# 深度学习第三次平时作业报告

## 2. RNN, LSTM, GRU模型解释

### RNN（循环神经网络）：

RNN是一种专门用于处理序列数据的神经网络。它通过在神经元之间建立有向环来维持状态，从而能够捕捉输入序列中的时间依赖关系。然而，标准RNN难以处理长距离依赖问题，因为它容易出现梯度消失或爆炸的问题。

### LSTM（长短时记忆网络）：

LSTM是RNN的一种特殊形式，旨在解决长期依赖问题。它引入了记忆单元和三个门控机制：输入门、遗忘门和输出门。这些门控机制共同作用以控制信息的流动，允许网络学习何时记住或忘记信息。

### GRU（门控循环单元）：

GRU可以看作是LSTM的一个变种，它将输入门和遗忘门合并为一个更新门，并且去除了输出门，简化了结构。虽然结构上更为简单，但GRU在很多任务中表现与LSTM相当，同时计算效率更高。

## 3. 诗歌生成过程叙述

本作业使用基于RNN的模型进行诗歌生成。具体步骤如下：

①数据预处理：首先对原始诗歌文本进行处理，包括去除无关字符、添加起始(bos)和结束(eos)标记等。然后构建词典(word2id)并将文本转换成ID形式。

②模型训练：采用RNN作为基础模型，通过输入诗歌的前缀部分预测下一个字，利用交叉熵损失函数优化模型参数。经过多次迭代，模型逐渐学会根据已有的文字预测接下来的文字。

③诗歌生成：当模型训练完成后，可以通过给定的开头词汇（如“日”、“红”等），调用模型逐步生成后续的文字直到遇到结束标记(eos)为止。这一步骤通常涉及到采样策略的选择，例如贪婪搜索或随机采样。

## 4. 生成诗歌示例

以下是使用指定的开始词汇生成的部分诗歌示例：

import random

def generate\_poem(begin\_word, model, w2id, id2w, poem\_type=random.choice([5,7])):

    state = model.rnncell.get\_initial\_state(batch\_size=1, dtype=tf.float32)

    current\_word = tf.constant([w2id[begin\_word]], dtype=tf.int32)

    poem\_lines = []

    current\_line = [begin\_word]

    # 忽略的符号列表（模型可能生成的无效字符）

    banned\_tokens = ['，', '。', 'eos', 'UNK']

    for \_ in range(4 \* poem\_type \* 4):  # 放宽生成次数限制

        current\_word, state = model.get\_next\_token(current\_word, state)

        word = id2w.get(current\_word.numpy()[0], 'UNK')

        # 关键逻辑：过滤无效符号

        if word in banned\_tokens:

            continue  # 直接跳过，不加入诗句

        current\_line.append(word)

        # 每句长度达标后添加标点

        if len(current\_line) == poem\_type:

            punctuation = '，' if len(poem\_lines) < 3 else '。'

            poem\_lines.append(''.join(current\_line) + punctuation)

            current\_line = []

        # 生成四句后强制终止

        if len(poem\_lines) >= 4:

            break

    # 补全最后一句（如果未完成）

    if len(poem\_lines) < 4 and current\_line:

        current\_line = current\_line[:poem\_type]  # 截断至规定长度

        punctuation = '。' if len(poem\_lines) >= 3 else '，'

        poem\_lines.append(''.join(current\_line) + punctuation)

    # 确保最终为四句

    return ''.join(poem\_lines[:4])

begin\_words = ["日", "红", "山", "夜", "湖", "海", "月"]

for word in begin\_words:

    poem = generate\_poem(word, model, word2id, id2word, poem\_type=random.choice([5,7]))  # 生成七言诗

    print(f"【{word}】开头生成的诗：\n{poem}\n")

诗句生成结果如下：

