0.98；损失函数使用交叉熵损失函数。

实现的模型在 MyTransformer 文件夹中，直接运行 python train.py 就可以进行训练，训练过程中首先对数据做 batch 操作，随后喂入 Transformer 进行训练。

相比于成熟的 Transformer 模型，我所实现的 Transformer 模型没有众多的优化策略，其性能大受影响。在 Transformer 模型的论文中报的 BLEU 分数达到 34.4，但我所实现的 Transformer 模型与之差距较大。

在实现的两种结构中发现，post 结构的 transformer 在训练过程中不能收敛，这主要是由于缺少了 warmup 的学习率更新策略。更新不收敛程序运行截图：

```
epoch 0002, step 50/3205, loss: 6.200901, ppl: 493.193
epoch 0002, step 100/3205, loss: 6.466301, ppl: 643.1
epoch 0002, step 150/3205, loss: 6.324169, ppl: 557.894
epoch 0002, step 200/3205, loss: 6.687206, ppl: 802.078
epoch 0002, step 250/3205, loss: 6.759071, ppl: 861.841
epoch 0002, step 300/3205, loss: 6.198554, ppl: 492.037
epoch 0002, step 350/3205, loss: 6.885186, ppl: 977.683
epoch 0002, step 400/3205, loss: 6.369576, ppl: 583.811
epoch 0002, step 450/3205, loss: 6.174514, ppl: 480.35
epoch 0002, step 500/3205, loss: 6.572212, ppl: 714.949
epoch 0002, step 550/3205, loss: 6.296309, ppl: 542.566
epoch 0002, step 600/3205, loss: 6.452173, ppl: 634.079
epoch 0002, step 650/3205, loss: 6.418378, ppl: 613.008
epoch 0002, step 700/3205, loss: 6.428337, ppl: 619.144
epoch 0002, step 750/3205, loss: 6.303205, ppl: 546.32
epoch 0002, step 800/3205, loss: 6.128253, ppl: 458.634
epoch 0002, step 850/3205, loss: 6.329247, ppl: 560.734
epoch 0002, step 900/3205, loss: 6.360161, ppl: 578.339
epoch 0002, step 950/3205, loss: 6.658200, ppl: 779.147
epoch 0002, step 1000/3205, loss: 6.675417, ppl: 792.678
epoch 0002, step 1050/3205, loss: 6.474620, ppl: 648.473
epoch 0002, step 1100/3205, loss: 6.457202, ppl: 637.276
epoch 0002, step 1150/3205, loss: 6.996212, ppl: 1092.49
epoch 0002, step 1200/3205, loss: 6.196954, ppl: 491.251
epoch 0002, step 1250/3205, loss: 6.658268, ppl: 779.2
epoch 0002, step 1300/3205, loss: 6.304260, ppl: 546.897
epoch 0002, step 1350/3205, loss: 6.829151, ppl: 924.406
epoch 0002, step 1400/3205, loss: 6.437516, ppl: 624.853
epoch 0002, step 1450/3205, loss: 6.662418, ppl: 782.441
epoch 0002, step 1500/3205, loss: 6.390876, ppl: 596.379
epoch 0002, step 1550/3205, loss: 6.491255, ppl: 659.35
epoch 0002, step 1600/3205, loss: 6.233465, ppl: 509.518
epoch 0002, step 1650/3205, loss: 6.178353, ppl: 482.197
epoch 0002, step 1700/3205, loss: 6.424876, ppl: 617.004
epoch 0002, step 1750/3205, loss: 6.677576, ppl: 794.391
epoch 0002, step 1800/3205, loss: 6.918656, ppl: 1010.96
epoch 0002, step 1850/3205, loss: 6.546679, ppl: 696.926
```

随后进行了一次改进，将 post 结构更换为 pre 结构，在这种结构下没有 warmup 这种学习率更新策略也可以是模型收敛，pre 结构的训练过程截图：

```
epoch 0002, step 50/3205, loss: 5.446591, ppl: 231.966
epoch 0002, step 100/3205, loss: 5.017273, ppl: 150.999
epoch 0002, step 150/3205, loss: 4.597127, ppl: 99.199
epoch 0002, step 200/3205, loss: 4.928457, ppl: 138.166
epoch 0002, step 250/3205, loss: 4.713477, ppl: 111.439
epoch 0002, step 300/3205, loss: 4.814572, ppl: 123.294
epoch 0002, step 350/3205, loss: 4.911431, ppl: 135.834
epoch 0002, step 400/3205, loss: 5.135686, ppl: 169.981
epoch 0002, step 450/3205, loss: 5.083424, ppl: 161.326
epoch 0002, step 500/3205, loss: 4.756464, ppl: 116.334
epoch 0002, step 550/3205, loss: 4.719357, ppl: 112.096
epoch 0002, step 600/3205, loss: 4.860491, ppl: 129.088
epoch 0002, step 650/3205, loss: 4.730538, ppl: 113.357
epoch 0002, step 700/3205, loss: 4.884657, ppl: 132.245
epoch 0002, step 750/3205, loss: 4.007191, ppl: 54.9922
epoch 0002, step 800/3205, loss: 4.776597, ppl: 118.7
epoch 0002, step 850/3205, loss: 5.083020, ppl: 161.26
epoch 0002, step 900/3205, loss: 4.682045, ppl: 107.991
epoch 0002, step 950/3205, loss: 4.847039, ppl: 127.363
epoch 0002, step 1000/3205, loss: 5.126810, ppl: 168.479
epoch 0002, step 1050/3205, loss: 4.805781, ppl: 122.215
epoch 0002, step 1100/3205, loss: 4.946883, ppl: 140.736
epoch 0002, step 1150/3205, loss: 5.068424, ppl: 158.924
epoch 0002, step 1200/3205, loss: 4.826886, ppl: 124.822
epoch 0002, step 1250/3205, loss: 4.807099, ppl: 122.376
epoch 0002, step 1300/3205, loss: 4.768191, ppl: 117.706
epoch 0002, step 1350/3205, loss: 4.739317, ppl: 114.356
epoch 0002, step 1400/3205, loss: 5.208171, ppl: 182.759
epoch 0002, step 1450/3205, loss: 4.738221, ppl: 114.231
epoch 0002, step 1500/3205, loss: 5.205833, ppl: 182.333
epoch 0002, step 1550/3205, loss: 5.204076, ppl: 182.013
epoch 0002, step 1600/3205, loss: 4.681903, ppl: 107.975
epoch 0002, step 1650/3205, loss: 5.101395, ppl: 164.251
epoch 0002, step 1700/3205, loss: 5.101669, ppl: 164.296
epoch 0002, step 1750/3205, loss: 4.810496, ppl: 122.792
epoch 0002, step 1800/3205, loss: 4.418579, ppl: 82.9783
epoch 0002, step 1850/3205, loss: 4.545160, ppl: 94.1755
```

从训练的 loss 和 ppl 两个指标中可以清晰地发现，在没有 warmup 学习率更新策略下，pre 结构的 transformer 相比于 post 结构的 transformer 能够收敛，随着训练的进行 loss 和 ppl 都在逐渐的的降低。

3.3 复杂模型测试

3.3.1 DLCL 模型测试

DLCL 网络模型的超参数有：解码端均设置为 6 层；模型隐藏层维度为 512 维；前馈神经网络中间层为 1024；多头注意力机制设置为 4 头；Warmup 设置为 8000；学习率 Lr 设置为 0.0015；Batch 设置为 4096；梯度更新方法选择 Adam，并且将 Adam-β_1 设置为 0.9，将 Adam-β_2 设置为 0.98；损失函数使用 Label-smoothed 损失函数，且其参数 Label-smoothing 设置为 0.1；使用半精度训练；使用 share-all-emdeddings 方式，

即分享编码段的词嵌入层，解码端的词嵌入层以及解码端输出时的词嵌入层。

表 3.1 DLCL 模型超参数在 Iwslt14 上的调优过程

实验组数	Ln	Encoder	Dropout	Att-Dropout	Relu-Dropout	Rpr	BLEU(de-en)
1	Post	6	0.1	0	0	0	33.12
2	Post	6	0.5	0	0	0	31.14
3	Pre	6	0.1	0	0	0	33.85
4	Pre	6	0.1	0.1	0.1	0	34.21
5	Pre	6	0.1	0.1	0.1	20	35.25
6	Pre	6	0.1	0	0	20	34.84
7	Post	6	0.1	0	0	20	34.82
8	Pre	12	0.1	0.1	0.1	20	35.37
9	Pre	12	0.3	0.1	0.1	8	35.88
10	**Pre**	**6**	**0.3**	**0.1**	**0.1**	**8**	**35.94**

在表 3.1 中 BLEU 值代表使用 BLEU 评价指标对训练完成的模型进行评价时取得的分数，de-en 指的是使用的语料对为英德语料对，翻译方向为德语到英语。

表 3.2 DLCL 论文效果与复现效果在 Iwslt14 上的比照

名称	BLEU(de-en)
原论文 DLCL 模型	35.6
复现 DLCL 模型	**35.94**

通过表 3.2 可以得知，原论文中 DLCL 模型在 Iwslt14 数据集上获得的 BLEU 值为 35.6，而在复现过程中实现的 DLCL 模型在 Iwslt14 数据集上获得的 BLEU 值为 35.94，超过原论文 BLEU 值 0.34 点，说明实验方法无误，成功复现。

3.2.2 T-Fixup 模型测试

在这一阶段使用的 T-Fixup 参数初始化策略模型的超参数有：编码器和解码端均设置为 6 层；模型隐藏层维度为 512 维；前馈神经网络中间层为 1024 维；多头注意力机制设置为 4 头；学习率 Lr 设置为 0.0005；Batch 设置为 4096；梯度更新方法选择 Adam，并且将 Adam-β_1设置为 0.9，将 Adam-β_2设置为 0.98；损失函数使用 Label-smoothed 损失函数，且其参数 Label-smoothing 设置为 0.1；Dropout 设置为 0.5；

Att-Dropout 设置为 0；Relu-Dropout 设置为 0，使用半精度训练。其中，前馈神经网络中间层、多头注意力机制头数、学习率 Lr 以及 Dropout 这五个参数的更改主要原因是 Iwslt14 这个小数据集特性引起的。其他不同的参数在表 3.3 中列出。

表 3.3 T-Fixup 超参数在 Iwslt14 上的调优过程

实验组数	Share-all-embedding	Share-decoder-input-output-embed	BLEU(de-en)
1	1	0	33.77
2	**0**	**1**	**35.93**

在表 3.3 中 BLEU 值代表使用 BLEU 评价指标对训练完成的模型进行评价时取得的分数，de-en 指的是使用的语料对为英德语料对，翻译方向为德语到英语。从表 6.5 中可以得出，使用 T-fixup 该初始化策略对于词嵌入层的分享策略更适合的是 Share-decoder-input-output-embed。达到最好的机器译文效果的是第 2 组实验。

通过表 3.4 可以得知，原论文中 T-Fixup 在 Iwslt14 数据集上获得的 BLEU 值为 35.5，而在复现过程中实现的 T-Fixup 在 Iwslt14 数据集上获得的 BLEU 值为 35.94，超过原论文 BLEU 值 0.43 点，说明实验方法无误，成功复现。

表 3.4 T-Fixup 论文效果与复现效果在 Iwslt14 上的比照

名称	BLEU(de-en)
原论文 T-Fixup	35.5
复现 T-Fixup	**35.93**

3.4 新模型测试

3.4.1 浅层神经网络上的测试

在这一阶段中使用的 T-Fixup 参数初始化策略与 DLCL 网络模型融合模型的一些相同超参数有：使用 Pre-Norm 结构的网络模型；编码器和解码端均设置为 6 层；模型隐藏层维度为 512 维；前馈神经网络中间层为 1024 维；多头注意力机制设置为 4 头；Batch 设置为 4096；梯度更新方法选择 Adam，并且将 Adam-β_1设置为 0.9，将 Adam-β_2 设置为 0.98；损失函数使用 Label-smoothed 损失函数，且其参数 Label-smoothing 设置为 0.1；Rpr 设置为 8。其他不同的参数在表 6.10 中列出。

表 3.5 中 Share1 代表 Share-all-embedding；Share2 代表 Share-decoder-input-

output-embed；BLEU 值代表使用 BLEU 评价指标对训练完成的模型进行评价时取得的分数，de-en 指的是使用的语料对为英德语料对，翻译方向为德语到英语。从表 6.10 中的 1 和 2 组实验中可以看出，学习率过大会导致融合模型不能收敛，因此将学习率调小；3、4 和 6 组实验可以发现融合模型更加适合 Share2 的方式，且不使用 Share1 的方式；2 和 5 组实验说明提高 Dropout 有利于融合模型的训练；7 和 8 组实验说明学习率为 0.0003 比 0.0004 有利于融合模型的训练；3 和 9 实验说明适当提高 Relu-Dropout 和 Att-Dropout 有利于融合模型的训练；随后通过 6-16 共计 11 组实验进行控制变量实验，每次只进行调整一个参数的值，这样控制变量的好处是：能够直接准确地发现该参数对整个融合模型训练的贡献度。最后通过多次对比，确定 T-Fixup 参数初始化策略与 DLCL 网络模型融合模型的参数采用 12 组实验的设置。在实验过程中，虽然 12 组实验得到最后最优结果，但随后又进行多次实验，目的是确保 12 组实验确定的参数是最优的，而不是在得到最优参数的过程中，防止实验落入陷阱，把一个次优的参数调整当成最终的最优参数设置。

表 3.5 T-Fixup 与 DLCL 融合模型超参数在 Iwslt14 上的调优过程

实验组数	Lr	Dropout	Att-Dropout	Relu-Dropout	Share1	Share2	BLEU(de-en)
1	0.0015	0.3	0	0	1	1	Falied
2	0.0005	0.3	0	0	1	1	33.55
3	0.0004	0.3	0	0	0	1	34.69
4	0.0004	0.3	0	0	1	1	34.11
5	0.0005	0.5	0	0	0	1	34.71
6	0.0004	0.3	0	0	1	0	34.69
7	0.0004	0.5	0	0	0	1	34.65
8	0.0003	0.5	0	0	0	1	35.47
9	0.0004	0.3	0.1	0.1	0	1	35.49
10	0.0003	0.5	0	0	1	0	35.39
11	0.0003	0.3	0	0	0	1	34.47
12	**0.0003**	**0.5**	**0.1**	**0.1**	**0**	**1**	**36.1**
13	0.0002	0.5	0.1	0.1	0	1	35.85
14	0.0003	0.5	0.2	0.1	0	1	35.88

15	0.0003	0.5	0.1	0.2	0	1	36.07
16	0.0003	0.5	0.5	0.3	0	1	34.56

3.4.2 融合模型与独立模型对比分析

该小节针对 3.2 小节中已复现并且通过实验参数调优的 T-Fixup 参数初始化策略和 DLCL 网络模型与 3.3.1 小节中实现并且通过实验参数调优的融合模型的参数以及性能进行对比。

在这一阶段中 T-Fixup 参数初始化策略、DLCL 网络模型和它们的融合模型的相同参数有：编码器和解码端的层数均设置为 6 层；融合模型隐藏层维度设置为 512 维；融合模型的前馈神经网络中间层设置为 1024 维；融合模型的多头注意力机制设置为4头；每一次从参数更新数据量大小Batch设置为4096；梯度更新方法选择Adam，并且将 Adam-β_1设置为 0.9，将 Adam-β_2设置为 0.98；损失函数使用 Label-smoothed 损失函数，且其参数 Label-smoothing 设置为 0.1；Rpr 设置为 8；使用半精度训练；其他不同的参数在表 3.6 中列出。

表 3.6 T-Fixup、DLCL 与 T-Fixup 和 DLCL 的融合模型性能对比

参数名称	DLCL	T-Fixup	融合模型
Lr	00015	0.0005	**0.0003**
Ln	Pre	不使用	**不使用**
Warmup	8000	不使用	**不使用**
Dropout	0.3	0.5	**0.5**
Relu-Dropout	0.1	0	**0.1**
Att-Dropout	0.1	0	**0.1**
Share-all-embedding	1	0	**0**
Share-decoder-input-output-embed	0	1	**1**
收敛训练轮次	49	214	**262**
BLEU	35.76	35.82	**36.1**
BLEU(last-5)	35.94	35.93	**36.07**

通过表 3.6 中的 BLEU 值可知融合模型平均比分开的单独模型高 0.15+的 BLEU 值，这说明了两种算法融合在一起的时候是能够保证性能不损失的，融合成功。但同

时也存在一些问题，可以明显发现，融合模型最优表现时的学习率较低，比两个独立模型的学习率都低，尤其和 DLCL 网络模型相比。这也导致了另外一个问题，那就是增加了融合模型的训练代价，通过对比三个模型收敛达到最佳性能时的训练轮次，融合模型比两独立模型都要高，尤其和 DLCL 模型相比时其差距更大，这样的训练代价也是需要考虑的问题之一。由于 T-fixup 参数初始化策略将 Warmup 这种学习率预热策略删除，这会导致网络在刚刚开始训练时不能很快的找到一个较好状态，延长模型训练时间。融合模型的另外一个特点就是类 Dropout 参数都需要相对调高，只有在这种情况下才能使融合模型的性能更好，同样的因为 Dropout 类参数本身就是提高模型鲁棒性用的，融合模型的高 Dropout 也能够使得该模型的鲁棒性提高，具有更好的泛化能力。实验中还发现 T-Fixup 参数初始化策略和融合模型的最后几个检测点其内部参数分布较为接近，在使用最后 5 个检测点进行平均得到平均检测点时测得的分数也十分接近，不像 DLCL 模型的平均检测点会得到比较多的 BLEU 值的收益，这可能也是训练时间过长引起的，导致最后几个检测点分布接近。

将 T-Fixup 参数初始化策略同 DLDL 网络模型进行融合时在 6 层上的实验结果表明：T-Fixup 参数初始化策略可以在移除 Warmup 学习率预热策略、移除所有的层标准化 Norm 的情况下，对 DLCL 网络模型的参数矩阵进行缩放实现参数初始化，两种算法可以进行融合，且不会有明显的排斥反省，这其实和 DLCL 网络模型也相关。融合模型虽然移除了每一层中的层标准化 Norm，但实际上 DLCL 这种模型结构本身就会对融合信息表示进行 Norm 操作。在 DLCL 模型中，每一层的输入是前几层输入的叠加表示，这会导致数据变得相对较大，而 DLCL 网络模型为了防止这种现象的发生，会对融合后的结果进行一次层标准化 Norm。因此 T-Fixup 在应用于 DLCL 网络模型时相当于没有完全去除层标准化 Norm，对融合信息进行层标准化 Norm 也是 DLCL 网络模型的取得成功的关键之一。虽然消除了移除层标准化 Norm 带来的影响，但是却无法消除移除 Warmup 带来的影响，没有这种学习率预热策略，会导致模型训练前期很难快速找到较好的状态，而后学习率又不断减小，这导致融合模型的训练时间成倍增加。融合模型虽然在最后的机器译文上比两个独立模型都要高，但是高出的性能不是很多，平均只有 0.2 的 BLEU 值提升。通过对比最后 5 个检测点生成的平均检测点与 Best 单检测点的性能可以发现，融合模型的性能相对更加稳定，没有过大的波动。

3.4.3 深层神经网络上的测试

通过 3.1 小节中的 DLCL 模型调优，得到了 6 层的 DLCL 模型的最佳超参数。随后又通过 3.3.1 小节的控制变量实验，将 T-Fixup 参数初始化策略与 DLCL 网络模型的融合模型进行超参数调优，同样的到了最优的 6 层融合模型超参数。这样一来，通过上述大量的实验，我们就得到了在深层网络要进行对比实验的所有模型的最优超参数。虽然增加网络层数可能会导致在 6 层的超参数并不是最优的超参数，但是，在一定程度上同样也能够代表该网络在深层时的效果。在本小节中的后续实验如果没有特别指明，除了网络层数的其他超参数都将使用 3.3.1 中所描述的。

该小节将对 T-Fixup 参数初始化策略同 DLCL 网络模型的融合模型的深层网络进行探索实验，具体的网络层数和实验结果如表 3.7 所示。

表 3.7 融合模型的深层实验

实验组数	结构	Encoder	Decoder	训练轮次	BLEU	BLEU(last-5)
1.1	DLCL	6	6	49	35.76	35.94
1.2	**融合模型**	**6**	**6**	**262**	**36.1**	**36.07**
2.1	DLCL	12	12	25	35.03	35.59
2.2	**融合模型**	**12**	**12**	**96**	**35.7**	**35.73**
3.1	DLCL	20	20	25	35.14	35.6
3.2	**融合模型**	**20**	**20**	**46**	**35.31**	**35.54**
4.1	DLCL	12	6	49	35.48	35.88
4.2	**融合模型**	**12**	**6**	**137**	**36.07**	**35.98**
5.1	DLCL	20	6	49	35.67	35.95
5.2	**融合模型**	**20**	**6**	**96**	**35.75**	**35.81**
6.1	DLCL	30	6	49	35.45	35.63
6.2	**融合模型**	**30**	**6**	**65**	**35.65**	**35.72**

从表 3.7 中不难看出，随着 DLCL 网络模型的深度的增加，模型的性能出现了波动式的变化，并不是一味地增长也不是一味的下降。当一个复杂的深层 DLCL 网络模型在 Iwslt14 数据集上进行实验的时候会因为过拟合的原因导致从翻译结果的性能上出现波动式变化，因此不能单纯的从模型的 BLEU 值来评定模型的性能。因此在对比网络性能时只进行不同网络相同层数的对比。

根据实验中的训练轮次比较，我们不难发现，越深层的网络收敛速度越快，并且

不管层数变化多少，相同层数的 DLCL 网络模型比融合模型的收敛速度要快。但随着网络层数的加深，两者之间的收敛轮次差距在变小。编码器和解码器都是 6 层的时候，DLCL 网络模型收敛速度差不多是融合模型收敛速度的 5 倍；当编码器和解码器都是 20 层的时候，DLCL 网络模型收敛速度只是融合模型收敛速度的 2 倍。融合模型随着网络深度的加深，其收敛速度变化较为敏感，提高的较快，这里考虑其原因是融合模型没有了 Warmup 的影响，其收敛速度对网络深度形成了依赖。一个额外的发现，只提升编码端层数，不提高解码端层数，融合模型和 DLCL 网络模型都会呈现出性能下降较少或者提升的情况。

根据实验中的性能对比我们可以发现，相同层数的融合模型比 DLCL 网络模型的性能较好。虽然两者随着网络层数变深，其性能变化趋势相同，但是融合模型的性能一直优于 DLCL 网络模型。

4 总结

通过本次实践，让我对现有的成熟框架有了新的认识，曾经我以为他们实现的代码我一样可以成功写出来。经过本次实践我才明白，有很多地方我做不到，成熟的框架中有很多优化策略，它能够在当前的学习框架中脱颖而出是有一定的道理的，这些优化策略让我从头搭建还是十分困难的。

网络的超参数对网络的性能影响很大，在网络结构确定的情况下，不同的超参数也会使整个网络的性能有着天差地别，超参数的设置更能决定一个网络模型的性能。

网络的结构，初始化策略等因素对网络模型的稳定性训练有着重要的影响，将多种优化策略结合起来能够对模型的性能有一个较好的提高。