

# 作业1 CSC 277 / 477 端到端深度学习 2025年秋季

约翰·多伊 - [jdoe@ur.rochester.edu](mailto:jdoe@ur.rochester.edu)

**截止日期:** 见黑板

## 说明

你的作业解决方案必须键入并使用 $\text{\LaTeX}$ 准备。它必须输出为PDF格式。为了使用 $\text{\LaTeX}$ , 我们建议使用 <http://overleaf.com>, 这是免费的。

你的提交必须引用所有使用的参考文献(包括文章、书籍、代码、网站和个人通信)。所有解决方案必须用你自己的话写, 并且你必须自己编写算法。**如果你与他人合作, 你必须列出你合作的人。**将你的解决方案作为PDF提交到黑板。

你的程序必须用Python编写。你将在这次作业中使用 *PyTorch with Lightning Fabric*(PyTorch, 闪电织物)。所有交付内容应在提供的答案框内, 即 `\begin{answerbox}` 交付内容 `\end{answerbox}`。如果一个问题的交付内容需要代码, 那么代码应作为解决方案的一部分展示。在 $\text{\LaTeX}$ 中, 一个简单的方法是使用`verbatim`环境, 即 `\begin{verbatim}` 你的代码 `\end{verbatim}`。

**关于作业1:** 作业1旨在让你熟悉超参数调优、网络微调、使用WandB进行训练监控以及模型测试。请记住, 网络训练非常耗时, 所以请尽早开始! 将此模板复制粘贴到编辑器中, 例如 [www.overleaf.com](http://www.overleaf.com), 然后只需输入答案即可。你可以使用数学编辑器来简化这个过程, 例如CodeCogs公式编辑器或MathType。你可以使用AI(大型语言模型)插件帮助你在Overleaf中进行 $\text{\LaTeX}$ 格式化。

## 问题 1 - 使用WandB进行训练监控

训练神经网络涉及探索不同的模型架构、超参数和优化策略。监控这些选择对于理解和改进模型性能至关重要。在训练过程中记录实验结果有助于：

- 深入了解模型行为（例如，损失、准确率、收敛模式）。
- 通过评估超参数对稳定性和准确率的影响来优化超参数。
- 检测过拟合或欠拟合并进行必要的调整。

在这个问题中，你将使用ResNet-18模型对牛津-IIIT宠物数据集进行图像分类训练，同时探索各种超参数。你将使用权重和偏差（W&B）来记录你的实验，并根据结果优化你的方法。

### P第1部分：使用W&B实现实验记录（7分）

**准备数据集。**下载数据集。在 `train.py` 中完成数据集定义。创建一个自定义的 `Dataset` 类（示例，参考）或使用 `ImageFolder` 来完成此操作。使用 `oxford_pet_split.csv` 中定义的分割来定义训练、验证和测试分割。在数据集的预处理中，将图像调整到ResNet-18所需的224大小，并使用训练集或ImageNet的统计数据应用图像归一化。

**评估模型性能。**在模型训练过程中，验证集是防止过拟合的关键工具。在 `train.py` 中完成 `evaluate()` 函数，该函数接受一个模型和一个数据加载器作为输入，并输出模型在数据集上的准确度分数和交叉熵损失。

**集成W&B记录。**为了集成W&B进行实验记录，按照以下步骤操作，并在 `train.py` 中添加必要的代码：

1. 参考W&B官方教程以获取指导。
2. 在实验开始时按照教程的代码片段初始化一个新的运行。记录基本的实验**配置**，例如总训练周期、学习率、批量大小和调度器使用。确保运行**名称**是可解释的，并反映这些关键细节。
3. 在训练过程中，在每个小批量后记录训练损失和学习率。
4. 在每个周期结束后，记录验证损失和验证准确度。
5. 在训练结束时，记录模型在测试集上的性能，包括损失和准确度分数。

**实验与分析。**使用默认设置（在 `get_args` 函数中找到）执行实验。登录W&B网站以检查您的实现。

**交付内容：**

- 完成加载数据集的代码片段。
- 实验配置的截图（在W&B中选择特定的运行，然后选择概览标签页）

- 所有记录的图表的截图（在W&B中选择特定的运行，然后在工作区标签下选择图表）。
- 数据是否在W&B界面中准确记录？实验配置是否符合您的预期？
- 分析记录的图表，以确定训练是否已收敛。

答案：

## 第二部分：调整超参数

在本节中，您将实验关键超参数，如学习率和调度器，并研究它们对训练动态的影响。对于每个步骤，一次只更改一个配置。尽量不要修改其他超参数（批量大小除外，可以根据您的计算资源进行调整）。

### 使用Sweep进行学习率调优（5分）

学习率是一个关键的超参数，它显著影响模型的收敛和性能。使用W&B sweep运行训练脚本，并采用以下学习率： $1e-2$ 、 $1e-4$ 和 $1e-5$ 。同时，在您的分析中包含第一部分的默认学习率（ $1e-3$ ）。

#### 交付内容：

- 提供记录的图表截图，显示学习率、训练损失、验证准确度和最终测试准确度。每个图表应展示multiple runs的结果。确保标题和图例清晰且易于解读。
- 分析学习率如何影响训练过程和最终性能。
- 定义搜索空间的扫描配置代码。

答案：

### 梯度范数监控用于爆炸/消失梯度（4分）

在训练过程中监控梯度范数对于检测爆炸梯度（导致不稳定或NaN损失）或消失梯度（导致收敛缓慢/不收敛）至关重要，这些是深度学习实践中常见的严重问题。在 `train.py` 中，扩展您的W&B记录以计算并在每个批次后记录所有模型参数的平均L2梯度范数（例如，通过循环遍历 `model.parameters()`）。运行一个‘问题’变体（例如，将学习率设置为0.1或更大值以引发爆炸梯度）。比较问题变体和默认运行的范数图表和损失曲线。

### 交付内容：

- 显示两个运行过程中梯度范数和训练损失随步骤变化的W&B图表的截图。
- 你在问题运行中注意到爆炸/消失梯度的迹象了吗？如果有，请识别它们并解释你的推断。
- 计算和记录梯度范数的代码片段。

**提示：** 仔细考虑梯度范数计算在训练循环反向传播中的位置。

**答案：**

### 学习率调度器 (4分)

学习率调度器在训练过程中动态调整学习率，提高效率、收敛性和整体性能。在这一步中，你将在 `train.py` 中的 `get_scheduler()` 函数中实现 `OneCycleLR` 调度器。将结果与基线（默认设置）进行比较。如果正确实现，学习率将在训练初期增加，然后逐渐减少。

### 交付内容：

- 提供图表，比较新设置与基线：学习率、训练损失、验证准确度和最终测试准确度。
- 解释 `OneCycleLR` 调度器如何影响学习率、训练过程和最终性能，与基线相比。

**答案：**

### 第三部分：随批量大小缩放学习率 (4分)

如前几部分所观察到的，学习率的选择对于有效训练至关重要。随着批量大小的增加，参数空间中的有效步长也会增加，这需要调整学习率。在本节中，你将研究在批量大小变化时如何适当地缩放学习率。阅读这篇博客文章的前几段，以了解 `Adam`（默认使用）和 `SGD` 优化器的缩放规则。然后，进行实验以验证这些规则。首先，将批量大小加倍（或减半），但不改变学习率，并运行训练脚本。接下来，仅根据文章中的建议调整学习率。将这些结果与默认设置进行比较。请注意，由于总训练步数随批量大小而变化，因此你还应记录已见示例的数量，以创建用于比较的准确图表。

### 交付内容：

- 展示图表，显示：训练损失和验证准确度（x轴为`seen_examples`），以及最终测试准确度。确保图例清晰。你可以应用平滑以获得更好的可视化效果。
- 分析结果：它们是否符合博客文章中讨论的模式？

**答案：**

#### 第4部分：微调预训练模型（3分）

微调利用在大数据集上训练的模型的知识，通过调整其权重以适应新任务。在本节中，你将使用 `torchvision.models.resnet18(pretrained=True)` 微调一个在ImageNet上预训练的ResNet-18模型。修改分类头以匹配你任务中的类别数量，并替换原始代码中的模型定义。为了比较，保持其余设置默认。

**交付内容：**

- 展示图表，显示：训练损失、验证准确度和最终测试准确度。
- 分析预训练对模型学习过程和性能的影响。

**答案：**

## 问题 2 - 模型测试

与关注性能指标的模型评估不同，模型测试确保模型在特定条件下按预期行为。

- **预训练测试:** 在训练之前进行，这些测试识别模型架构、数据预处理或其他组件中的潜在问题，防止在存在缺陷的训练上浪费资源。
- **后训练测试:** 在训练之后进行，这些测试评估模型在各种场景下的行为，以确保其泛化良好，并在真实世界情境中按预期表现。

在这个问题中，你将检查一位前员工留下的代码和模型，该员工在工作中表现出缺乏责任感。代码可以在 Problem 2 文件夹中找到。此任务所需的预定义函数位于 `model_testing.py` 文件中。请按照该文件中提供的说明进行详细指导。

### 第一部分：预训练测试

在这一部分，你将应用预训练测试来验证实验正确性，在训练进行之前。**交付物：**对于第一部分的每个问题，提供以下清晰的交付物：

1. 结果的观察和分析。
2. 针对检测到的问题（如果有）的 suggested approaches。
3. 代码实现。

### 数据泄露检查（3分）

使用 `get_dataset()` 函数加载训练、验证和测试数据集。通过直接比较图像来检查这些集合之间的潜在数据泄露，因为未应用数据增强。由于在Python中相同的对象通常具有不同的哈希值，考虑使用图像哈希等技术进行此比较。

答案：

### 模型架构检查（2分）

使用 `get_model()` 函数初始化模型。验证模型的输出形状是否与标签格式匹配（提示：考虑数据集中的类别数量）。

答案：

### 梯度下降验证 (2分)

验证在单个数据批次的梯度步长后，模型的所有可训练参数是否都已更新。

答案：

### 学习率检查 (2分)

使用`pytorch-lr-finder`实现学习率范围测试。通过检查损失-学习率图来确定学习率是否设置得当。`torch_lr_finder.LRFinder` 所需的组件在 `model_testing.py`中提供。

答案：

## 第2部分：训练后测试

### 死亡ReLU检查 (4分)

在本节中，你将检查训练模型是否存在“死亡ReLU”。死亡ReLU是指ReLU神经元持续输出零，无法区分输入的情况。使用 `get_trained_model()` 函数加载训练模型，使用 `get_test_set()` 函数加载测试集。审查基于ResNet的模型架构，该架构可以在 `utils/trained_models.py`中找到。然后回答以下问题：

1. 识别可能发生死亡ReLU的层，并解释原因。
2. 描述你检测死亡ReLU神经元的 `approach`。
3. 确定训练模型中是否存在死亡ReLU神经元，并提供你的代码实现。

提示：考虑批量归一化操作如何影响死亡ReLU的出现。

答案：

### 模型鲁棒性测试 - 亮度 (4分)

在本节中，你将使用定义的亮度因子评估模型对图像亮度变化的鲁棒性。定义一个亮度因子  $\lambda$ ，通过将像素值乘以  $\lambda$  来确定图像亮度。具体来说， $\lambda = 1$  对应于原始图像的亮度。使用 `get_trained_model()` 函数加载训练模型，并使用 `get_test_set()` 函数加载测试数据集。通过将  $\lambda$  从 0.2 调整到 1.0，以 0.2 的增量来研究模型在不同亮度级别上的性能。

**交付内容：**

1. 绘制一条曲线，显示模型准确性如何随亮度级别变化。
2. 分析关系并讨论观察到的任何趋势。

**答案：**

**模型鲁棒性测试 - 旋转（4分）**

评估模型对图像旋转变化的鲁棒性。将输入图像从0度旋转到300度，每次增加60度。同样，使用 `get_trained_model()` 函数加载训练模型，使用 `get_test_set()` 函数加载测试集。

**交付内容：**

1. 绘制一条曲线，显示旋转角度与模型准确性之间的关系。
2. 分析趋势并讨论任何观察到的模式。
3. 提出潜在改进以增强模型鲁棒性

**答案：**

**归一化不匹配（2分）**

使用 `get_test_set()` 函数加载测试集。假设用于归一化测试数据的均值和标准差（std）与应用于训练数据的均值和标准差不同。

**交付内容：**

1. 计算并报告加载的测试集（教程）中图像的均值和标准差。将这些值与适当归一化后的预期均值和标准差进行比较。
2. 讨论这种不正确归一化对模型性能或预测的一个潜在影响。

**答案：**