实验报告

一、摘要（Abstract）

本实验解决CIFAR-10 数据集上的图像分类问题。训练了两个神经网络Lenet和简化版本的Resnet，并对两个网络的超参数都做了专门调整，以提高分类效果。在Lenet上，模型最高准确率可以达到65.75%;在简化版本的Resnet上,模型最高准确率可以达到89.56%。另外本实验实现了两种减少过拟合的正则化方法：L2正则化和Dropout 正则化，并且验证了正则化的效果。通过实验，初步明确了神经网络一般调优策略，积累了在单卡和多卡上训练模型的经验。

二、引言（Introduction）

1. 本实验解决CIFAR-10 数据集上的图像分类问题。CIFAR-10 数据集包含 60,000 张 32×32 像素的彩色图像，分为 10 个不同的类别。实验目标时在 CIFAR-10 数据集上训练一个神经网络，了解训练和推理的基本流程、探究神经网络的设计（包括网络结构、超参数）对模型性能的影响。

2.相关技术背景有两种正则化方法：L2正则化，限制权重的大小来防止模型复杂性，增强泛化能力；Dropout 正则化，通过随机丢弃神经元，减少过拟合，使模型更具鲁棒性；ResNet,2015 年Kaiming He 等人提出d 残差网络（ResNet）,解决深层网络中的训练困难（如梯度消失和过拟合等问题）;数据并行,每个计算单元上复制同一模型,这些模型在结构和参数上是相同的，但它们各自处理不同的数据，通过并行计算加快运算速度。

3.我的实验思路：

针对Task1,将project1.ipynb中提供的代码复制到Original_version_model.py中，作为实验的baseline,训练后得到的相应的模型权重文件，训练过程loss的json文件，训练过程中loss的变化图像存储在modelset/original_version_model文件夹中；考虑到原始版本的代码运行速度较慢，故想采用GPU并行训练加快训练速度，但原始版本的代码只支持CPU运算，无法直接用多张GPU并行训练，所以修改了代码，采用数据并行，并存储为Original_version_model_multiGPU.py,在4张A100上进行了测试，可以做到不损失准确率且加快训练速度；

针对Task2,分别试验L2正则化和Dropout 正则化方法，在Original_version_model_multiGPU.py基础上修改已加入正则化功能，将修改后的代码分别存储为L2_regularization.py和Dropout_ regularization.py，在4张A100上进行了测试，验证两种正则化方法都有助于预防过拟合；

针对Task3,调整超参数,在Original_version_model_multiGPU.py中设置不同的config，调节num_epochs,lr,momentum,batch_size这些参数，在4张A100上进行测试，观察损失函数以及模型在训练集和测试集上的性能变化；

针对Task4,借鉴Resnet,实现了一个18层的简化版本的Resnet,观察分类准确率和训练时间，并且也设置了不同的超参数config，调整num_epochs,lr,momentum这些参数，尽可能提高模型分类的性能。

4.主要实验结论：

一般调优策略：

（1）优先调学习率：优先确定合适的学习率，因为它直接影响模型是否能有效收敛。大学习率会让模型学习更快，但容易跳过最优解，甚至使训练过程不稳定。小学习率训练更稳定，但训练速度会变慢，可能陷入局部最优。可以使用学习率调度器，例如

ReduceLROnPlateau，可以在模型表现停滞时自动减小学习率。但学习率调度器包含有新的参数，仍然需要调参，故实验中直接将学习率作为参数，未使用学习率调度器。

（2）**确定批大小**：在保证显存足够的情况下，选择较大的 batch_size，这样能更好地并行处理并且稳定训练。

（3）**训练轮数**：过少的训练轮数会导致欠拟合，模型在训练数据上表现不佳：增加训练轮数会让模型有更多机会学习，但也可能导致过拟合，模型在训练集上准确率不断提升，在测试集上效果准确率不变或者降低。如果模型在更早的轮数后效果已经停止提升或开始恶化，可以适当减少。

（4）**动量**：增大动量值可以帮助模型更快地收敛到较优的解；动量值较小则训练更为稳定，但收敛速度可能变慢。

另外，**模型选择的重要性大于对模型参数调整的重要性**，实验中选择 Resnet 后效果明显好于 Lenet。

三、方法（Methodology）

本实验相关采用两种正则化方法：**L2 正则化和 Dropout 正则化**。

L2 正则化：通过向损失函数添加权重的平方和来防止模型过拟合。具体而言，它在优化过程中引入一个惩罚项，使得模型的权重不会过大。通过限制权重的大小，L2 正则化促使模型学习到更加平滑的函数，从而增强模型的泛化能力，降低过拟合的风险。

Dropout 正则化：是一种在训练过程中随机忽略（“丢弃”）一部分神经元的方法，以减少神经网络的复杂性和防止过拟合。 在每次训练迭代中，以一定的概率 p 随机选择某些神经元，并将其输出设置为零。这样的操作会使得网络在每次前向传播时都以不同的结构进行学习。通过随机丢弃神经元，Dropout 防止了神经元之间的依赖性，使得模型变得更加鲁棒。训练结束后，所有神经元都参与推理，但输出会根据训练时的丢弃比例进行缩放，以保持一致性。

Task4 采用 **ResNet 实现**，残差网络是由多个残差块（residual blocks）构成的深度卷积神经网络。每个残差块通过跳跃连接将输入直接添加到输出，从而形成如下结构：

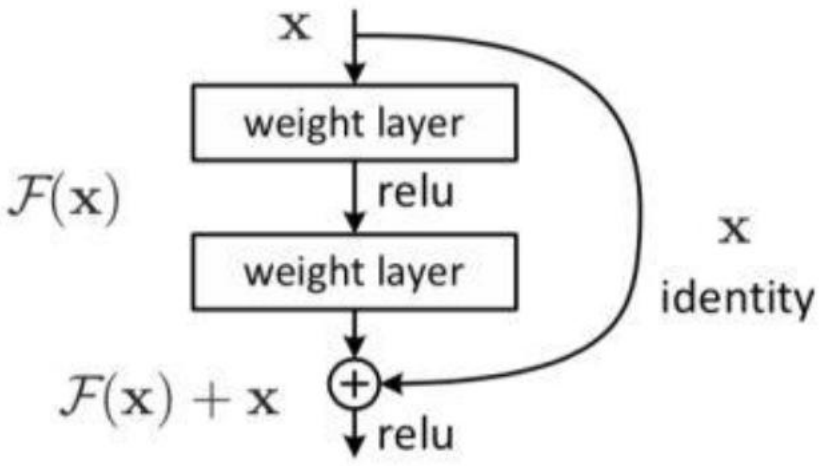


Figure 2. Residual learning: a building block.[1]

其中，F(x) 是经过卷积和激活函数处理的输入 x。这样的结构允许网络学习输入与输出之间的“残差”。

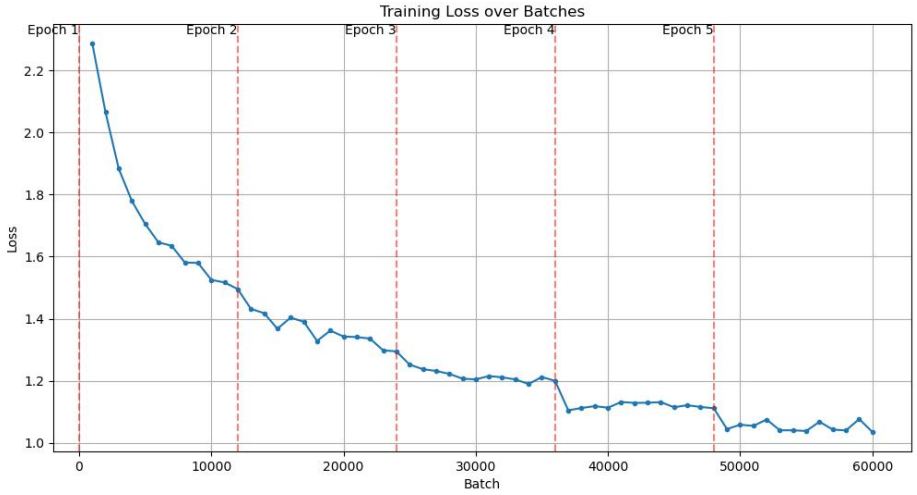
采用残差网络结构优越性在于:1.易于训练:残差连接使得网络能够直接学习恒等映射，当更深的网络不需要改进时，网络可以轻松地学习到相同的输出。2.有效的梯度传播：跳跃连接使得梯度在反向传播时更容易通过网络流动，从而缓解了梯度消失的问题。3.支持极深网络:ResNet 使得构建非常深的网络成为可能，如 152 层甚至更深，而不会出现训练困难。**总之，通过残差学习和跳跃连接，ResNet 能够有效地训练更深的网络，提高模型的性能和泛化能力，本次实验中采用此结构，实现了一个 18 层的简化版本的 Resnet,分类准确率有明显提升。**

四、实验（Experiments）

1.针对 Task1,将 project1.ipynb 中提供的代码复制到 Original_version_model.py 中，作为实验的 baseline。

实验设置（Experimental Setup）： CPU；数据集是 CIFAR-10 数据集；模型是 Lenet；超参数设置(lr=0.001, momentum=0.9,num_epochs = 5)。

实验结果（Main Results）：训练后得到的相应的模型权重文件，训练过程 loss 的 json 文件，训练过程中 loss 的变化图像存储在 modelset/original_version_model 文件夹中。**训练五轮 loss 随训练整体呈下降趋势。测试集上的准确率 60%。**




```
epoch 5: batch  9000 loss: 1.032
epoch 5: batch 10000 loss: 1.056
epoch 5: batch 11000 loss: 1.028
epoch 5: batch 12000 loss: 1.056
Finished Training
测试集中的准确率为: 60 %
```

2.针对 Task3,调整参数，考虑到原始版本的代码运行速度较慢，故想采用 GPU 并行训练**加快训练速度**，但原始版本的代码只支持 CPU 运算，无法直接用**多张 GPU 并行训练，**所以修改了代码，采用**数据并行，**并存储为 Original_version_model_multiGPU.py。在 4 张 A100 上进行了测试，可以做到不损失准确率且加快训练速度。

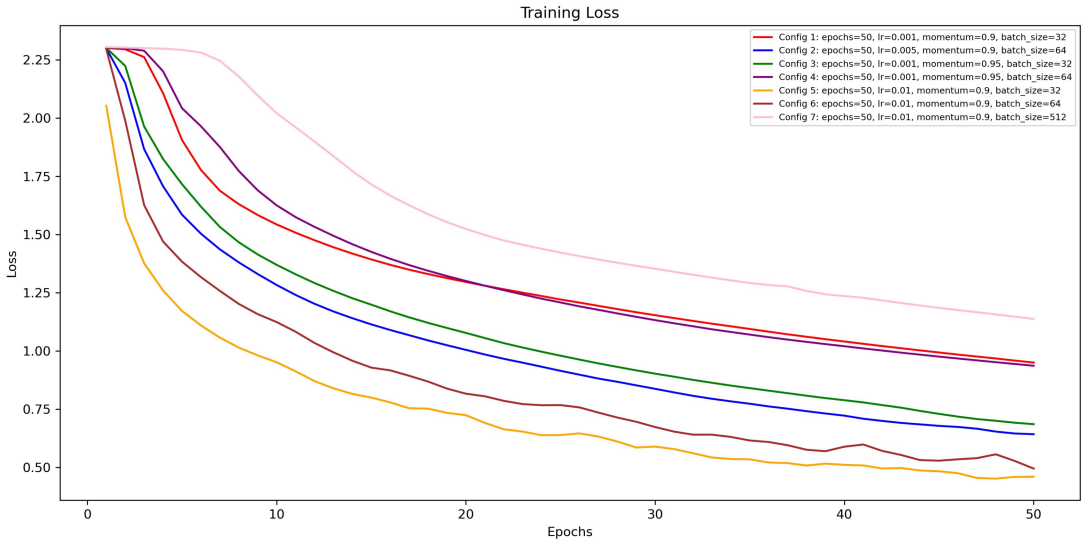
实验设置（Experimental Setup）： 4 张 A100；数据集是 CIFAR-10 数据集；模型是 Lenet；超参数设置如下图。

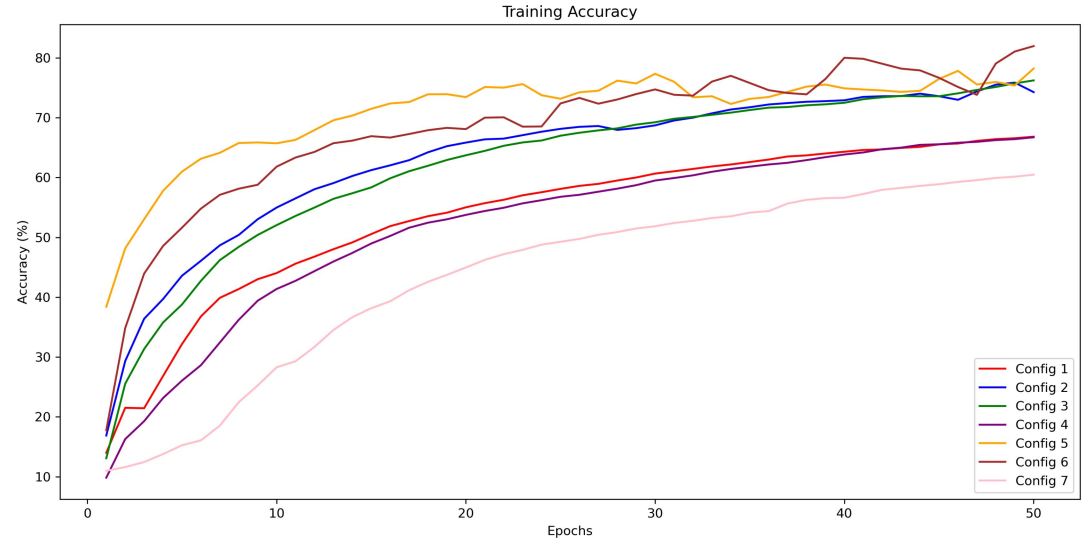
实验结果（Main Results）： 训练后得到的相应的模型权重文件，训练过程 training_result 的 json 文件,训练过程中 loss,训练集上准确率和测试集上准确率的变化图像存储在 modelset/original_version_model_multiGPU 文件夹中。**发现测试集上准确率**

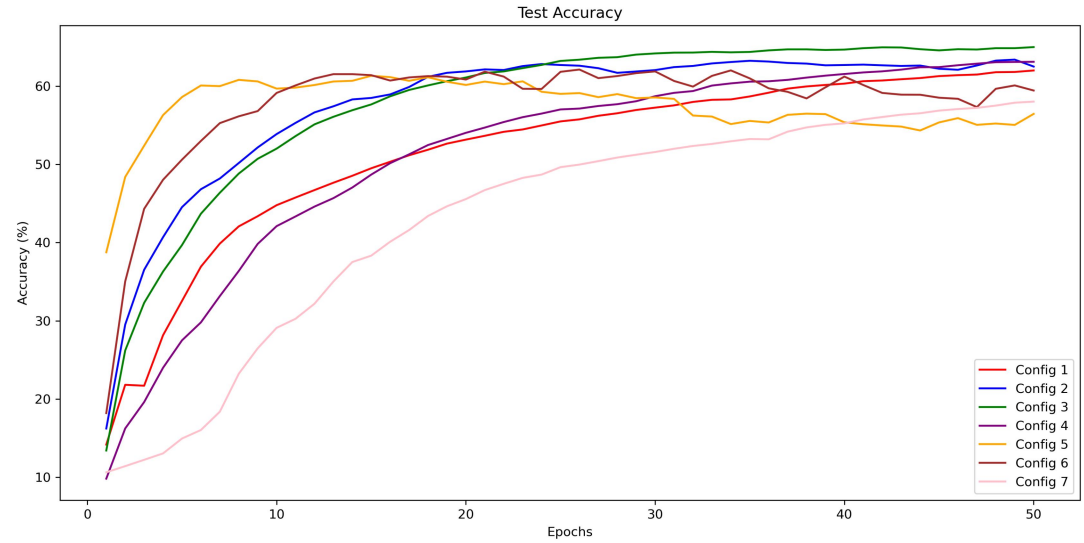
最高可以达到65.75%（num_epochs=200,lr=0.001,momentum=0.95）；另外根据绘出的图观察到测试集上准确率随训练过程趋于一致，落在55%--70%的区间，继续调整超参数对模型预测准确率的提高并不明显。

```
Training completed. Results summary:

config_1:
Configuration: {'num_epochs': 50, 'lr': 0.001, 'momentum': 0.9}
Final test accuracy: 47.63%

config_2:
Configuration: {'num_epochs': 100, 'lr': 0.01, 'momentum': 0.9}
Final test accuracy: 60.25%

config_3:
Configuration: {'num_epochs': 200, 'lr': 0.001, 'momentum': 0.95}
Final test accuracy: 65.75%
```

```
Training completed. Results summary:

config_1:
Configuration: {'num_epochs': 50, 'lr': 0.001, 'momentum': 0.9, 'batch_size': 32}
Final test accuracy: 62.00%
Training time: 263.94 seconds

config_2:
Configuration: {'num_epochs': 50, 'lr': 0.005, 'momentum': 0.9, 'batch_size': 64}
Final test accuracy: 62.50%
Training time: 181.47 seconds

config_3:
Configuration: {'num_epochs': 50, 'lr': 0.001, 'momentum': 0.95, 'batch_size': 32}
Final test accuracy: 64.99%
Training time: 259.07 seconds

config_4:
Configuration: {'num_epochs': 50, 'lr': 0.001, 'momentum': 0.95, 'batch_size': 64}
Final test accuracy: 63.11%
Training time: 180.19 seconds

config_5:
Configuration: {'num_epochs': 50, 'lr': 0.01, 'momentum': 0.9, 'batch_size': 32}
Final test accuracy: 56.45%
Training time: 260.93 seconds

config_6:
Configuration: {'num_epochs': 50, 'lr': 0.01, 'momentum': 0.9, 'batch_size': 64}
Final test accuracy: 59.44%
Training time: 181.00 seconds

config_7:
Configuration: {'num_epochs': 50, 'lr': 0.01, 'momentum': 0.9, 'batch_size': 512}
Final test accuracy: 58.02%
Training time: 163.23 seconds
```







3. 针对 Task2,分别试验 **L2 正则化和 Dropout 正则化方法**，在 Original_version_model_multiGPU.py 基础上修改已加入正则化功能，将修改后的代码分别存储为 L2_regularization.py 和 Dropout_ regularization.py。

实验设置（Experimental Setup）：4 张 A100；数据集是 CIFAR-10 数据集；模型是正则化后的 Lenet；

L2 正则化：在配置中添加了 weight_decay 参数；在创建优化器时，将 weight_decay 参数传递给了 SGD 优化器：

```
optimizer = optim.SGD(net.parameters(), lr=lr, momentum=momentum, weight_decay=weight_decay)
```

这里的 weight_decay 参数实际上就是 L2 正则化系数 λ。当使用 weight_decay 时，优化器会在每次参数更新时自动将 L2 正则化项添加到损失函数中。具体来说，它会修改梯度更新规则如下：

```
w = w - learning_rate * (gradient + weight_decay * w)
```

这等效于在损失函数中添加了一个 L2 正则化项：

```
L = original_loss + (weight_decay/2) * ||w||^2
```

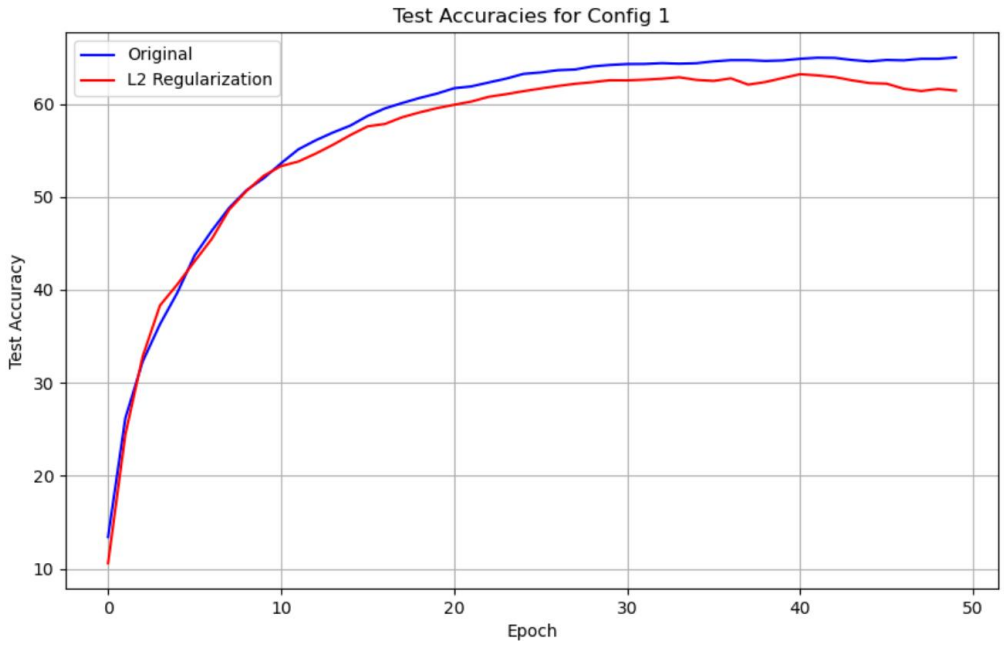
其中 ||w||^2 是模型参数的 L2 范数的平方。通过这种方式，L2 正则化鼓励模型学习更小的权重，从而减少过拟合的风险。weight_decay 的值（在这个例子中是 1e-4）控制了正则化的强度：值越大，正则化效果越强；值越小，正则化效果越弱。

Dropout 正则化：在全连接层之间应用了 Dropout，这会在训练过程中随机"丢弃"一部分神经元，防止网络过度依赖某些特征，从而减少过拟合。默认的 Dropout 率设置为 0.5：

```
def __init__(self, dropout_rate=0.5):
```

这意味着在每次前向传播时，有 50% 的神经元会被随机关闭。

实验结果（Main Results）：训练后得到的相应的模型权重文件，训练过程 training_result 的 json 文件存储在 modelset/L2_regularization_model 文件夹和 modelset\Dropout_regularization 中。**根据验证两种正则化方法都有助于预防过拟合；但会略微降低准确率（下图为 num_epochs：50，lr：0.001，momentum：0.95）。**



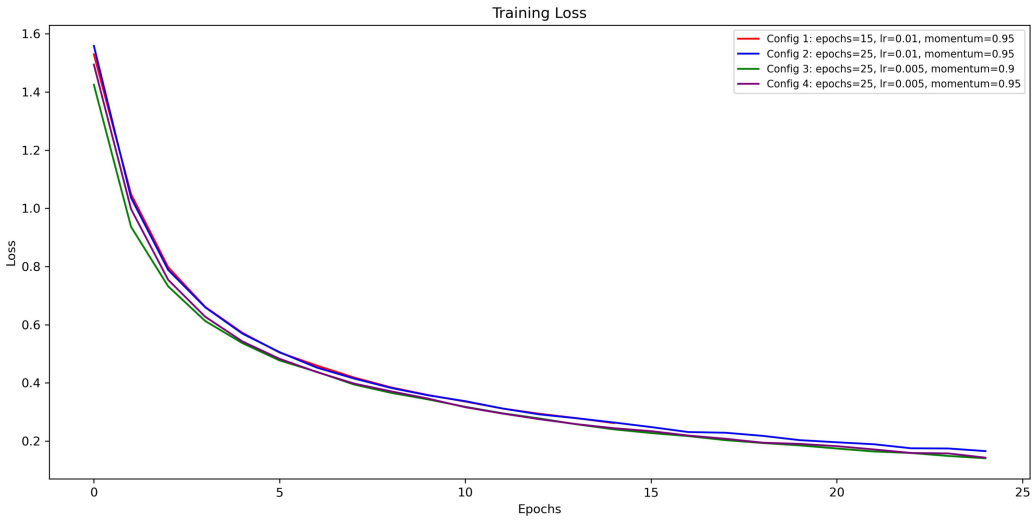
4.针对 Task4,借鉴 Resnet,实现了一个 18 层的简化版本的 Resnet,**观察分类准确率和训练时间，**并且也设置了不同的超参数 config，调整 num_epochs,lr,momentum 这些参数，尽可能提高模型分类的性能。

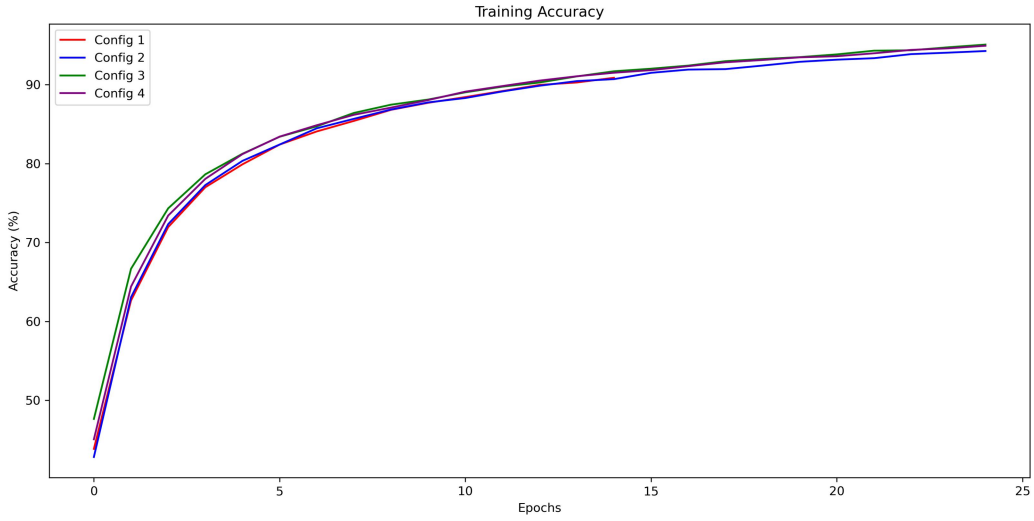
实验设置（Experimental Setup）：1 张 A100；数据集是 CIFAR-10 数据集；模型是 18 层的简化版本的 Resnet；超参数设置如下图。

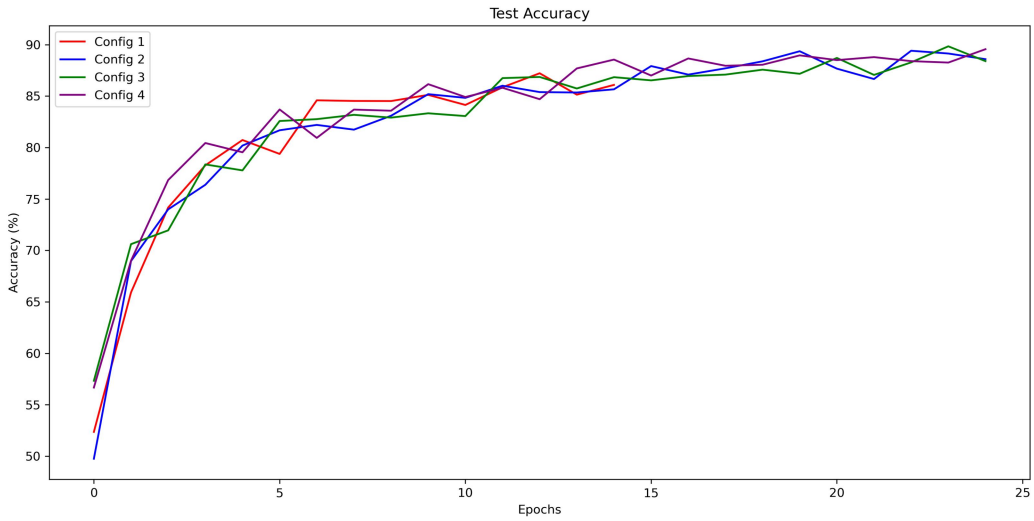
实验结果（Main Results）：训练后得到的相应的模型权重文件，训练过程 Resnet_model_result 的 json 文件,训练过程中 loss,训练集上准确率和测试集上准确率的变化图像存储在 modelset/Resnet 文件夹中。发现测试集上准确率最高可以达到 89.56%（num_epochs=25,lr=0.005,momentum=0.95）。

模型选择的重要性大于对模型参数调整的重要性，实验中选择 Resnet 后效果明显好于 Lenet。另外训练时间方面，由于使用 GPU 数量不同，直接比较单个模型最终训练时间意义不大，但 1 张 A100 上的 Resnet 与 4 张 A100 上的 Lenet 达到相同准确率时时间接近甚至更短，初步说明使用 Resnet 在模型准确率和训练时间方面均具有优越性。

```
Training completed. Results summary:

config_1:
Configuration: {'epochs': 15, 'lr': 0.01, 'momentum': 0.95}
Final test accuracy: 86.09%
Training time: 186.00 seconds

config_2:
Configuration: {'epochs': 25, 'lr': 0.01, 'momentum': 0.95}
Final test accuracy: 88.59%
Training time: 300.44 seconds

config_3:
Configuration: {'epochs': 25, 'lr': 0.005, 'momentum': 0.9}
Final test accuracy: 88.40%
Training time: 298.37 seconds

config_4:
Configuration: {'epochs': 25, 'lr': 0.005, 'momentum': 0.95}
Final test accuracy: 89.56%
Training time: 305.84 seconds
```







五、总结（Conclusion）

通过实验，初步明确了神经网络一般调优策略，认识到模型选择的重要性大于对模型参数调整的重要性，并且积累了在单卡和多卡上训练模型的经验。

参考资料：

1 有关 SGD 优化器的文档
https://pytorch.org/docs/stable/generated/torch.optim.SGD.html#sgd
2 有关 Dropout 层的文档
https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html#dropout
3 动手学深度学习 Resnet
https://zh.d2l.ai/chapter_convolutional-modern/index.html