音乐生成模型训练实验报告

报告日期：2025 年 10 月 29 日**实验平台**：Google Colab **实验跟踪工具**：Comet.ml **实验链接**：https://www.comet.com/jason-wei/6s191-lab1-part2/2e9b7e00811442a8950294e6a686312d **关联 notebook**：

一、实验概述

本次实验基于深度学习框架构建音乐生成模型，通过迭代训练优化模型参数，目标是让模型从音乐序列数据中学习内在规律，最终具备生成新音乐的能力。实验全程使用 Comet.ml 跟踪记录，包括训练指标、参数配置及实验资产，确保实验过程可追溯、结果可复现。

二、模型参数配置

注：黑色字体为默认参数值，红色为修改后的参数值

参数类别	参数名称	参数值	说明
训练参数	批量大小（batch_size）	8	每次迭代输入模型的样本数量，平衡训练效率与内存占用，避免内存溢出。
训练参数	训练迭代次数（num_training_iterations）	3000	模型总计训练步数，总训练样本量达 24000（3000×8），保证训练充分性。
训练参数	学习率（learning_rate）	0.0001、0.005、0.01	控制参数更新幅度，设置为适中水平，兼顾收敛速度与训练稳定性。数值大于或者小于该数值较多时都会出现明显的最终损失增大情况，因此判断 0.005 是范围

参数类别	参数名称	参数值	说明
			内较为适中的一个学习率
模型结构参数	嵌入维度（embedding_dim）	256	将离散音乐符号转换为连续向量的维度，影响音乐特征的表达能力。
模型结构参数	隐藏层大小（hidden_size）	1024、2048	模型核心拟合层的容量，2048 的设置提供较强特征学习能力，适配复杂音乐数据。参数修改为 2048 后起始 Loss 和最终 Loss 都有所下降，但不是特别明显
数据参数	序列长度（seq_length）	100	单次输入模型的音乐序列长度，决定模型对“上下文”信息的捕捉范围。

三、训练结果与关键指标

3.1 核心指标表现

- **损失值（loss）**：训练全程记录 3300 个损失数据点，初始损失值为 4.736，最终降至 0.506，下降幅度达 89.3%，表明模型在训练数据上有效学习到音乐序列规律。

- **训练稳定性**：损失值从高到低持续下降，无明显震荡或回升现象，说明学习率、批量大小等参数配置合理，训练过程稳定。

3.2 结果分析

1. **模型拟合能力**：2048 的隐藏层大小为模型提供充足容量，能够捕捉音乐数据中的复杂模式，是损失值大幅下降的核心因素之一。
2. **训练充分性**：3000 次迭代配合 8 的批量大小，确保模型充分接触训练数据，避免因训练步数不足导致的欠拟合问题。
3. **潜在风险**：较大的隐藏层容量（2048）可能带来过拟合风险，需通过后续验证环节确认模型在未见过数据上的泛化能力。

四、后续操作建议

4.1 音乐生成实践

基于训练完成的模型，可通过以下代码生成音乐，生成后需将序列转换为 MIDI 等音频格式，通过听觉判断质量（重点关注旋律连贯性、和声合理性）：

python

运行

```
import torch


def generate_music(model, start_sequence, temperature=1.0, generate_length=1000):
    """
    音乐生成函数
    参数说明:
    - model：训练完成的音乐生成模型
    - start_sequence：生成音乐的起始序列（需与训练数据格式一致）
    - temperature：随机性控制参数（>1 增加创意性，<1 增加稳定性，=1 保持原概率分布）
    - generate_length：生成音乐的总长度（单位：序列符号数）
    """
    model.eval()  # 切换至评估模式，关闭训练特有的层（如 dropout）
```

```
    with torch.no_grad():  # 禁用梯度计算，提升生成速度
        generated_sequence = start_sequence.copy()
        # 循环生成后续序列
        for _ in range(generate_length - len(start_sequence)):
            # 1. 转换当前序列为模型输入格式
            input_tensor = torch.tensor(generated_sequence[-seq_length:], dtype=torch.long).unsqueeze(0)
            # 2. 模型预测下一个符号的概率分布
            output = model(input_tensor)
            # 3. 根据温度调整概率分布
            output = output / temperature
            probabilities = torch.softmax(output[-1], dim=0)
            # 4. 采样获取下一个符号
            next_symbol = torch.multinomial(probabilities, num_samples=1).item()
            # 5. 新增符号加入生成序列
            generated_sequence.append(next_symbol)
    return generated_sequence
```

4.2 模型优化方向

1. **引入验证机制**：划分训练集与验证集，监控验证损失（val_loss）。若 val_loss 上升而 train_loss 下降，需添加 dropout（如 dropout=0.2）或减小隐藏层大小，缓解过拟合。
2. **优化学习率策略**：使用学习率调度器，训练后期自动减小学习率，进一步优化参数，代码示例如下：

python

运行

```
from torch.optim.lr_scheduler import ReduceLROnPlateau

# 初始化调度器（基于验证损失调整）
```

scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=50, verbose=True)

每个训练周期结束后更新学习率

scheduler.step(val_loss)

3. **调整生成参数**：尝试 0.7、1.2、1.5 等不同温度值，生成多版音乐并对比，确定符合需求的随机性水平。

4.3 实验资产管理

1. 访问 Comet.ml 实验链接，查看完整损失变化曲线、参数记录及资产文件（如模型权重），便于复盘实验过程。
2. 将生成的音频文件（MIDI/WAV 格式）上传至 Comet 实验，与训练参数关联存档，方便后续对比不同实验版本的效果。

五、总结

本次音乐生成模型训练实验效果显著，损失值从 4.736 降至 0.506，模型成功学习到音乐序列规律。后续可通过生成实践验证泛化能力，并结合验证机制、学习率调度等方法优化模型，为生成高质量音乐提供支持。