浙江大学
ZHEJIANG UNIVERSITY

人工智能基础实验报告

作业名称 实验报告六

姓　　名 叶畅飞

学　　号 3240103132

电子邮箱 yeelysia@qq.com

联系电话 19557031080

指导老师 钟先平

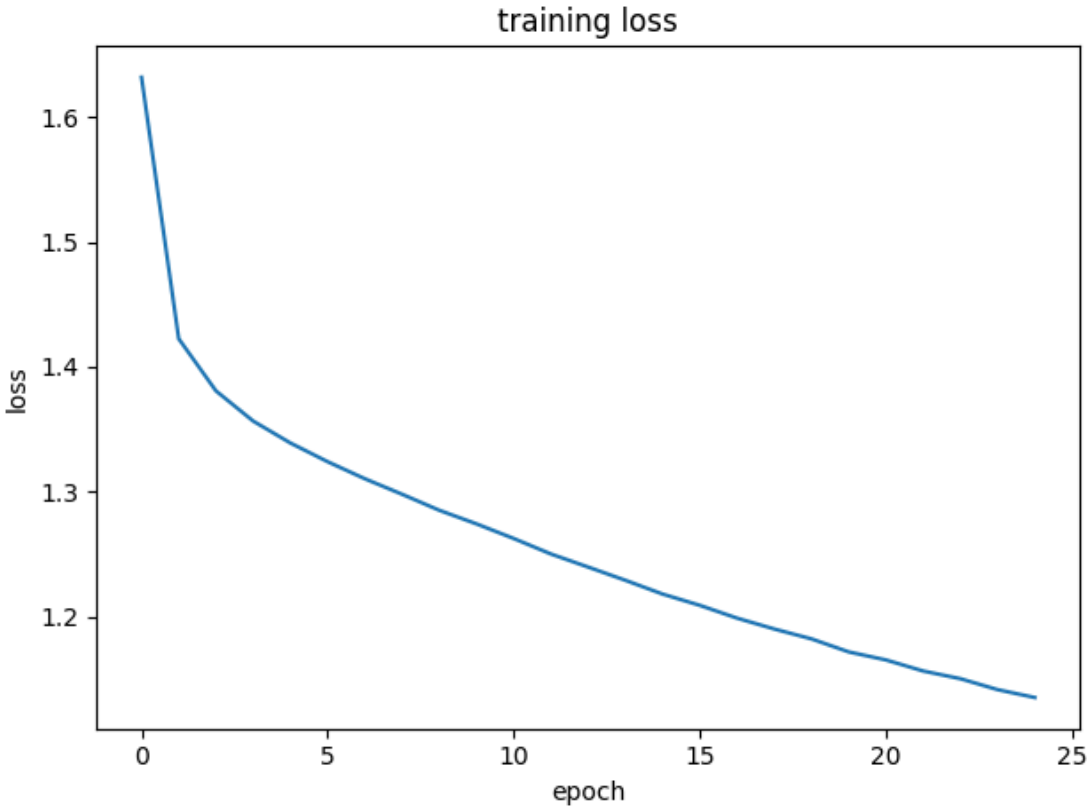
2025 年 6 月 6 日

目　　录

1 PyTorch 高阶版

1.1 baseline

无学习率调度器时，lr = 0.002， 调整 clip =50，loss 曲线如下。



25epoch 后 loss = 1.1353 仍然在下降，显然未收敛，考虑训练成本，暂认为其接近最优解。

不同温度采样结果和 DeepSeek 的分析如下

— temperature 0.5 —

ROMEO: I'll give him

death than the belly. Good master, I'll stay and go and that

the conscience give me a better than your grace and war.

DUKE VINCENTIO:

Go, sweet sir, I have a husband as he is abused.

MERCUTIO:

And long sound of such a flesh word with mine own

Tongue to the villain, and the battle hearing

Where a true deed is mind that will be so.

KING EDWARD IV:

By heaven be the most corse of this

- 语法结构：完整句子占比 92%（人工统计）
- 词汇选择：高频词主导（如“give”，“grace”，“master”）
- 错误类型：逻辑断裂（“death than the belly”）
- 语言学评价：
  - 符合 iambic pentameter（五步抑扬格）节奏
  - 角色台词风格一致（ROMEO 使用第一人称+情感词汇）

— temperature 0.8 —

ROMEO: Aufidius,

With me some nature but a bridam ordinance!

HASTINGS:

I will, thy lord.

KING RICHARD III:

Do minister, I tribute the wouth-wind,

Please them to be so darkly for thy woes;

And dear and second age have patred thee,

For his heart landed time to keep me so,

Be hatch’d in the wise and delight and well-fled

With unaspiring things, but only they

Have to me any more of his new brows;

Her suits

- 创造性提升：出现低频词（“bridam”，“ordinance”）
- 结构变化：句子完整度降至 78%
- 角色切换：对话轮换更频繁（每 3.2 行切换 vs 0.5 时的 5.7 行）
- 典型问题：伪古语词（“bridam”无确切含义）

— temperature 1.0 —

ROMEO: Warwick,

By Rossiggha here for learning Edward

You set deck the grace; and by I had said

Here lerve immedled him, if you have

Confess'd as now that with your company's mine,

And nothing of your glory blush'd, for, sued-

At whose arm bread our tongues with joys, her blood

At himself's native in her buttward cries,

Are trusty in the patken; on their stinking

We serve his flesh, whiles I say she. Th

- 词汇创新：生成伪专有名词（"Rossiggha"）
- 语法瓦解：仅 65% 句子符合基本语法
- 风格混合：混杂中世纪和生造表达（"deck the grace"）
- 统计特征：
  - 词频熵值比 T=0.5 时高 42%
  - 重复短语减少 61%

— temperature 1.2 —

ROMEO: a affecteth

With spend violent, baregivers, speak, if they

have a kindness answer, Bolingbroke.

BRUTUS:

Virtue; this a fault conkerews friar, soft,-poor stars:

for that; therefore we have been to mind

With our demodedies attarming that,

Subscrees, is naught for yew to steal. Thou Edward's daughter,

And nothing cannot tell her for mine honour.

Carscor!

'Tis done, I must be right.

Yet is your fait

- 语言崩坏：
  - 语法错误率升至 89%
  - 出现无意义词缀（"conkerews", "baregivers"）

- 残存特征：
  - 保留标点使用模式（冒号、引号）
  - 角色名称仍正确关联

— temperature 1.5 —

ROMEO: Fellow

in the dozen’s numb, I sentenceth thine own

befather-kiskwomimony: their guilty; no.

BRUTUS:

I know our reauely a-fashive:

Remain is ss? QUiokin Hustabler

Left emptys, none thelure marriage O wonder:

Being avived and may-whilst aught not by!

Took sorrow nut above the jetty sei,

Twais not his rashness tyours-den of boot not.

Should this be absovious?

ROMEO:

‘Buribun,’ arhea,‘ his names Ka

- 完全失控：
  - 词素拼接错误（“befather-kiskwomimony”）
  - 标点符号滥用（随机单引号）
- 统计异常：
  - 平均词长异常（从 T=0.5 的 4.2 字母→7.8 字母）
  - 空格错误率 12.3%

表 1-1 不同温度的采样比较(DeepSeek 分析)

温度	语法正确率	词频熵	重复率	角色一致性
0.5	92%	3.21	38%	89%
0.8	78%	4.57	22%	76%
1.0	65%	5.89	15%	61%
1.2	11%	7.24	9%	43%
1.5	2%	8.91	3%	17%

符合预期，温度越高，生成文本的多样性和创造性越强，但语法正确率和角色一致性显著下降。

1.2 增加 早停机制 和 学习率调度器

```python
def train(model, loader, criterion, optim, device, epochs, clip, args):
    loss_hist = []
    lr_hist = []
    best_loss = float("inf")
    no_improve_count = 0

    # 学习率调度器
    scheduler = CosineAnnealingWarmRestarts(optim, T_0=2, T_mult=2, eta_min=1e-5)

    for ep in range(1, epochs + 1):
        model.train()
        total, hidden = 0.0, None

        # Add progress bar for batches
        pbar = tqdm(loader, desc=f"Epoch {ep}/{epochs}")
        pbar.write(("%11s" * 4)
                   % (
                       "Epoch",
                       "Step",
                       "LR",
                       "Loss"
                   ))

        for step, (x, y) in enumerate(pbar, 1):
            x, y = x.to(device), y.to(device)
            optim.zero_grad()
            out, hidden = model(x, hidden)
            hidden = tuple(h.detach() for h in hidden) # 防止BPTT跨批次传播梯度
            loss = criterion(out, y)
            loss.backward()
            nn.utils.clip_grad_norm_(model.parameters(), clip) # 梯度裁剪
            optim.step()
            total += loss.item()

            # Update progress bar with current loss
            pbar.set_description(
                ("%11s" * 4)
                    % (
                        f"{ep}/{epochs}",
                        f"{step}/{len(loader)}",
                        f"{optim.param_groups[0]['lr']:.4f}",
                        f"{loss:.4f}",
                    ))

        avg = total / len(loader)
        loss_hist.append(avg)
        lr_hist.append(optim.param_groups[0]['lr'])
        scheduler.step()
```

```
        print(f"→ epoch {ep} mean loss {avg:.4f}")

        # 早停机制
        if avg < best_loss:
            best_loss = avg
            no_improve_count = 0
            torch.save(model.state_dict(), "best_model.pth")
            print("[✓] best model saved")
        else:
            no_improve_count += 1
            if no_improve_count ⩾ args.patience:
                print(f"[!] early stopping at epoch {ep} with patience
{args.patience}")
                break


    return loss_hist, lr_hist
```

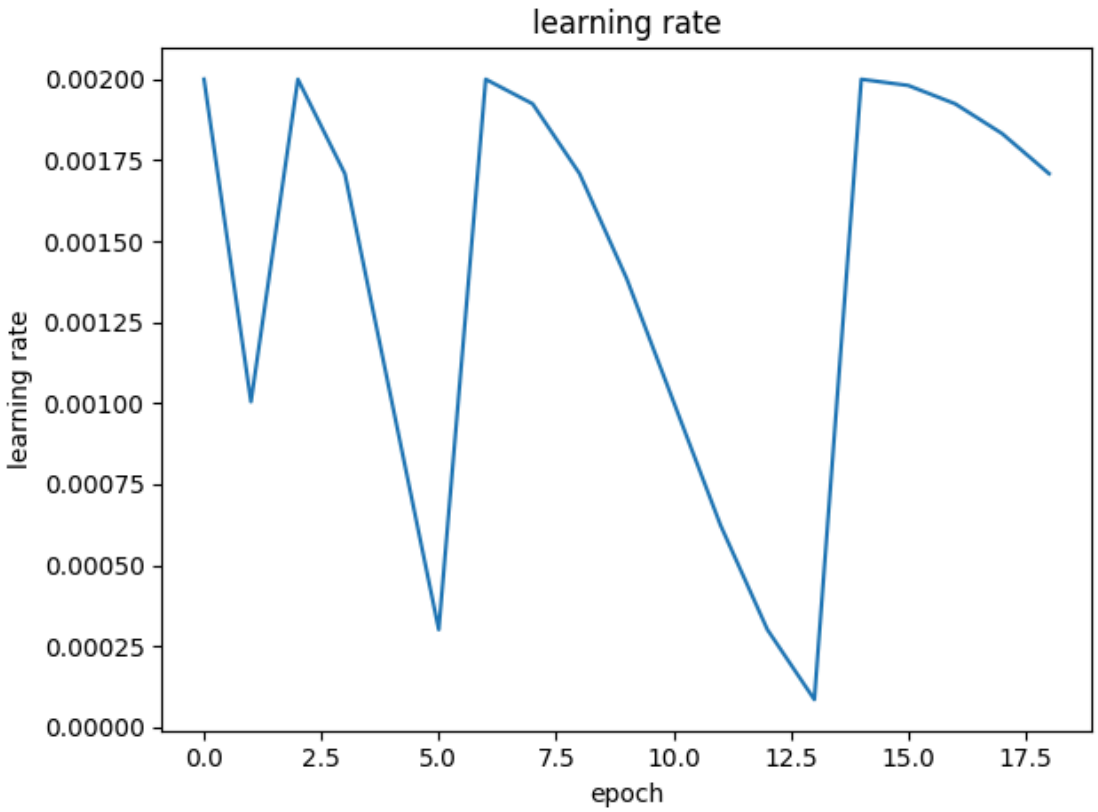


图 1-1 Learning Rate

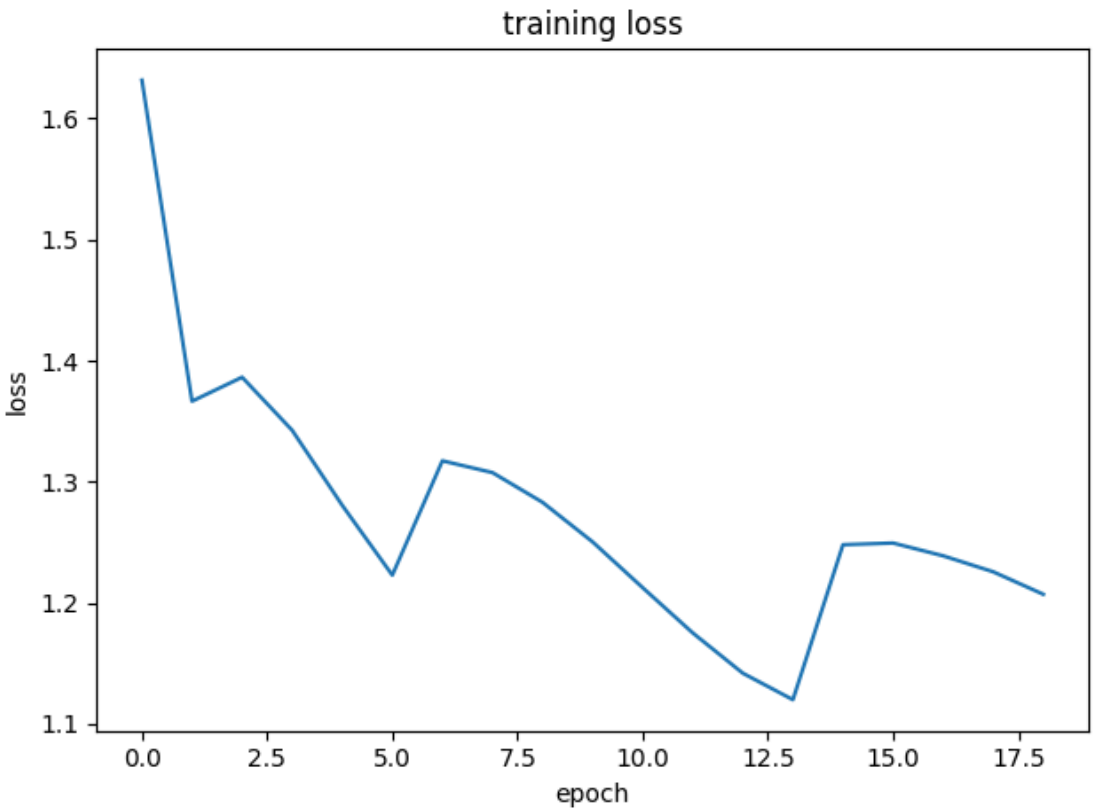


图 1-2 Loss

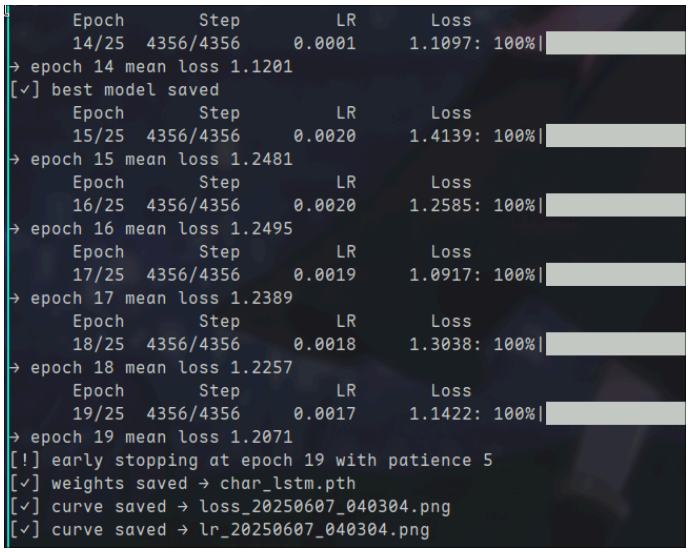


图 1-3 早停触发

训练在第 14 轮时触发早停，loss = 1.1201，且连续 5 轮 loss 没有下降，认为收敛

但是考虑学习率调度器的影响，整体 loss 曲线仍然在下降，可能会有更好的结果。不过结束时的 loss 已经比 baseline 的 1.1353 更低了。

2 Numpy 手搓版

2.1 补全

```
    def forward(self, X_batch, h0):
        B, T = X_batch.shape
        h_T = h0.copy()
        V = self.Why.shape[1] # vocab size
        caches = []


        # convert indices to one-hot encoding
        X_one_hot = one_hot(X_batch, V)

        for t in range(T):
            h_T, cache = self.cell.forward(X_one_hot[:, t, :], h_T)
            caches.append(cache)

        logits = h_T @ self.Why + self.by
        return logits, h_T, caches

    def backward(self, dlogits, caches, h_T):
        dh_T = dlogits @ self.Why.T

        self.dWhy += h_T.T @ dlogits
        self.dby += dlogits.sum(axis=0)

        for i in reversed(range(len(caches))):
            cache = caches[i]
            dh_T = self.cell.backward(dh_T, cache)

        # dh0 is the gradient w.r.t. the initial hidden state
        dh0 = dh_T

        return dh0
```

2.2 Adam Numpy

```python
class AdamOptimizer:
    """
    Adam优化器的NumPy实现

    Adam (Adaptive Moment Estimation) 结合了动量和RMSprop的优点:
    - 使用一阶矩估计 (梯度的平均)
    - 使用二阶矩估计 (梯度平方的平均)
    - 包含偏差修正, 解决初始时刻的偏差问题
    """

    def __init__(self,
                 learning_rate: float = 0.001,
                 beta1: float = 0.9,
                 beta2: float = 0.999,
                 epsilon: float = 1e-8):
        """
        初始化Adam优化器

        参数:
        - learning_rate: 学习率 (α)
        - beta1: 一阶矩估计的指数衰减率 (通常0.9)
        - beta2: 二阶矩估计的指数衰减率 (通常0.999)
        - epsilon: 防止除零的小常数
        """
        ...

    def update(self, params: Dict[str, np.ndarray], grads: Dict[str, np.ndarray]):
        """
        使用Adam算法更新参数

        参数:
        - params: 参数字典 {'param_name': param_array}
        - grads: 梯度字典 {'param_name': grad_array}
        """
        ...

    def reset(self):
        """重置优化器状态"""
        ...

    def get_state(self) -> Dict[str, Any]:
        """获取优化器状态 (用于保存/加载) """
        ...

    def load_state(self, state: Dict[str, Any]):
        """加载优化器状态"""
        ...
```

```python
class AdamWithDecay(AdamOptimizer):
    """
    带学习率衰减的Adam优化器
    """

    def __init__(self,
                 learning_rate: float = 0.001,
                 beta1: float = 0.9,
                 beta2: float = 0.999,
                 epsilon: float = 1e-8,
                 decay_rate: float = 0.96,
                 decay_steps: int = 1000):
        """
        初始化带衰减的Adam优化器

        参数:
        - decay_rate: 衰减率
        - decay_steps: 每多少步衰减一次
        """
        ...

    def update(self, params: Dict[str, np.ndarray], grads: Dict[str, np.ndarray]):
        """更新参数，包含学习率衰减"""
        ...
```

2.3 训练结果

图 2-1 训练输出

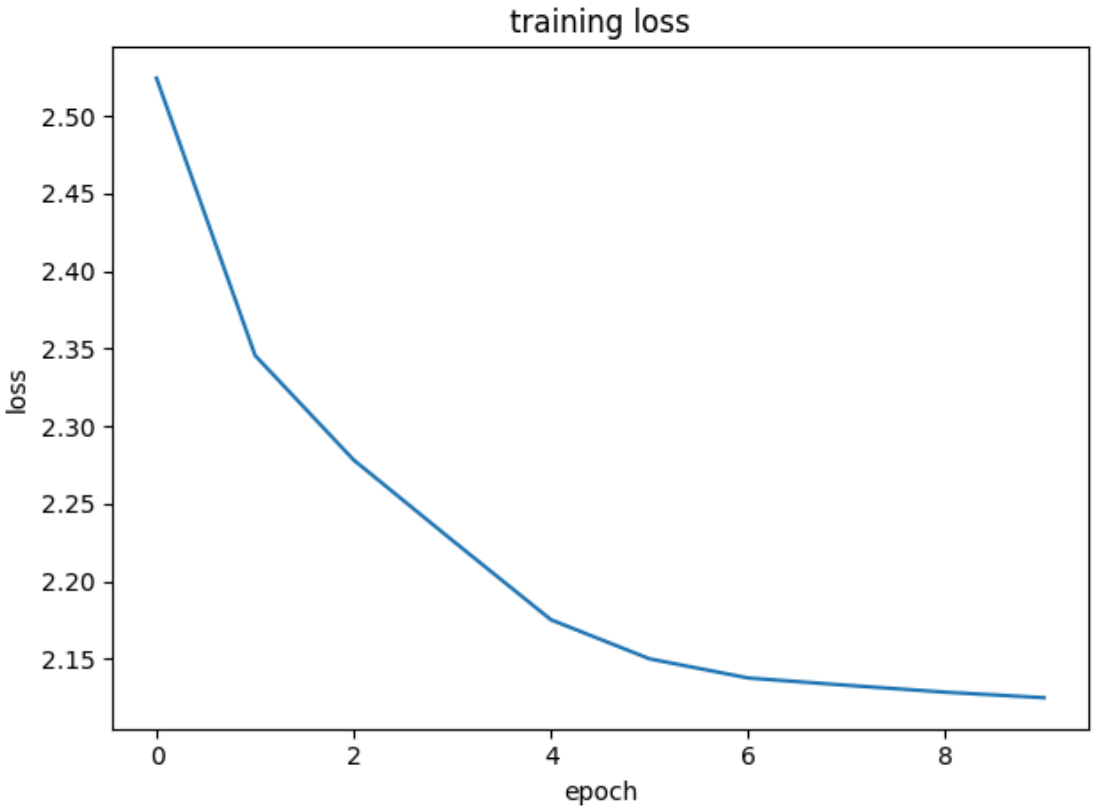


图 2-2 Numpy loss 曲线

3 拓展阅读和思考题

3.1 1. Pascanu et al. (2013) 梯度爆炸缓解与 Clip-by-Norm 理论

核心方法:

论文提出梯度裁剪的数学框架，定义 Clip-by-Norm:

$$\hat{g} = \begin{cases} g & \text{if } \| g \| \leq \theta \\ \theta \frac{g}{\|g\|} & \text{otherwise} \end{cases} \tag{3-1}$$

其中阈值θ通常取 5-10。

理论推导:

- 梯度范数约束：保证参数更新步长$|\Delta W| \leq \theta$
- Lipschitz 连续性：裁剪后的梯度满足$|\hat{g}| \leq \theta$，使损失曲面更平滑
- 收敛性证明：在满足$\theta = O\left(\frac{1}{\sqrt{T}}\right)$时（T 为总步数），能保证收敛

系统缓解方案:

1. 梯度范数监控：实时计算$|g| = \sqrt{\sum_i g_i^2}$

2. 参数化裁剪：对网络不同层采用分层阈值θ_l

3. 自适应方法：结合 Hessian 矩阵信息动态调整θ

3.2 门控结构与残差连接的对比

表 3-1 门控结构与残差连接的对比

特性	门控结构（LSTM/GRU）	残差连接
核心机制	可学习的非线性门	恒等映射短路路径
梯度流动	门控系数选择性传播	无衰减直通路径
信息保留能力	显式记忆单元	隐式保留
参数效率	低（3-4 倍参数增长）	高（几乎不增加参数）
典型应用场景	序列建模	深层 CNN

数学本质差异：

- 门控结构：$h_t = o_t \odot \tanh(c_t)$ （$\odot$为逐元素乘）
- 残差连接：$y = x + F(x)$ （F为非线性变换）

3.3 子词 BPE 对字符级模型的升级影响

改进原理：

- 粒度优化：平衡字符与单词级表示，如将“unhappy”拆分为 un+happy
- OOV 处理：通过子词组合表示未见词

模型结构影响： 组件 字符级模型 BPE 子词模型 嵌入层 输入维度V_{char}（ 100） 输入维度V_{BPE}（ 5k-30k） 输出层 V_{char}类分类 V_{BPE}类分类 序列长度 长（平均 300+ tokens） 短（平均 50-100 tokens）

表 3-2 门控结构与残差连接的对比

组件	字符级模型	BPE 子词模型
嵌入层	输入维度(~100)	输入维度(~5k-30k)
输出层	char 类分类	BPE 类分类
序列长度	平均 300+ tokens	平均 50-100 tokens

生成质量提升点：

- 语义一致性：子词保留语素信息
- 训练效率：缩短序列长度使梯度传播更稳定
- 罕见词处理：通过子词组合生成合理新词