第二次实验：基于神经网络的情感分类、命名实体识别与机器翻译

实验内容

本次实验中我们完成了：

任务\模型	CNN	RNN	Transformer
情感分类	✓	✓	✓
命名实体识别		✓	✓
机器翻译			?

其中RNN代表LSTM和GRU两种，包含可选的双向配置；对于情感分类和命名实体识别任务，Transformer使用了类似BERT的Encoder-Only结构，只是作为提取特征的主干网络；对于机器翻译任务，Transformer使用了完全体的Encoder-Decoder结构。其中，机器翻译任务由于时间和算力的限制，没有来得及完成。

实验原理

情感分类

情感分类任务旨在根据输入文本的内容，预测其情感类别（如正面、负面）。情感分类模型通过提取文本中的特征，捕获其语义信息，并利用分类器完成情感类别的预测。情感分类是以句子为单位的整体性的识别任务，所以CNN、RNN、BERT都比较合适。

本实验中，情感分类任务的模型框架被统一在 `SentimentModel` 中，输入词ID序列，输出情感类别ID。其中包含一个嵌入层，一个提取情感特征的可替换主干网络，还有一个由多层感知机组成的分类器。

命名实体识别

命名实体识别（NER）任务的目标是从文本中提取特定类别的实体（如人名、地名、组织名等），并对这些实体进行标注。NER 通常是一个序列标注问题，要求模型结合上下文信息为每个词或字符分配一个预定义的标签（如 B-PER、I-PER、O 等）。在本实验中，采用基于条件随机场（CRF）的NER，先给出每个词或字符对应每个标签的概率$P(t_i|w_*)$，然后用CRF结合标签转移概率$P(t_{i+1}|t_i)$，利用Viterbi搜索给出最大概率标签序列$\arg\max_{t_{0:l}} P(t_{0:l}|w_{0:l})$.

本实验中，命名实体识别任务框架被统一在 `CRF_NER` 中，输入词ID序列，输出标签ID序列。其中包含一个嵌入层，一个用于提取特征的可替换的主干网络，一个由多层感知机组成的分类器，还有一个CRF。

机器翻译

机器翻译是将源语言句子自动翻译成目标语言句子的任务。在本实验中，使用了 Transformer 模型实现机器翻译。

本实验中，机器翻译任务框架被统一在 `Translater` 中，输入源语言ID序列，输出目标语言ID序列。其中包含一个嵌入层，一个用于将源嵌入序列和目标嵌入序列映射到下一个目标嵌入的主干网络，一个由多层感知机组成的分类器，用于将嵌入表示映射到目标语言ID概率分布。

CNN

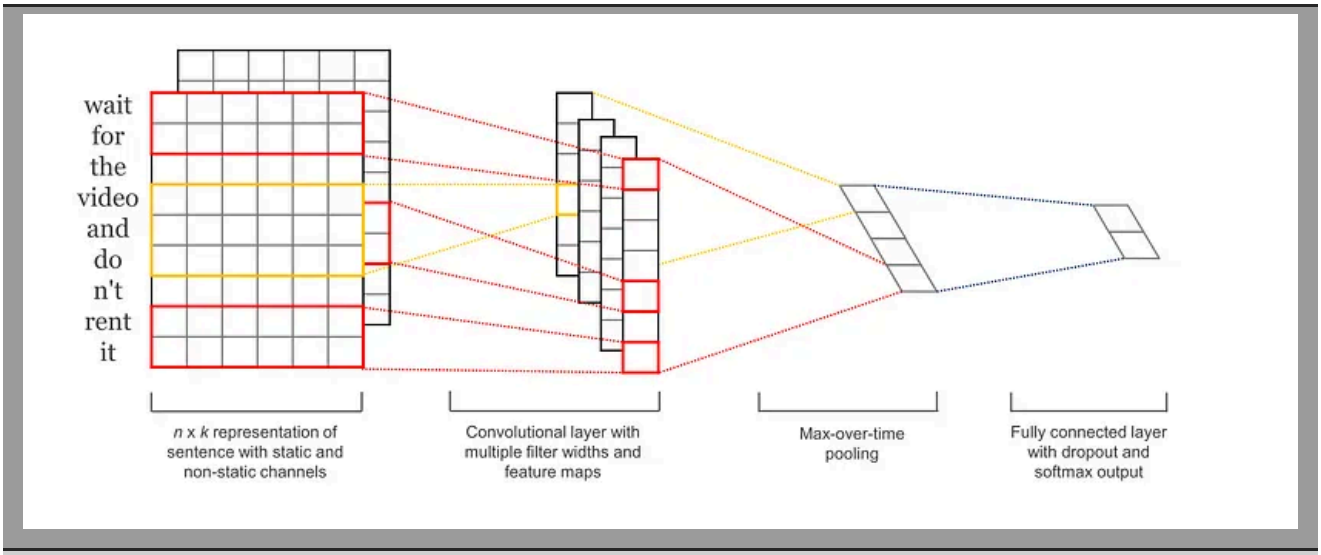


图1 TextCNN用于文本特征提取

CNN 本来用于图像领域，使用二维卷积可以很好地提取图像中的特征。对于文本这样的一维序列，TextCNN 使用一维卷积，对并排的词向量序列进行卷积，就可以完成对于文本信息的特征提取。本实验使用 `nn.Conv1d` 实现这一过程。

RNN（递归神经网络）

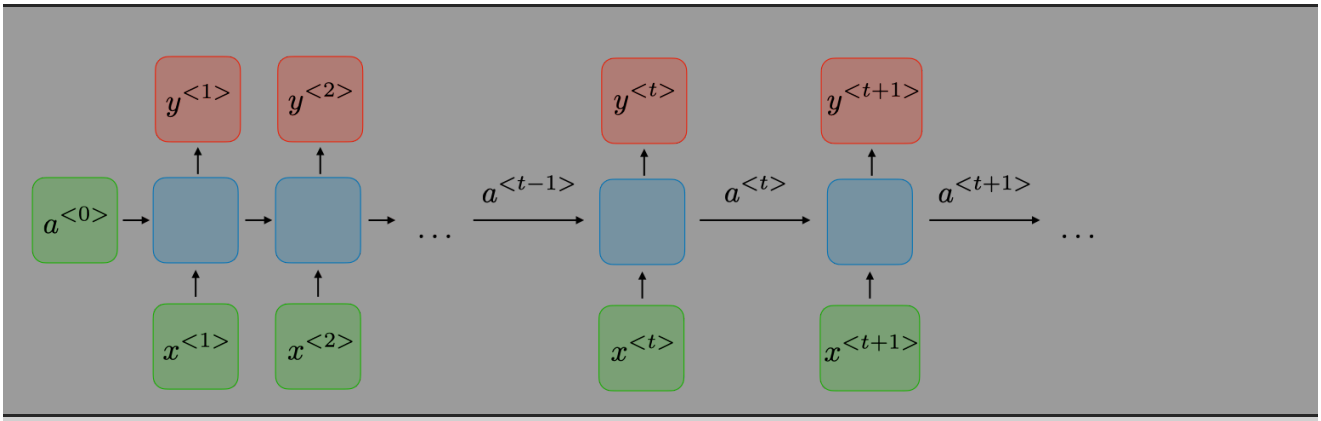


图2 （单向）RNN原理，其中x为token，y为特征

RNN 是处理序列数据的典型模型，通过循环结构逐步处理每个时间步的输入。改进版本包括 LSTM（长短期记忆网络）和 GRU（门控循环单元），它们通过门机制解决了普通 RNN

的梯度消失和梯度爆炸问题。RNN 能捕捉长距离依赖信息，适合序列建模任务（如 NER）。本实验中我们使用 `nn.RNN` 、 `nn.GRU` 和 `nn.LSTM` 实现多种RNN。

Transformer

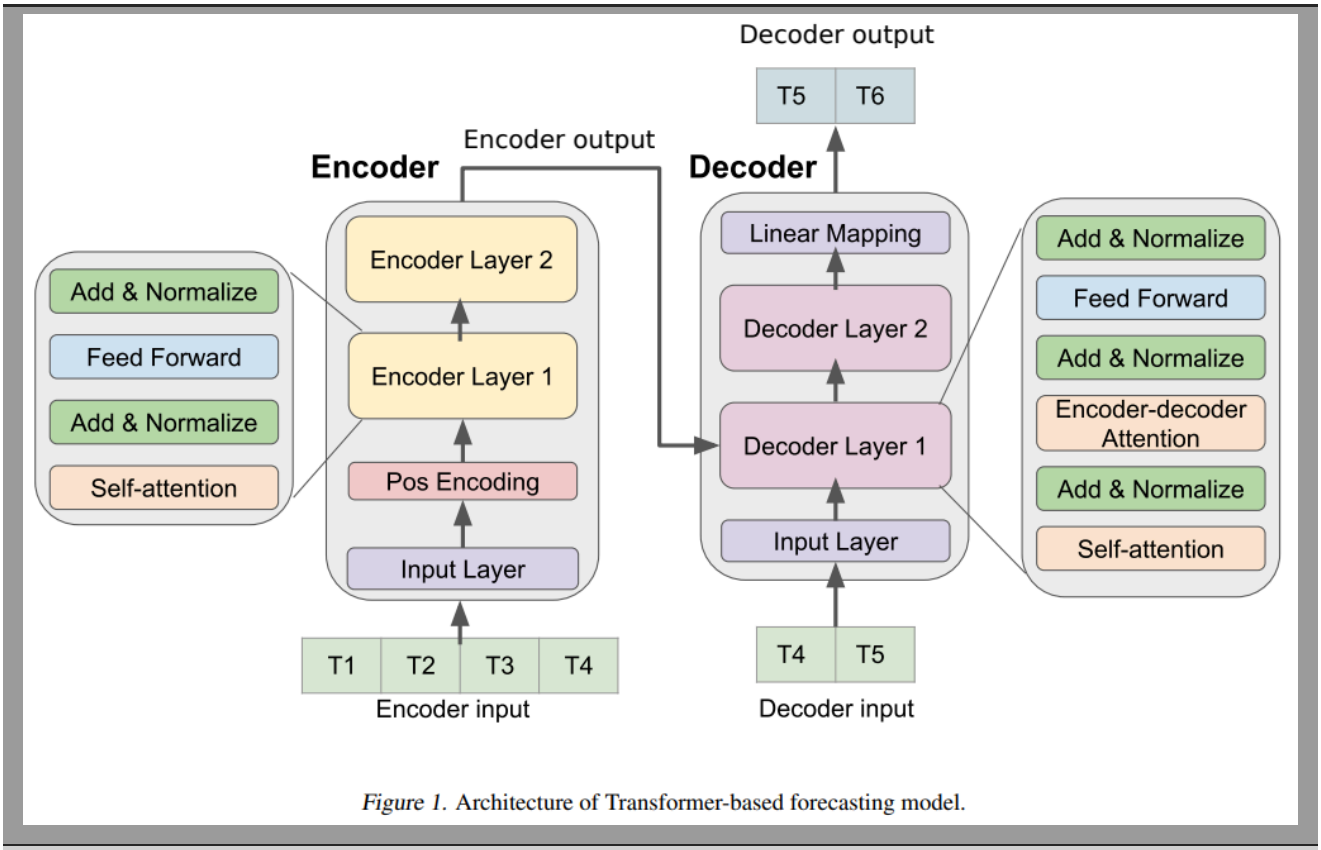


Figure 1. Architecture of Transformer-based forecasting model.

图3 Transformer的结构

Transformer 是一种基于自注意力机制的模型，彻底抛弃了 RNN 的循环结构。其核心组件是多头自注意力和前馈神经网络：

- 自注意力机制：能全局建模序列中任意两个位置的依赖关系。输入序列$\mathbf{X} \in \mathbb{R}^{n \times d}$：

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V$$

  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$分别被称为查询、键和值矩阵。
- 位置编码：Transformer 并不像 RNN 那样接受上文依次输入，也不像 CNN 那样通过卷积核天然建模空间位置信息，所以为了捕捉序列位置信息，需要加入位置编码。本实验中我们使用正弦-余弦位置编码：

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

  本实验中，对于情感分类和NER任务，我们使用上图的Encoder部分，也就是 `nn.TransformerEncoder` 来实现。对于情感分类，加入特殊token `[CLS]` ，取其注意力输出作为特征；对于NER，取每个token的注意力输出作为每个token的特征；对于机器翻译任务，我们使用 `nn.Transformer` 来实现。值得注意的是， `nn.Transformer` 和 `nn.TransformerEncoder` 并不包含图3中Encoder 的位置编码！

代码结构

```
.
├── .gitignore
├── checkpoints/  # 模型检查点
│   ├── ner/  # 命名实体识别模型检查点
│   ├── sentiment/  # 情感分析模型检查点
│   └── translation/  # 翻译模型检查点
├── configs/  # 配置文件
│   ├── __init__.py
│   ├── base_config.py  # 基础配置
│   ├── model_config.py  # 模型配置
│   ├── parser.py  # 配置解析器
│   ├── pipeline_config.py  # 流水线配置
│   └── task_config.py  # 任务配置
├── data/  # 数据文件夹
│   ├── ner/  # 命名实体识别数据
│   ├── sentiment/  # 情感分析数据
│   └── translation/  # 翻译数据
├── data_utils/  # 数据处理工具与数据集
│   ├── __init__.py
│   ├── ner.py  # 命名实体识别数据处理与数据集
│   ├── sentiment.py  # 情感分析数据处理与数据集
│   └── translation.py  # 翻译数据处理与数据集
├── environment.yml  # 环境配置文件
├── LICENSE  # 许可证文件
├── logs/  # 日志文件夹
│   ├── ner/  # 命名实体识别日志
│   ├── sentiment/  # 情感分析日志
│   └── translation/  # 翻译日志
├── main.py  # 主程序入口
├── metrics/  # 评估指标
│   ├── ner_metrics.py  # 命名实体识别评估指标
│   ├── sentiment_metrics.py  # 情感分析评估指标
│   └── translation_metrics.py  # 翻译评估指标
├── models/  # 模型文件夹
│   ├── __init__.py
│   ├── backbone.py  # 模型骨干选择
│   ├── crf.py  # 条件随机场模型
│   ├── encoder.py  # 编码器模型
│   ├── encoder_decoder.py  # 编码器-解码器模型
│   ├── ner_crf.py  # 命名实体识别CRF模型
│   ├── sentiment_model.py  # 情感分析模型
│   ├── text_cnn.py  # 文本CNN模型
│   ├── text_rnn.py  # 文本RNN模型
│   └── translater.py  # 翻译模型
├── README.md  # 项目说明文件
├── scripts/  # 脚本文件夹
├── temp.txt  # 临时文件
├── test.py  # 测试文件
├── tests/  # 测试文件夹
│   ├── ner_rnn.py  # 命名实体识别RNN测试
│   ├── ner_trans.py  # 命名实体识别Transformer测试
│   ├── sentiment_cnn.py  # 情感分析CNN测试
│   ├── sentiment_rnn.py  # 情感分析RNN测试
│   ├── sentiment_trans.py  # 情感分析Transformer测试
│   └── trans_trans.py  # 翻译Transformer测试
└── utils/  # 工具文件夹
    └── example.py  # 用于生成与保存测试结果
```

关于如何运行代码，见 `README.md` 。

共享框架

本次实验我们尝试使涉及的三个任务共享同一套框架，依托于 `transformers.Trainer` ，框架包含配置（ `configs` ）、模型（ `models` ）、数据处理（ `datasets` ）、指标（ `metrics` ）等模块。根据传入参数 `task` ，使用参数 `model` 对应的模型，根据参数 `mode` 执行训练、测试等模式。对于特定任务，只需根据统一接口编写子类。

RNN掩码实现

每个句子后填充的长度不一的掩码会影响RNN的效果，因为在经过长掩码序列后，RNN可能会遗忘先前的内容。我们使用 `torch.nn.utils.rnn.pack_padded_sequence` 来避免这种情况，同时保留并行计算的速度。但这个函数要求序列长度降序排列，具体实现如下：

```
def forward(self, emb: torch.Tensor,
                attention_mask: torch.LongTensor|None=None) -> torch.Tensor:
        # 计算每个序列的有效长度
        if attention_mask is not None:
            lengths = attention_mask.sum(dim=1)  # (batch_size, )
        else:
            lengths = torch.full((emb.size(0),), emb.size(1),
                                 dtype=torch.long, device=emb.device)
        lengths = lengths.clamp(min=1)  # 确保长度至少为1，避免索引错误
        # 将输入按序列长度降序排序
        lengths_sorted, sorted_idx = lengths.sort(descending=True)
        emb_sorted = emb[sorted_idx]
        # 打包序列，忽略填充部分
        packed_input = pack_padded_sequence(emb_sorted,
                                lengths_sorted.cpu(),
                                        batch_first=True, enforce_sorted=True)
        # 通过RNN处理
        packed_output, _ = self.rnn(packed_input)
        # 解包输出
        output, _ = pad_packed_sequence(packed_output, batch_first=True)
        # 恢复原始排序
        _, original_idx = sorted_idx.sort()
        output = output[original_idx]
        ...
```

实验过程

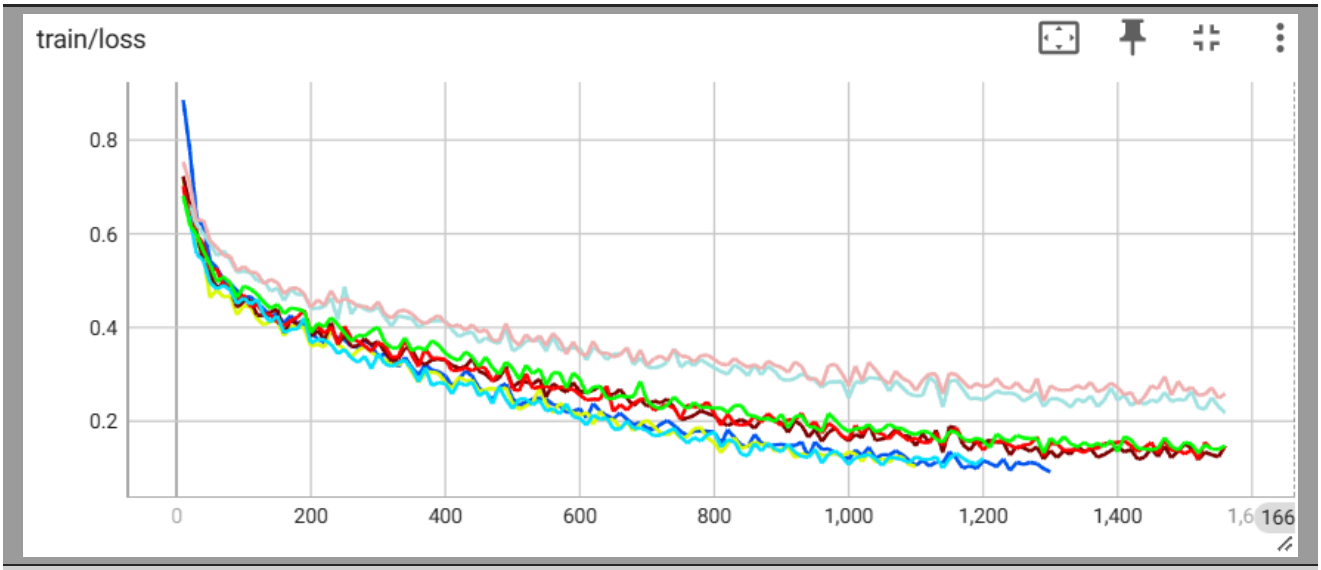
情感分类

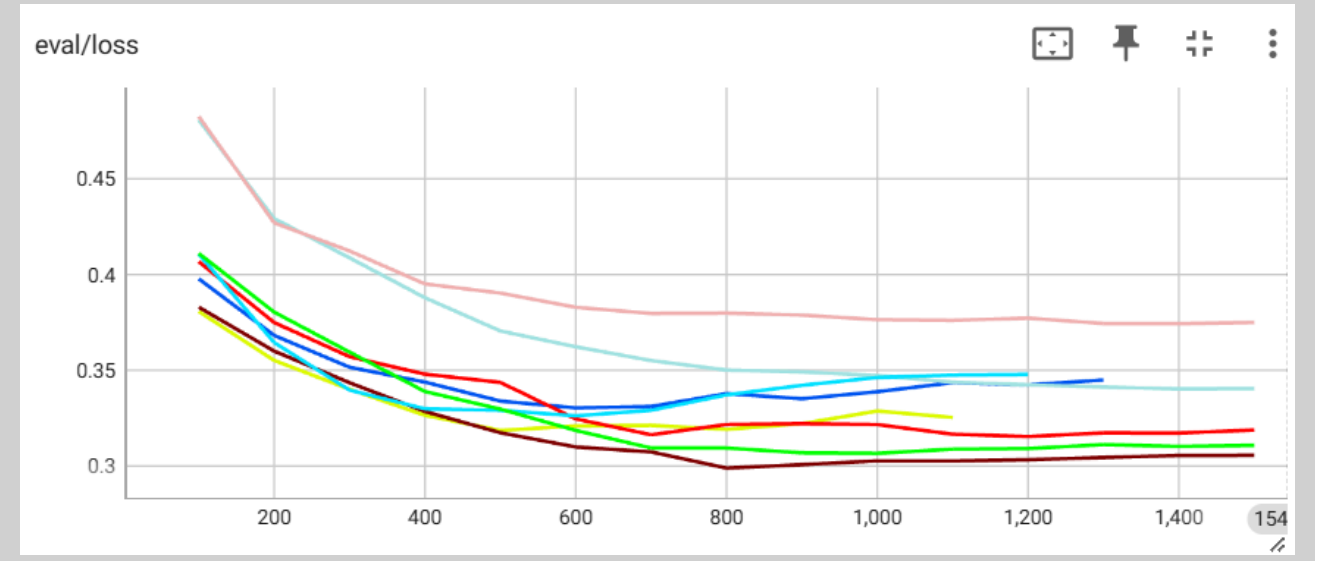
CNN

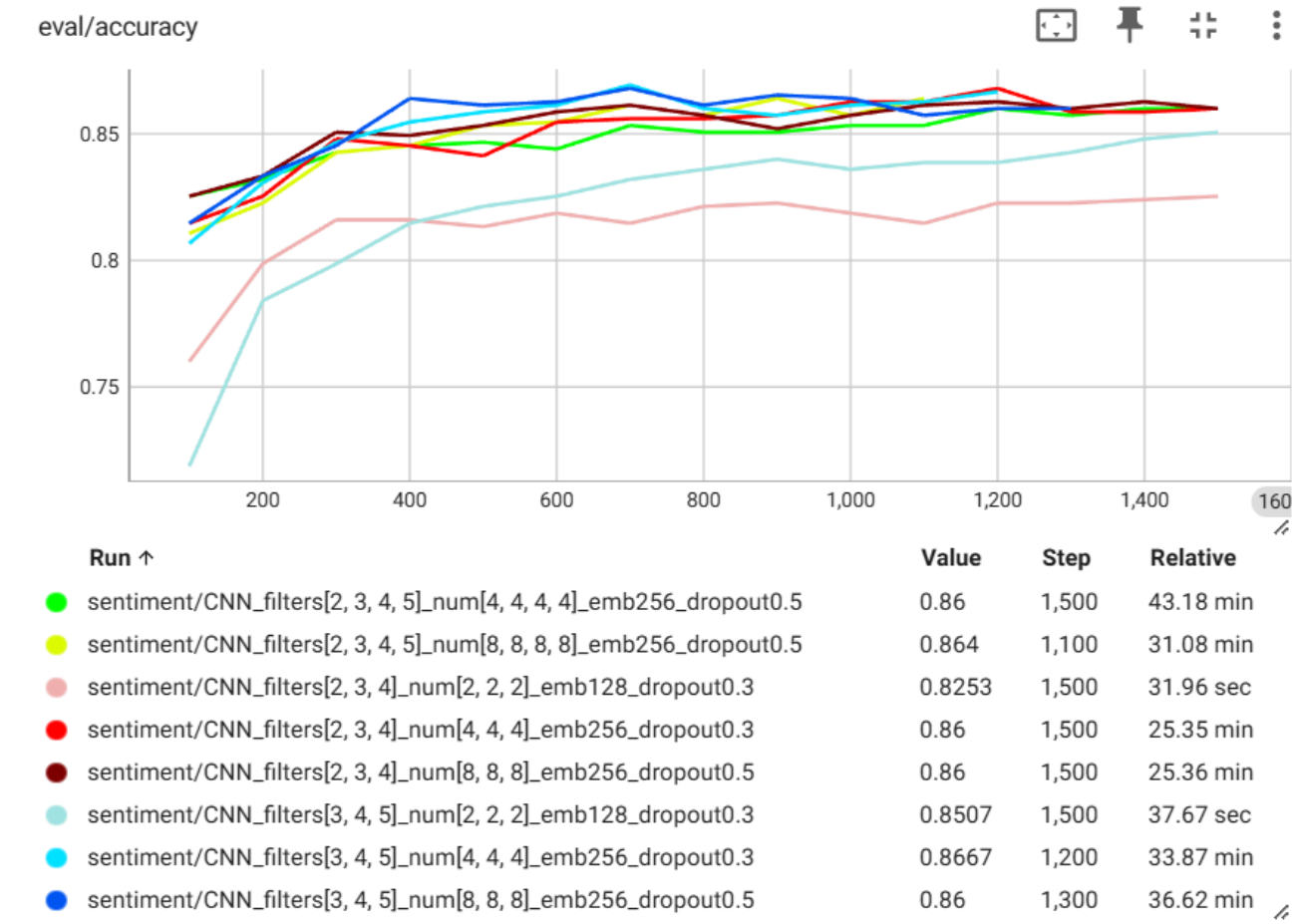
我们首先尝试了如下配置：

```
# 轻量配置
CNNConfig(emb_dim=128, dropout=0.3, filter_sizes=[2, 3, 4], num_filters=[2, 2,
2]),
CNNConfig(emb_dim=128, dropout=0.3, filter_sizes=[3, 4, 5], num_filters=[2, 2,
2]),
# 标准配置
CNNConfig(emb_dim=256, dropout=0.3, filter_sizes=[2, 3, 4], num_filters=[4, 4,
4]),
CNNConfig(emb_dim=256, dropout=0.3, filter_sizes=[3, 4, 5], num_filters=[4, 4,
4]),
# 增强配置，增加过滤器数量和更高的 dropout
CNNConfig(emb_dim=256, dropout=0.5, filter_sizes=[2, 3, 4], num_filters=[8, 8,
8]),
CNNConfig(emb_dim=256, dropout=0.5, filter_sizes=[3, 4, 5], num_filters=[8, 8,
8]),
# 更强特征提取配置，适合高资源场景
CNNConfig(emb_dim=256, dropout=0.5, filter_sizes=[2,3,4,5], num_filters=
[4,4,4,4]),
CNNConfig(emb_dim=256, dropout=0.5, filter_sizes=[2,3,4,5], num_filters=
[8,8,8,8]),
```

得到如下损失与准确率曲线：


train/loss
0.8
0.6
0.4
0.2
0
200
400
600
800
1,000
1,200
1,400
166


eval/loss
0.45
0.4
0.35
0.3
200
400
600
800
1,000
1,200
1,400
154



Run ↑	Value	Step	Relative
sentiment/CNN_filters[2, 3, 4, 5]_num[4, 4, 4, 4]_emb256_dropout0.5	0.86	1,500	43.18 min
sentiment/CNN_filters[2, 3, 4, 5]_num[8, 8, 8, 8]_emb256_dropout0.5	0.864	1,100	31.08 min
sentiment/CNN_filters[2, 3, 4]_num[2, 2, 2]_emb128_dropout0.3	0.8253	1,500	31.96 sec
sentiment/CNN_filters[2, 3, 4]_num[4, 4, 4]_emb256_dropout0.3	0.86	1,500	25.35 min
sentiment/CNN_filters[2, 3, 4]_num[8, 8, 8]_emb256_dropout0.5	0.86	1,500	25.36 min
sentiment/CNN_filters[3, 4, 5]_num[2, 2, 2]_emb128_dropout0.3	0.8507	1,500	37.67 sec
sentiment/CNN_filters[3, 4, 5]_num[4, 4, 4]_emb256_dropout0.3	0.8667	1,200	33.87 min
sentiment/CNN_filters[3, 4, 5]_num[8, 8, 8]_emb256_dropout0.5	0.86	1,300	36.62 min

图4 各种配置CNN的情感分类损失与准确率曲线

结果显示，在卷积核大小为 (2, 3, 4) 且每个卷积核数量为 4 时，模型在验证集上达到了最佳表现，进一步增加卷积核数量或使用更大的窗口组合（如 (3, 4, 5)）并未显著提升准确率，反而在 600 步左右开始出现过拟合现象，即便使用了 Dropout（0.5）和早停策略也未能缓解。分析认为，这是因为增加卷积核数量后，模型的特征提取能力已经饱和，进一步增加容量引入了冗余特征，同时更大的窗口组合捕获了无关或冗余的上下文信息，导致泛化性能下降。这表明，在本任务和数据集规模下，较小的卷积核数量（每个 4 个）和窗口大小组合 (2, 3, 4) 更能平衡模型的特征提取能力与过拟合风险。进一步，我们分别对四种尺寸的卷积各12个的配置进行了测试，损失曲线如图：

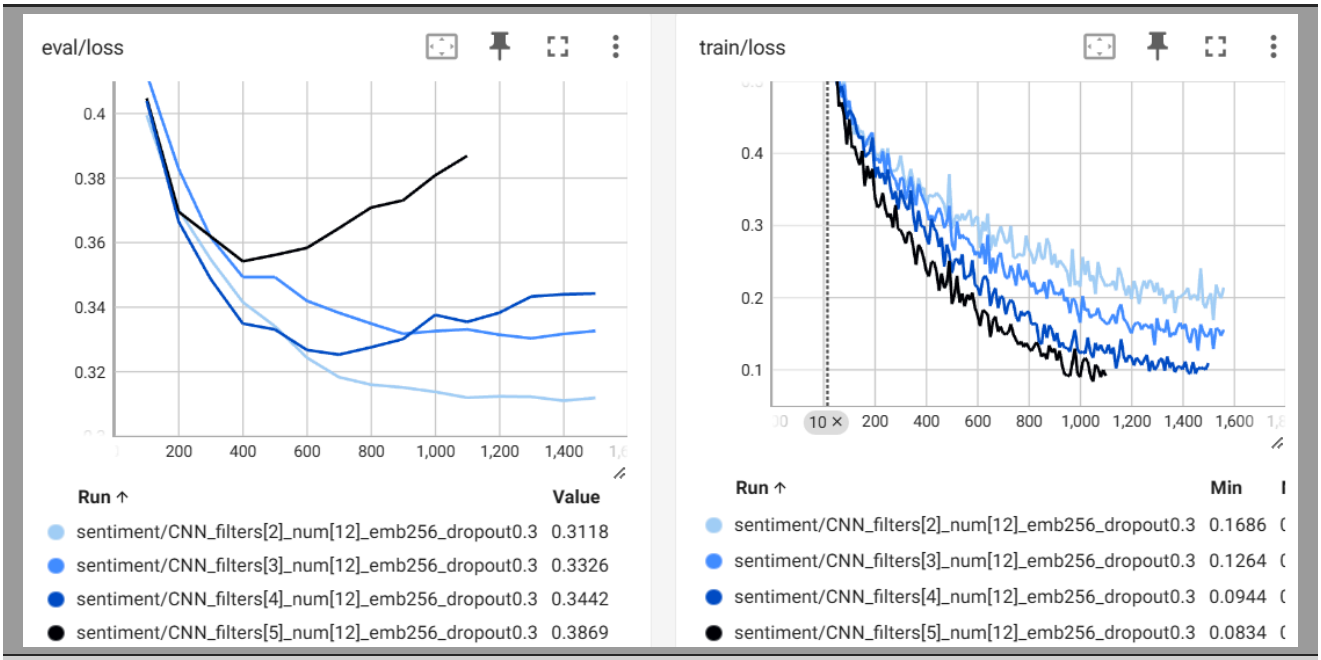


图5 单一尺寸卷积核情感分类的损失曲线

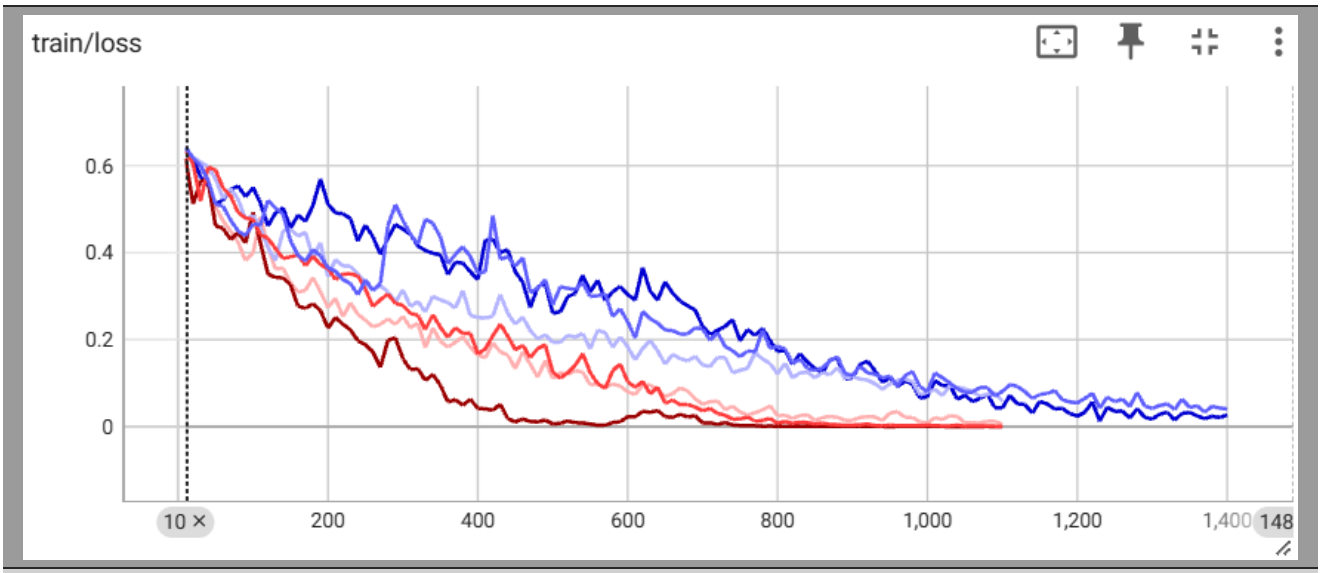
结果比较出人意料，即使只使用尺寸为 2 的卷积核，模型依然能够达到非常优异的性能，甚至表现最佳，而尺寸较大的卷积核（如 4 和 5）则出现了显著的过拟合现象。可以看出，尺寸为 2 的卷积核提取的短依赖特征对情感分类任务至关重要，能够有效捕捉模式。相比之下，较大尺寸的卷积核由于覆盖了更多上下文，虽然引入了一些有用的特征，但同时增加了参数量，捕获了大量冗余或无关的信息，从而降低了泛化性能。这一结果表明，对于以短文本为主的情感分类任务，短窗口卷积核足以提取关键特征，而更大的卷积核容易导致模型过复杂化和过拟合。这说明情感分类任务实际上并不要求太多的上下文信息。

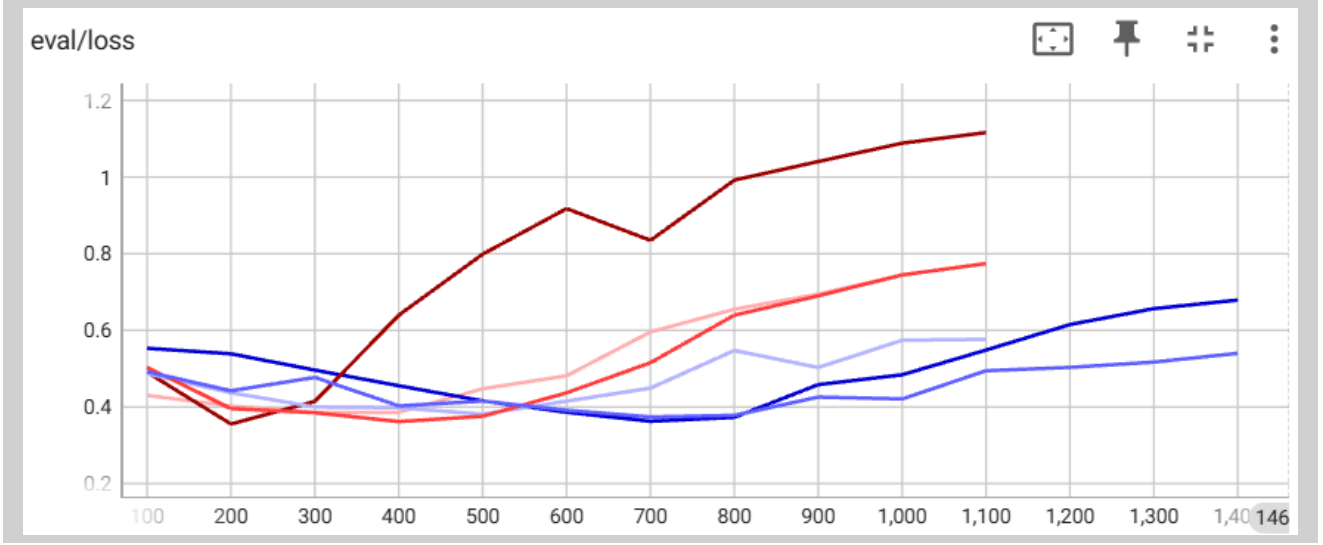
RNN

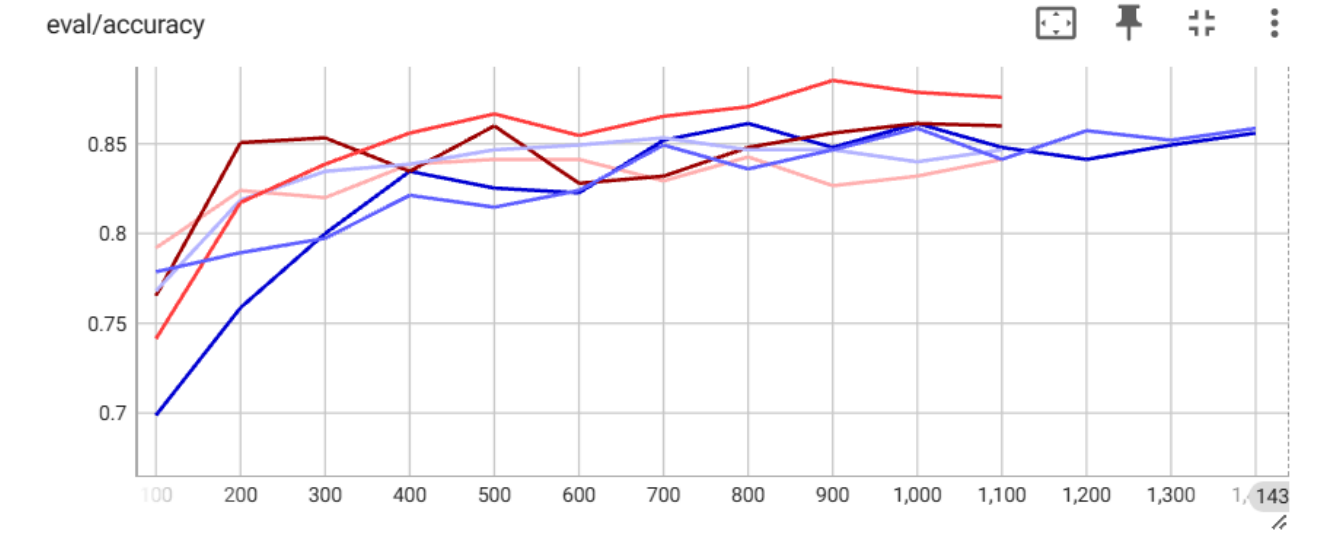
我们对LSTM和GRU分别尝试了如下配置，每种配置都分别尝试了单向和双向两种RNN：

```
# 轻量配置，适合快速实验
RNNConfig(emb_dim=64, hidden_size=128, num_layers=1, dropout=0.2),
# 增加嵌入和隐藏层
RNNConfig(emb_dim=128, hidden_size=256, num_layers=1, dropout=0.2),
# 增加层数
RNNConfig(emb_dim=128, hidden_size=256, num_layers=2, dropout=0.3),
# 增加嵌入，较大隐藏层
RNNConfig(emb_dim=256, hidden_size=512, num_layers=2, dropout=0.3),
```

各种配置LSTM的情感分类训练结果如图：





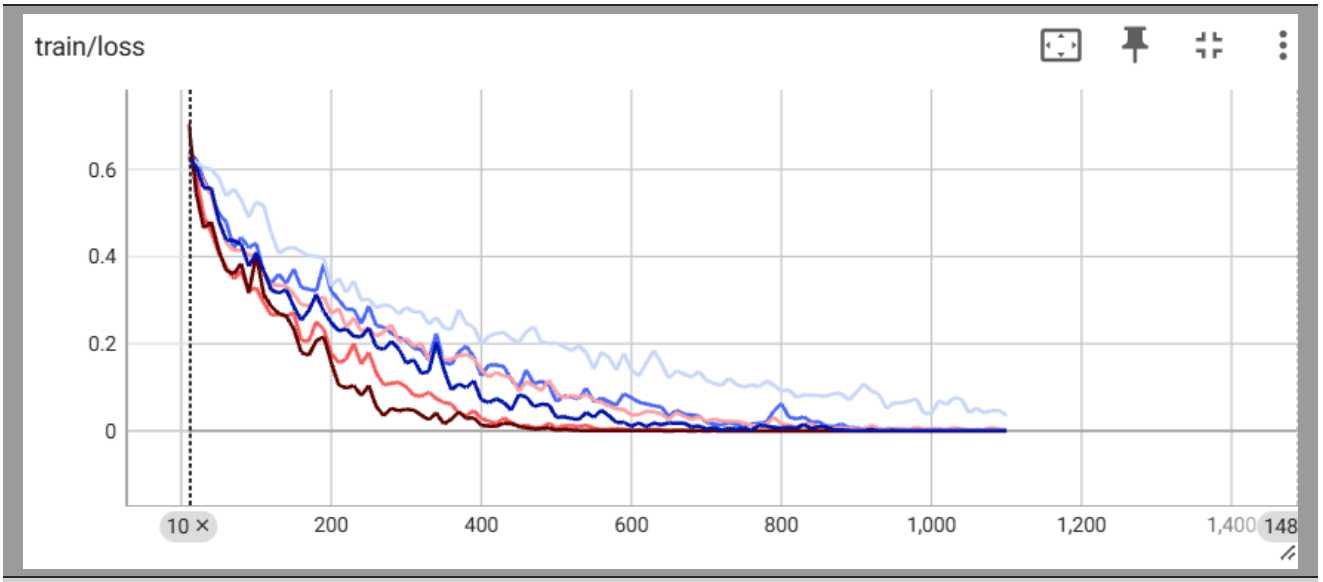


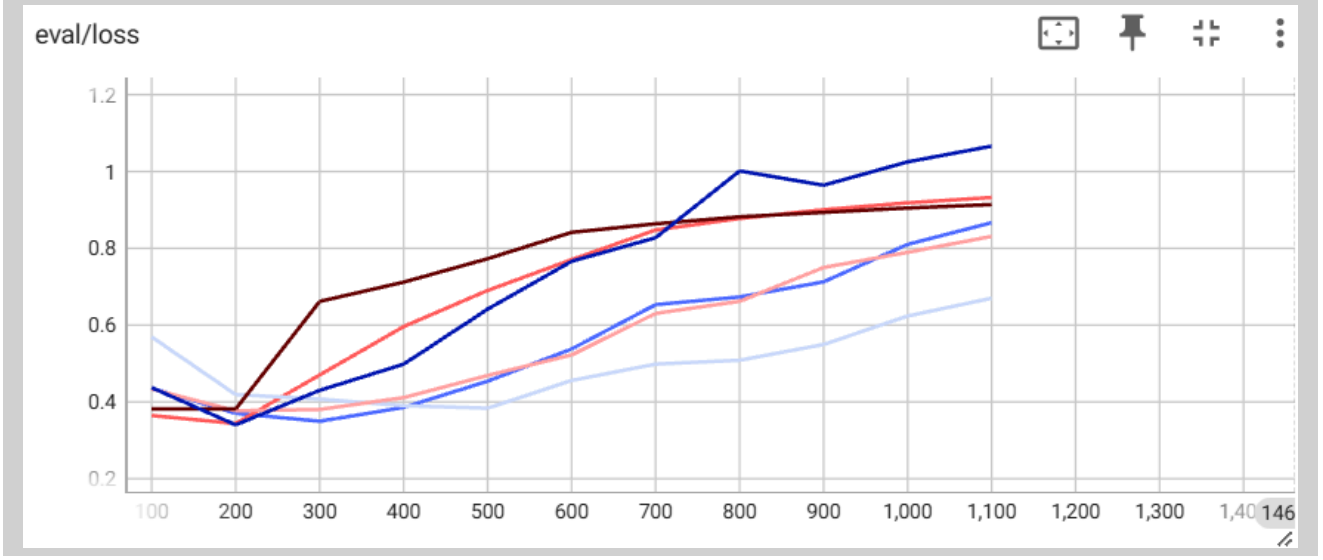
Run ↑	Value	Step	Relative
sentiment/LSTM_layers1_hidden128_emb64_dropout0.2	0.8467	1,100	39.31 sec
sentiment/LSTM_layers1_hidden128_emb64_dropout0.2_bi	0.8413	1,100	55.85 sec
sentiment/LSTM_layers1_hidden256_emb128_dropout0.2	0.8587	1,400	53.87 sec
sentiment/LSTM_layers1_hidden256_emb128_dropout0.2_bi	0.876	1,100	1.089 min
sentiment/LSTM_layers2_hidden256_emb128_dropout0.3	0.856	1,400	1.212 min
sentiment/LSTM_layers2_hidden512_emb256_dropout0.3_bi	0.86	1,100	2.029 min

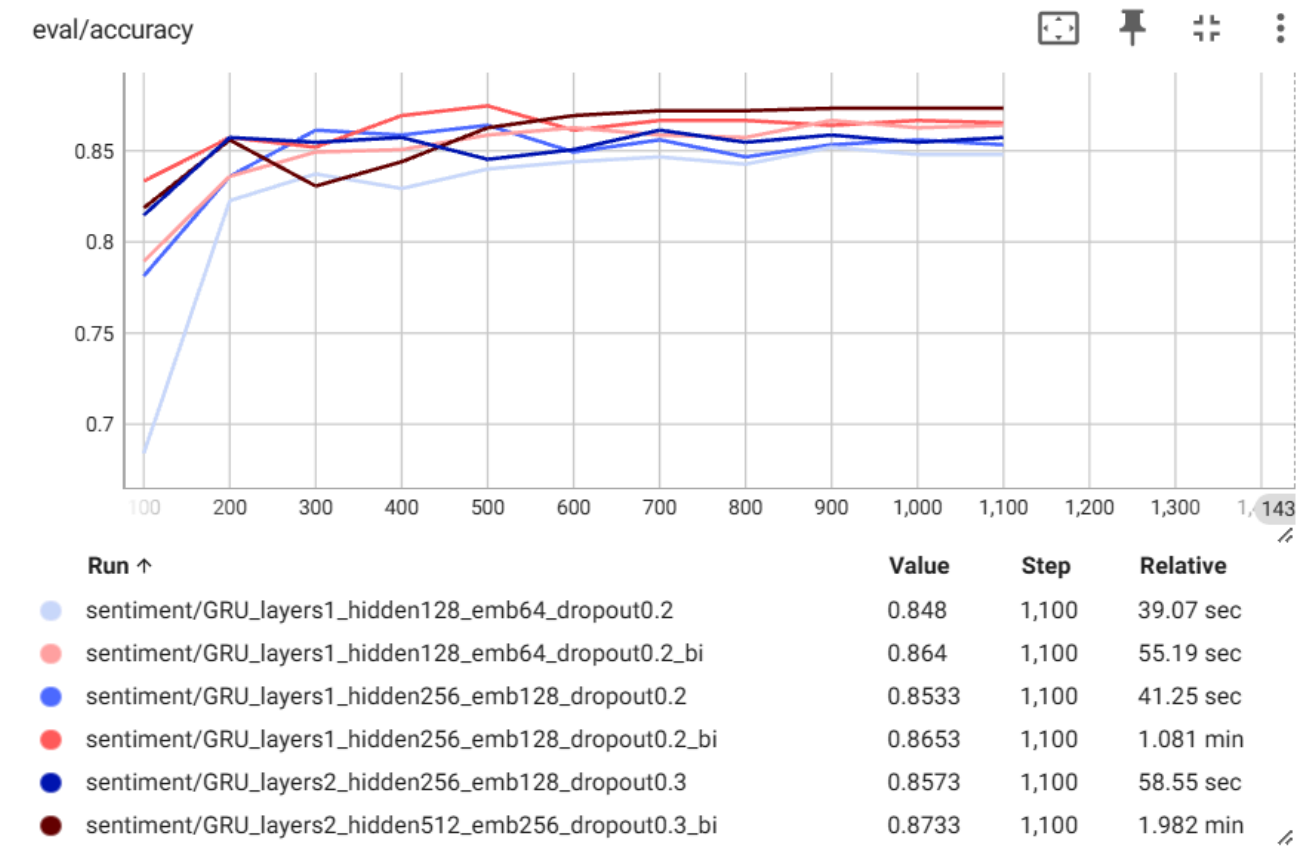
图6 各种配置LSTM的情感分类损失与准确率曲线

结果表明，双向LSTM训练损失曲线在单向LSTM下方，说明其收敛更快；但是其验证损失曲线却更快地上升，说明其更快出现了过拟合现象，参数越大，过拟合越严重，并且相比于单向LSTM过拟合现象的严重程度是显著的；对于验证准确率，平均来说，双向LSTM的准确率更高，但提升相比于代价并不显著（2%以内）。

各种配置GRU的情感分类训练结果如图：







Run ↑	Value	Step	Relative
sentiment/GRU_layers1_hidden128_emb64_dropout0.2	0.848	1,100	39.07 sec
sentiment/GRU_layers1_hidden128_emb64_dropout0.2_bi	0.864	1,100	55.19 sec
sentiment/GRU_layers1_hidden256_emb128_dropout0.2	0.8533	1,100	41.25 sec
sentiment/GRU_layers1_hidden256_emb128_dropout0.2_bi	0.8653	1,100	1.081 min
sentiment/GRU_layers2_hidden256_emb128_dropout0.3	0.8573	1,100	58.55 sec
sentiment/GRU_layers2_hidden512_emb256_dropout0.3_bi	0.8733	1,100	1.982 min

图7 各种配置GRU的情感分类损失与准确率曲线

结果表明，与LSTM的情况相似，双向GRU收敛更快、更快过拟合，并且准确率极小幅提升。

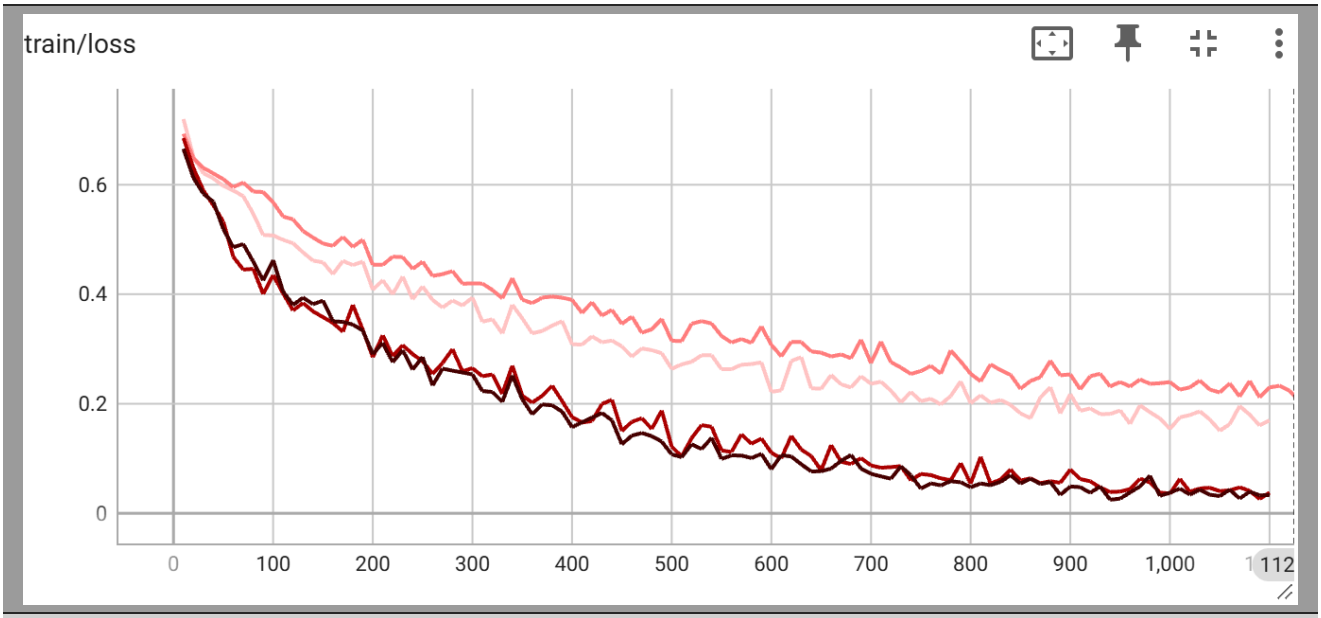
此外，从 LSTM 和 GRU 的对比来看，GRU 在所有配置下的收敛速度均快于 LSTM，但也更早出现过拟合。在 400 步左右，中高配置的双向 GRU 基本已收敛，而对应的 LSTM 仍处于收敛过程；在 800 步左右，中高配置的单向 GRU 已基本收敛，而对应的 LSTM 尚未完全收敛。这表明 GRU 由于结构更简单，在训练过程中表现出更高的效率，同时也说明情感分类任务相对简单，LSTM 在捕获复杂特征上的优势并未得到体现。综合来看，单向GRU 在训练效率和性能上更适合本任务。同时，我们发现，虽然很快验证损失就剧烈上升，但模型的验证准确率并不会剧烈下降，而是维持稳定甚至增长。

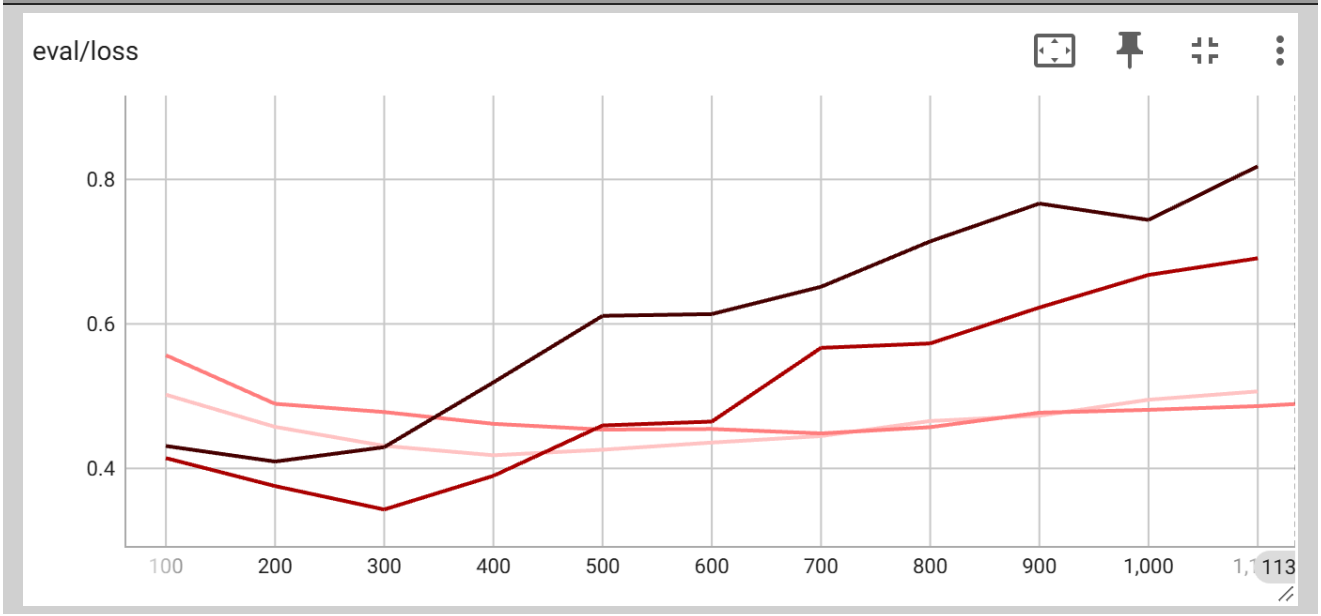
Transformer

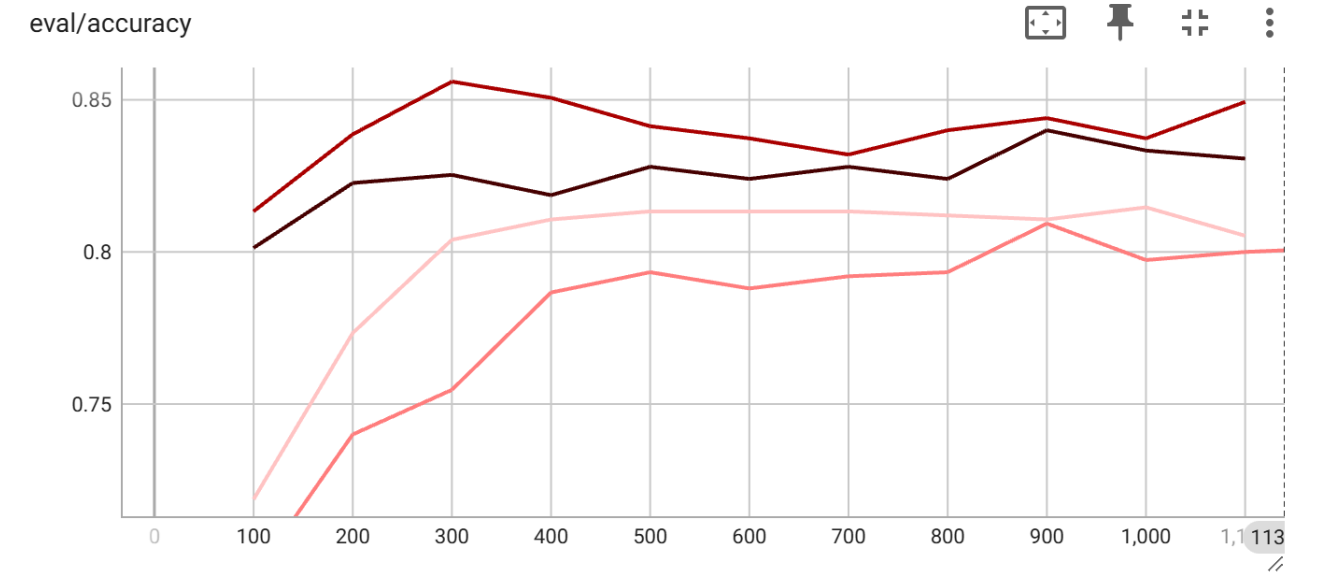
我们对Encoder-Only Transformer分别尝试了如下配置：

```
TransformerConfig(emb_dim=32, ffn_size=64, num_heads=2, num_layers=1,
dropout=0.1),
TransformerConfig(emb_dim=32, ffn_size=128, num_heads=2, num_layers=1,
dropout=0.2),
TransformerConfig(emb_dim=64, ffn_size=256, num_heads=2, num_layers=2,
dropout=0.2),
TransformerConfig(emb_dim=64, ffn_size=256, num_heads=4, num_layers=2,
dropout=0.2),
```

得到损失曲线与准确率曲线如图：







Run ↑	Value	Step	Relative
sentiment/Transformer_layers1_ffnsize128_heads2_emb32_dropout0.2	0.8	1,100	11.19 sec
sentiment/Transformer_layers1_ffnsize64_heads2_emb32_dropout0.1	0.8053	1,100	11.06 sec
sentiment/Transformer_layers2_ffnsize256_heads2_emb64_dropout0.2	0.8493	1,100	20.77 sec
sentiment/Transformer_layers2_ffnsize256_heads4_emb64_dropout0.2	0.8307	1,100	26.82 sec

图8 各种配置Encoder-Only Transformer的情感分类损失与准确率曲线

实验结果表明，随着模型容量的增加（如线性层大小增大、网络层数增加、嵌入维度上升以及多头注意力头数增加），模型的学习能力显著增强，训练损失快速下降，但从配置3到配置4的提升已经不明显。然而，这种增强的学习能力也带来了更高的过拟合风险，验证损失在训练后期明显上升，尤其是在中高配置下（如 `emb_dim=64, ffn_size=256, num_heads=4, num_layers=2` ）；同时，dropout从0.1到0.2，即使隐藏层大小增大一倍，学习能力也显著降低；虽然中高配置在训练后期有较为严重的过拟合，但验证准确率并没有明显下降。

在本实验中，我们对情感分类任务分别使用了 CNN、RNN（包括 LSTM 和 GRU）以及 Encoder-Only Transformer 模型进行测试，对其配置与性能进行了详细分析。实验结果表明，各模型在特定配置下均能够取得较好的分类效果（85%左右），但它们在特征提取能力、训练效率和过拟合表现上存在显著差异。由于情感分类任务较为简单，对上下文特征提取能力的要求低，所以 RNN 与 Transformer 在该任务上特征捕捉能力过于强了，导致过拟合问题很严重，基本上最低配置也会出现过拟合；反而小窗口的 CNN 能够兼顾计算效率与特征捕捉能力。

我们猜测，这可能是因为嵌入层也是待训练的参数，将代表不同情感色彩的词向量，移动到嵌入空间的不同位置，可以补充单层小尺寸卷积核对语义的特征提取作用，导致 CNN 的性能也十分出色。

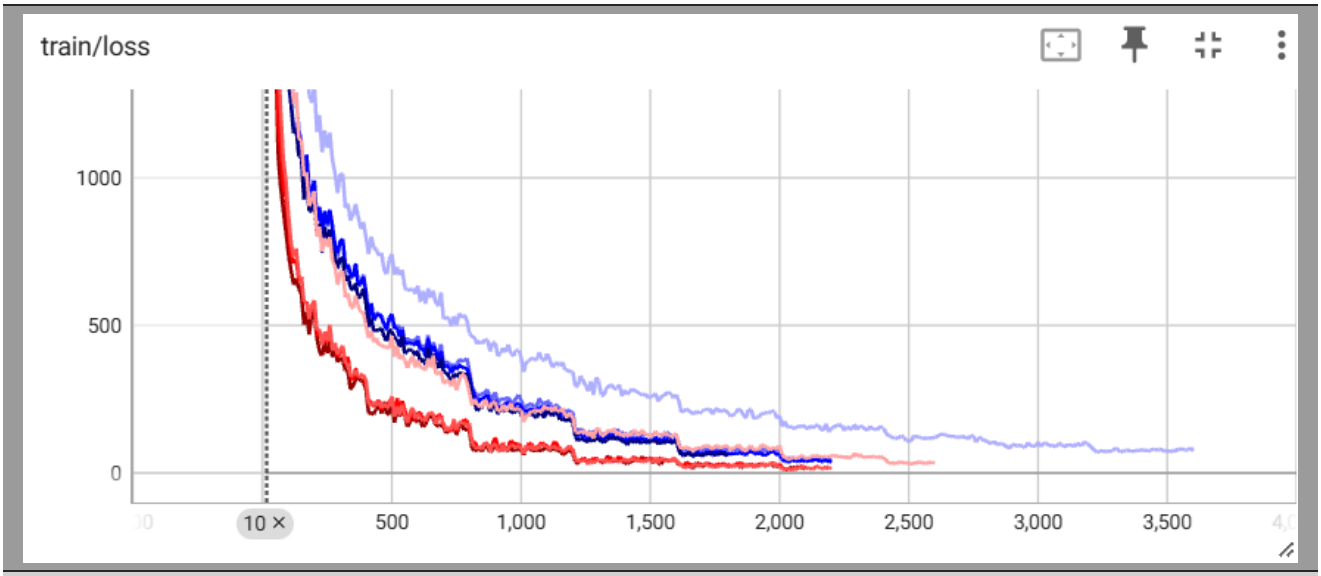
命名实体识别

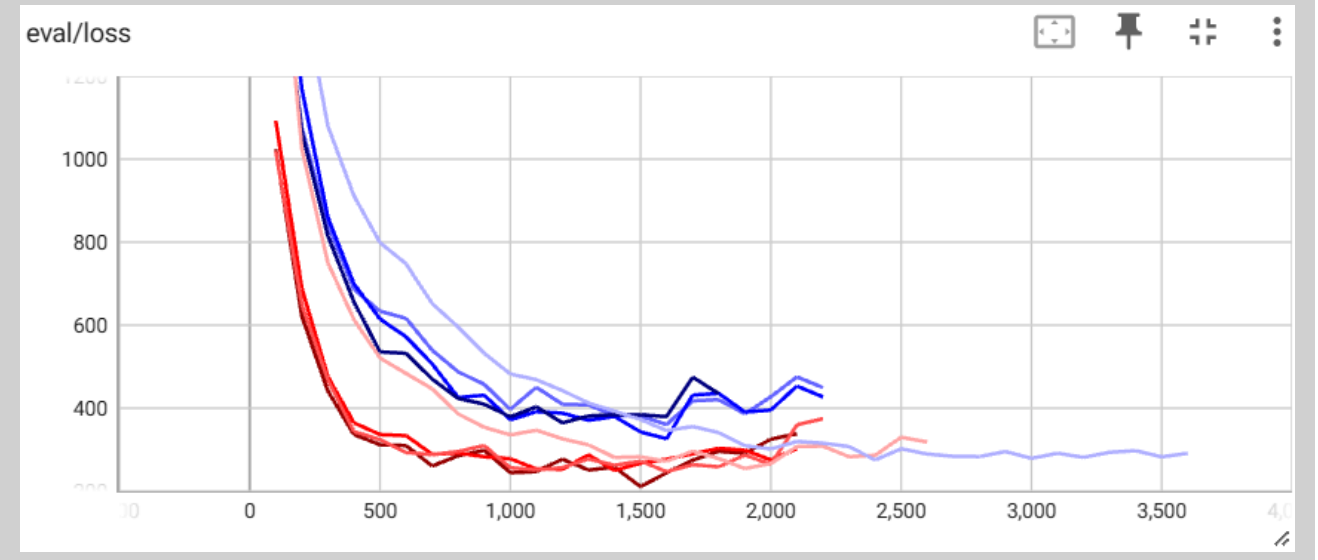
RNN

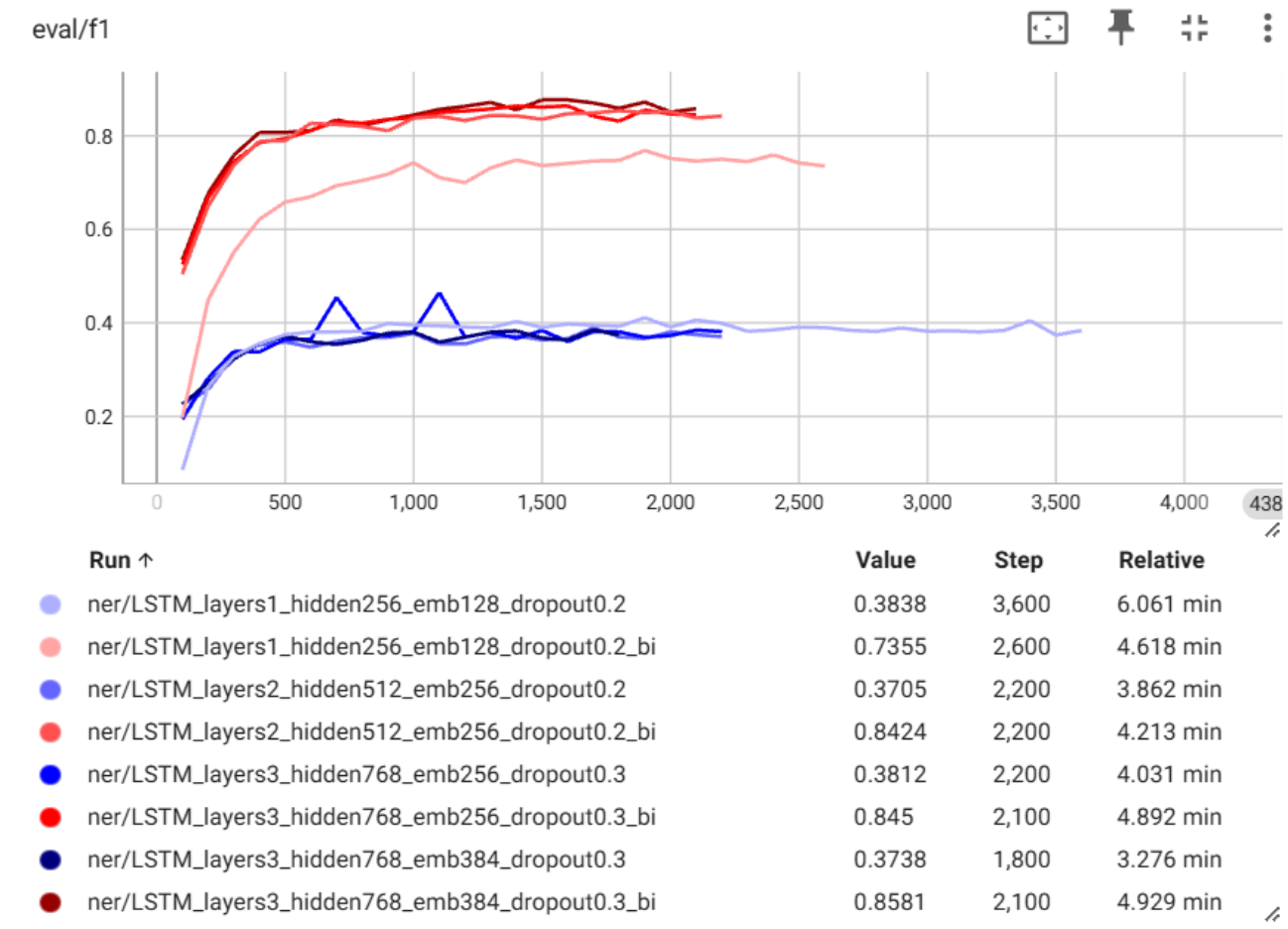
我们对LSTM和GRU分别尝试了如下配置，每种配置都分别尝试了单向和双向两种RNN：

```
# 轻量配置
RNNConfig(emb_dim=128, hidden_size=256, num_layers=1, dropout=0.2),
# 标准配置
RNNConfig(emb_dim=256, hidden_size=512, num_layers=2, dropout=0.2),
# 增强配置
RNNConfig(emb_dim=256, hidden_size=768, num_layers=3, dropout=0.3),
# 强特征捕捉
RNNConfig(emb_dim=384, hidden_size=768, num_layers=3, dropout=0.3)
```

各种配置LSTM的NER训练结果如图：


train/loss
1000
500
0
10 ×
500
1,000
1,500
2,000
2,500
3,000
3,500


eval/loss
1000
800
600
400
0
500
1,000
1,500
2,000
2,500
3,000
3,500

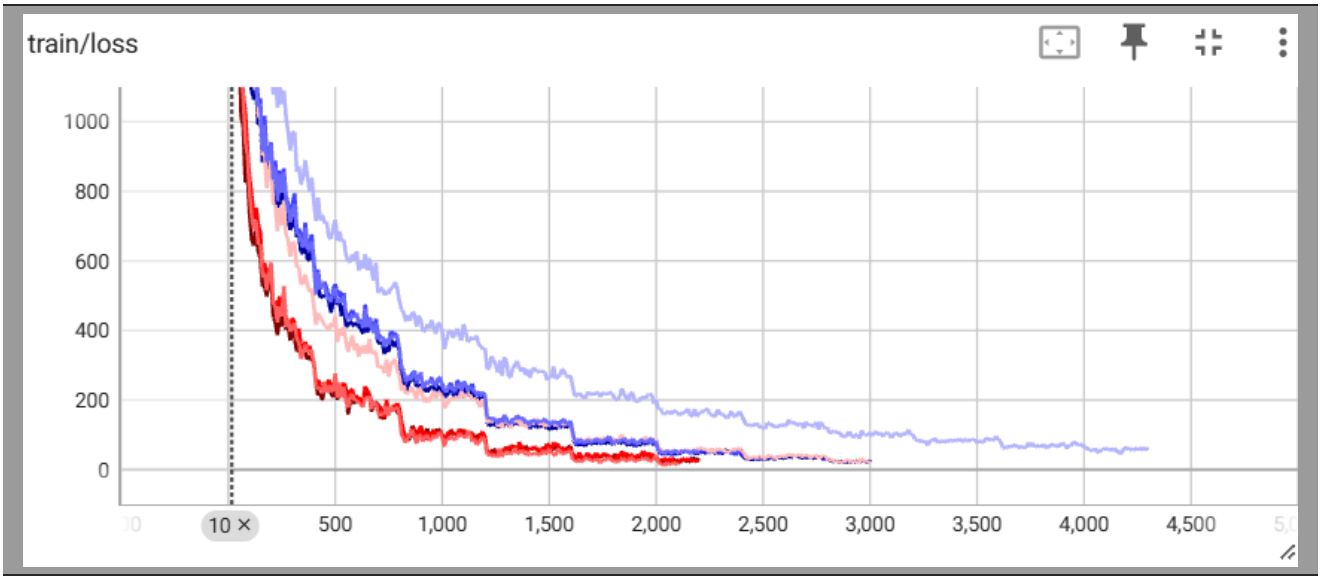


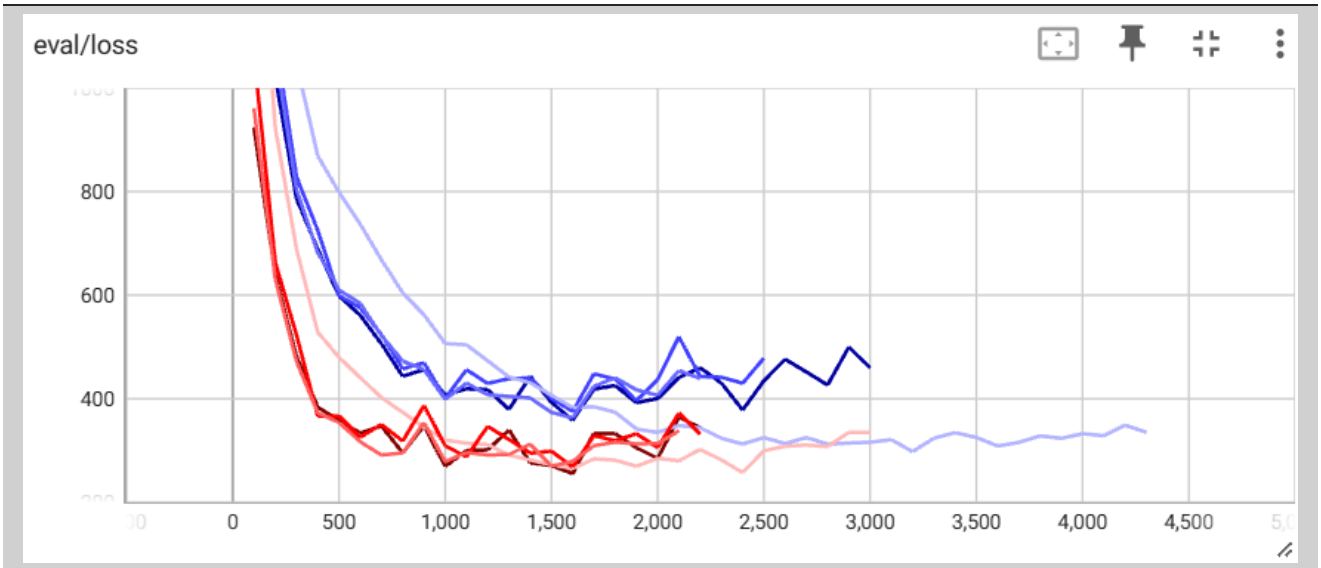
Run ↑	Value	Step	Relative
ner/LSTM_layers1_hidden256_emb128_dropout0.2	0.3838	3,600	6.061 min
ner/LSTM_layers1_hidden256_emb128_dropout0.2_bi	0.7355	2,600	4.618 min
ner/LSTM_layers2_hidden512_emb256_dropout0.2	0.3705	2,200	3.862 min
ner/LSTM_layers2_hidden512_emb256_dropout0.2_bi	0.8424	2,200	4.213 min
ner/LSTM_layers3_hidden768_emb256_dropout0.3	0.3812	2,200	4.031 min
ner/LSTM_layers3_hidden768_emb256_dropout0.3_bi	0.845	2,100	4.892 min
ner/LSTM_layers3_hidden768_emb384_dropout0.3	0.3738	1,800	3.276 min
ner/LSTM_layers3_hidden768_emb384_dropout0.3_bi	0.8581	2,100	4.929 min

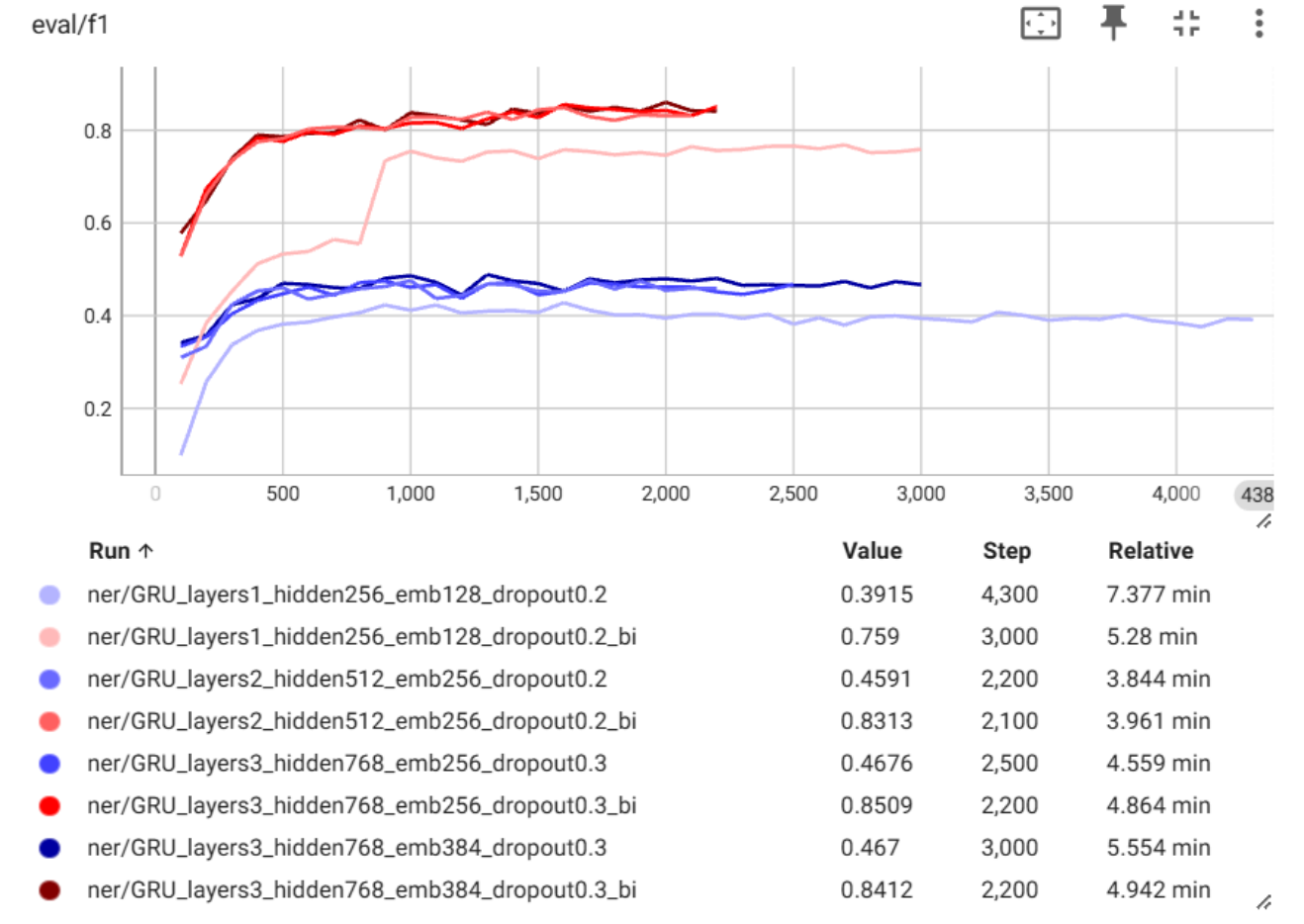
图9 各种配置LSTM的NER损失与f1曲线

可以看到，双向LSTM训练损失曲线几乎都在单向LSTM下方，说明其收敛更快；都收敛后，双向LSTM的训练和验证损失都比单向LSTM更低，说明双向LSTM更好地学习到了NER所需的上下文信息；对于验证f1，双向LSTM的f1显著高于单向LSTM，可以达到两倍的差距，这说明双向提取的上下文信息对于NER任务是十分必要的。

各种配置GRU的NER训练结果如图：







Run ↑	Value	Step	Relative
ner/GRU_layers1_hidden256_emb128_dropout0.2	0.3915	4,300	7.377 min
ner/GRU_layers1_hidden256_emb128_dropout0.2_bi	0.759	3,000	5.28 min
ner/GRU_layers2_hidden512_emb256_dropout0.2	0.4591	2,200	3.844 min
ner/GRU_layers2_hidden512_emb256_dropout0.2_bi	0.8313	2,100	3.961 min
ner/GRU_layers3_hidden768_emb256_dropout0.3	0.4676	2,500	4.559 min
ner/GRU_layers3_hidden768_emb256_dropout0.3_bi	0.8509	2,200	4.864 min
ner/GRU_layers3_hidden768_emb384_dropout0.3	0.467	3,000	5.554 min
ner/GRU_layers3_hidden768_emb384_dropout0.3_bi	0.8412	2,200	4.942 min

图10 各种配置GRU的NER损失与f1曲线

结果与LSTM的情况相似，双向GRU更好地学习了上下文特征，显著更好地完成了NER任务。训练速度上，GRU与LSTM几乎一致，差别没有在情感分类任务中那么大。

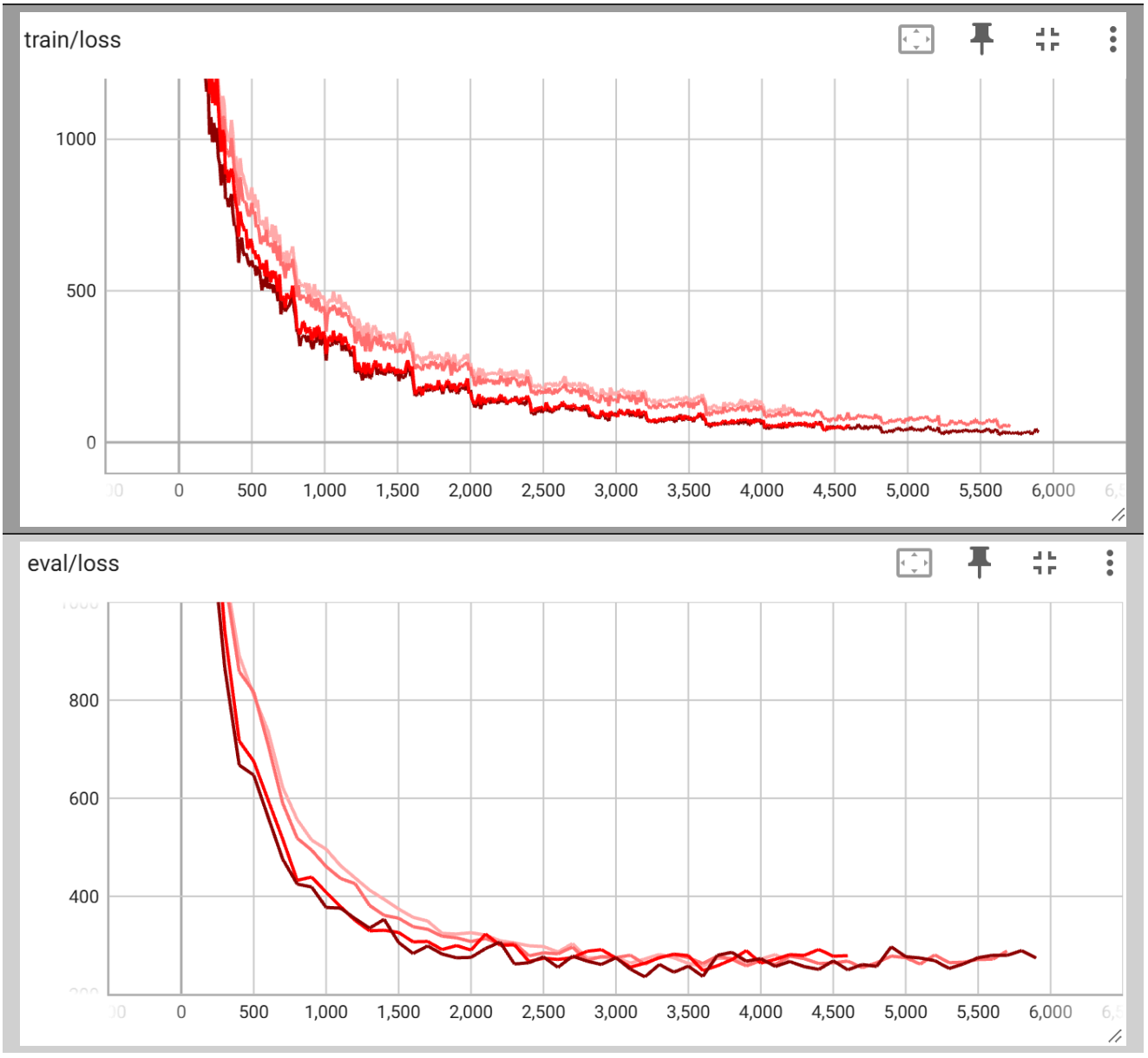
Transformer

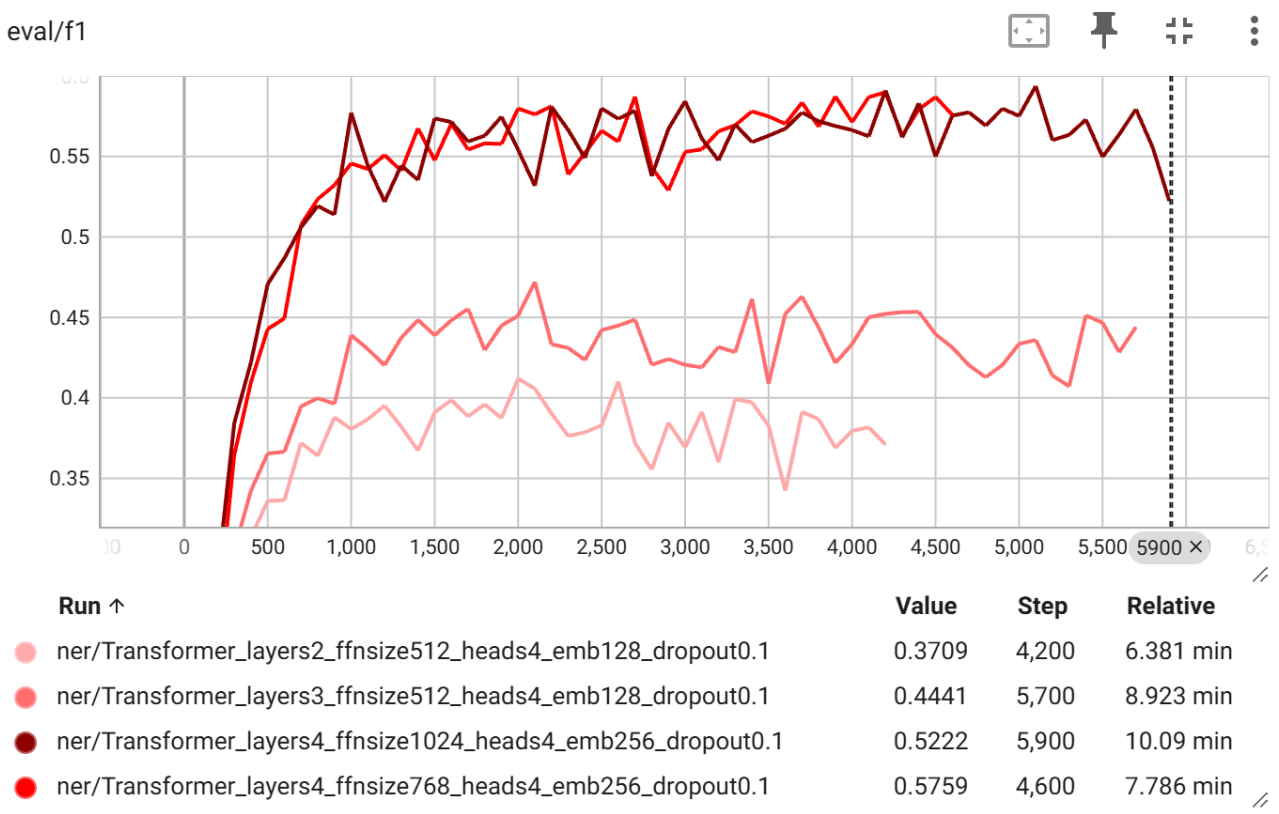
我们对Encoder-Only Transformer分别尝试了如下配置：

```
TransformerConfig(emb_dim=128, ffn_size=512, num_heads=4, num_layers=2,
dropout=0.1),
TransformerConfig(emb_dim=128, ffn_size=512, num_heads=4, num_layers=3,
```

```
dropout=0.1),
TransformerConfig(emb_dim=256, ffn_size=768, num_heads=4, num_layers=4,
dropout=0.1),
TransformerConfig(emb_dim=256, ffn_size=1024, num_heads=4, num_layers=4,
dropout=0.1),
```

得到损失与f1曲线如图：





Run ↑	Value	Step	Relative
ner/Transformer_layers2_ffnsize512_heads4_emb128_dropout0.1	0.3709	4,200	6.381 min
ner/Transformer_layers3_ffnsize512_heads4_emb128_dropout0.1	0.4441	5,700	8.923 min
ner/Transformer_layers4_ffnsize1024_heads4_emb256_dropout0.1	0.5222	5,900	10.09 min
ner/Transformer_layers4_ffnsize768_heads4_emb256_dropout0.1	0.5759	4,600	7.786 min

图11 各种配置Encoder-Only Transformer的情感分类损失与准确率曲线

Encoder-Only Transformer 的实验结果显示，随着模型容量的增加（如嵌入维度、网络层数、线性层大小的增加），模型的学习能力显著增强，训练损失快速下降。特别是在从配置 2（`emb_dim=128, ffn_size=512, num_layers=3`）到配置 3（`emb_dim=256, ffn_size=768, num_layers=4`）的提升中，验证 F1 分数增长较为明显。然而，从配置 3 到配置 4（`emb_dim=256, ffn_size=1024, num_layers=4`）的性能提升趋于饱和，表明进一步增加模型容量对性能的提升有限。此外，Transformer 在整个训练过程中未出现明显的过拟合现象，这可能与 NER 数据集规模较大或任务本身的复杂性较高有关。但最终 Transformer 的最高验证 F1 分数（约 0.55）仍低于双向 RNN。

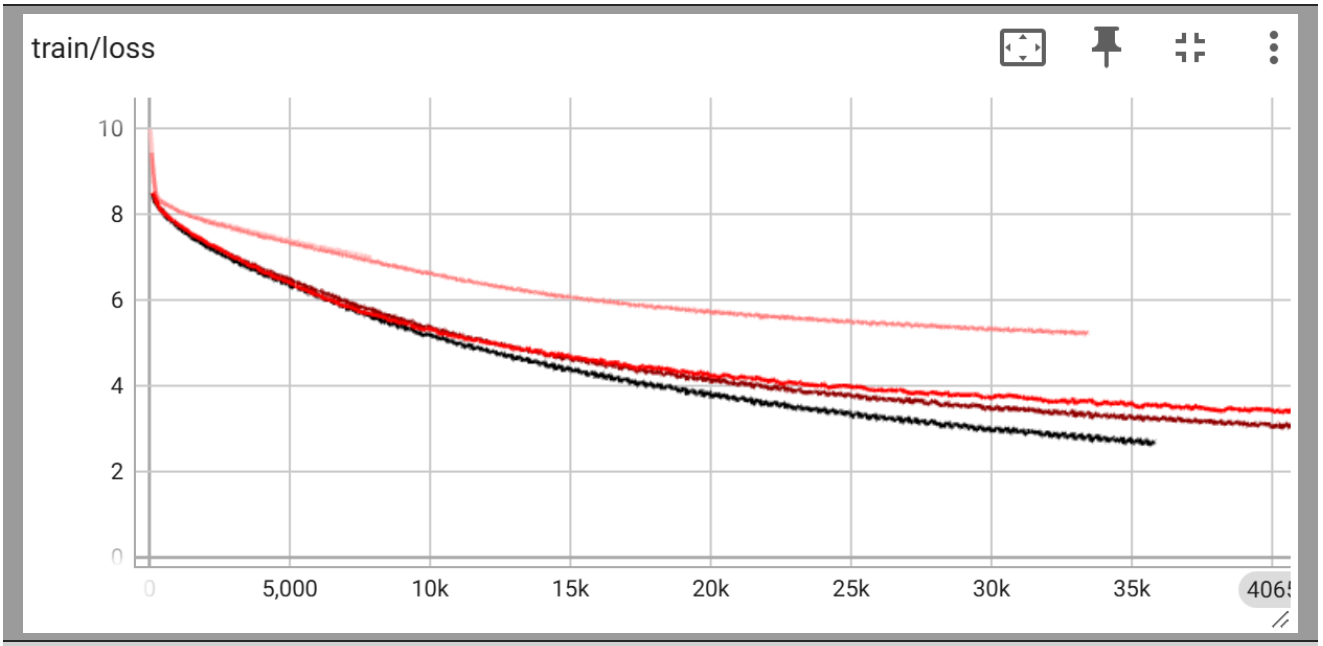
在 NER 任务中，双向 RNN（特别是 LSTM）由于能够有效捕捉序列的上下文信息，在验证 F1 分数上表现最佳，充分展现了其对序列建模的优势。虽然 Transformer 的性能高于单向 RNN，但由于对数据和计算资源的需求较高，且性能提升逐步趋于平缓，在当前数据规模和任务设置下，双向 RNN 是更适合的选择。

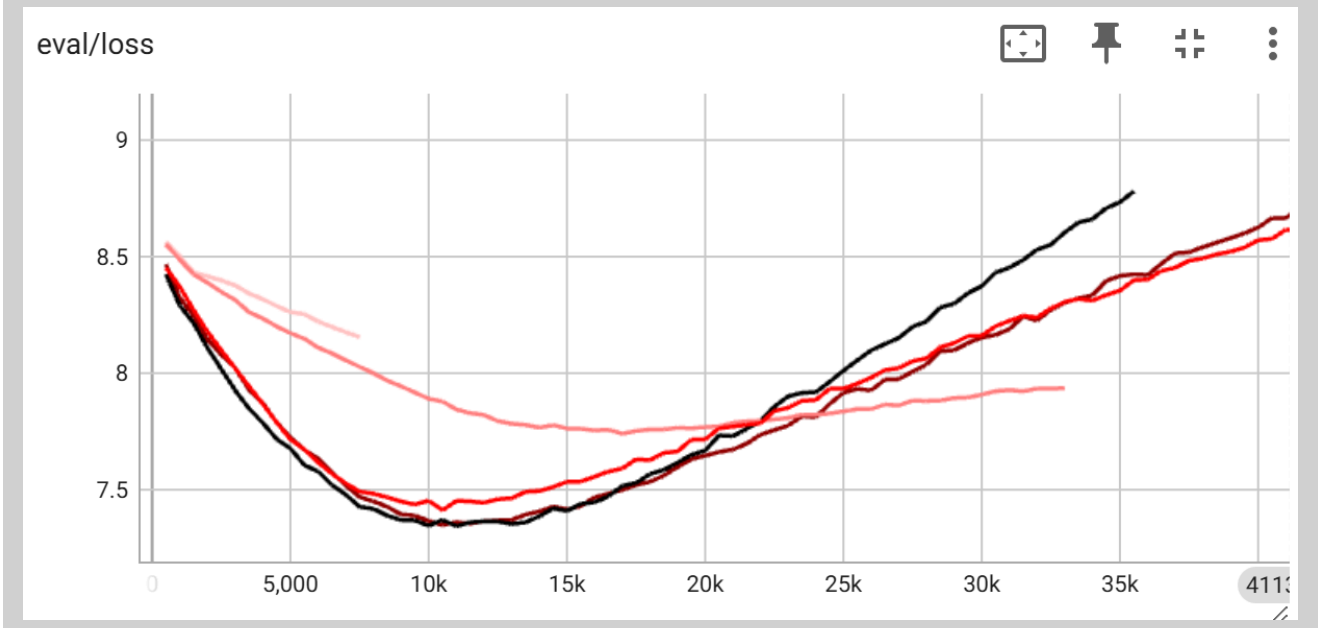
此外，NER 任务中未出现明显的过拟合现象，与情感分类任务的结果形成对比。这可能是由于 NER 数据集规模较大且任务更复杂，使得模型在训练中能够更充分地学习数据模式，而不是记住训练样本的细节。

机器翻译

我们实现了基于Encoder-Decoder Transformer的机器翻译算法，位于`models/translater.py`与`models/encoder_decoder.py`。但是最开始我们的BLEU评价指标的

计算出现了错误，把分词的 @@ 错当成token了导致BLEU虚高，后续我们修复了算法。由于时间和算力的限制，我们只能使用之前的结果。这是部分结果的损失与准确率曲线：





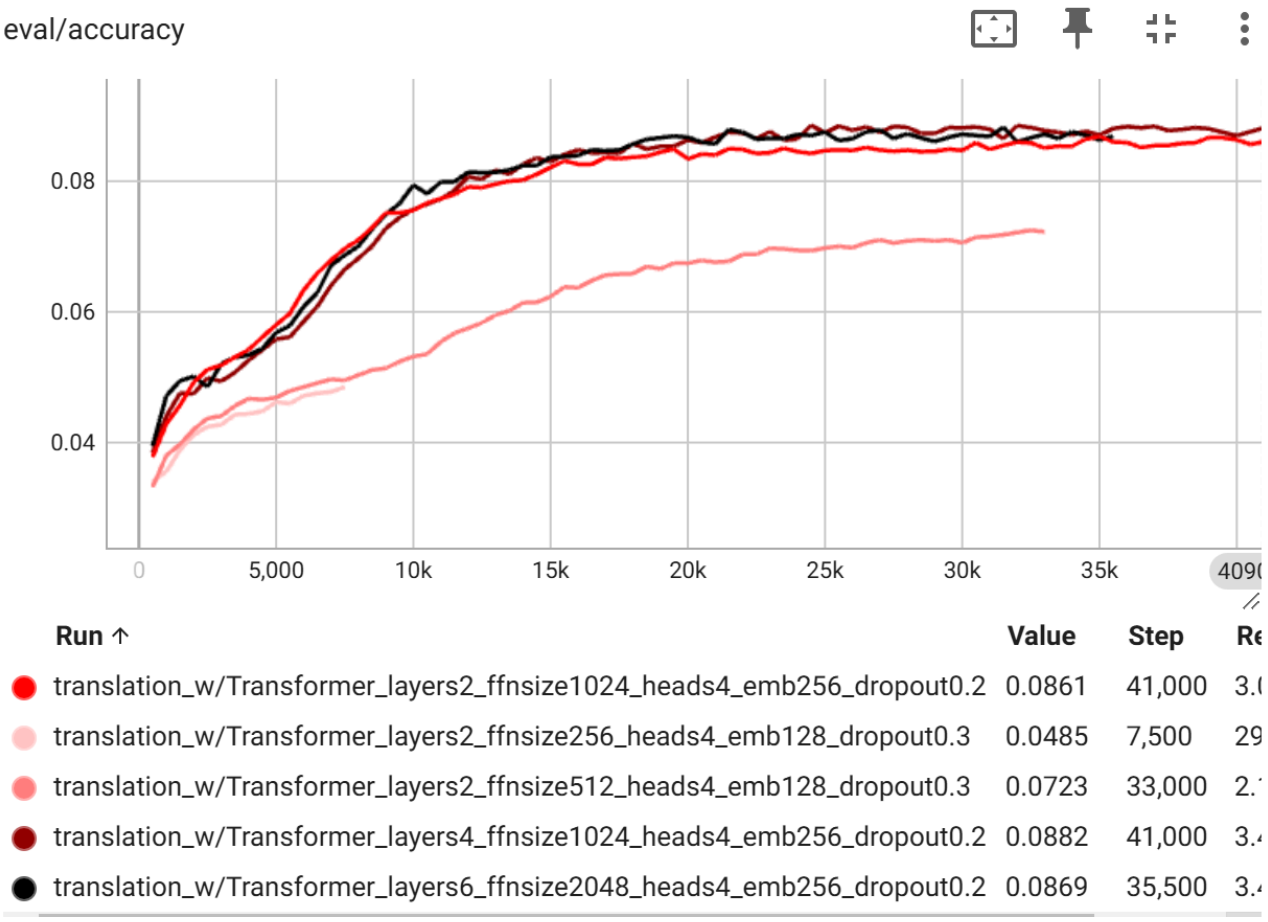


图12 Transformer机器翻译损失与准确率曲线

可以看到，dropout为0.3的两个配置相比0.2的，验证准确率显著更低，验证损失更高并且发生过拟合之前达到的最低点更高，训练损失下降慢，说明0.3的dropout可能影响到了模型的学习能力了，而dropout为0.2的一组。当验证损失急剧上升，验证准确率并没有马上下降，说明可能只是输出的概率分布趋于平缓。然而最终0.08的准确率还是不尽人意。

实验结果及分析

情感分类

给定的数据有些本身带有乱码，这是影响模型正确率的一个因素：

```
1    &#35828;&#23454;&#35805;，&#23545;景&#21306;酒店的硬件我&#27809;有&#25253;太高的期望，又是&#40644;金周期&#38388;，价格肯定偏&#36149;。但&#23454;&#38469;入住后，感&#35273;&#36824;是很不&#38169;
1    ᴛ腹隔腹タ癸ɪ隔1-3だ牧诀初娩暗诀初ペぃ25じRMB衡ス惠10だ牧逗ゑ耕T
```

还有一些数据我认为不太符合常识，比如：

```
1    只有一点，我住的南楼没有电梯，房间还很贵
```

这个我感觉应该是负面情感，但是标签给他打的是正面。

我们统计了三种模型判断错误的例子，画出Venn图：

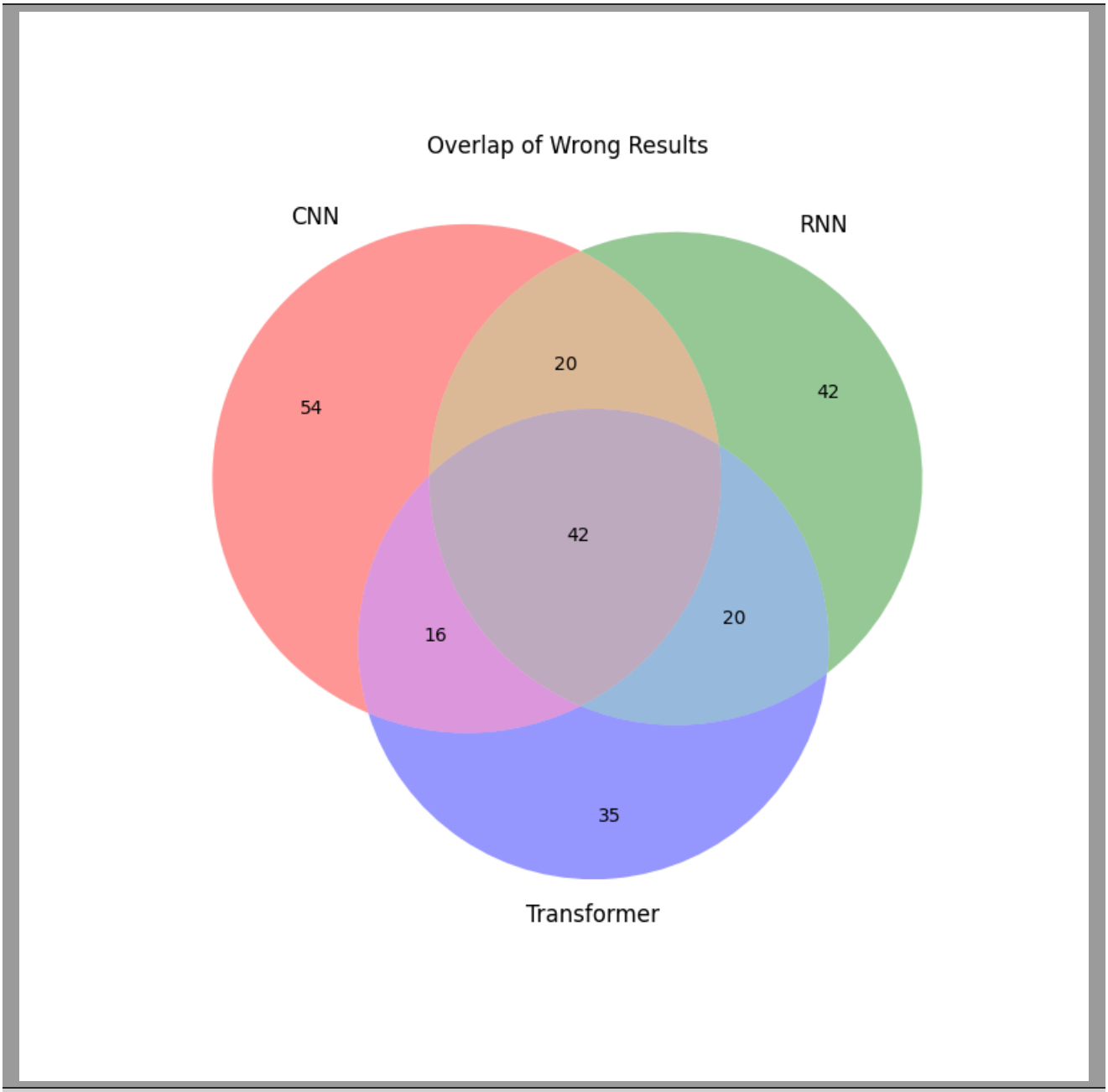


图12 三种模型在情感分类任务的失败样本

三者其实能力比较均衡，有许多例子是共同错误，也有一些是只有一个模型错误的。我们提取出几个共同错误样本：


```
{
        "input": "早餐很一般.房间里还算干净,就是噪音太大,价格相对来说还是贵的",
        "label": "negative",
        "pred": "positive"
},
{
        "input": "夜景不错，房间不错，早餐很不错，离海边和商业区有点距离，服务UNK也有
些距离。",
        "label": "negative",
        "pred": "positive"
},
{
        "input": "房间还可以，早餐不大好，周围比较偏僻。服务态度一般。",
        "label": "negative",
```

```
        "pred": "positive"
},
```


可以看出大多数共同错误样本情感比较模糊，同时包含正面评价和负面评价，很多人类都可能判错，或许这些评价就不是可以用正面和负面二分评价的。

然后我们取仅有CNN错判的样本：


```
{
        "input": "此酒店简直就是糟糕！环境很UNK，离市区还有点远！住在里面的是UNK的旅行社的团队，乱糟糟的。上网的话价格贵的吓死你！对于携程优先推介此酒店非常不满，绝对是给的UNK比较多。不如住火车站对面的如家快捷商务酒店，上网还免费！不知道携程怎么搞的！自己的UNK酒店不推介，UNK这个破酒店。",
        "label": "negative",
        "pred": "positive"
},
{
        "input": "房间临街，隔音很差，加上外面施工，吵得很！房间门锁似乎有故障，UNK门卡显示UNK但就是打不开，服务员的门卡UNK马上就好了。由于只在携程定了1天，临时有事要加一天，前台办理时UNK，第二天总是UNKUNK退房，很烦。宾馆反馈2008年5月20日：您好,给您的住宿带来不便,酒店向您表示诚挚的歉意!您提到的问题,酒店都UNK了改进和跟踪,并将尽最大努力减少影响您入住行程的小问题.并请您继续给予我们信任和支持!酒店周边的施工问题,我们已经进行了协商,夜间施工将得到有效的控制,我们一定UNK如家真诚的服务理念,为您提供更好的服务.",
        "label": "negative",
        "pred": "positive"
},
```


可以发现，CNN错判的样本大多数是含有情感关键词，但含有一些整句的干扰信息，例如“不如住火车站对面的如家快捷商务酒店，上网还免费！”和酒店的友善回复，由于窗口太短无法提取上下文信息，导致错误判断了情感。

取只有RNN错判的样本：


```
{
        "input": "真是可笑,入住该酒店结帐时要多付UNKUNK的费用,他们UNK自己的物品数量,出了差错,UNK的竟是消费者.请问各位:这样的地方你们敢去吗?我选择坚决不再去.如果要我给该酒店评分,我给它0分,他们的服务给客户的好感全被结帐时UNK的UNK所UNK.补充点评2008年4月15日:建议大家到日照时选择其他酒店,祝大家在日照开心.",
        "label": "negative",
        "pred": "positive"
},
{
        "input": "我第一天晚上是在森林公园门口的UNK山庄过的，才住了一个晚上，就忙着搬到武陵源宾馆，主要原因是住宿在森林公园门口太不方便，冷冷清清的。人都跑到武陵源景区门口的宾馆了。但是UNK武陵源宾馆，才UNK，先将武陵源宾馆的UNKUNK如下：优点：1.地理位置方便—跟景区门口距离不到200米，一眼就可以看到景区门口2.占地面积大，UNK舒服—宾馆的占地面积让我吓了UNK，从门口到前台还有40米。够UNK，感觉很好！3.洗手间布局合理，有热水供应，还有个大浴缸—热水供应很足，而大浴缸又能在我们洗完UNK的热水澡之后，再泡泡脚。经过了一天的UNK，简直是无比的享受4.餐厅—餐厅手艺不错，而且价格合理。我们第一天在森林公园门口那边吃了2UNK馆子，竟然给我们还是UNK！！！！毫不夸张地说，就武陵源宾馆的免费早餐都要在森林公园门口那边100块钱一顿的馆子要吃得好！缺点：隔音—房与房之间的隔音做得不太好。但是，换个角度，也希望UNK能避免在走廊里，甚至在方面里大声喧哗，武陵源宾馆很多日本人韩国人入住，别在UNK面前丢了咱们中国的UNK。",
        "label": "positive",
        "pred": "negative"
},
```


对于有些结果，评价的文字很长，比如先说优点再说缺点，或者最后提了一下有强烈情感色彩的词语（例如“开心”），由于我们只提取最后一次的输出作为特征，所以前文的信息容易被遗忘，可能导致RNN错判。

总体来说三个模型已经做到这个任务的较好水平了，因为很多评价的的情感是模糊复杂的。

NER

由于 Transformer 的效果相比 GRU 还有很大差距，这里我们就选取GRU的测试结果进行分析，呈现高频结果：

```
Missed entities: Counter({'月': 13, '包公': 10, '喀喇昆仑': 9, '毕加索': 9, '马拉维': 7, '界首': 6, '昆仑': 5, '熊壁岩': 4, '喀喇昆仑山': 4, '炳奎': 4, '皖': 4, '孤儿院': 4, '高尔基大街': 4, '湘': 3, '土家': 3, '英': 3, 'PATA': 3, '中环': 3, '合肥': 3, '满地可': 3, '毕加索旧居': 3, '函馆': 3, '张浩': 2, '前所': 2, '臧克和': 2, '约翰·拉贝': 2, '纳兰性德': 2, '毕桂发': 2, '闻韶派出所': 2, '闻韶': 2, '派出所': 2, '打铁桥村': 2, '萨利马': 2, '马拉维湖': 2, '约堡': 2, '克夫拉达悬崖': 2, '上帝': 2, '蒙市': 2, '扎琳娜': 2, '张家界市': 1, '张家界': 1, '湖北': 1, '共产国际': 1, '青云路': 1, '沈阳监狱': 1, '查阜西': 1, '延边': 1, ...})
Added entities: Counter({'包': 4, '黄': 3, '昆仑山': 3, '昆仑': 3, '巴': 3, '燕': 2, '张': 2, '英': 2, '德': 2, '红楼': 2, '毕桂': 2, '广场': 2, '什': 2, '公安': 2, '联防队': 2, '铁桥村': 2, '皖K': 2, '全国省市妇联': 2, '妇联': 2, '香港妇女组织': 2, '约堡市': 2, '克夫拉达': 2, '美': 2, '高尔基': 2, '北张家界市': 1, '市委': 1, '湘西': 1, '春联': 1, '九都': 1, '张浩传奇': 1, '湖北农村': 1, '沈阳': 1, '公园': 1, '太平': 1, '西海': 1, '膀': 1, '黄灿': 1, '夏天': 1, '抹汗': 1, '润家园': 1, ...})
```

可以看到，漏检现象主要集中在一些高频实体和复杂实体上，例如“月”（13次）、“包公”（10次）和“喀喇昆仑”（9次）等多次未被识别。可能因为“月”本身具有复杂的含义——和数字在一起表示时间以及单独表示月亮，比较难学习；“包公”的包除了作为姓氏还有广泛的使用；“喀喇昆仑”在训练集没有出现过，本身也比较生僻。基本是由于模型未能充分理解这些实体的上下文特征，或者训练数据对相关领域的覆盖不足。此外，像“孤儿院”和“毕加索旧居”这样的多词长实体也频繁漏检，表明模型在处理复杂实体时的能力有限。对于地名和机构类别，模型的识别效果尤其不理想，可能需要更丰富的领域数据来增强泛化能力。

误检方面，模型对部分普通名词（如“公安”、“妇联”）的判断存在过度泛化的问题，这些词在训练数据中可能被标注为实体，导致模型在实际应用中产生误判；此外，一些多词实体如“喀喇昆仑山”被错误分割为“喀喇昆仑”和“山”，显示出模型在多词实体边界识别上的不足；“黄”、“巴”、“燕”、“张”等姓氏与“包”的情况相同，除姓氏外还有广泛用途。整体来看，模型对上下文信息的依赖性不足，对低频和特定领域实体的表现较弱,未来可通过数据增强、上下文优化和后处理规则进一步提升识别性能。

机器翻译

利用训练失败模型的测试结果，分析失败原因：

```
{
        "src": "28-Year-Old Chef Found Dead at San Francisco Mall ",
        "tgt": "<GO> 28岁厨师被发现死于旧金山一家商场 ",
        "pred": "28-的儿子师旧金山旧金山圣马服务 传统的渔旧金山场"
},
{
        "src": "a 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week. ",
        "tgt": "<GO> 近日刚搬至旧金山的一位28岁厨师本周被发现死于当地一家商场的楼梯间。 ",
        "pred": "“几年一次走出家旧金山,他们次谁分钟寻师,所以办法到的发现,比胖那个公安比赛场。 的当地, 房间"
```

```
},
{
        "src": "but the victim's brother says he can't think of anyone who would
want to hurt him, saying, \"Things were finally going well for him.\" ",
        "tgt": "<GO> 但受害人的哥哥表示想不出有谁会想要加害于他，并称“一切终于好起来
了。” ",
        "pred": "但现在者愿们愿意还是希望任何人钱,。」安慰高兴,便千万。\"他们的说:“谢
事情。  ,好好地了。” "
},
{
        "src": "the body found at the Westfield Mall Wednesday morning was
identified as 28-year-old San Francisco resident Frank Galicia, the San
Francisco Medical Examiner's Office said. ",
        "tgt": "<GO> 旧金山验尸官办公室表示，周三早上于西田购物中心发现的尸体确认为28
岁旧金山居民FrankGalicia。 ",
        "pred": "在卡滩体邸的在医院六,最佳的十二瓜园,从圣马旧金山区泳切瑞士8,卡晚
fovy,K"
},
```


从实验结果中可以看出，模型的翻译表现存在较大的问题，主要体现在语言结构混乱、语义不完整、关键实体错误和上下文联系不佳等方面。例如，在生成的句子中，经常出现无意义的字符拼凑和非目标语言的内容（如拼音或随机字母），这些问题导致生成的句子无法传达正确的语义，也与目标翻译存在显著差距。特别是在专有名词的翻译上，比如地名等，会产生比较多的乱码。但也可以看出模型还是学习到了一些单词的中英对照，例如“旧金山”等。

造成这些问题的可能原因包括数据质量不足、模型能力有限和训练过程中的问题。首先，数据集规模较小，仅包含70k对句子，可能不足以支撑模型学习复杂的语言映射规则。此外，模型并没有使用很智能的分词，我们发现多个相同意思的token，例如 工具 、 工具@@ 、 工具。 @@ 和 工具，@@ 可能就占用了四个词向量的位置，这也就导致了语义更加难以学习。其次，模型本身可能未能充分学习目标语言的语法规则，或由于训练参数设置不佳导致优化效果有限。

为了改进翻译质量，可以从多个方面入手。首先，优化数据质量，清理数据集中的噪声，确保句对的准确性和对齐性。同时，增加训练数据量，以提升模型的泛化能力。最后，通过优化训练过程（如增加训练轮次和调整超参数）以及引入后处理模块，可以进一步提升翻译的流畅性和准确性。这些改进措施有助于模型在翻译任务中的整体表现，减少现有的错误生成现象。

实验总结

本次实验中，我们针对情感分类、命名实体识别（NER）和机器翻译三个任务，分别使用了 CNN、RNN（包括 LSTM 和 GRU）以及 Transformer 模型进行实验和分析。在情感分类和 NER 任务中，Transformer 采用了 Encoder-Only 结构，而在机器翻译任务中使用了完整的 Encoder-Decoder 架构。实验探索了多种模型配置和训练策略，并对不同模型在任务中的表现进行了详细的比较和分析。

在情感分类任务中，CNN、RNN 和 Transformer 均表现良好，验证准确率达到了 85% 左右。实验结果表明，CNN 的小卷积核（如窗口为 2）在提取关键情感特征时表现优异，而更大的卷积核容易引入冗余信息，导致泛化能力下降。RNN（尤其是 GRU）虽然具有较强的特征提取能力，但容易过拟合，特别是双向 RNN 在捕捉上下文信息时虽然性能更高，但其代价是更快地出现过拟合。Transformer 尽管特征提取能力强，但由于情感分类任务对上下文依赖

较弱，其复杂模型的优势未能完全体现。同时，任务中的许多模糊样本（如同时包含正负评价的文本）进一步限制了模型的表现。

在命名实体识别任务中，双向 RNN（包括 LSTM 和 GRU）表现优于其他模型，验证 F1 分数接近 0.7，展现了其在序列标注任务中的强大特性。相比之下，Transformer 在 NER 任务中的性能稍显逊色，验证 F1 分数约为 0.55，可能是因为对上下文的依赖方式不同，以及模型未能充分利用训练数据。分析发现，漏检现象主要集中于生僻实体、高频实体和长多词实体，而误检往往发生在普通名词泛化为实体的情况下。此外，多词实体的边界识别能力较弱也导致了一定的错误率。未来可以通过引入更多领域数据、优化上下文处理和加入后处理规则进一步提升 NER 性能。

机器翻译任务的实验中，由于时间和算力限制，模型表现不佳，验证准确率和 BLEU 分数较低。分析发现，模型生成的句子存在语义混乱、关键实体错误和上下文联系不佳等问题。这些问题主要源于训练数据质量和规模的限制，以及分词策略和模型配置的不足。尽管模型对部分短语的翻译较为准确，但在整体句子结构和语义表达上仍存在较大差距。改进方向包括清理数据集噪声、优化分词策略、扩充训练数据，以及通过调整模型配置和引入后处理模块提升翻译质量。

综合来看，不同任务的最佳模型选择依赖于任务特性。情感分类任务对上下文依赖较低，CNN 以其简单高效的特征提取能力表现优异；NER 任务中，双向 RNN 能更好地捕捉上下文信息，是当前最佳选择；而在机器翻译任务中，Transformer 的潜力尚未完全体现，需优化数据和模型配置以进一步提升性能。实验也表明，数据质量和规模对模型性能的影响显著，未来研究需更加注重数据的预处理和扩充。

通过本次实验，我们不仅对各类模型在不同任务中的表现有了深入理解，也体会到了模型选择、配置调优和数据处理对实验成功的重要性。同时，基于共享框架的实验设计提升了开发和分析效率，为后续任务扩展和改进奠定了基础。

附录

情感分类

CNN

利用 `checkpoints/sentiment/CNN_filters[2, 3, 4]_num[4, 4, 4]_emb256_dropout0.3` 进行测试


```
{
        "input": "酒店有两种房间，一种是UNK的，是主楼，一种是商务型的，在副楼，但事情别选择副楼，附楼有点像民宅的感觉，有UNK，主楼还是不错的，符合四星级的标准，更值得一提的是距离UNK高速很近，非常方便。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "住的套房，但是价格仍然很便宜！这个是这个地区最好的了，但是仍然需要提高！但是已经是最好的了，各位，只能选择这里！",
        "label": "positive",
        "pred": "positive"
},
{
```

```
        "input": "整体环境不错,离市区较远，出租车到火车站15快。我住南楼，298的价位跟房间环境UNK。结账遇到很大麻烦，先说我用了付费饮料及食品。后来又说浴袍没了。通常大家住店都不会检查其是否存在。尤其是在四星酒店。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "房间的装修还可以,早餐一般.位置离火车站也就一个出租车起步价.大堂看着很气派.",
        "label": "positive",
        "pred": "positive"
},
```


一些负面例子：


```
{
        "input": "西楼海景房实际面对的是一片工地，正在施工，不过还好，不影响睡眠。免费升级到了一个家庭套房，有3张床。不过卫生间小点，尤其是台盆。酒店服务还是不错，热情。就是Checkout怎么到中午高峰时段只有一个人。酒店housekeeping也不会补拖鞋。",
        "label": "positive",
        "pred": "negative"
},
{
        "input": "房间挺大，设施也还好用，但装修很差，UNK部分墙壁的壁纸都脱落了，很难看。卫生间比较简陋，但各种用具都还能正常使用。",
        "label": "positive",
        "pred": "negative"
},
{
        "input": "门前在修路,UNK的，又吵，第一感觉不太好。里面的设施、服务一般化，UNK借这个地方，整体水平都不咋样，也算过得去了。",
        "label": "positive",
        "pred": "negative"
},
{
        "input": "三星级的硬件，五星级的服务。服务真的没话说。离解放碑还是稍微远了一点，没有想象的近。不能用走的，还得打车。打车到只要5块钱。地段设施也只是过得去，但是它的服务能让我们两个在外面UNK了一个多礼拜并且UNK重庆插头司机拒载UNK的UNK游客感动得UNK。UNKUNKUNK话真没错。首先，UNK打车到达酒店或是从酒店出发的客人，门童不但负责开门拿行李以外，还会及时将UNK的车牌号UNK在一张小卡片上交给客人。UNK客人UNK物品在出租车上。（貌似重庆插头没有给发票的习惯）酒店checkin的柜台是UNK，客人和前台小姐都是坐着得，感觉非常亲切。虽然是自助早餐，但是你只要对服务员说一声你要的鸡蛋是什么样的，保管五分钟内UNK的给你端上来，而且绝对不会搞错UNKUNK。自助早餐的质量也比我们住的成都的一家类似价格的号称四星的酒店好多了。房间里的拖鞋是那种厚厚的UNK的，不是UNK做的一次性的。卫生间里配有可UNK的UNK。UNKUNK小肥皂，外加一包UNK洗衣粉。对出行有任何疑问只要致电总台就能得到详尽的答案。我们的UNK问题包括附",
        "label": "positive",
        "pred": "negative"
},
```


RNN

checkpoints/sentiment/GRU_layers1_hidden256_emb128_dropout0.2_bi


```
{
        "input": "当地最好的酒店，房间环境都很好，地理位置稍微欠缺一点，下次还会入住。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "只有一点，我住的南楼没有电梯，房间还很贵",
        "label": "positive",
        "pred": "negative"
},
{
        "input": "西楼海景房实际面对的是一片工地，正在施工，不过还好，不影响睡眠。免费
```

```
升级到了一个家庭套房，有3张床。不过卫生间小点，尤其是台盆。酒店服务还是不错，热情。就是Checkout怎么到中午高峰时段只有一个人。酒店housekeeping也不会补拖鞋。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "该酒店是UNK目前最好UNK的酒店.早餐还可以.房间也不错.不过就周围没有娱乐和购物地方...出门不是很方便",
        "label": "positive",
        "pred": "positive"
},
```


Transformer

checkpoints/sentiment/Transformer_layers2_ffnsize256_heads2_emb64_dropout0.2


```
{
        "input": "当地最好的酒店，房间环境都很好，地理位置稍微欠缺一点，下次还会入住。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "只有一点，我住的南楼没有电梯，房间还很贵",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "西楼海景房实际面对的是一片工地，正在施工，不过还好，不影响睡眠。免费升级到了一个家庭套房，有3张床。不过卫生间小点，尤其是台盆。酒店服务还是不错，热情。就是Checkout怎么到中午高峰时段只有一个人。酒店housekeeping也不会补拖鞋。",
        "label": "positive",
        "pred": "positive"
},
{
        "input": "该酒店是UNK目前最好UNK的酒店.早餐还可以.房间也不错.不过就周围没有娱乐和购物地方...出门不是很方便",
        "label": "positive",
        "pred": "positive"
},
```


命名实体识别

checkpoints/ner/GRU_layers2_hidden512_emb256_dropout0.2_bi


```
{
        "input": "如有关紫菜抗癌的说法，来源于日本一家研究机构，该机构医生通过一次一组动物实验，就声称“紫菜含有抗癌成分”；",
        "label_entities": [
                "日本"
        ],
        "pred_entities": [
                "日本"
        ]
},
{
        "input": "但结论公布仅一周，日本早稻田大学的五位权威医学家就指出，这种说法很可能“依据不足”，随后他们通过同样的动物实验，用数据否定了前种结论。",
        "label_entities": [
                "日本早稻田大学"
        ],
        "pred_entities": [
                "日本早稻田大学"
        ]
},
{
        "input": "还有一些报刊发表“中老年人不宜吃鸡蛋”的告诫性文章，说蛋黄中含有较高的
```

```
胆固醇，它是诱发高血压、冠心病、动脉粥样硬化及脑中风病的元凶。",
        "label_entities": [],
        "pred_entities": [
                "黄"
        ]
},
{
        "input": "一些老年人看了这类文章后，就不敢吃鸡蛋。",
        "label_entities": [],
        "pred_entities": []
},
```


部分错误案例：


```
{
        "input": "1994年，我从工作了10年的《经济日报》到湘西北张家界市挂职，当了两年
市委副书记。",
        "label_entities": [
                "经济日报",
                "湘",
                "张家界市"
        ],
        "pred_entities": [
                "经济日报",
                "北张家界市",
                "市委"
        ]
},
{
        "input": "张家界是新开发的旅游胜地，辖境内仍有不少土家族聚居的偏远山区。",
        "label_entities": [
                "张家界",
                "土家"
        ],
        "pred_entities": [
                "张"
        ]
},
{
        "input": "这是土家族山民在原本没有路的地方踩出的路，是他们与外界沟通的生存之
路，看来也是制约他们发展的主要因素。",
        "label_entities": [
                "土家"
        ],
        "pred_entities": []
},
{
        "input": "爬上熊壁岩，我们走村串户。",
        "label_entities": [
                "熊壁岩"
        ],
        "pred_entities": []
},
```


Transformer

checkpoints/ner/Transformer_layers4_ffnsize768_heads4_emb256_dropout0.1


```
{
        "input": "如有关紫菜抗癌的说法，来源于日本一家研究机构，该机构医生通过一次一组
动物实验，就声称“紫菜含有抗癌成分”；",
        "label_entities": [
                "日本"
        ],
        "pred_entities": [
                "日本"
        ]
```

```
},
{
        "input": "但结论公布仅一周，日本早稻田大学的五位权威医学家就指出，这种说法很可能“依据不足”，随后他们通过同样的动物实验，用数据否定了前种结论。",
        "label_entities": [
                "日本早稻田大学"
        ],
        "pred_entities": [
                "日本早稻田大学"
        ]
},
{
        "input": "还有一些报刊发表“中老年人不宜吃鸡蛋”的告诫性文章，说蛋黄中含有较高的胆固醇，它是诱发高血压、冠心病、动脉粥样硬化及脑中风病的元凶。",
        "label_entities": [],
        "pred_entities": []
},
{
        "input": "一些老年人看了这类文章后，就不敢吃鸡蛋。",
        "label_entities": [],
        "pred_entities": []
},
```