CS336-assignment1模型训练记录

一、调整学习率

要求：（1）对学习率进行超参数搜索，给出最终损失；（2）民间智慧认为，最佳学习率是“稳定的边缘”，研究学习率发散点与最佳学习率的联系。

> 💡 **basic hyperparameters**
>
> - vocab_size 10000
> - context_length 256
> - d_model 512
> - d_ff 1344
> - RoPE theta parameter Θ 10000
> - number of layers and heads 4 layers, 16 heads
> - total tokens processed 327,680,000 (your batch size × total step count × context length should equal roughly this value)

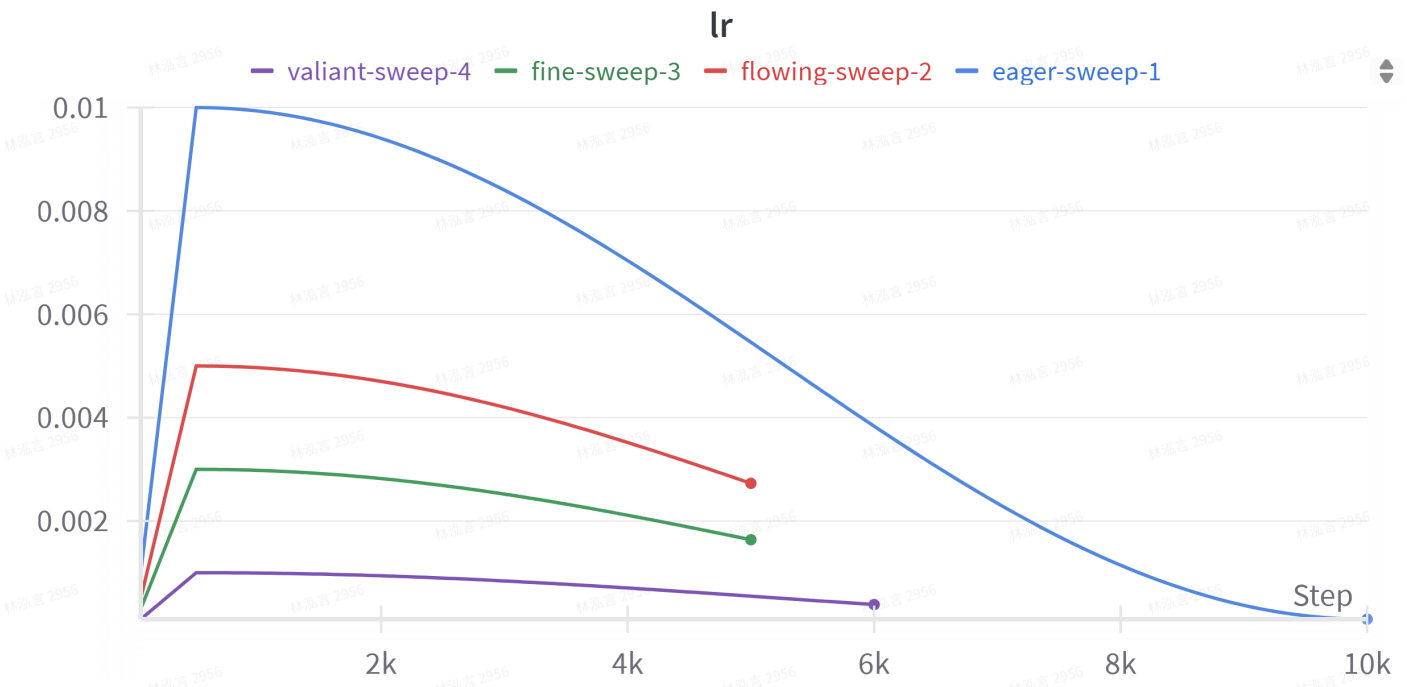
我设置的 batch size = 128，所以 total step count = 10000。

Adam 优化器使用默认的参数，lr = 1e-3，betas = [0.9, 0.999]，eps = 1e-8，weight_decay = 0.01。也就是默认的学习率是 1e-3，我使用超参数网格搜索，指定四个学习率搜素值，分别是 1e-2，5e-3，3e-3，1e-3，最多进行 10000 次迭代，如果在过程中 val_loss < 1.45，则提前终止迭代，继续用下一个学习率进行训练。

实验发现，lr = 1e-2 结果是发散的，lr = 5e-3 和 lr = 3e-3 在第 5000 次迭代的时候 val_loss 就小于 1.45 终止迭代了，lr = 1e-3 在第 6000 次的时候停止迭代。

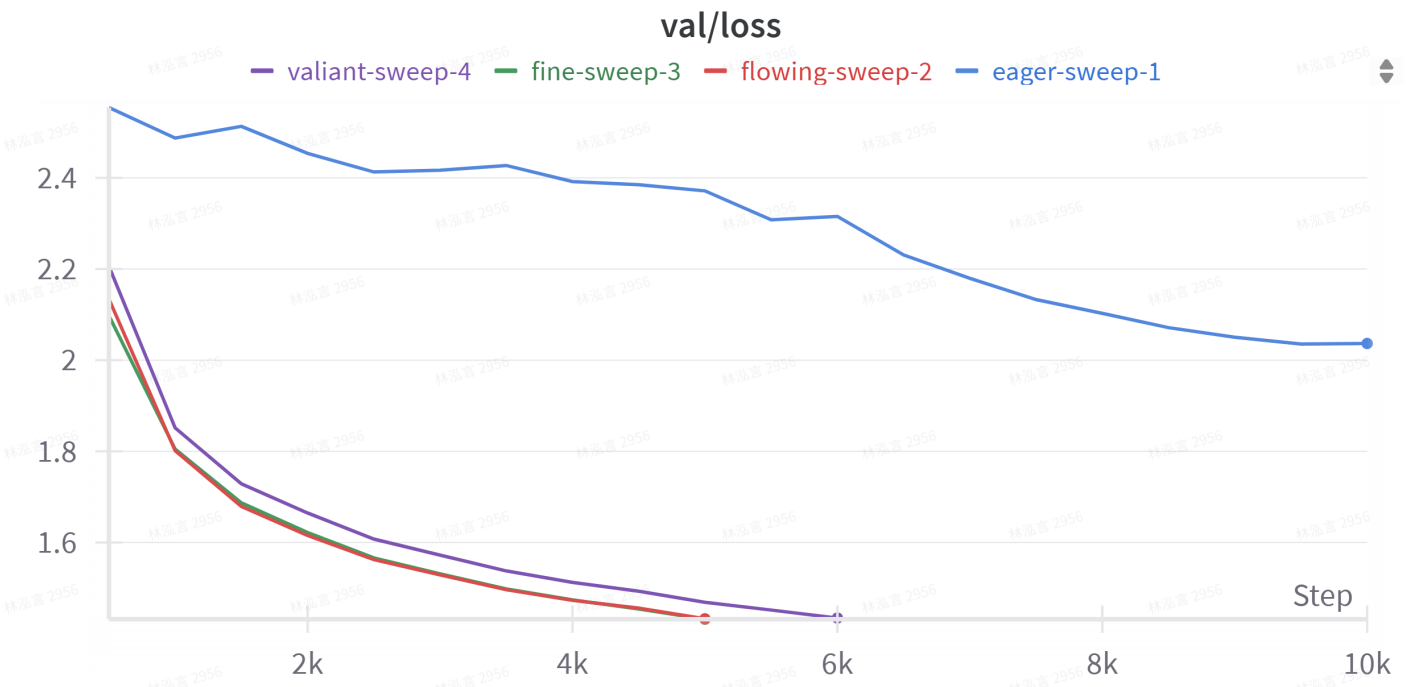
训练过程记录在 wandb，链接： https://wandb.ai/hongyanlin29-x_heart/cs336_ass1/sweeps/bz5t6u5a?nw=nwuserhongyanlin29

它们的 lr 变化曲线如下图所示：



train/loss 曲线：

val/loss 曲线：

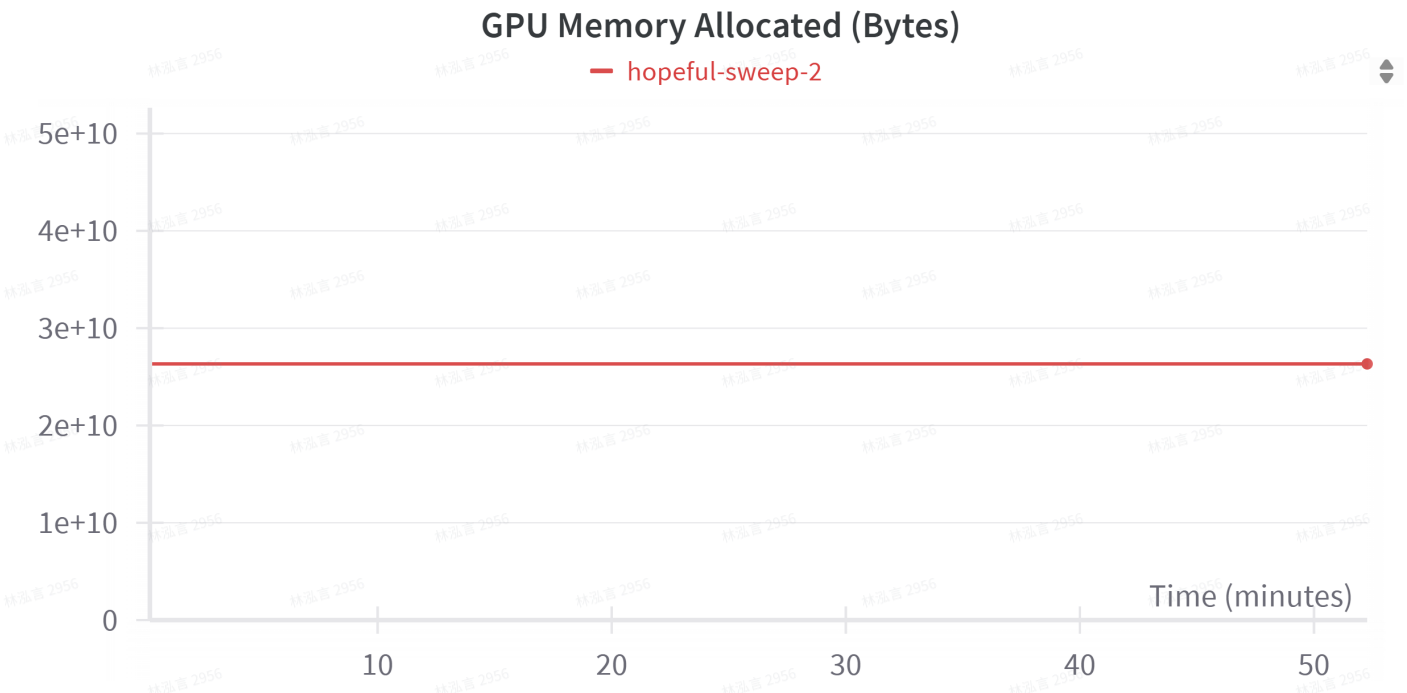


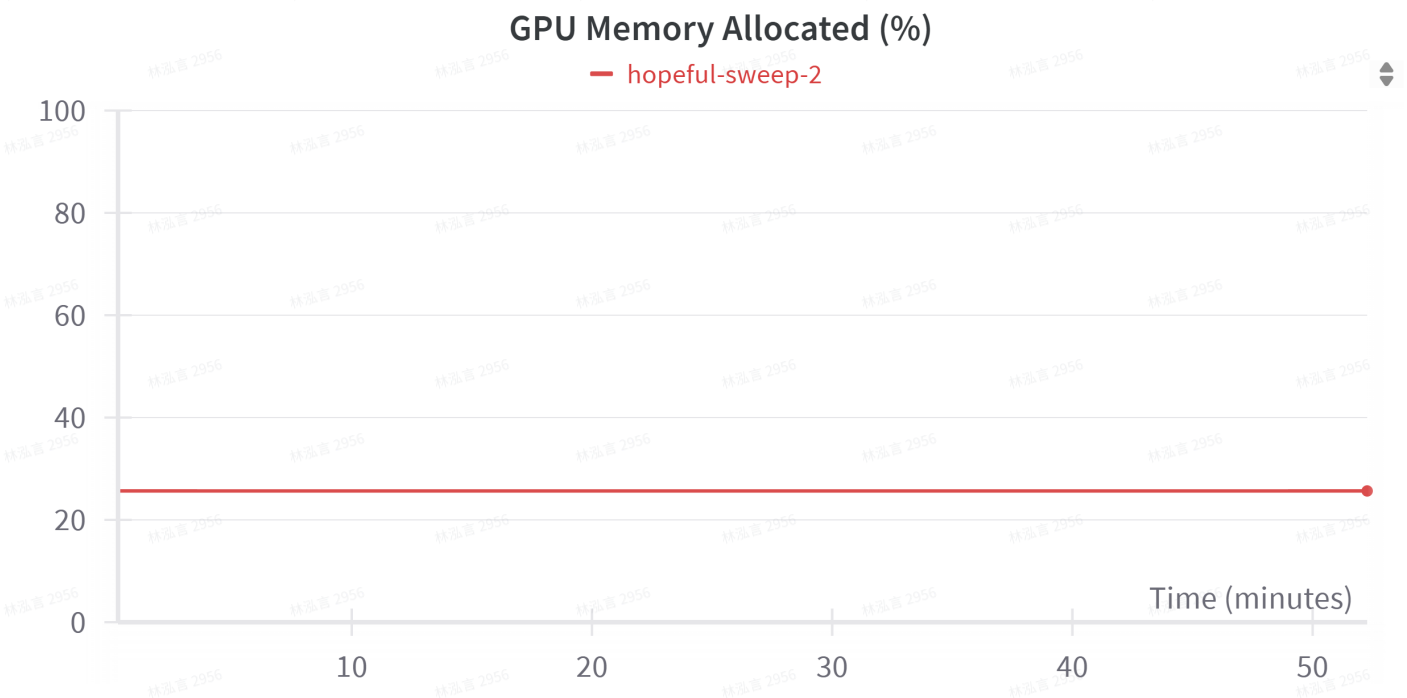
由上面实验得知，lr 的发散点是 1e-2，在第 5000 次训练迭代时，lr = 0.003 的 val/loss = 1.43164，低于 lr = 0.005 的 val/loss = 1.43249，所以当 batch_size = 128 时，最优的学习率 lr = 0.003。

二、批次大小的变化

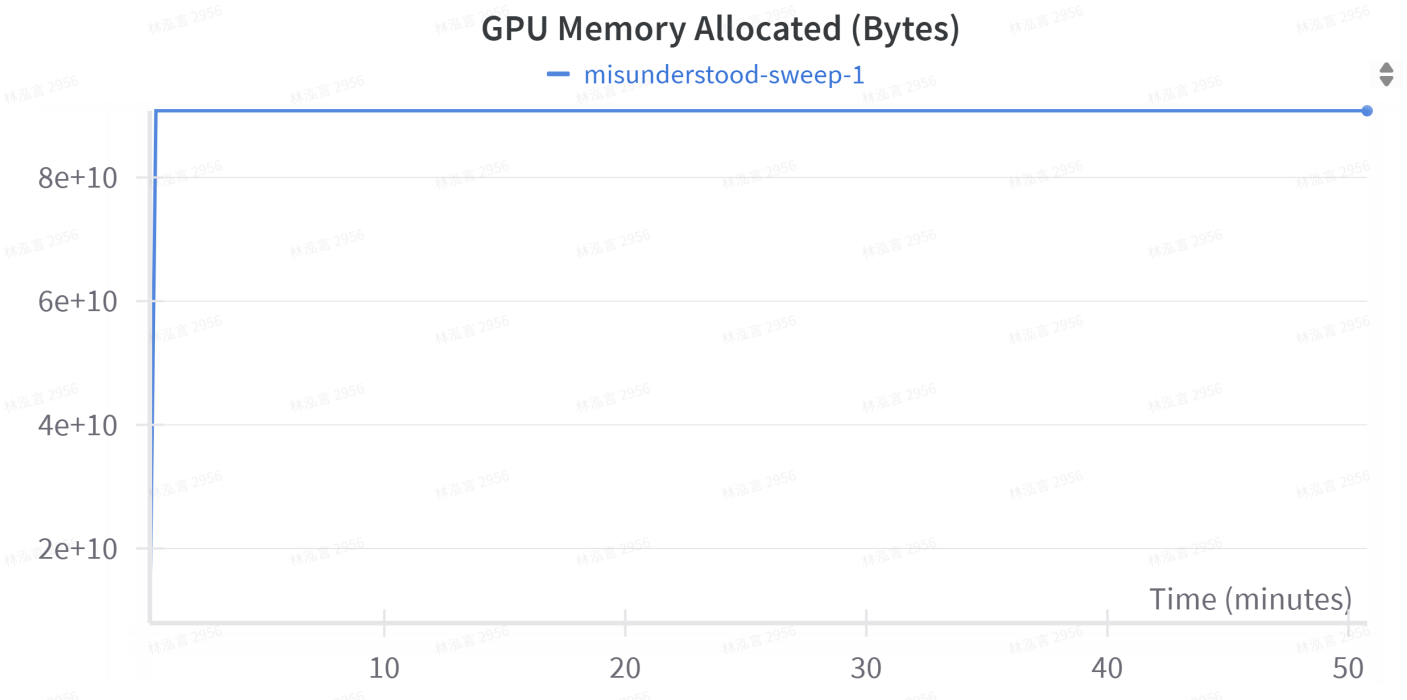
要求：调整批次大小直到 GPU 内存的上限。

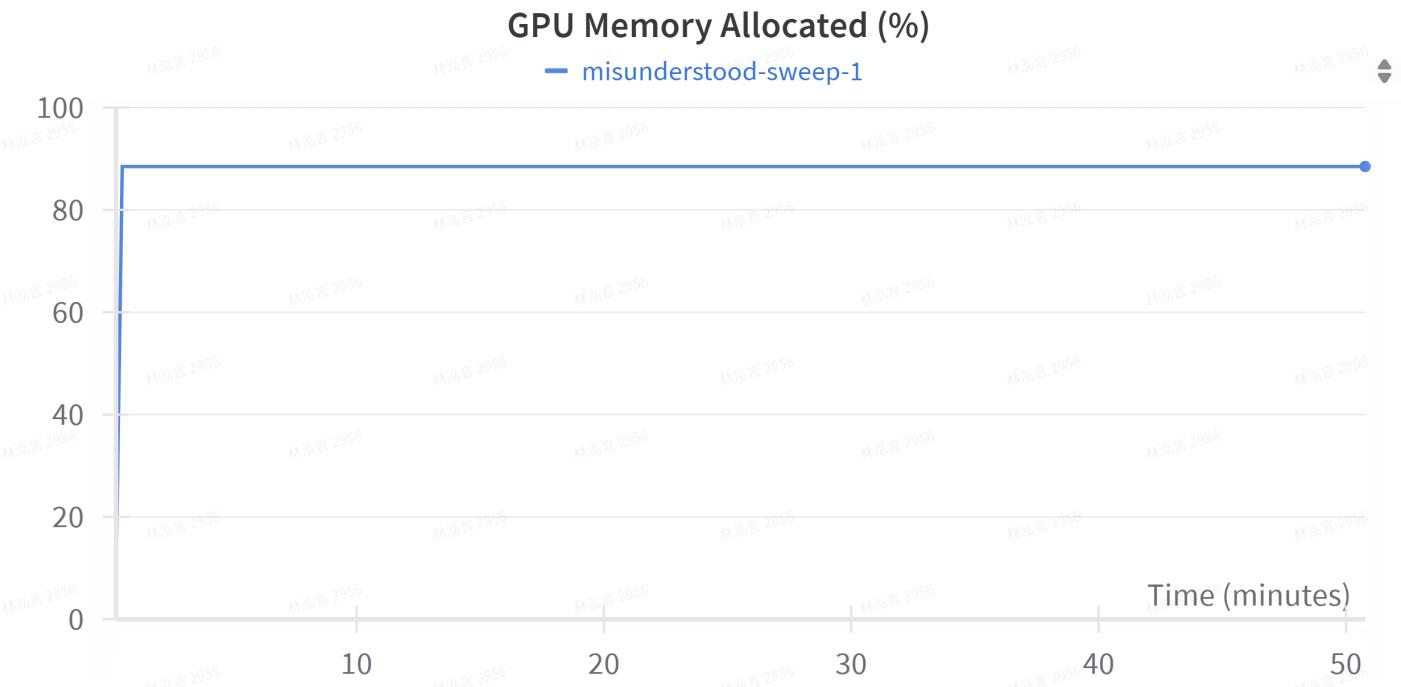
当 batch_size = 128 时，wandb 记录的 GPU 显存分配大小（字节）和百分比（%）分别是 26GB 和 25%，远未达到 GPU 内存上限。



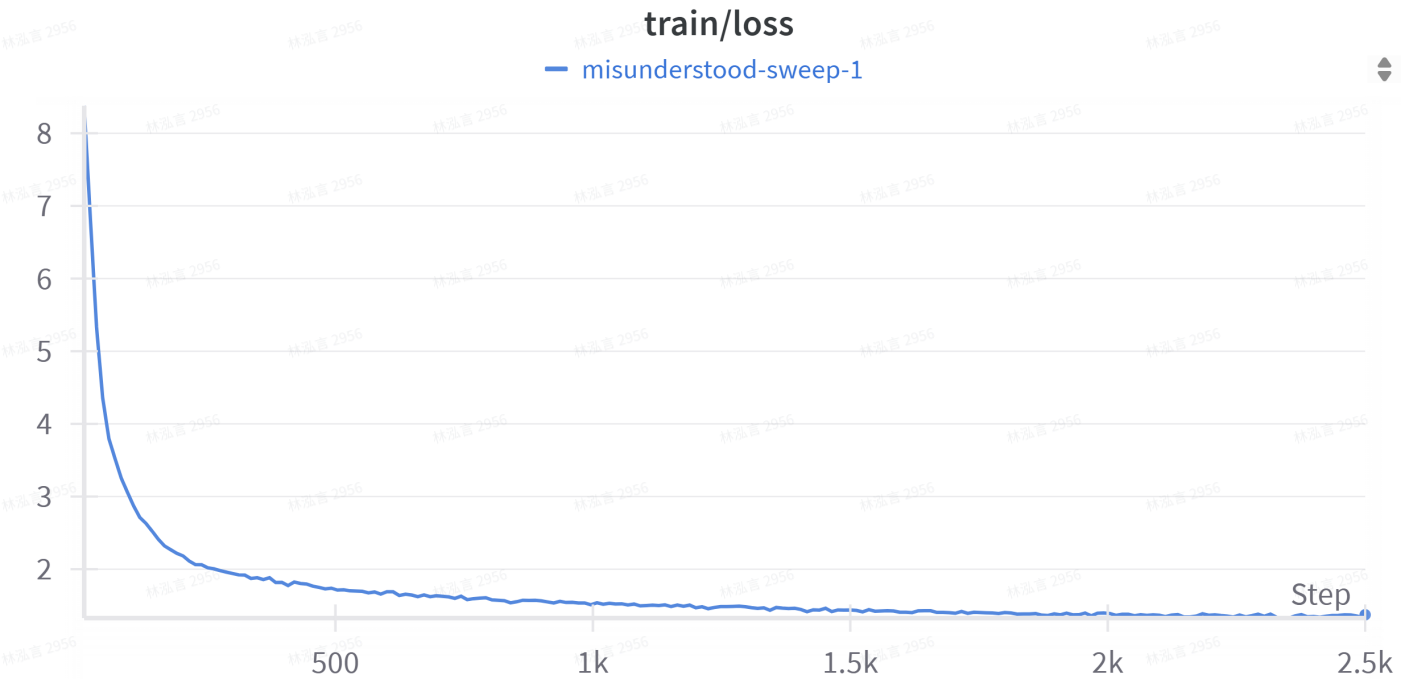


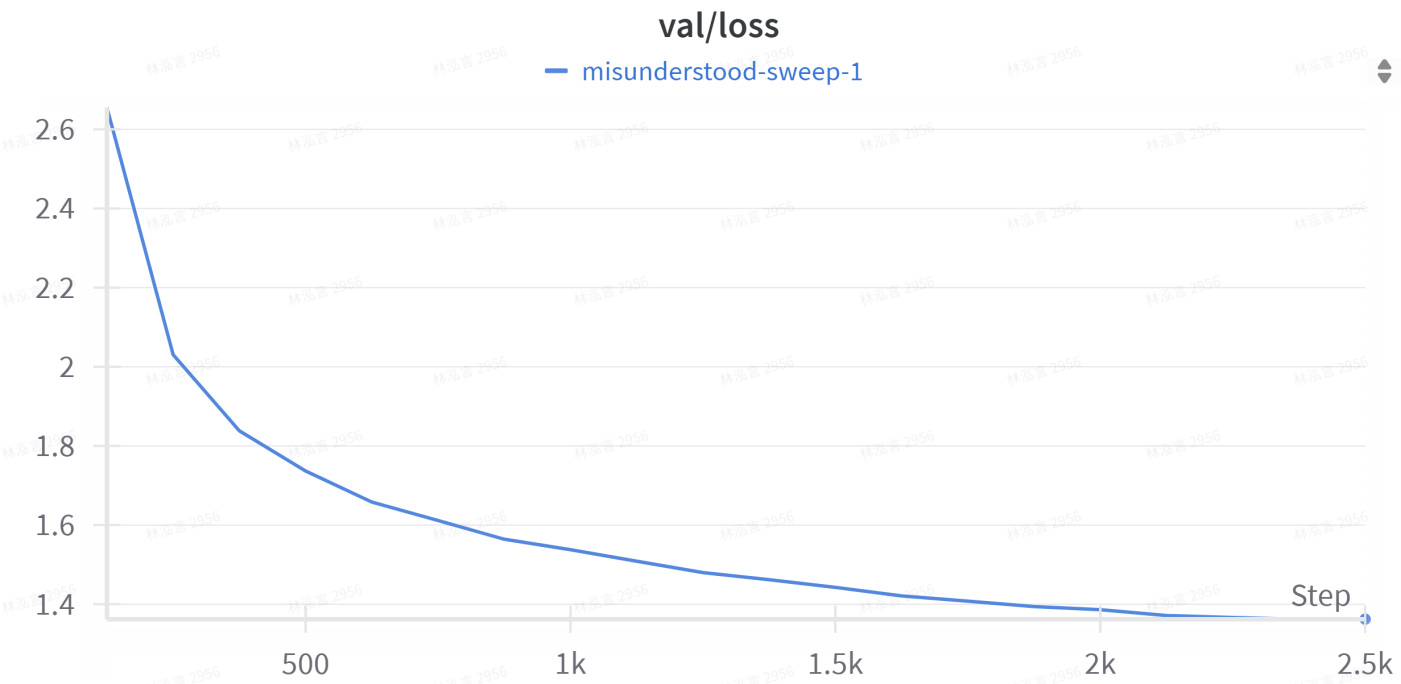
根据 GPU 分配 25% 的显存情况，将 batch_size 调整为 4 * 128 = 512，调整后重新训练，现在 GPU 显存分配 90GB，分配占比 88%。wandb 链接：https://wandb.ai/hongyanlin29-x_heart/cs336_ass1?nw=nwuserhongyanlin29





训练迭代过程的 train/loss 和 val/loss 曲线如下所示：





三、生成文本

使用 temperature = 0.8，top_p = 0.95，prompt_str = "Once upon a time"，生成 256 个 tokens。生成的文本如下：

Once upon a time, there was a little girl named Sue. Sue loved to play with her toys and eat yummy food. One day, she found a big, round onion in the kitchen. Sue thought the onion was very tasty and wanted to eat it.

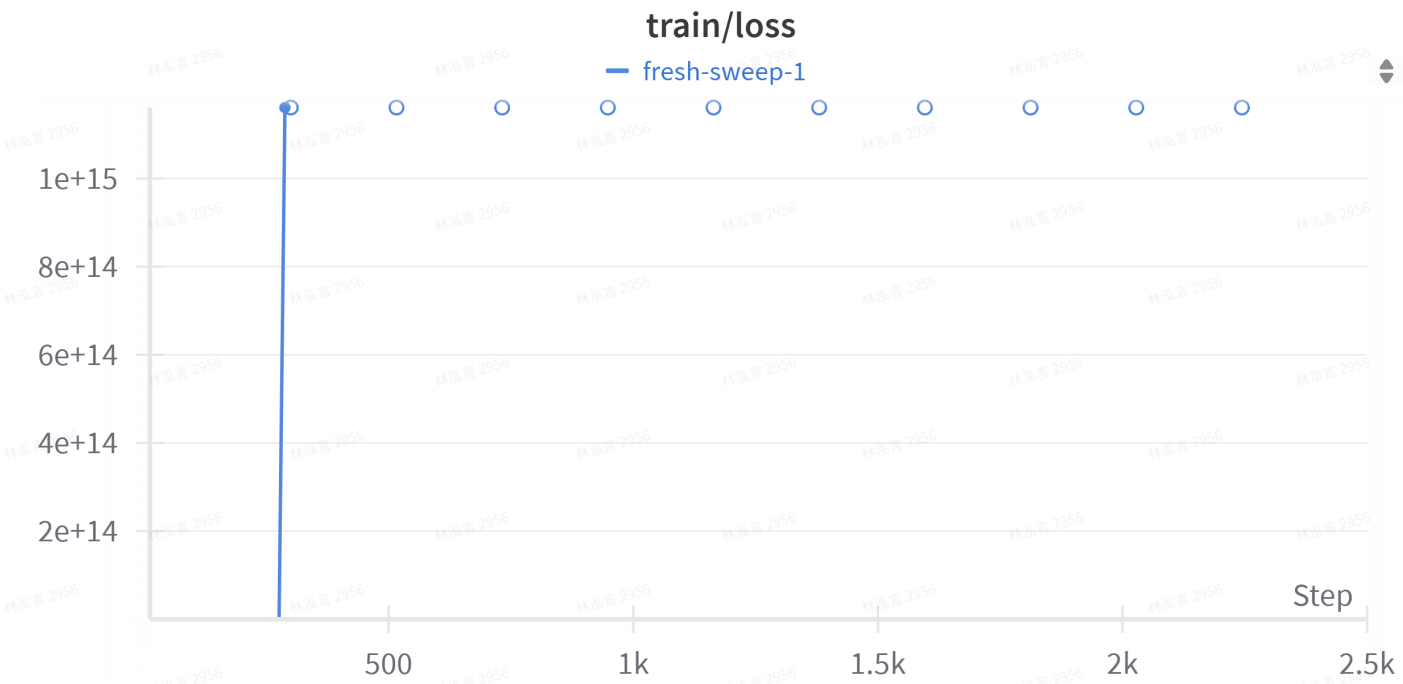
Sue's mom saw her with the onion and said, "Be careful with that onion, it is very sharp!" Sue did not listen and took a big bite of the onion. Suddenly, the onion turned into a big, round ball! Sue was very surprised.

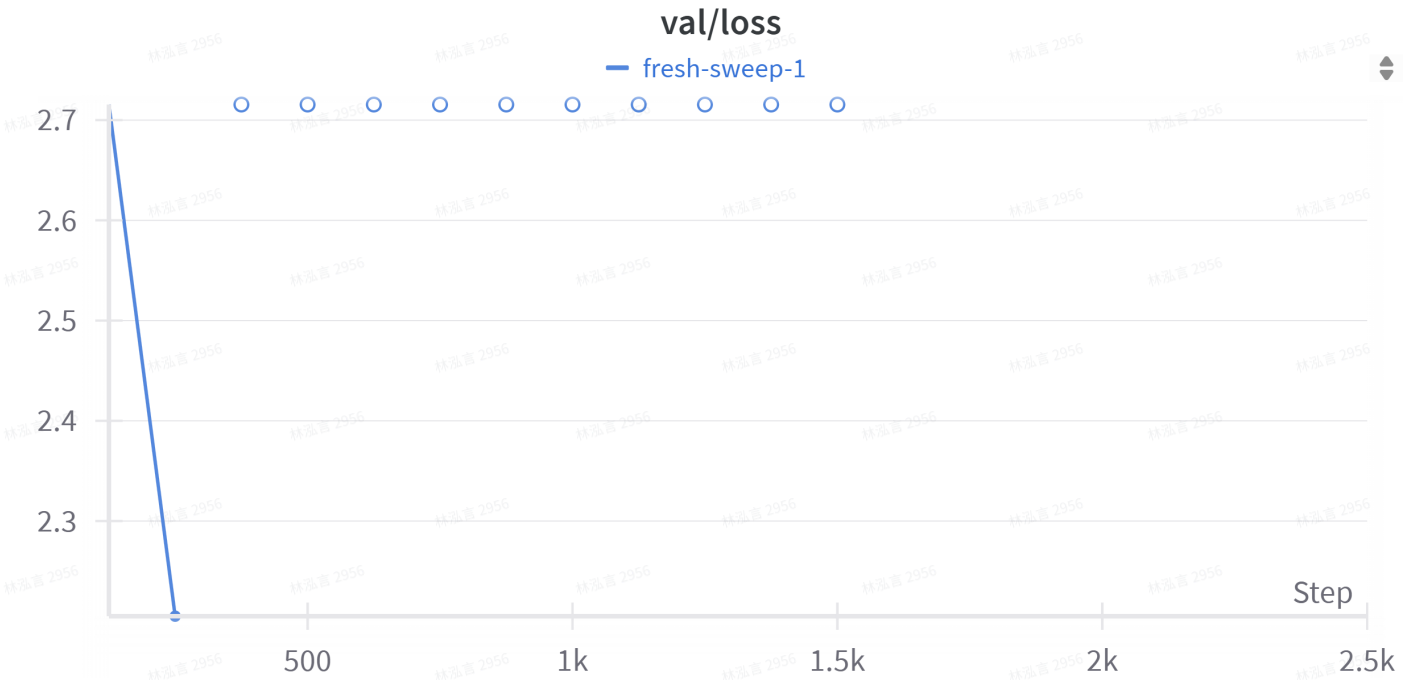
Sue's mom came into the kitchen and saw the big, round ball. She laughed and said, "Sue, that was not a ball! That was a ball! Now we have to clean it up!" They cleaned up the mess and Sue's mom washed the onion. From that day on, Sue learned to listen to her mom and never ate a tasty onion again.

四、移除 RMSNorm 并训练

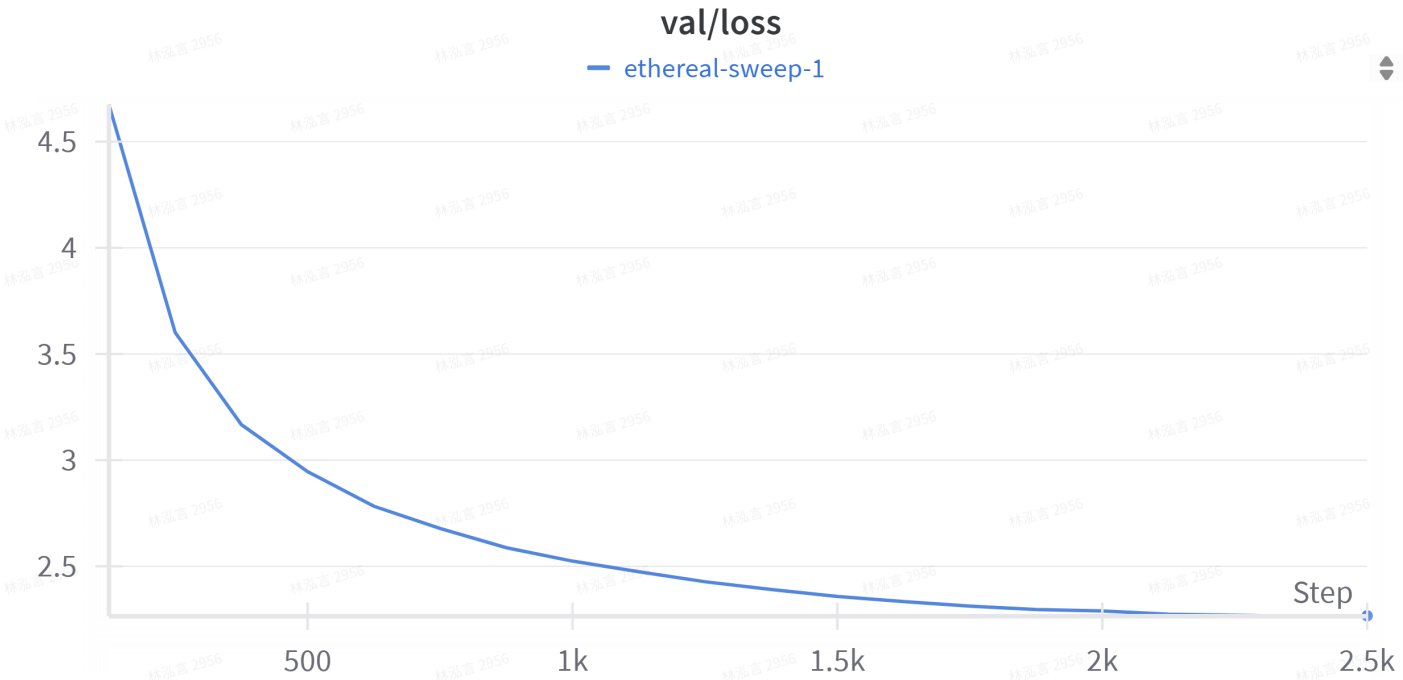
要求：从 Transformer 中移除所有的 RMSNorm 并进行训练，使用之前最佳的 lr 会发生什么？使用一个更低的 lr 能获得稳定吗？

移除 RMSNorm 之后，使用之前最佳的 lr = 0.003，train/loss 和 val/loss 的值在训练过程中都变成了 NaN。





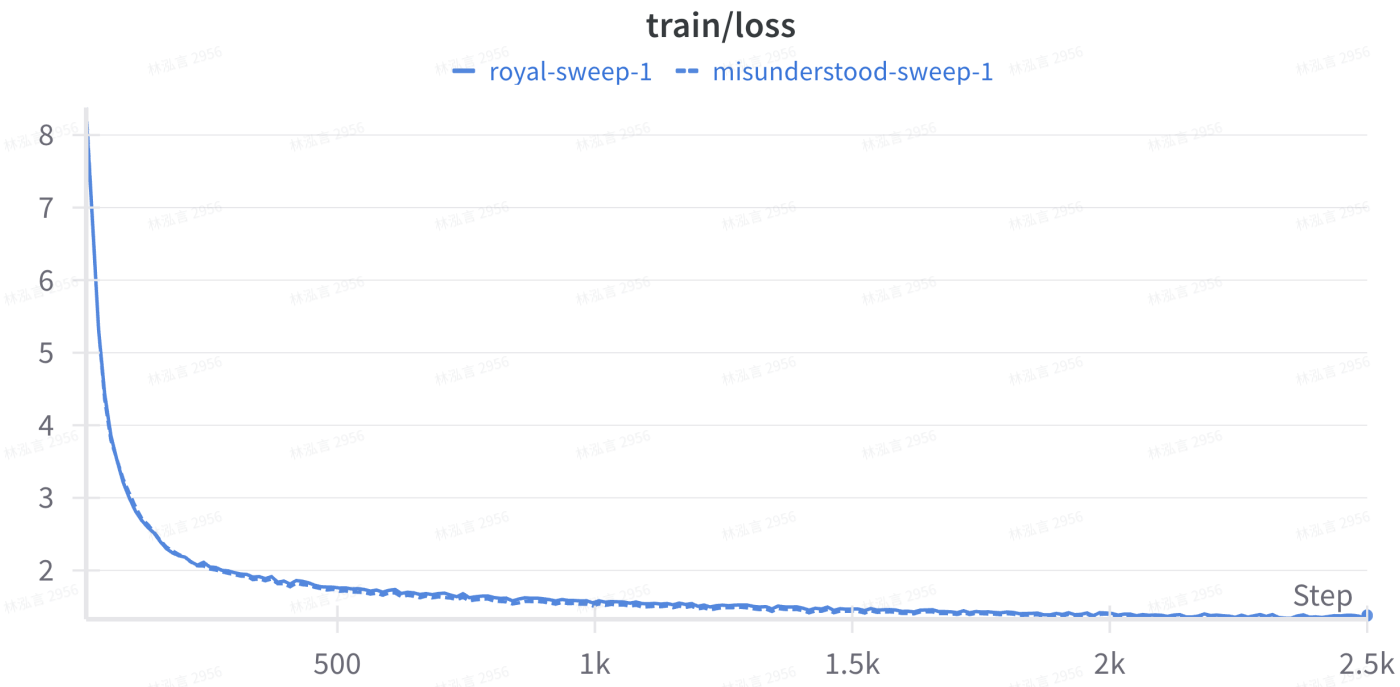
现在将 lr 降低到 0.0001，重新训练，损失值得变化如下图所示。可以看到，使用一个更低的 lr 能获得稳定。

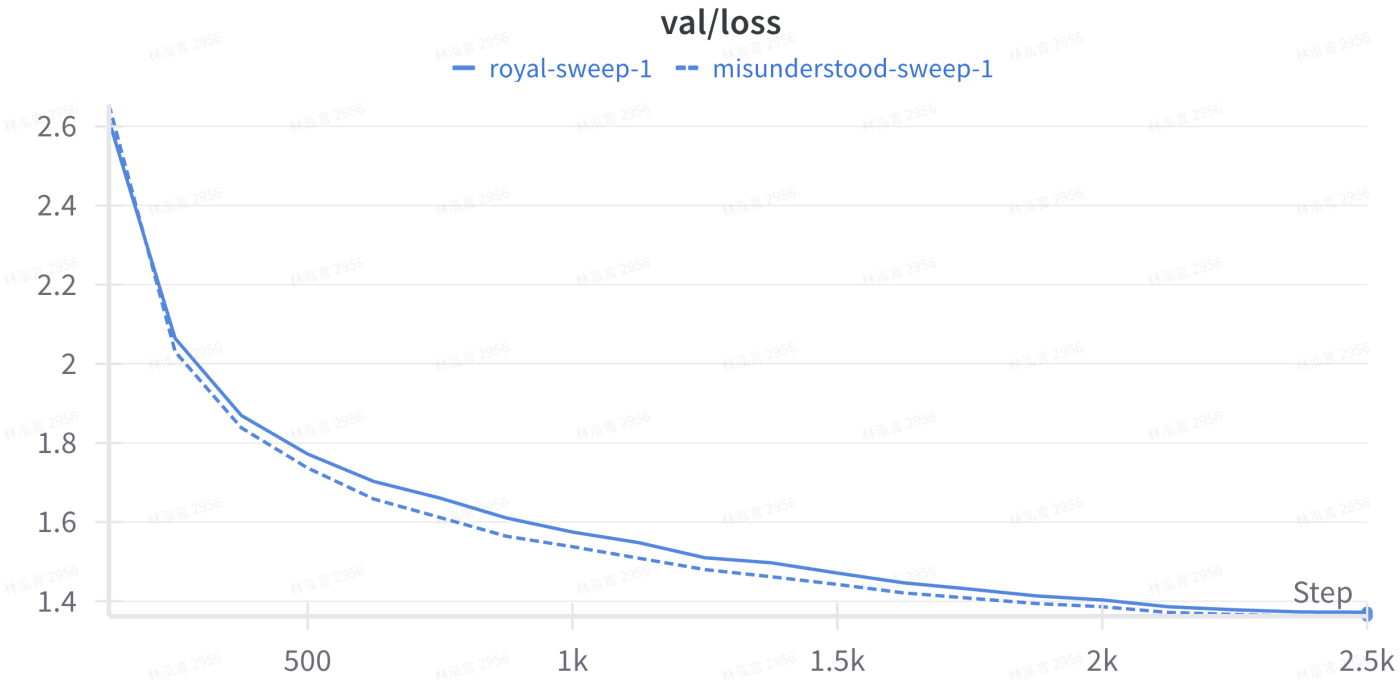


RMSNorm 的其中一个作用是稳定激活值得数值范围，移除它以后，模型在 forward 过程中产生了极大的激活值，最终导致交叉熵损失的溢出。

五、实现 post-norm 并训练

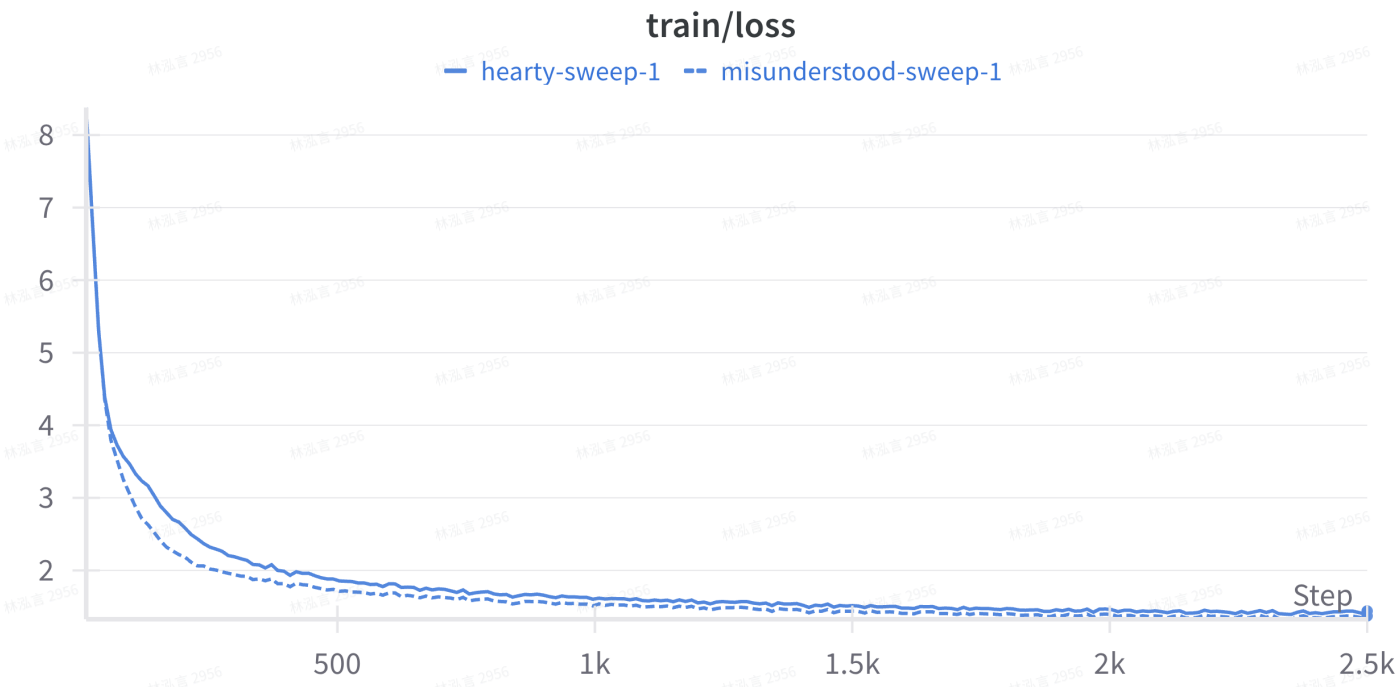
下图实线是 pre-norm 的曲线，虚线是 post-norm 的曲线。两者的 train/loss 曲线基本一样，val/loss 曲线有点不同，在前面的训练迭代中，pre-norm 的验证损失高于 post-norm，但在训练的最后两者基本一样。

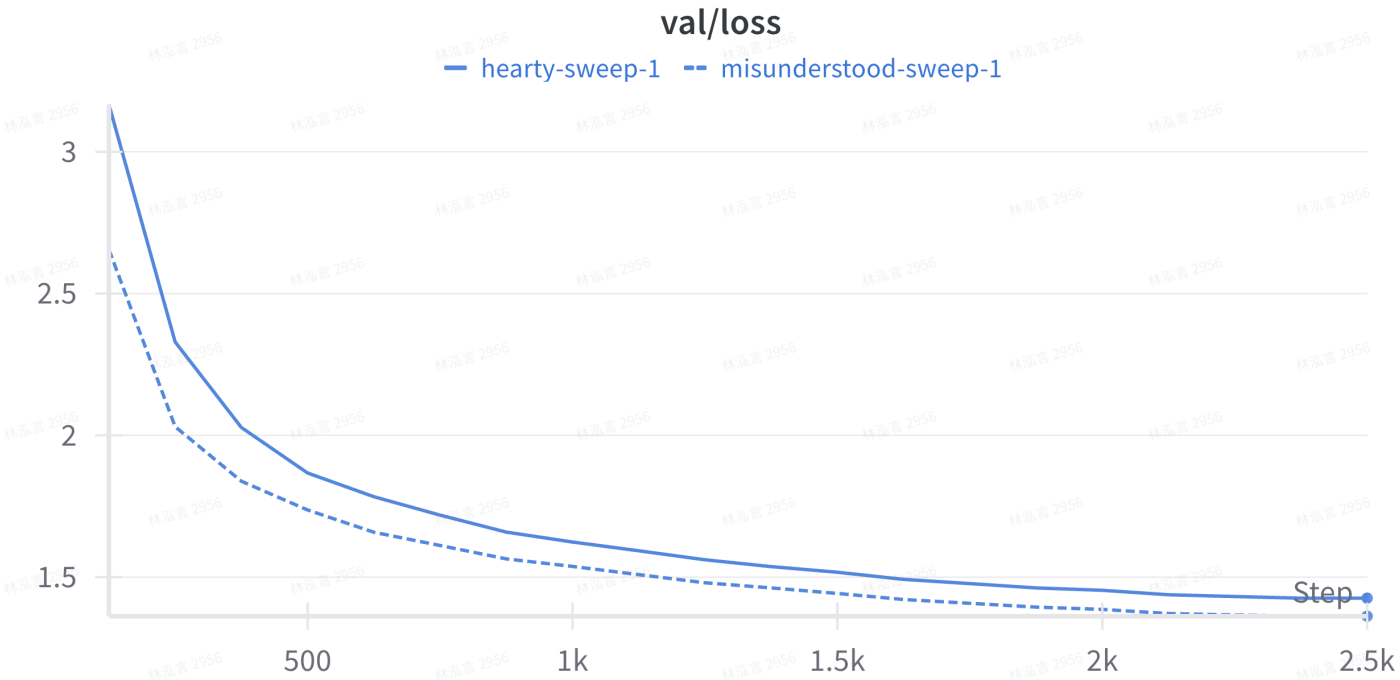




六、不使用 RoPE 编码训练

下图实线是不使用 RoPE 编码的曲线，虚线是使用 RoPE 的曲线。不使用 RoPE 的训练和验证损失都会更高。

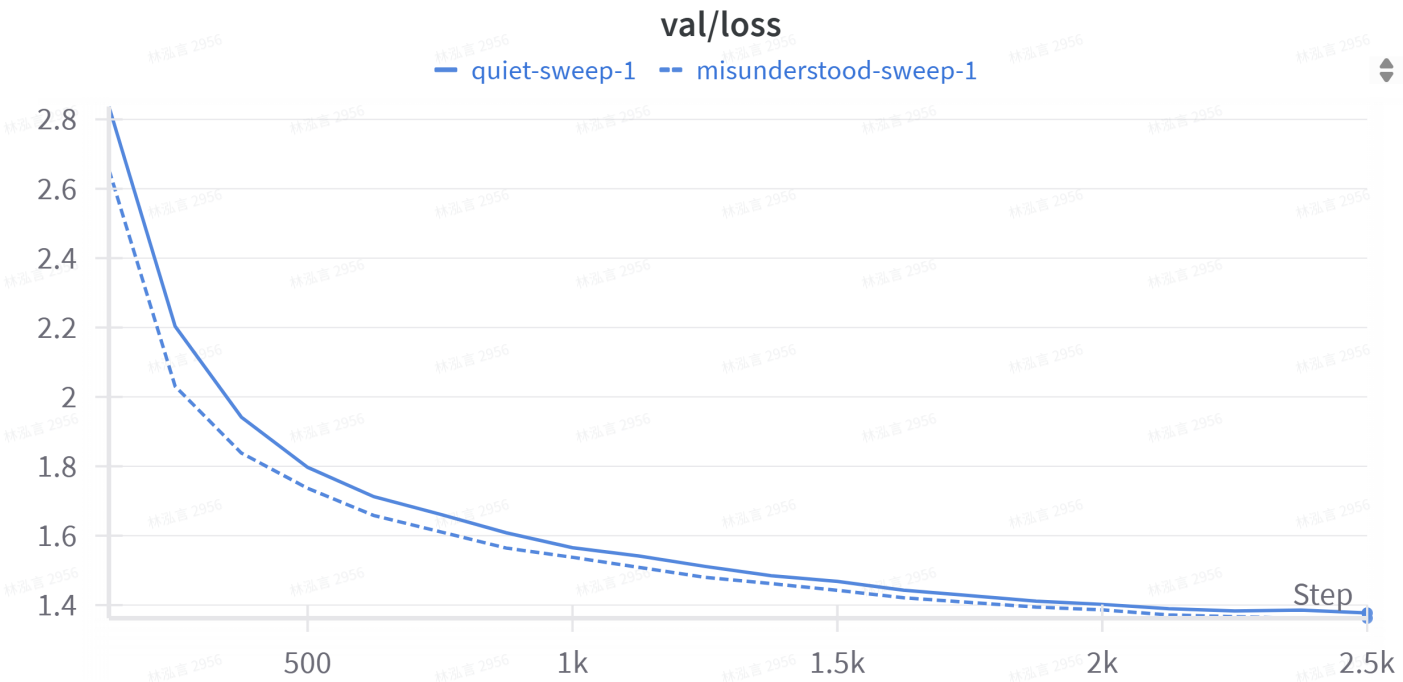




七、SwiGLU vs SiLU

注意：在 SwiGLU 的实现中，我们将内部前馈层的 d_ff 维度设置为大约 8/3 倍的 d_model，同时确保 d_ff mod 64 = 0。在 SiLU 实现中，为了将参数数量大致匹配 SwiGLU 的前馈层参数，应该将 d_ff 设为 4 倍的 d_model。

下图实线是使用 SiLU 的曲线，虚线是使用 SwiGLU 的曲线。两者的 loss 差别不大，SwiGLU 的 loss 稍微低一点。



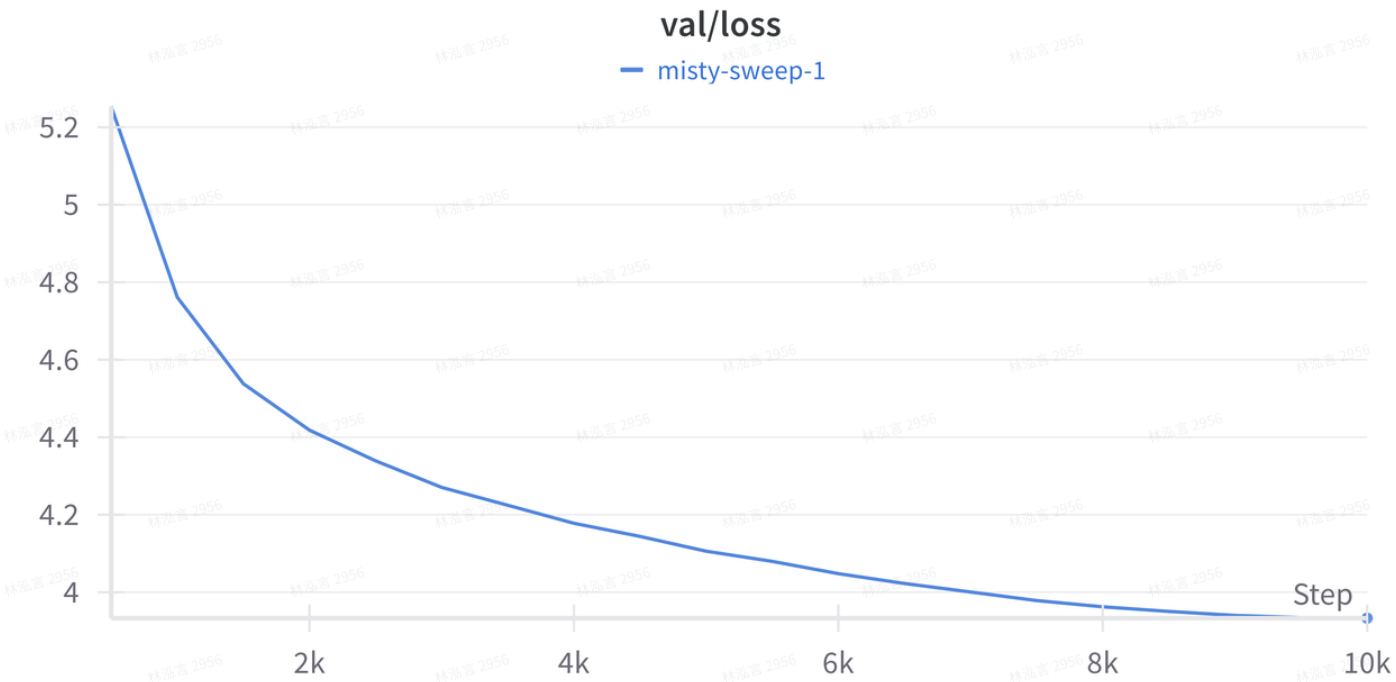
八、Experiment on OWT

要求：在 OpenWebText 上使用跟 TinyStories 相同的模型架构和总迭代次数训练一个语言模型。

train/loss = 3.914，val/loss = 3.933

https://wandb.ai/hongyanlin29-x_heart/cs336_ass1?nw=nwuserhongyanlin29


train/loss
misty-sweep-1
7
6
5
4
Step
2k
4k
6k
8k
10k


val/loss
misty-sweep-1
5.2
5
4.8
4.6
4.4
4.2
4
Step
2k
4k
6k
8k
10k