题目：使用PyTorch实现线性回归模型（自定义数据集）

任务描述：

在本作业中，你将使用PyTorch生成一个自定义的数据集，并实现一个简单的线性回归模型。你需要完成以下步骤：

1. 数据集生成与自定义Dataset类

- 生成一个二维输入特征的数据集，输出为一维。数据集应基于一个平面函数生成，并在输出中加入一定的噪声。
- 平面函数的形式为：$y = w_1 x_1 + w_2 x_2 + b$，其中 w_1, w_2 是权重，b 是偏置，x_1, x_2 是输入特征。
- 在生成数据集时，请确保加入适量的噪声，以模拟真实数据中的随机性。
- 使用 `torch.utils.data.Dataset` 实现一个自定义数据集类，要求：
  - 实现 `__len__` 方法，返回数据集的大小。
  - 实现 `__getitem__` 方法，根据索引返回一个样本（特征和标签）。

2. 数据集分割

- 将生成的数据集分割为训练集和验证集，建议按照80%训练集和20%验证集的比例进行分割。
- 使用 `torch.utils.data.DataLoader` 创建训练集和验证集的数据加载器，设置合适的批量大小（batch size）。

3. 线性回归模型实现

- 使用PyTorch实现一个简单的线性回归模型。模型应包含一个全连接层（Linear Layer），用于将二维输入映射到一维输出。
- 定义损失函数为均方误差损失（MSELoss），优化器可以选择SGD或Adam。

4. 模型训练与参数调试

- 使用训练集对模型进行训练，调试学习率、批量大小等超参数，观察模型在训练过程中的表现。
- 在每个epoch中，记录训练集和验证集的损失值。

5. 训练与验证结果分析

- 分别绘制训练集和验证集的损失曲线（loss curve）。
- 分析训练结果，讨论模型是否过拟合或欠拟合，并给出结论。

提交要求：

- **文件格式**：提交一个Jupyter Notebook文件（ `.ipynb` ），包含以下内容：
  i. **代码**：完整的PyTorch代码，包括数据生成、自定义Dataset类、模型定义、训练和验证过程。
  ii. **运行结果**：代码的运行结果，包括损失值、模型参数等。
  iii. **曲线图**：训练集和验证集的损失曲线图。
  iv. **讨论与结论**：以Markdown格式撰写，包含以下内容：
    - 数据生成的步骤和参数设置。
    - 模型的结构和超参数设置。
    - 对损失曲线的分析。
    - 训练结果的讨论和结论。

评分标准：

- 数据集生成与自定义Dataset类实现（20%）
- 模型实现正确性（20%）
- 训练过程与参数调试（20%）
- 损失曲线绘制与分析（20%）
- 报告完整性与结论合理性（20%）

提示：

- 使用 `torch.randn` 生成噪声。
- 使用 `torch.utils.data.DataLoader` 管理数据集。
- 使用 `torch.nn.MSELoss` 作为损失函数。
- 使用 `torch.optim.SGD` 或 `torch.optim.Adam` 作为优化器。

示例代码结构：

```
import torch
from torch.utils.data import Dataset, DataLoader

# 自定义Dataset类
class CustomDataset(Dataset):
    def __init__(self, features, outputs):
        self.features = features
        self.outputs = outputs

    def __len__(self):
        return len(self.features)

    def __getitem__(self, idx):
        return self.features[idx], self.outputs[idx]

# 数据生成
def generate_data(num_samples, noise_std=0.1):
    # 生成特征和标签
    pass

# 模型定义
class LinearRegressionModel(torch.nn.Module):
    # 定义必要的模型结构和参数

# 训练与验证
def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):
    # 训练和验证逻辑
    pass

# 主程序
if __name__ == "__main__":
    # 生成数据
    # 创建Dataset和DataLoader
    # 初始化模型、损失函数和优化器
    # 训练模型
    # 绘制损失曲线
    # 输出结论
```