微调在 ImageNet 上预训练的卷积神经网络

实现 Caltech-101 分类

金潇睿

项目主页: https://github.com/Anderasderry/2025-CV-Midterm-Project

一、摘要

在图像分类任务中，深度卷积神经网络（CNN）由于其出色的特征提取能力，已成为主流方法。然而，在样本数量有限的情况下，从零开始训练 CNN 模型容易过拟合，泛化能力不足。迁移学习（Transfer Learning）通过利用在大规模数据集（如 ImageNet）上预训练的模型参数，在下游任务中仅需较少数据便可获得良好性能。

本实验以 Caltech-101 图像分类任务为例，探究了微调 ResNet-18 预训练模型与用随机初始化参数训练模型在小样本分类任务中的性能差异。

二、 实验环境

- **操作系统**：Windows 11
- **Python 版本**：3.11
- **PyTorch 版本**：2.0.0
- **CUDA 版本**：11.8
- **GPU**：NVIDIA RTX 4060
- **CPU**：Intel Core i9
- **内存**：16 GB RAM

三、数据集与预处理

- **数据集**：Caltech-101 是由加州理工学院发布的一个经典图像分类数据集，包含 101 个物体类别与 1 个背景类别，总计约 9000 张图像，每个类别包含 40 到 800 张图像不等，图像尺寸不统一，背景相对干净，适合进行图像分类任务的初步研究。
- **划分策略**：遵循 Caltech-101 的标准做法，在每个类别选取 30 张图像用于训练集，其中 6 张划入验证集，其余图像放入测试集。

- **数据增强与预处理**：为了提升训练效果，增强模型鲁棒性，在训练前对图像进行预处理，人为增加样本多样性，从而缓解过拟合现象。对于训练集，将图像缩放为 256×256 后，再随机裁剪为 224×224，并进行随机水平翻转和归一化；对于验证集和测试集，将图像缩放为 224×224 后，直接归一化处理。

四、训练策略与超参数设置

- **模型构建**：本实验选用ResNet-18 作为基础模型，使用 PyTorch 提供的 `torchvision.models.resnet18` 接口加载模型。
  - **带预训练版本**：加载在 ImageNet 数据集上预训练得到的权重，对输出层（fc）进行替换，以适应 Caltech-101 的 101 个类别；
  - **不带预训练版本**：随机初始化全部参数，训练过程完全依赖 Caltech-101 数据。
- **优化器**：Adam（Adaptive Moment Estimation）优化器综合了 Momentum 和 RMSProp 的优点，能够根据每个参数的自适应学习率动态调整更新幅度。在图像分类任务中，Adam 收敛速度快，适合小型数据集（如 Caltech-101），并且鲁棒性强，因此本实验选取Adam作为优化器。
- **损失函数**：交叉熵损失（CrossEntropyLoss）
- **学习率设置**：在微调阶段，采用分组学习率策略：对新初始化的全连接层使用较大的学习率 `lr_fc`（如 1×10^{-3}），对预训练的网络主干使用较小学习率 `lr_base`（如 1×10^{-4} 或 1×10^{-5}），以实现“快速学习输出层，缓慢调整特征层”的目标。
- **学习率调度器**：StepLR（每 10 个 epoch 衰减至 0.1 倍）
- **批量大小** `batch_size`：32
- **训练轮数** `lr_base`：20 或 30
- **可视化工具**：TensorBoard

五、实验结果与分析

实验一： 微调预训练模型，网格搜索超参数

对 `epoch`，`lr_fc`，`lr_base` 三个超参数执行网格搜索，网格设置如下，共 8 种组合；

```
epochs_list = [20, 30]
lr_fc_list = [1e-3, 5e-4]
lr_base_list = [1e-4, 1e-5]
```

每一轮训练在验证集上评估准确率，结果保存在`grid_search_results.csv`，并保存准确率最高的模型权重； 最后在测试集上评估最佳模型性能。结果显示，利用超参数组合`epochs = 30, lr_fc = 1e-3, lr_base = 1e-4`训练的模型最终在测试集上的准确率达到 96.33%，是本次实验中准确率最高的模型。下表为不同超参数组合的准确率。

Experiment	Epochs	LR_FC	LR_Base	Best_Val_Acc	Test_Acc
1	20	1e-3	1e-4	0.9472	0.9591
2	20	1e-3	1e-5	0.9389	0.9543
3	20	5e-4	1e-4	0.9521	0.9614
4	20	5e-4	1e-5	0.9406	0.9511
5	30	1e-3	1e-4	0.9472	0.9633
6	30	1e-3	1e-5	0.9323	0.9522
7	30	5e-4	1e-4	0.9488	0.9603
8	30	5e-4	1e-5	0.9389	0.9509

实验二： 随机初始化参数，从头开始训练（Training from Scratch）

使用了在实验一中表现最优的超参数组合，即`epochs = 30, lr_fc = 1e-3, lr_base = 1e-4`，用随机初始化的网络参数在 Caltech-101 上开始训练，训练完成的模型在测试集上准确率为 51.35%, 可以说明在本实验中，预训练能够显著提高模型性能。

可视化分析:

实验利用 TensorBoard 可视化了模型训练过程中，在训练集和验证集上的 loss 曲线和验证集上的 accuracy 变化，所有参数组合的训练记录已上传至 Github项目主页 `task1/runs/`目录下，在此仅展示`epochs=30`的部分训练记录。

- **不同参数组合对比**

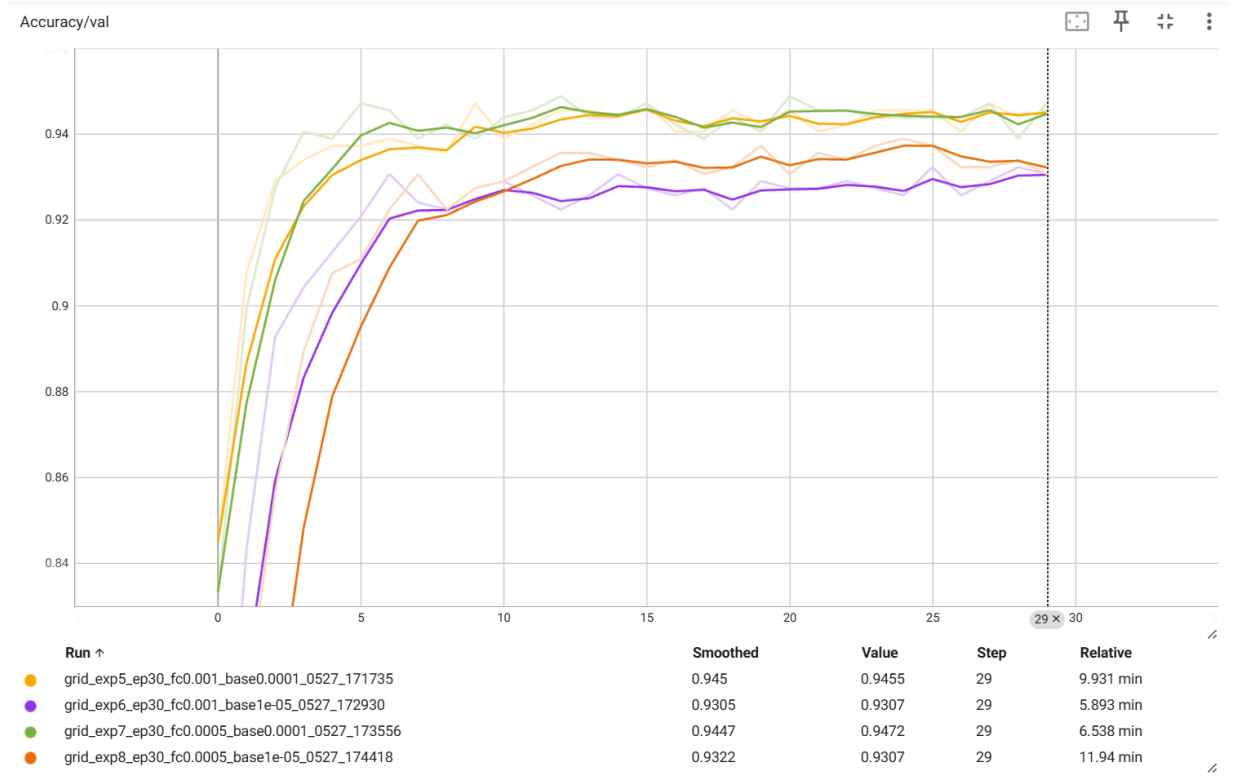

图1：epochs=30，在验证集上的准确率图像

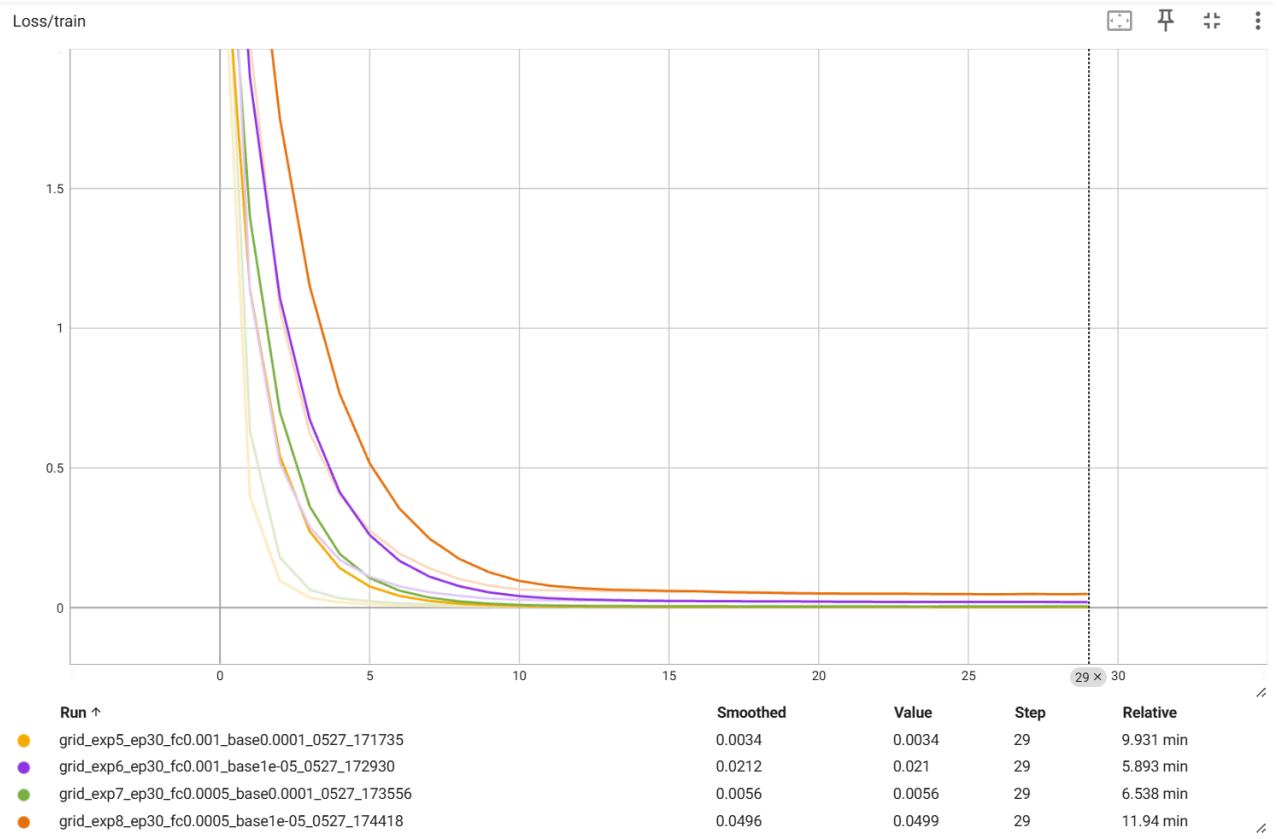

图2：epochs=30，在训练集上的损失函数图像

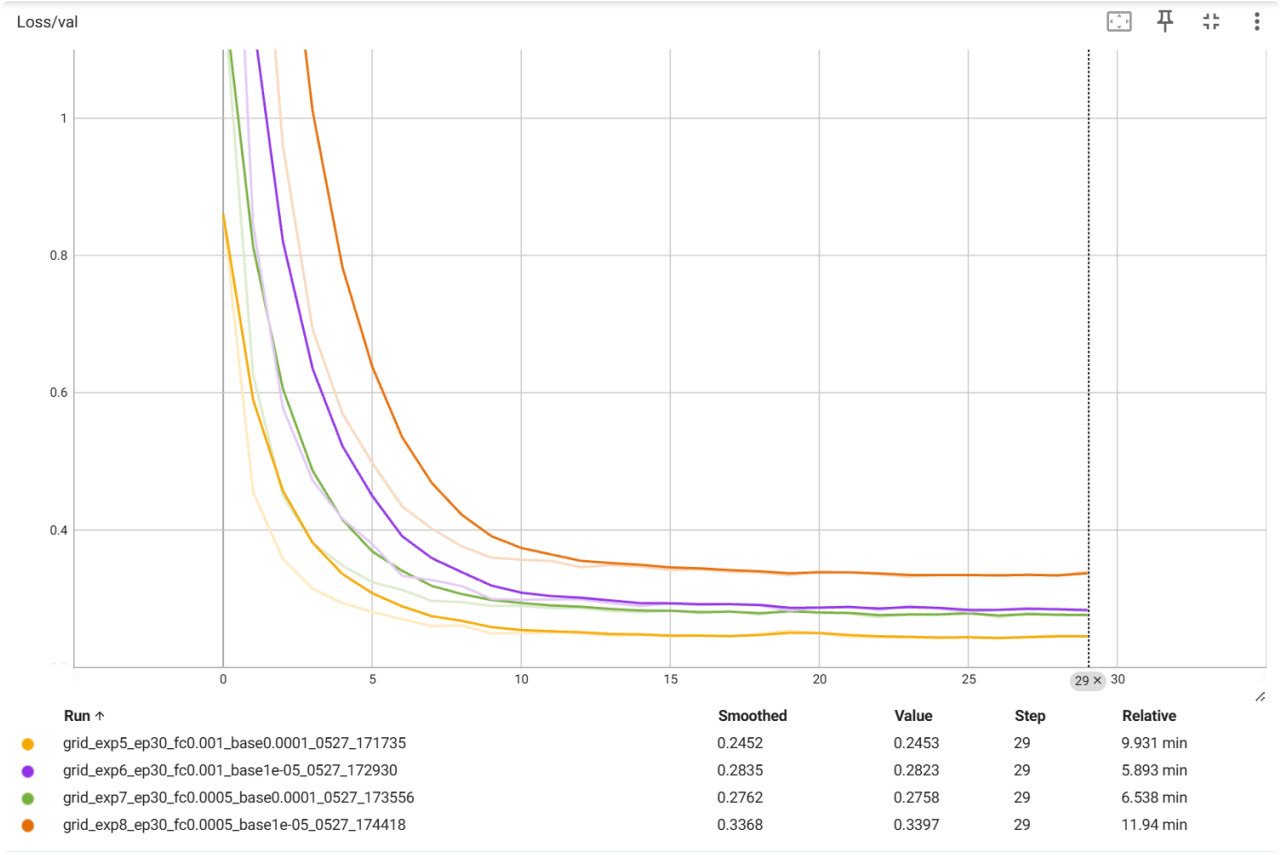

图3：epochs=30，在验证集上的损失函数图像

可以验证，准确率总体呈先上升后平稳的趋势，表明训练初期模型学习能力强，之后准确率收敛到较高水平。损失函数在训练集和验证集上皆呈下降趋势，在训练集上能逐渐收敛至接近于 0. 其中 `lr_base = 1e-5` 的两组与 `lr_base = 1e-4` 的两组明显表现更差，可能是因为基础层的学习率设置得太低，而输出层学习能力较强，导致参数更新不协调，模型性能不佳。

- **预训练与随机初始化对比**

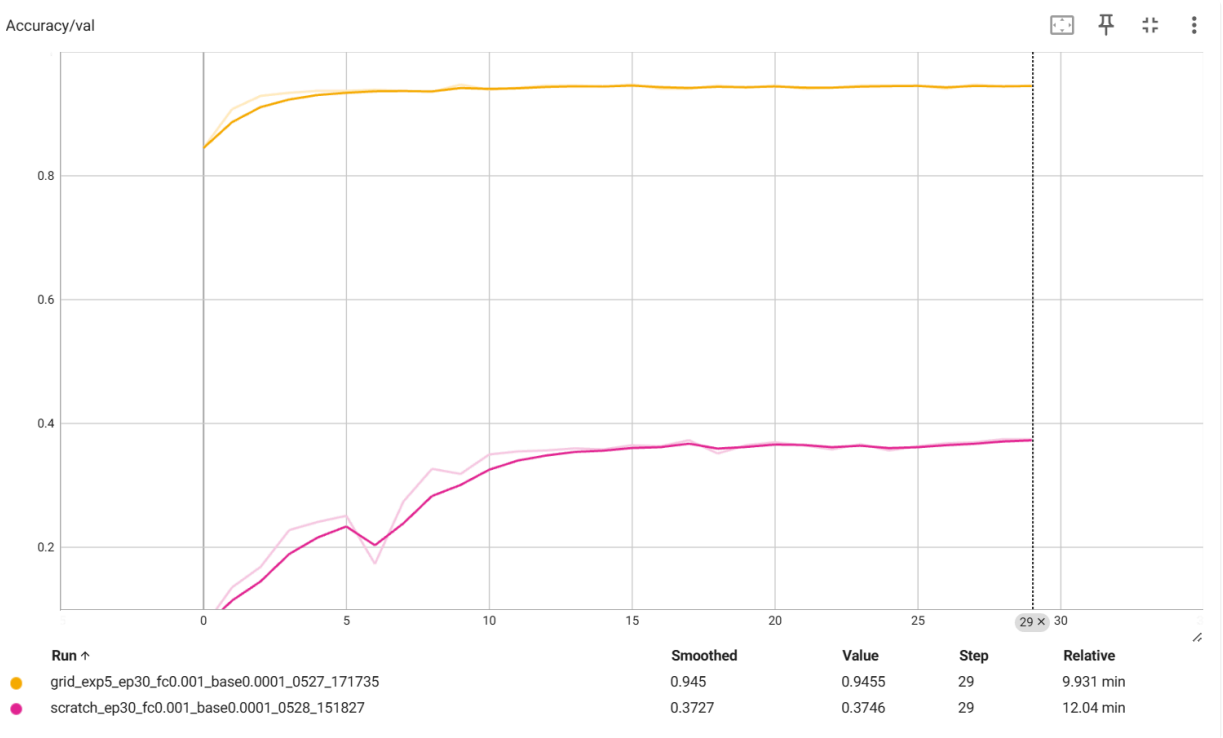

图4：预训练与无预训练模型在验证集上的准确率图像

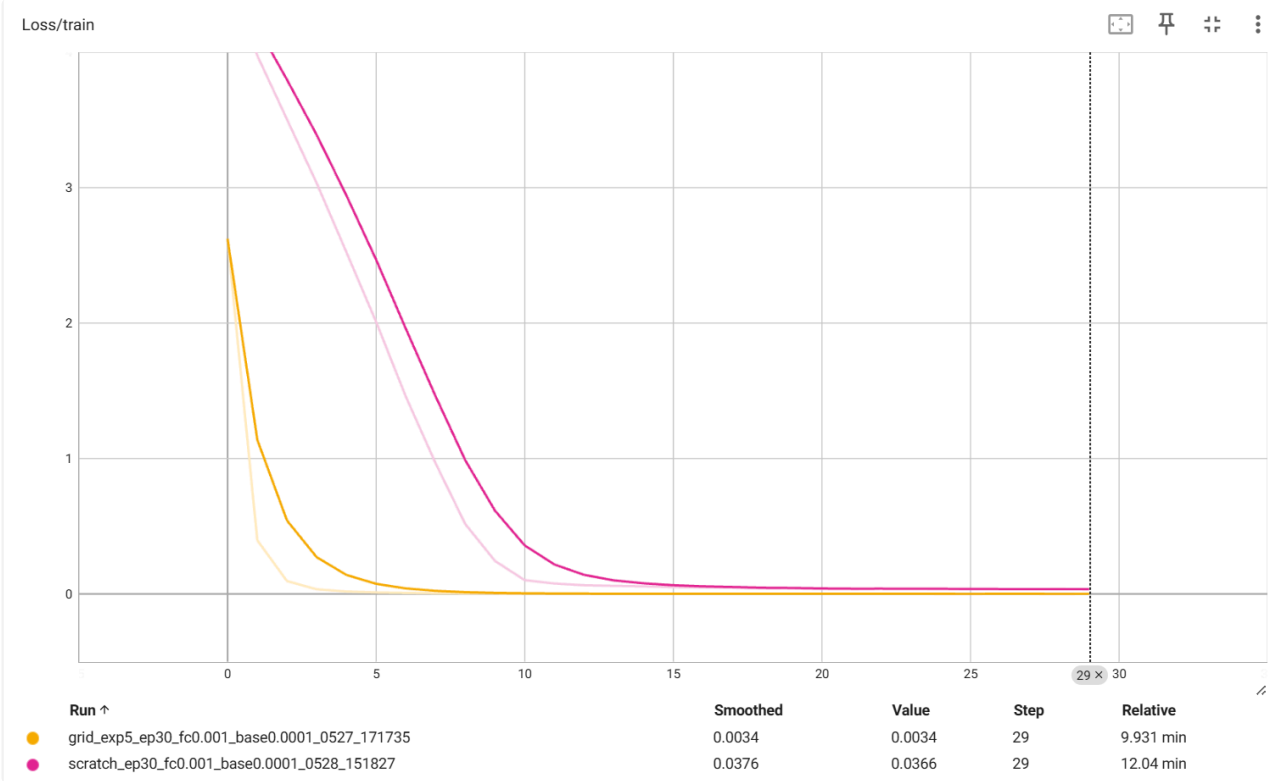

图5：预训练与无预训练模型在训练集上的损失函数图像

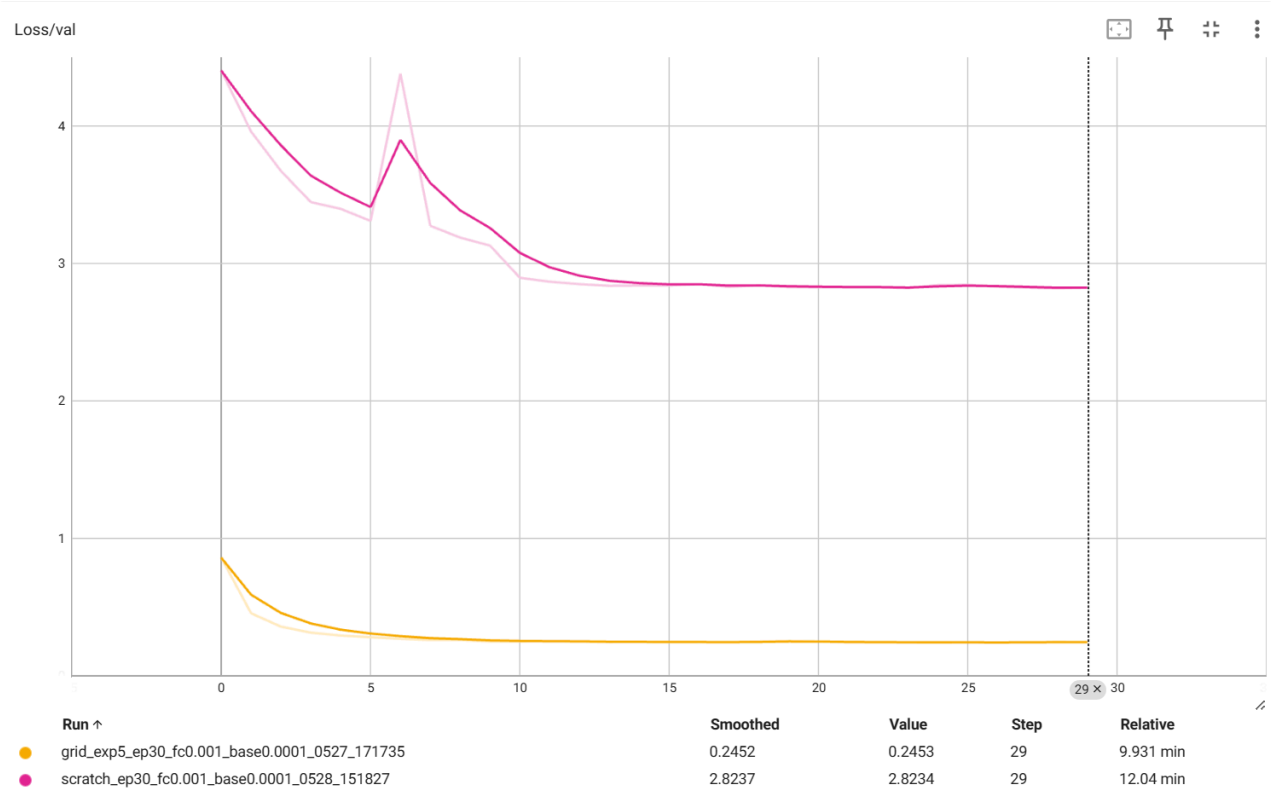

图6：预训练与无预训练模型在验证集上的损失函数图像

可以看出，在使用相同的超参数组合时，未经过预训练的模型在训练过程中，明显准确率更低，损失函数更大，图像的光滑性差，并且训练相同的轮次需要花费更长的时间。因此实验表明，预训练模型在图像分类任务中能够显著提高训练效率和模型性能。

六、附录

- **GitHub Repo**：https://github.com/Anderasderry/2025-CV-Midterm-Project
- **模型权重**：百度云网盘, 提取码: bj4k