利用 **Pytorch** 构建与优化 **AlexNet** 深层卷积神经网络
技术报告

作者：梨花先雪

摘要

本研究探讨了不同归一化方法（BN、LN、GN）对 AlexNet 卷积神经网络性能的影响。通过在不同批次大小下进行实验，结果表明，归一化方法能显著提高模型的收敛速度和准确率。无归一化的网络在较小批次下表现较差，而在较大批次下性能大幅提升。BN 在 batch_size=64 时达到最高准确率 81.31%，LN 在同样批次下表现略优，准确率为 82.66%。GN 在不同批次下表现稳定，总体性能良好。实验结果表明，归一化方法有效提升了模型性能，尤其在大批次训练时表现突出。

关键词— AlexNet, BN, LN, GN,性能

1. 引言

卷积神经网络（CNN）作为深度学习领域的核心技术之一，已在图像分类、目标检测等任务中取得了显著的成果。AlexNet 作为深度学习历史上具有里程碑意义的网络之一，凭借其强大的图像处理能力和较深的网络结构，开启了卷积神经网络在计算机视觉中的广泛应用。然而，随着网络结构的不断深入，模型训练的效率和性能也面临着越来越多的挑战，尤其是梯度消失、梯度爆炸以及训练收敛速度慢等问题，这些问题限制了网络的进一步优化和应用。

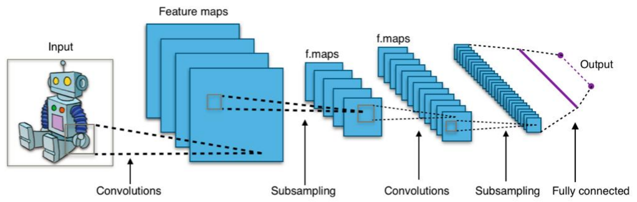


图 1 CNN 工作原理的示意图

为了改善这些问题，研究者们提出了多种网络改进方法，其中归一化技术（如批量归一化 BN、层归一化 LN 和组归一化 GN）被广泛应用于提升网络的训练稳定性和加速收敛。批量归一化 BN 最初被提出时，成功缓解了梯度消失问题，显著提高了训练速度和准确率；而 LN 和 GN 等方法则通过对每一层或特征组进行归一化，进一步提高了模型在不同数据分布下的鲁棒性。

尽管已有大量研究探索了归一化方法对模型性能的提升，但对于不同归一化方法在不同批次大小下的效果比较和具体应用依然较少研究。不同批次大小对训练的影响以及归一化方法的选择，仍然是影响 CNN 性能的关键因素。因此，本研究的主要目标是通过对 AlexNet 网络引入 BN、LN 和 GN 三种归一化方法，结合不同批次大小，探索其在收敛速度、准确率和训练稳定性方面的表现，并通过实验对比分析，提出改进的训练策略。

在本节的最后一段中，我将对本技术报告的主要贡献总结如下：
1) 系统评估了不同归一化方法（BN、LN、GN）对 AlexNet 性能的影响，深入分析了不同批次大小下的训练效果。
2) 验证了归一化技术在加速训练和提升模型性能方面的关键作用。
3) 提供了关于如何选择归一化方法和批次大小的新的参考和实践指导，为今后的网络优化和应用提供了理论依据和实际帮助。

2. 研究方法

（1）研究思路

- **归一化方法的引入：**在原始 AlexNet 中，使用了局部响应归一化（Local Response Normalization，LRN）来增强模型的泛化能力。然而，随着新的归一化技术的提出，我们引入了批量归一化（Batch Normalization，BN）、层归一化（Layer Normalization，LN）和组归一化（Group Normalization，GN）来替代 LRN，评估其对模型性能的影响。
- **批次大小（Batch Size）的调整：**批次大小对模型的训练效果有显著影响。我们在不同的批次大小下，评估了不同归一化方法对模型性能的影响，以确定最佳的训练配置。

（2）网络架构

我们采用了经典的 AlexNet 架构，该网络包含 5 个卷积层和 3 个全连接层。具体结构如下：

- **输入层**：接受尺寸为 224×224×3 的彩色图像。
- **卷积层**：前 5 层为卷积层，使用不同数量和大小的卷积核提取图像特征。
- **激活函数**：每个卷积层后应用 ReLU 激活函数，引入非线性。
- **池化层**：部分卷积层后接最大池化层，减少特征图尺寸。
- **全连接层**：最后 3 层为全连接层，将提取的特征映射到分类结果。
- **输出层**：使用 softmax 函数输出对各类别的预测概率。

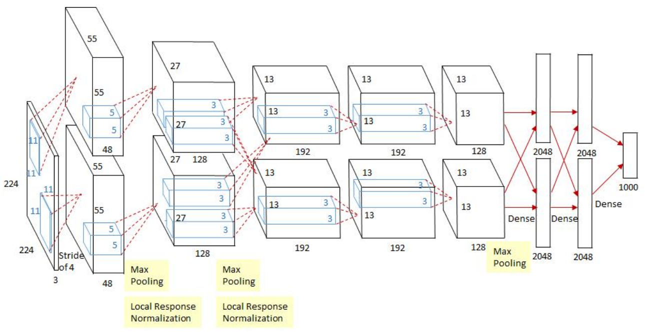


图 2 AlexNet 工作原理的示意图

在不同的实验中，我们在卷积层后插入了不同的归一化层（如 BN、LN、GN），以评估其对模型性能的影响。

（3）优化原则

在训练过程中，我们遵循以下优化原则：

- **学习率（Learning Rate）和动量（Momentum）**：通过实验确定最佳的学习率和动量组合，以确保模型有效收敛。
- **损失函数**：使用交叉熵损失函数，衡量模型预测与真实标签之间的差异。
- **优化算法**：采用随机梯度下降（SGD）算法进行参数更新。
- **正则化**：在全连接层使用 Dropout 技术，防止过拟合。

（4）实验设置

我们在不同的批次大小（如 8、32、64）下，分别对原始的 AlexNet 和加入不同归一化方法的 AlexNet 进行训练和测试。每组实验均运行相同的训练周期，记录模型的收敛速度和最终准确率。

通过上述方法，我们系统地评估了不同归一化方法和批次大小对 AlexNet 性能的影响，为模型的优化提供了有价值的参考。

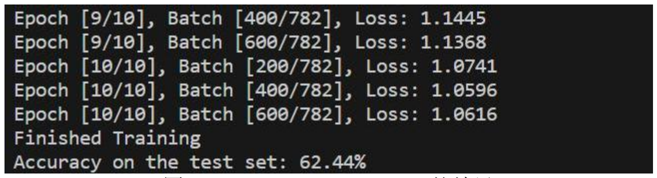
3. 实验与结果分析

实验数据集：Cifar-10 数据集
实验平台：windows
实验内容：对比实验

（1）未经过改进的 AlexNet

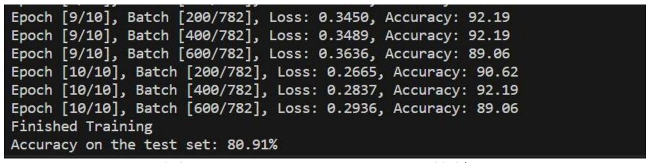
第一个卷积层的输入是尺寸为 224×224×3 的图像，采用了 96 个内核进行处理，每个内核的大小为 11×11，且步长设置为 4。接着，第二个卷积层接收第一个卷积层输出并通过池化操作进行降维，之后使用 256 个内核，每个内核的尺寸为 5×5×48，对处理后的图像进行卷积操作。第三、第四和第五个卷积层直接相连，中间没有池化或归一化操作。第三个卷积层具有 384 个内核，每个内核的大小为 3×3×256，其输入来自第二个卷积层的输出。第四个卷积层也使用了 384 个内核，且每个内核的尺寸为 3×3×192。第五个卷积层则包含 256 个内核，每个内核的大小为 3×3×192。最后的全连接层包含 4096 个神经元。

原始的 AlexNet 模型并未使用归一化层，且在训练时设置了批量大小（batch_size）为 64。在经过 5 小时的训练后，得到了以下预测结果：



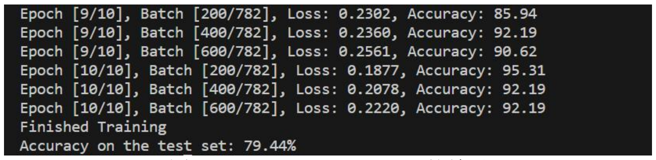
```
Epoch [9/10], Batch [400/782], Loss: 1.1445
Epoch [9/10], Batch [600/782], Loss: 1.1368
Epoch [10/10], Batch [200/782], Loss: 1.0741
Epoch [10/10], Batch [400/782], Loss: 1.0596
Epoch [10/10], Batch [600/782], Loss: 1.0616
Finished Training
Accuracy on the test set: 62.44%
```

图 3 lr=0.001, momentum=0.9 的结果



```
Epoch [9/10], Batch [200/782], Loss: 0.3450, Accuracy: 92.19
Epoch [9/10], Batch [400/782], Loss: 0.3489, Accuracy: 92.19
Epoch [9/10], Batch [600/782], Loss: 0.3636, Accuracy: 89.06
Epoch [10/10], Batch [200/782], Loss: 0.2665, Accuracy: 90.62
Epoch [10/10], Batch [400/782], Loss: 0.2837, Accuracy: 92.19
Epoch [10/10], Batch [600/782], Loss: 0.2936, Accuracy: 89.06
Finished Training
Accuracy on the test set: 80.91%
```

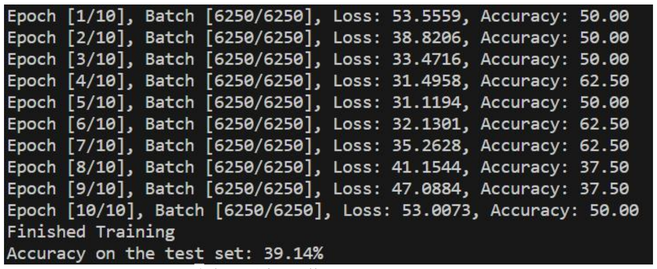
图 4 lr=0.005, momentum=0.9 的结果



```
 Epoch [9/10], Batch [200/782], Loss: 0.2302, Accuracy: 85.94
 Epoch [9/10], Batch [400/782], Loss: 0.2360, Accuracy: 92.19
 Epoch [9/10], Batch [600/782], Loss: 0.2561, Accuracy: 90.62
 Epoch [10/10], Batch [200/782], Loss: 0.1877, Accuracy: 95.31
 Epoch [10/10], Batch [400/782], Loss: 0.2078, Accuracy: 92.19
 Epoch [10/10], Batch [600/782], Loss: 0.2220, Accuracy: 92.19
 Finished Training
 Accuracy on the test set: 79.44%
```

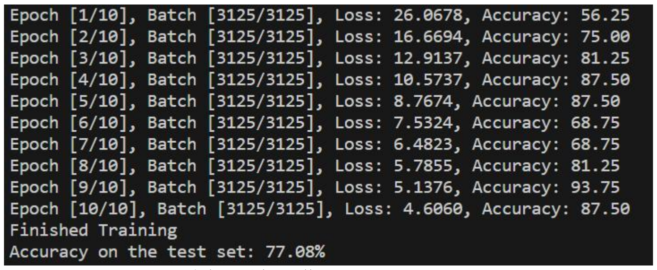
图 5 lr=0.01, momentum=0.9 的结果

从以上三组实验结果可以看出，当 lr 设为 0.001 且 momentum 设为 0.9 时，AlexNet 模型尚未收敛。而后两组情况的收敛结果相差不大，且表现较为稳定。因此，我决定选择学习率 lr = 0.005 和动量 momentum = 0.9 作为训练参数，继续对 AlexNet 神经网络进行改进。为了提高训练效率，我将代码改为 GPU 加速版本，并调整了训练过程中输出准确率的频率，每一轮训练后输出一次。此外，对于没有归一化层的情况，我还在不同批次下测试了训练结果。



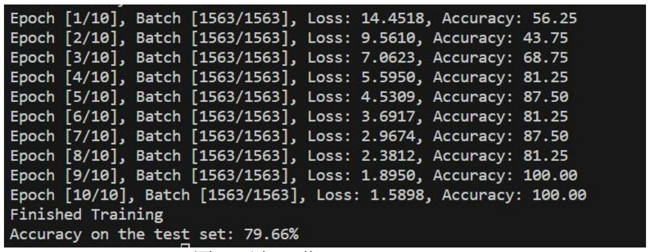
```
Epoch [1/10], Batch [6250/6250], Loss: 53.5559, Accuracy: 50.00
Epoch [2/10], Batch [6250/6250], Loss: 38.8206, Accuracy: 50.00
Epoch [3/10], Batch [6250/6250], Loss: 33.4716, Accuracy: 50.00
Epoch [4/10], Batch [6250/6250], Loss: 31.4958, Accuracy: 62.50
Epoch [5/10], Batch [6250/6250], Loss: 31.1194, Accuracy: 50.00
Epoch [6/10], Batch [6250/6250], Loss: 32.1301, Accuracy: 62.50
Epoch [7/10], Batch [6250/6250], Loss: 35.2628, Accuracy: 62.50
Epoch [8/10], Batch [6250/6250], Loss: 41.1544, Accuracy: 37.50
Epoch [9/10], Batch [6250/6250], Loss: 47.0884, Accuracy: 37.50
Epoch [10/10], Batch [6250/6250], Loss: 53.0073, Accuracy: 50.00
Finished Training
Accuracy on the test set: 39.14%
```

图 6 无归一化，batch_size=8



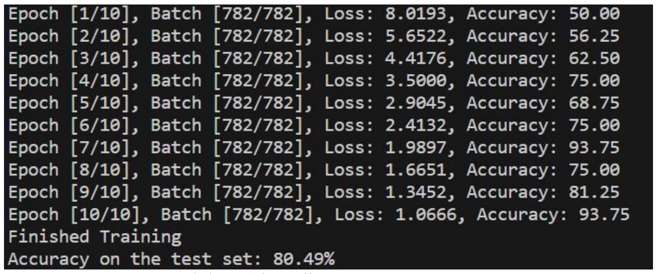
```
Epoch [1/10], Batch [3125/3125], Loss: 26.0678, Accuracy: 56.25
Epoch [2/10], Batch [3125/3125], Loss: 16.6694, Accuracy: 75.00
Epoch [3/10], Batch [3125/3125], Loss: 12.9137, Accuracy: 81.25
Epoch [4/10], Batch [3125/3125], Loss: 10.5737, Accuracy: 87.50
Epoch [5/10], Batch [3125/3125], Loss: 8.7674, Accuracy: 87.50
Epoch [6/10], Batch [3125/3125], Loss: 7.5324, Accuracy: 68.75
Epoch [7/10], Batch [3125/3125], Loss: 6.4823, Accuracy: 68.75
Epoch [8/10], Batch [3125/3125], Loss: 5.7855, Accuracy: 81.25
Epoch [9/10], Batch [3125/3125], Loss: 5.1376, Accuracy: 93.75
Epoch [10/10], Batch [3125/3125], Loss: 4.6060, Accuracy: 87.50
Finished Training
Accuracy on the test set: 77.08%
```

图 7 无归一化，batch_size=16



```
Epoch [1/10], Batch [1563/1563], Loss: 14.4518, Accuracy: 56.25
Epoch [2/10], Batch [1563/1563], Loss: 9.5610, Accuracy: 43.75
Epoch [3/10], Batch [1563/1563], Loss: 7.0623, Accuracy: 68.75
Epoch [4/10], Batch [1563/1563], Loss: 5.5950, Accuracy: 81.25
Epoch [5/10], Batch [1563/1563], Loss: 4.5309, Accuracy: 87.50
Epoch [6/10], Batch [1563/1563], Loss: 3.6917, Accuracy: 81.25
Epoch [7/10], Batch [1563/1563], Loss: 2.9674, Accuracy: 87.50
Epoch [8/10], Batch [1563/1563], Loss: 2.3812, Accuracy: 81.25
Epoch [9/10], Batch [1563/1563], Loss: 1.8950, Accuracy: 100.00
Epoch [10/10], Batch [1563/1563], Loss: 1.5898, Accuracy: 100.00
Finished Training
Accuracy on the test set: 79.66%
```

图 8 无归一化，batch_size=32

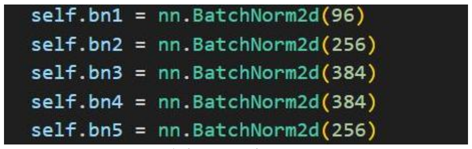


```
Epoch [1/10], Batch [782/782], Loss: 8.0193, Accuracy: 50.00
Epoch [2/10], Batch [782/782], Loss: 5.6522, Accuracy: 56.25
Epoch [3/10], Batch [782/782], Loss: 4.4176, Accuracy: 62.50
Epoch [4/10], Batch [782/782], Loss: 3.5000, Accuracy: 75.00
Epoch [5/10], Batch [782/782], Loss: 2.9045, Accuracy: 68.75
Epoch [6/10], Batch [782/782], Loss: 2.4132, Accuracy: 75.00
Epoch [7/10], Batch [782/782], Loss: 1.9897, Accuracy: 93.75
Epoch [8/10], Batch [782/782], Loss: 1.6651, Accuracy: 75.00
Epoch [9/10], Batch [782/782], Loss: 1.3452, Accuracy: 81.25
Epoch [10/10], Batch [782/782], Loss: 1.0666, Accuracy: 93.75
Finished Training
Accuracy on the test set: 80.49%
```

图 9 无归一化，batch_size=64

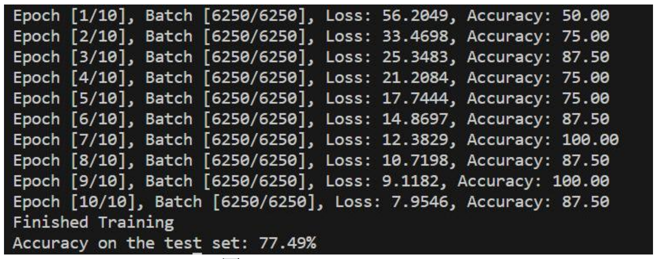
（2）加入归一化

接下来，我分别在原始的 AlexNet 卷积神经网络中加入了 BN、LN 和 GN 三种不同的归一化方法，并在不同的 batch_size 设置下进行了训练。具体的代码实现和细节可以参考 Alexnet.py 文件。



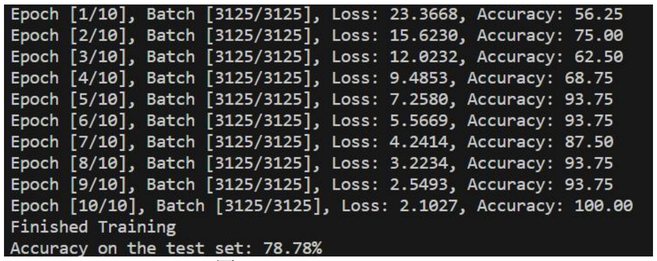
```
self.bn1 = nn.BatchNorm2d(96)
self.bn2 = nn.BatchNorm2d(256)
self.bn3 = nn.BatchNorm2d(384)
self.bn4 = nn.BatchNorm2d(384)
self.bn5 = nn.BatchNorm2d(256)
```

图 10 五次 BN



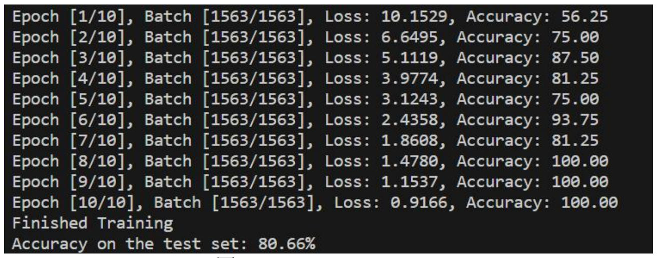
```
Epoch [1/10], Batch [6250/6250], Loss: 56.2049, Accuracy: 50.00
Epoch [2/10], Batch [6250/6250], Loss: 33.4698, Accuracy: 75.00
Epoch [3/10], Batch [6250/6250], Loss: 25.3483, Accuracy: 87.50
Epoch [4/10], Batch [6250/6250], Loss: 21.2084, Accuracy: 75.00
Epoch [5/10], Batch [6250/6250], Loss: 17.7444, Accuracy: 75.00
Epoch [6/10], Batch [6250/6250], Loss: 14.8697, Accuracy: 87.50
Epoch [7/10], Batch [6250/6250], Loss: 12.3829, Accuracy: 100.00
Epoch [8/10], Batch [6250/6250], Loss: 10.7198, Accuracy: 87.50
Epoch [9/10], Batch [6250/6250], Loss: 9.1182, Accuracy: 100.00
Epoch [10/10], Batch [6250/6250], Loss: 7.9546, Accuracy: 87.50
Finished Training
Accuracy on the test set: 77.49%
```

图 11 BN，batch_size=8



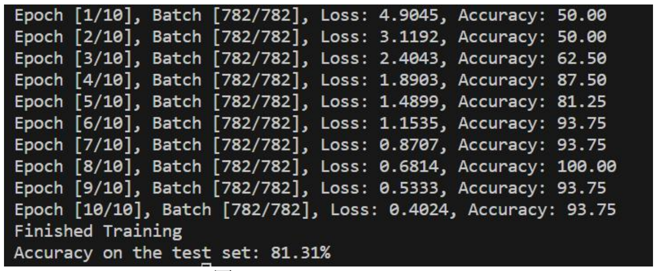
```
Epoch [1/10], Batch [3125/3125], Loss: 23.3668, Accuracy: 56.25
Epoch [2/10], Batch [3125/3125], Loss: 15.6230, Accuracy: 75.00
Epoch [3/10], Batch [3125/3125], Loss: 12.0232, Accuracy: 62.50
Epoch [4/10], Batch [3125/3125], Loss: 9.4853, Accuracy: 68.75
Epoch [5/10], Batch [3125/3125], Loss: 7.2580, Accuracy: 93.75
Epoch [6/10], Batch [3125/3125], Loss: 5.5669, Accuracy: 93.75
Epoch [7/10], Batch [3125/3125], Loss: 4.2414, Accuracy: 87.50
Epoch [8/10], Batch [3125/3125], Loss: 3.2234, Accuracy: 93.75
Epoch [9/10], Batch [3125/3125], Loss: 2.5493, Accuracy: 93.75
Epoch [10/10], Batch [3125/3125], Loss: 2.1027, Accuracy: 100.00
Finished Training
Accuracy on the test set: 78.78%
```

图 12 BN，batch_size=16



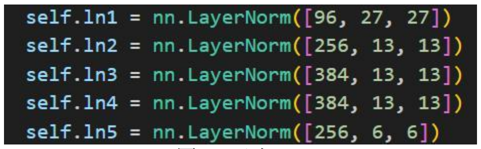
```
Epoch [1/10], Batch [1563/1563], Loss: 10.1529, Accuracy: 56.25
Epoch [2/10], Batch [1563/1563], Loss: 6.6495, Accuracy: 75.00
Epoch [3/10], Batch [1563/1563], Loss: 5.1119, Accuracy: 87.50
Epoch [4/10], Batch [1563/1563], Loss: 3.9774, Accuracy: 81.25
Epoch [5/10], Batch [1563/1563], Loss: 3.1243, Accuracy: 75.00
Epoch [6/10], Batch [1563/1563], Loss: 2.4358, Accuracy: 93.75
Epoch [7/10], Batch [1563/1563], Loss: 1.8608, Accuracy: 81.25
Epoch [8/10], Batch [1563/1563], Loss: 1.4780, Accuracy: 100.00
Epoch [9/10], Batch [1563/1563], Loss: 1.1537, Accuracy: 100.00
Epoch [10/10], Batch [1563/1563], Loss: 0.9166, Accuracy: 100.00
Finished Training
Accuracy on the test set: 80.66%
```

图 13 BN，batch_size=32



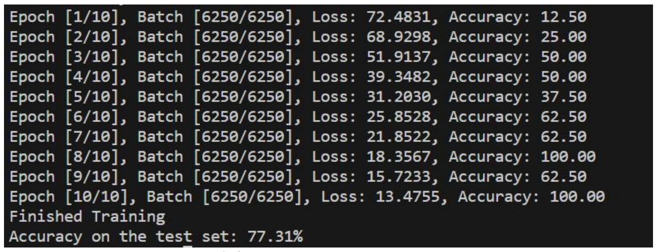
```
Epoch [1/10], Batch [782/782], Loss: 4.9045, Accuracy: 50.00
Epoch [2/10], Batch [782/782], Loss: 3.1192, Accuracy: 50.00
Epoch [3/10], Batch [782/782], Loss: 2.4043, Accuracy: 62.50
Epoch [4/10], Batch [782/782], Loss: 1.8903, Accuracy: 87.50
Epoch [5/10], Batch [782/782], Loss: 1.4899, Accuracy: 81.25
Epoch [6/10], Batch [782/782], Loss: 1.1535, Accuracy: 93.75
Epoch [7/10], Batch [782/782], Loss: 0.8707, Accuracy: 93.75
Epoch [8/10], Batch [782/782], Loss: 0.6814, Accuracy: 100.00
Epoch [9/10], Batch [782/782], Loss: 0.5333, Accuracy: 93.75
Epoch [10/10], Batch [782/782], Loss: 0.4024, Accuracy: 93.75
Finished Training
Accuracy on the test set: 81.31%
```

图 14 BN，batch_size=64



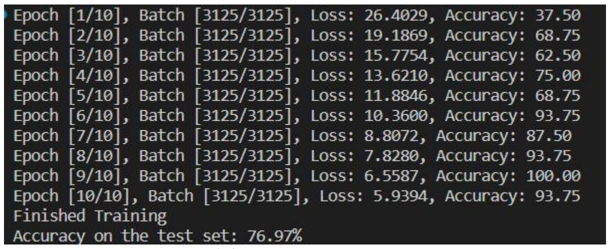
```
self.ln1 = nn.LayerNorm([96, 27, 27])
self.ln2 = nn.LayerNorm([256, 13, 13])
self.ln3 = nn.LayerNorm([384, 13, 13])
self.ln4 = nn.LayerNorm([384, 13, 13])
self.ln5 = nn.LayerNorm([256, 6, 6])
```

图 15 五次 LN



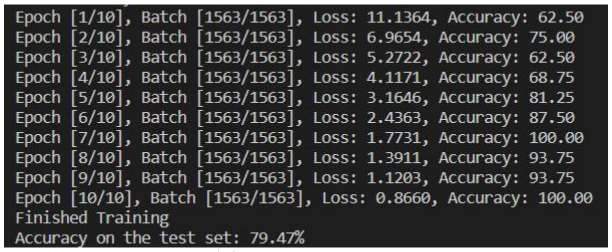
```
Epoch [1/10], Batch [6250/6250], Loss: 72.4831, Accuracy: 12.50
Epoch [2/10], Batch [6250/6250], Loss: 68.9298, Accuracy: 25.00
Epoch [3/10], Batch [6250/6250], Loss: 51.9137, Accuracy: 50.00
Epoch [4/10], Batch [6250/6250], Loss: 39.3482, Accuracy: 50.00
Epoch [5/10], Batch [6250/6250], Loss: 31.2030, Accuracy: 37.50
Epoch [6/10], Batch [6250/6250], Loss: 25.8528, Accuracy: 62.50
Epoch [7/10], Batch [6250/6250], Loss: 21.8522, Accuracy: 62.50
Epoch [8/10], Batch [6250/6250], Loss: 18.3567, Accuracy: 100.00
Epoch [9/10], Batch [6250/6250], Loss: 15.7233, Accuracy: 62.50
Epoch [10/10], Batch [6250/6250], Loss: 13.4755, Accuracy: 100.00
Finished Training
Accuracy on the test set: 77.31%
```

图 16 LN，batch_size=8



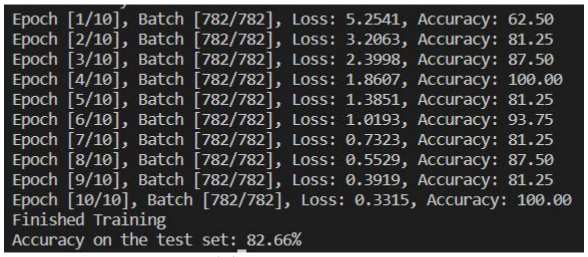
```
Epoch [1/10], Batch [3125/3125], Loss: 26.4029, Accuracy: 37.50
Epoch [2/10], Batch [3125/3125], Loss: 19.1869, Accuracy: 68.75
Epoch [3/10], Batch [3125/3125], Loss: 15.7754, Accuracy: 62.50
Epoch [4/10], Batch [3125/3125], Loss: 13.6210, Accuracy: 75.00
Epoch [5/10], Batch [3125/3125], Loss: 11.8846, Accuracy: 68.75
Epoch [6/10], Batch [3125/3125], Loss: 10.3600, Accuracy: 93.75
Epoch [7/10], Batch [3125/3125], Loss: 8.8072, Accuracy: 87.50
Epoch [8/10], Batch [3125/3125], Loss: 7.8280, Accuracy: 93.75
Epoch [9/10], Batch [3125/3125], Loss: 6.5587, Accuracy: 100.00
Epoch [10/10], Batch [3125/3125], Loss: 5.9394, Accuracy: 93.75
Finished Training
Accuracy on the test set: 76.97%
```

图 17 LN，batch_size=16



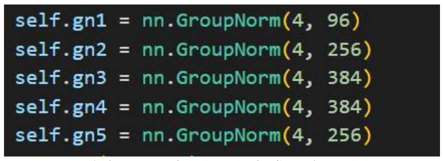
```
Epoch [1/10], Batch [1563/1563], Loss: 11.1364, Accuracy: 62.50
Epoch [2/10], Batch [1563/1563], Loss: 6.9654, Accuracy: 75.00
Epoch [3/10], Batch [1563/1563], Loss: 5.2722, Accuracy: 62.50
Epoch [4/10], Batch [1563/1563], Loss: 4.1171, Accuracy: 68.75
Epoch [5/10], Batch [1563/1563], Loss: 3.1646, Accuracy: 81.25
Epoch [6/10], Batch [1563/1563], Loss: 2.4363, Accuracy: 87.50
Epoch [7/10], Batch [1563/1563], Loss: 1.7731, Accuracy: 100.00
Epoch [8/10], Batch [1563/1563], Loss: 1.3911, Accuracy: 93.75
Epoch [9/10], Batch [1563/1563], Loss: 1.1203, Accuracy: 93.75
Epoch [10/10], Batch [1563/1563], Loss: 0.8660, Accuracy: 100.00
Finished Training
Accuracy on the test set: 79.47%
```

图 18 LN，batch_size=32



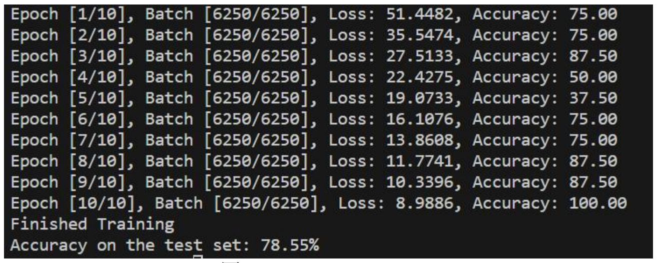
```
Epoch [1/10], Batch [782/782], Loss: 5.2541, Accuracy: 62.50
Epoch [2/10], Batch [782/782], Loss: 3.2063, Accuracy: 81.25
Epoch [3/10], Batch [782/782], Loss: 2.3998, Accuracy: 87.50
Epoch [4/10], Batch [782/782], Loss: 1.8607, Accuracy: 100.00
Epoch [5/10], Batch [782/782], Loss: 1.3851, Accuracy: 81.25
Epoch [6/10], Batch [782/782], Loss: 1.0193, Accuracy: 93.75
Epoch [7/10], Batch [782/782], Loss: 0.7323, Accuracy: 81.25
Epoch [8/10], Batch [782/782], Loss: 0.5529, Accuracy: 87.50
Epoch [9/10], Batch [782/782], Loss: 0.3919, Accuracy: 81.25
Epoch [10/10], Batch [782/782], Loss: 0.3315, Accuracy: 100.00
Finished Training
Accuracy on the test set: 82.66%
```

图 19 LN，batch_size=64



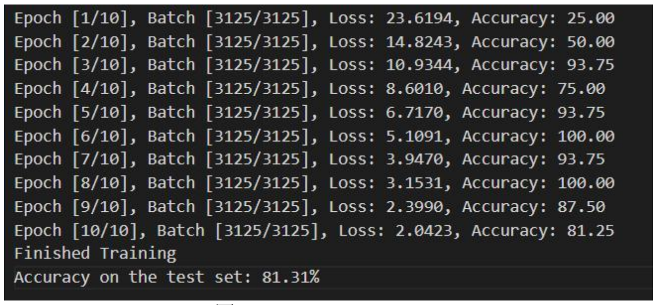
```
self.gn1 = nn.GroupNorm(4, 96)
self.gn2 = nn.GroupNorm(4, 256)
self.gn3 = nn.GroupNorm(4, 384)
self.gn4 = nn.GroupNorm(4, 384)
self.gn5 = nn.GroupNorm(4, 256)
```

图 20 五次 GN（分成 4 组）



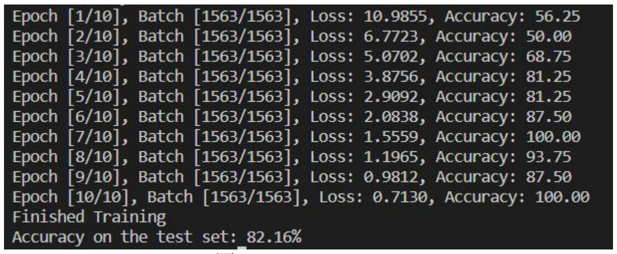
```
Epoch [1/10], Batch [6250/6250], Loss: 51.4482, Accuracy: 75.00
Epoch [2/10], Batch [6250/6250], Loss: 35.5474, Accuracy: 75.00
Epoch [3/10], Batch [6250/6250], Loss: 27.5133, Accuracy: 87.50
Epoch [4/10], Batch [6250/6250], Loss: 22.4275, Accuracy: 50.00
Epoch [5/10], Batch [6250/6250], Loss: 19.0733, Accuracy: 37.50
Epoch [6/10], Batch [6250/6250], Loss: 16.1076, Accuracy: 75.00
Epoch [7/10], Batch [6250/6250], Loss: 13.8608, Accuracy: 75.00
Epoch [8/10], Batch [6250/6250], Loss: 11.7741, Accuracy: 87.50
Epoch [9/10], Batch [6250/6250], Loss: 10.3396, Accuracy: 87.50
Epoch [10/10], Batch [6250/6250], Loss: 8.9886, Accuracy: 100.00
Finished Training
Accuracy on the test set: 78.55%
```

图 21 GN，batch_size=8



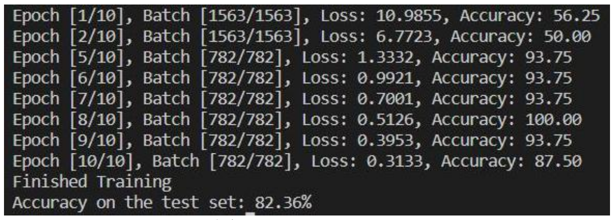
```
Epoch [1/10], Batch [3125/3125], Loss: 23.6194, Accuracy: 25.00
Epoch [2/10], Batch [3125/3125], Loss: 14.8243, Accuracy: 50.00
Epoch [3/10], Batch [3125/3125], Loss: 10.9344, Accuracy: 93.75
Epoch [4/10], Batch [3125/3125], Loss: 8.6010, Accuracy: 75.00
Epoch [5/10], Batch [3125/3125], Loss: 6.7170, Accuracy: 93.75
Epoch [6/10], Batch [3125/3125], Loss: 5.1091, Accuracy: 100.00
Epoch [7/10], Batch [3125/3125], Loss: 3.9470, Accuracy: 93.75
Epoch [8/10], Batch [3125/3125], Loss: 3.1531, Accuracy: 100.00
Epoch [9/10], Batch [3125/3125], Loss: 2.3990, Accuracy: 87.50
Epoch [10/10], Batch [3125/3125], Loss: 2.0423, Accuracy: 81.25
Finished Training
Accuracy on the test set: 81.31%
```

图 22 GN，batch_size=16



```
Epoch [1/10], Batch [1563/1563], Loss: 10.9855, Accuracy: 56.25
Epoch [2/10], Batch [1563/1563], Loss: 6.7723, Accuracy: 50.00
Epoch [3/10], Batch [1563/1563], Loss: 5.0702, Accuracy: 68.75
Epoch [4/10], Batch [1563/1563], Loss: 3.8756, Accuracy: 81.25
Epoch [5/10], Batch [1563/1563], Loss: 2.9092, Accuracy: 81.25
Epoch [6/10], Batch [1563/1563], Loss: 2.0838, Accuracy: 87.50
Epoch [7/10], Batch [1563/1563], Loss: 1.5559, Accuracy: 100.00
Epoch [8/10], Batch [1563/1563], Loss: 1.1965, Accuracy: 93.75
Epoch [9/10], Batch [1563/1563], Loss: 0.9812, Accuracy: 87.50
Epoch [10/10], Batch [1563/1563], Loss: 0.7130, Accuracy: 100.00
Finished Training
Accuracy on the test set: 82.16%
```

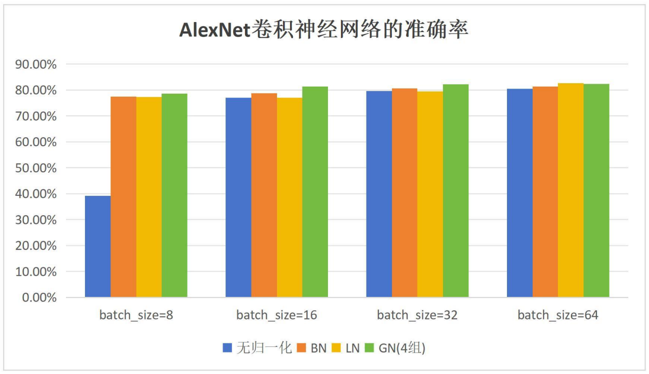
图 23 GN，batch_size=32



```
Epoch [1/10], Batch [1563/1563], Loss: 10.9855, Accuracy: 56.25
Epoch [2/10], Batch [1563/1563], Loss: 6.7723, Accuracy: 50.00
Epoch [5/10], Batch [782/782], Loss: 1.3332, Accuracy: 93.75
Epoch [6/10], Batch [782/782], Loss: 0.9921, Accuracy: 93.75
Epoch [7/10], Batch [782/782], Loss: 0.7001, Accuracy: 93.75
Epoch [8/10], Batch [782/782], Loss: 0.5126, Accuracy: 100.00
Epoch [9/10], Batch [782/782], Loss: 0.3953, Accuracy: 93.75
Epoch [10/10], Batch [782/782], Loss: 0.3133, Accuracy: 87.50
Finished Training
Accuracy on the test set: 82.36%
```

图 24 GN，batch_size=64

4. 结论



通过在不同归一化方法和不同批次大小下对 AlexNet 卷积神经网络进行实验，我们绘制了上述的簇状柱形图。从中可以直观地看到，加入 BN、LN 和 GN 三种归一化方法后，AlexNet 的表现得到了显著提升，收敛速度和准确率均有所改善。这主要是因为三种归一化方法通过不同的方式对特征进行标准化，进而加速了训练过程并提升了模型的性能。

• **无归一化：** 在较小的批次（batch_size = 8）下，模型的性能较差，准确率为 39.14%。然而，在批次较大的情况下（batch_size = 32 和 64），模型的性能显著提升，准确率分别达到了 79.66% 和 80.49%。这可能是由于较小的批次无法提供足够的统计信息，导致训练效果较差；而较大的批次能够更好地利用批次内的统计信息，从而提高了模型性能。

• **BN：** 在所有批次大小下，BN 归一化均表现稳定，并在大多数情况下提供了优异的性能。特别是在较大的批次（batch_size = 64）时，准确率可达 81.31%。BN 通过

规范化每个特征的分布，缓解了梯度消失和梯度爆炸的问题，加速了训练并提升了模型的表现。

• **LN**：LN 归一化在大部分情况下表现较为一致。尽管在小批次下略低于 BN 的性能，但在批次大小为 64 时，准确率超过了 BN，达到了 82.66%。LN 通过对每一层的所有神经元进行规范化，使得它对批次大小的变化更加稳定。

• **GN**：GN 在较大批次下表现出较高的准确率，而在批次大小为 8 时，准确率为 78.55%，相较于其他方法略有下降。总体来说，GN 在不同批次大小下表现较为稳定，因为它将特征划分为多个组并在每个组内进行归一化，这使得它在面对不同批次大小时能保持较为一致的性能。

5. 参考文献

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Advances in Neural Information Processing Systems, vol. 25, pp. 1097-1105, 2012.