从零开始的三层神经网络图像分类器建构

——基于卷积神经网络

CV Assignment 1 Report

姓 名: 黄 亦 绪
学 号: 23307110412
DATA130051.01 计算机视觉
(春季, 2025)

复旦大学
大数据学院

2025 年 4 月 13 日

目 录

1 引言 **5**

1.1 实验背景 5

1.2 实验环境：GPU 5

1.3 实验目标 6

2 数据集 **7**

2.1 数据集简介 7

2.2 数据预处理 7

2.2.1 数据加载 7

2.2.2 数据维度重构 7

2.2.3 像素值归一化 7

2.2.4 数据随机打乱 8

2.2.5 数据划分 8

3 模型设计 **9**

3.1 网络结构 9

3.2 激活函数 10

3.2.1 ReLU 激活函数 (`relu`) 10

3.2.2 Softmax 激活函数 (`softmax`) 10

3.3 卷积层 (`conv_forward`, `conv_backward`) 11

3.4 最大池化层 (`maxpool_forward`, `maxpool_backward`) 12

3.5 批量归一化 (`batch_norm_forward`, `batch_norm_backward`) . 13

3.6 全连接层 (`fc_forward`, `fc_backward`) 15

3.7 初始化 16

4 训练过程 **17**

4.1 优化器 17

4.2 损失函数 17

4.3 训练技巧 19

4.3.1 Dropout 19

4.3.2 Early Exiting 19

4.3.3 梯度裁剪 20

4.4 模型保存 20

5 参数查找与调优 **21**
5.1 超参数 21
5.1.1 学习率 21
5.1.2 正则化强度 22
5.1.3 隐藏层尺寸 22
5.1.4 Dropout 率 22
5.2 搜索策略 23
5.2.1 学习率、正则化强度和隐藏层大小的网格搜索 23
5.2.2 Dropout 率的随机搜索 23
5.3 调优结果与可视化 23

6 实验结果 **26**
6.1 训练过程可视化 26
6.1.1 Loss 曲线 26
6.1.2 Accuracy 曲线 26
6.2 测试集性能 27
6.3 模型最优网格参数可视化 28
6.4 模型输出特征可视化 32

7 讨论与分析 **33**
7.1 模型优势 33
7.1.1 高效的 GPU 加速计算 33
7.1.2 模块化与灵活的网络架构 33
7.1.3 批归一化提升训练稳定性 34
7.1.4 鲁棒的正则化机制 34
7.1.5 指数学习率与早停策略 34
7.1.6 全面的超参数调优支持 35
7.1.7 精确的梯度裁剪 35
7.2 不足 35
7.2.1 计算效率较低 35
7.2.2 模型容量有限 35
7.2.3 缺乏数据增强 35
7.2.4 优化器局限性 35
7.3 改进方向 36
7.3.1 增加卷积层深度 36
7.3.2 采用 Adam 优化器 36
7.3.3 引入数据增强技术 36

8 结论 **37**
8.1 模型结构和参数 37
8.2 超参数选择 37
8.3 优化策略 38
8.4 训练流程其他参数 38

9 代码与资源 **39**
9.1 Github 链接 39
9.2 模型权重 39

1 引言

1.1 实验背景

卷积神经网络（CNN）是图像分类领域的核心技术，通过卷积层提取局部特征、池化层降低维度并增强平移不变性，最终通过全连接层完成分类任务。自 LeNet 首次应用于手写数字识别以来，CNN 经历了 AlexNet 、VGGNet 、GoogLeNet 和 ResNet 等关键模型的发展，逐步提升网络深度和特征提取能力，显著提高了分类准确率。CNN 在医学、遥感以及自然场景图像等领域表现出色，能够自动学习图像的分层特征，对大规模数据集具有良好的适应性，并在平移、缩放和旋转等变换下保持鲁棒性。其强大的特征提取能力和广泛的适用性使其成为现代计算机视觉研究和应用的基石。

此次作业中，我将从零开始是用 `python` 手工实现 CNN 。其意义在于深入理解神经网络的数学原理和优化过程，掌握梯度计算、链式法则以及权重更新的核心机制。由此能够学会验证深度学习框架的正确性，调试模型中的异常问题，并针对特定需求优化性能。此外，这一过程还能提高我的编程能力，锻炼数学公式代码化的技巧，同时培养系统性思维，帮助更好地设计和优化复杂模型。

1.2 实验环境：GPU

GPU 加速的优势在于计算加速、减少数据传输开销、提升训练效率以及支持大规模深度学习任务，它为高效处理复杂数据集和模型提供了强有力的支持。它在我们 CNN 的训练中也发挥了关键作用，主要通过 `CuPy` 库实现。在处理 CIFAR-10 数据集时，数据加载和预处理直接在 GPU 上完成，减少了 CPU 与 GPU 之间的数据传输瓶颈。模型计算中的卷积、最大池化（Maxpooling）、全连接（FC）和批量归一化（BN）等操作，利用 GPU 的并行计算能力显著提升了效率。训练过程中的前向传播、反向传播和参数更新均在 GPU 上执行，避免了不必要的数据传输，进一步加速了训练。

本次我使用的 GPU 是 Nvidia 的 L20 ， `Driver Version: 535.183.01, CUDA Version: 12.2` 。由于内存限制和训练速度的双重要求，多次实验后，我选择在代码中并用 `cp.tensordot` 和 `cp.max` 等函数（速度要求）以及 `for` 循环和 `cp.get_default_memory_pool().free_all_blocks()` 函数（GPU 内存节省）进行卷积和池化的前向和反向传播。

1.3 实验目标

现有深度学习框架 `pytorch` 简单实现的三层 CNN 预期达到 72.84 % 的测试准确率(`lr=0.005, reg=0.001`), 如下图所示。据此可以设立目标, **我希望从零开始实现一个三层 CNN 图像分类器，测试准确率目标为 65%-70%**。

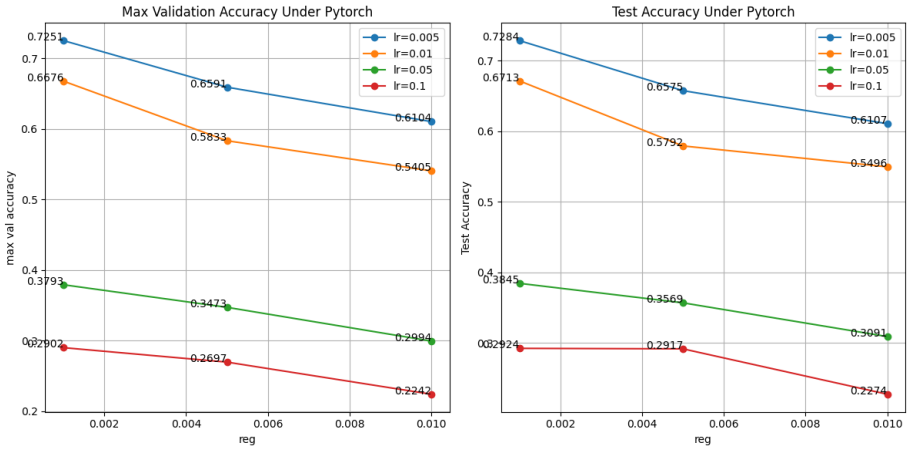

图 1: pytorch 三层 CNN 实现效果

2 数据集

2.1 数据集简介

CIFAR-10[1] 数据集由加拿大高级研究所（CIFAR）创建，包含 60,000 张 RGB 格式（$3 \times 32 \times 32$）的彩色图像，分为 10 个类别，每个类别有 6000 张图像。这些图像涵盖了日常生活中常见的物体，如飞机、汽车、鸟类、猫、鹿、狗、青蛙、马、船和卡车等。数据集被分为训练集和测试集，分别包含 50000 张和 10000 张图像，用于模型的训练和评估。CIFAR-10 数据集的特点是图像尺寸较小，但类别多样，这使得它成为研究简单的视觉分类问题的理想选择。它广泛应用于图像分类任务的训练与测试。

2.2 数据预处理

2.2.1 数据加载

CIFAR-10 数据集以批次形式存储，每个批次包含 10,000 张图像。我们依次加载每个批次的数据。

2.2.2 数据维度重构

为适配卷积神经网络的输入格式，加载的 CIFAR-10 数据在预处理时进行了维度重构。原始数据为一维向量（3072 维，包含 $32 \times 32 \times 3$ 的像素值），重构后转换为四维张量：

$$\mathbf{x}_{\text{reshaped}} \in \mathbb{R}^{N \times C \times H \times W},$$

其中 N 为批次大小，$C = 3$ 为颜色通道数，$H = 32$、$W = 32$ 分别为图像高度和宽度。此重构确保数据与卷积操作兼容，同时保留空间结构，便于特征提取和后续池化处理。

2.2.3 像素值归一化

我通过将加载的数据集的像素值做以下处理，

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x}}{255},$$

将这些像素值归一化到 0,1 范围内。归一化有助于加速模型收敛并提高训练稳定性。

2.2.4 数据随机打乱

为了打破数据的原始顺序，避免模型对数据顺序的依赖，我在加载数据后对数据集进行了随机打乱。这样的随机打乱能确保数据分布的均匀性。

2.2.5 数据划分

CIFAR-10 原始文件中，训练集和测试集已经分开存储。训练集包含 50,000 张图像，测试集包含 10,000 张图像。我进一步将训练集划分为训练集和验证集，比例为 80% 和 20% 。训练集用于训练模型，使之学习数据中的模式、规律和特征；验证集用于评估模型的泛化能力，并在训练过程中用于超参数调整和模型选择。

3 模型设计

3.1 网络结构

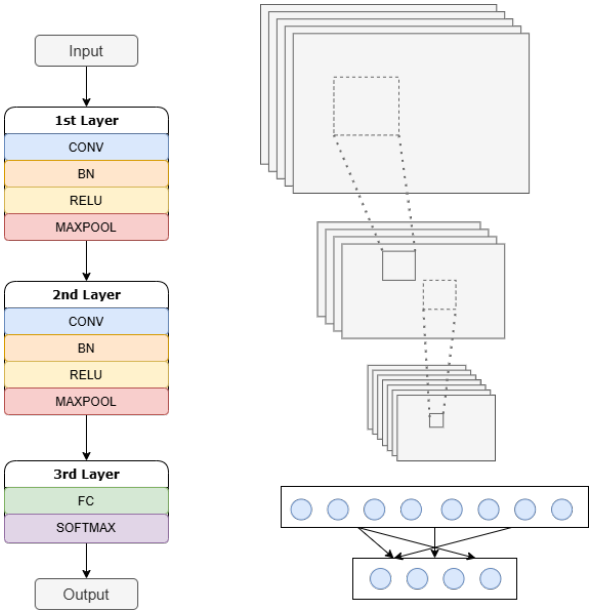

图 2: 网络结构

我的 3 层 CNN 的网络结构如图 2 所示，总结如下：

- **输入：** 输入维度为 $(n \times 3 \times 32 \times 32)$ （n 为批量大小，3 为输入通道数，32 为图像高度和宽度）
- **第一层卷积：** 使用 `num_filters1` 3×3 卷积核提取低级特征，经过 BN 和 ReLU 处理后，通过 2×2 池化减小尺寸，输出维度为 $(n\times32\times16\times16)$
- **第二层卷积：** 使用 `num_filters2` 个 3×3 卷积核提取更高级特征，经过 BN 和 ReLU 处理后，再次通过 2×2 池化进一步下采样，输出维度为 $(n \times 64 \times 8 \times 8)$
- **全连接层：** 将池化后的特征图展平，输入全连接层，输出 10 个类别的分类结果
- **输出：** 输出维度为 $(n \times 10)$（n 为批量大小，10 为类别数）

3.2 激活函数

3.2.1 ReLU 激活函数 (`relu`)

ReLU（Rectified Linear Unit）激活函数的公式为：

$$f(x) = \max(0, x),$$

其特点是将输入值小于 0 的部分置为 0，而大于 0 的部分保持不变。ReLU 以其简单而高效的设计在深度学习中展现出显著优势。它通过稀疏激活机制生成真实的零值，有效减少了计算冗余，同时缓解了梯度消失问题，使得深层网络的训练更加稳定和高效。实验结果表明，ReLU 在自然稀疏数据上的表现尤为突出，例如在图像分类和文本分析任务中均取得了优异的性能（Glorot et al., 2011）[2]。ReLU 在这里被置于卷积层和池化层之后，能进一步提升神经网络对 CIFAR-10 数据集特征的提取能力。

- **前向传播**：我们对输入张量 x 的每个元素应用 ReLU，保留大于 0 的值，负值置为 0 。最后输出与输入形状相同的张量，负值置零。
- **反向传播**：核心是计算 ReLU 的梯度，公式为：

  $$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \cdot \mathbb{1}(x > 0)$$

  我们根据输入 x 的符号，生成布尔掩码 $\mathbb{1}(x > 0)$，将上层梯度 $\frac{\partial L}{\partial y}$ 乘以掩码，仅传递正值部分的梯度。输出与输入形状相同的梯度张量。

3.2.2 Softmax 激活函数 (`softmax`)

Softmax 激活函数将全连接层的输出（logits）转换为概率分布，适用于多分类任务，其公式为

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}.$$

Softmax 确保输出的概率和为 1，能够直观地表示每个类别的相对置信度。在 CIFAR-10 任务中，Softmax 作为网络的最后一层，将特征映射到 10 个类别的概率分布，为分类决策提供了基础。Softmax 的设计通过指数化增强了模型对类别差异的敏感性，同时与交叉熵损失结合使用，能够有效优化模型参数。

- **前向传播**：其功能是将全连接层的输出转换为概率分布，用于多分类任务。首先通过输入为形状 (N, C) 的 logits（N 为样本数，C 为类别

数)。为了为数值稳定性，减去每行最大值：

$$\hat{x} = x - \max(x, \text{axis} = 1)$$

然后计算指数和归一化：

$$\text{softmax}(x)_i = \frac{\exp(\hat{x}_i)}{\sum_j \exp(\hat{x}_j)}$$

最终输出每个样本的概率分布，形状为 (N, C)，每行和为 1。

- **反向传播**：Softmax 的反向传播通常与交叉熵损失结合计算，在模型的 `backward` 方法中实现。

3.3 卷积层 (`conv_forward`, `conv_backward`)

卷积层是卷积神经网络的核心组件，通过滑动窗口机制提取输入图像的空间特征，如边缘、纹理等。卷积操作通过滤波器与输入数据的局部区域进行点积，生成特征图，能够有效捕捉图像的局部相关性，同时保持空间结构。卷积层的参数共享机制显著减少了计算量，使得深层网络在处理高维图像数据时更加高效。在 CIFAR-10 任务中，卷积层通过多通道滤波器提取多样化的特征，为后续的分类任务提供了丰富的表示能力（LeCun et al., 1998）[3].

- **前向传播**：

  输入为形状 (N, C, H, W) 的数据 x、形状 (F, C, FH, FW) 的卷积核 w 和形状 $(F,)$ 的偏置 b，其中 N 为样本数，C 为通道数，H 和 W 为高度和宽度，F 为滤波器数，FH 和 FW 为滤波器尺寸。若 $\text{pad} > 0$，对 x 进行零填充，得到 x_{padded}。输出尺寸为

$$H_{\text{out}} = \lfloor \frac{H + 2 \cdot \text{pad} - FH}{\text{stride}} \rfloor + 1$$
$$W_{\text{out}} = \lfloor \frac{W + 2 \cdot \text{pad} - FW}{\text{stride}} \rfloor + 1.$$

  遍历输出空间 (i, j)，提取输入窗口并使用 `CuPy` 的 `tensordot` 计算窗口与卷积核的点积：

$$x_{\text{padded}}[:, :, i \cdot \text{stride} : i \cdot \text{stride} + FH, j \cdot \text{stride} : j \cdot \text{stride} + FW]$$
$$\text{out}[:, :, i, j] = \sum_{c,fh,fw} x_{\text{padded}}[:, c, i \cdot \text{stride} + fh, j \cdot \text{stride} + fw] \cdot w[:, c, fh, fw] + b.$$

输出形状为 $(N, F, H_{\text{out}}, W_{\text{out}})$ 的特征图。

- **反向传播**:

  给定上层梯度 $\frac{\partial L}{\partial \text{out}} \in \mathbb{R}^{N\times F\times H_{\text{out}}\times W_{\text{out}}}$、原始输入 $\mathbf{x} \in \mathbb{R}^{N\times C\times H\times W}$ 和卷积核 $\mathbf{w} \in \mathbb{R}^{F\times C\times FH\times FW}$，计算以下梯度:

  – 偏置梯度:

$$\frac{\partial L}{\partial \mathbf{b}}[f] = \sum_{n=0}^{N-1}\sum_{i=0}^{H_{\text{out}}-1}\sum_{j=0}^{W_{\text{out}}-1}\frac{\partial L}{\partial \text{out}}[n, f, i, j].$$

  – 输入梯度:

  初始化 $\frac{\partial L}{\partial \mathbf{x}_{\text{padded}}} \in \mathbb{R}^{N\times C\times (H+2\cdot\text{pad})\times (W+2\cdot\text{pad})}$, 对输出空间 (i, j) 遍历:

$$\frac{\partial L}{\partial \mathbf{x}_{\text{padded}}}[n, c, i\cdot\text{stride} : i\cdot\text{stride}+FH, j\cdot\text{stride} : j\cdot\text{stride}+FW] \mathrel{+}= \sum_{f=0}^{F-1}\frac{\partial L}{\partial \text{out}}[n,f,i,j]\cdot\mathbf{w}[f, c, :, :].$$

  若 $\text{pad} > 0$，裁剪 $\frac{\partial L}{\partial \mathbf{x}_{\text{padded}}}$ 得到 $\frac{\partial L}{\partial \mathbf{x}} \in \mathbb{R}^{N\times C\times H\times W}$。

  – 权重梯度:

$$\frac{\partial L}{\partial \mathbf{w}}[f,c,fh,fw] = \sum_{n=0}^{N-1}\sum_{i=0}^{H_{\text{out}}-1}\sum_{j=0}^{W_{\text{out}}-1}\frac{\partial L}{\partial \text{out}}[n,f,i,j]\cdot\mathbf{x}_{\text{padded}}[n,c,i\cdot\text{stride}+fh, j\cdot\text{stride}+fw].$$

  – 输出梯度: $\frac{\partial L}{\partial \mathbf{x}}$、$\frac{\partial L}{\partial \mathbf{w}}$ 和 $\frac{\partial L}{\partial \mathbf{b}}$。

3.4 最大池化层 (`maxpool_forward`, `maxpool_backward`)

最大池化层通过选择局部区域的最大值进行下采样，减少空间维度，保留显著特征。其固定窗口和步幅操作增强了网络对平移、旋转等变换的不变性，降低计算复杂度和过拟合风险。在 CIFAR-10 任务中，最大池化层置于卷积层后，压缩特征图尺寸，突出关键特征。

- **前向传播**:

  给定输入特征图 $\mathbf{x} \in \mathbb{R}^{N\times C\times H\times W}$、池化窗口大小 pool_size 和步幅

stride，输出尺寸为：

$$H_{\text{out}} = \left\lfloor \frac{H - \text{pool\_size}}{\text{stride}} \right\rfloor + 1,$$
$$W_{\text{out}} = \left\lfloor \frac{W - \text{pool\_size}}{\text{stride}} \right\rfloor + 1.$$

对于输出空间的每个位置 (i, j)，提取池化窗口：

$$\mathbf{x}_{\text{window}} = \mathbf{x}[:, :, i \cdot \text{stride} : i \cdot \text{stride} + \text{pool\_size}, j \cdot \text{stride} : j \cdot \text{stride} + \text{pool\_size}],$$

计算最大值：

$$\text{out}[n, c, i, j] = \max_{fh=0,\ldots,\text{pool\_size}-1,\, fw=0,\ldots,\text{pool\_size}-1} \mathbf{x}[n, c, i \cdot \text{stride} + fh, j \cdot \text{stride} + fw],$$

并记录最大值位置生成掩码 $\text{mask} \in \mathbb{R}^{N \times C \times H \times W}$，其中最大值位置为 1，其余为 0。输出特征图 $\text{out} \in \mathbb{R}^{N \times C \times H_{\text{out}} \times W_{\text{out}}}$ 和掩码 mask。

- **反向传播**：

  给定上层梯度 $\frac{\partial L}{\partial \text{out}} \in \mathbb{R}^{N \times C \times H_{\text{out}} \times W_{\text{out}}}$、原始输入 $\mathbf{x} \in \mathbb{R}^{N \times C \times H \times W}$ 和掩码 mask，初始化输入梯度 $\frac{\partial L}{\partial \mathbf{x}} \in \mathbb{R}^{N \times C \times H \times W}$。对输出空间 (i, j) 遍历：

  $$\frac{\partial L}{\partial \mathbf{x}}[:, :, i \cdot \text{stride} : i \cdot \text{stride} + \text{pool\_size}, j \cdot \text{stride} : j \cdot \text{stride} + \text{pool\_size}] += \frac{\partial L}{\partial \text{out}}[:, :, i, j] \cdot \text{mask}[:,$$

  输出输入梯度 $\frac{\partial L}{\partial \mathbf{x}}$。

3.5 批量归一化 (`batch_norm_forward`, `batch_norm_backward`)

批量归一化通过标准化每层输入，加速训练并提高稳定性。其对数据进行均值和方差归一化，并引入可学习参数 γ 和 β：

$$\text{out} = \gamma \cdot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta.$$

这个操作能够减少内部协变量偏移，提升网络对参数和学习率的鲁棒性，并缓解梯度消失问题。在我们的任务中，BN 置于卷积层后，能提升收敛速度和分类精度。

- **前向传播**：给定输入特征图 $\mathbf{x} \in \mathbb{R}^{N \times C \times H \times W}$、缩放参数 $\gamma \in \mathbb{R}^C$、偏移参数 $\beta \in \mathbb{R}^C$、防止除零的 ϵ、运行均值和方差的动量 momentum 以及训练标志 training。重塑 $\mathbf{x}$ 为 $\mathbf{x}_{\text{flat}} \in \mathbb{R}^{(N \cdot H \cdot W) \times C}$。在训练模式下：

– 计算批次均值和方差：

$$\mu[c] = \frac{1}{N \cdot H \cdot W} \sum_{n=0}^{N-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} \mathbf{x}[n, c, h, w],$$
$$\sigma^2[c] = \frac{1}{N \cdot H \cdot W} \sum_{n=0}^{N-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} \left(\mathbf{x}[n, c, h, w] - \mu[c]\right)^2.$$

– 归一化并缩放：

$$\hat{\mathbf{x}} = \frac{\mathbf{x}_{\text{flat}} - \mu}{\sqrt{\sigma^2 + \epsilon}},$$
$$\text{out}_{\text{flat}} = \gamma \cdot \hat{\mathbf{x}} + \beta.$$

– 更新运行均值和方差：

$$\text{running\_mean} = \text{momentum} \cdot \text{running\_mean} + (1 - \text{momentum}) \cdot \mu,$$
$$\text{running\_var} = \text{momentum} \cdot \text{running\_var} + (1 - \text{momentum}) \cdot \sigma^2.$$

– 存储 μ、σ^2、$\hat{\mathbf{x}}$ 至 cache。

在测试模式下：

$$\hat{\mathbf{x}} = \frac{\mathbf{x}_{\text{flat}} - \text{running\_mean}}{\sqrt{\text{running\_var} + \epsilon}},$$
$$\text{out}_{\text{flat}} = \gamma \cdot \hat{\mathbf{x}} + \beta.$$

重塑 out_{flat} 为 $\text{out} \in \mathbb{R}^{N \times C \times H \times W}$，输出归一化特征图和 cache。

- **反向传播**：给定上层梯度 $\frac{\partial L}{\partial \text{out}} \in \mathbb{R}^{N \times C \times H \times W}$、缩放参数 γ 和 cache（包含 μ、σ^2、$\hat{\mathbf{x}}$、$\mathbf{x}$）。重塑 $\frac{\partial L}{\partial \text{out}}$ 为 $\frac{\partial L}{\partial \text{out}_{\text{flat}}} \in \mathbb{R}^{(N \cdot H \cdot W) \times C}$。计算：

  – 参数梯度：

$$\frac{\partial L}{\partial \gamma}[c] = \sum_{k=0}^{N \cdot H \cdot W - 1} \frac{\partial L}{\partial \text{out}_{\text{flat}}}[k, c] \cdot \hat{\mathbf{x}}[k, c],$$
$$\frac{\partial L}{\partial \beta}[c] = \sum_{k=0}^{N \cdot H \cdot W - 1} \frac{\partial L}{\partial \text{out}_{\text{flat}}}[k, c].$$

– 输入梯度：

$$\begin{aligned}
\frac{\partial L}{\partial \hat{\mathbf{x}}}[k,c] &= \frac{\partial L}{\partial \text{out}_{\text{flat}}}[k,c] \cdot \gamma[c], \\
\frac{\partial L}{\partial \sigma^2}[c] &= \sum_{k=0}^{N\cdot H\cdot W-1} \frac{\partial L}{\partial \hat{\mathbf{x}}}[k,c] \cdot (\mathbf{x}_{\text{flat}}[k,c] - \mu[c]) \cdot \left(-\frac{1}{2}\right) \left(\sigma^2[c] + \epsilon\right)^{-3/2}, \\
\frac{\partial L}{\partial \mu}[c] &= \sum_{k=0}^{N\cdot H\cdot W-1} \frac{\partial L}{\partial \hat{\mathbf{x}}}[k,c] \cdot \left(-\frac{1}{\sqrt{\sigma^2[c]+\epsilon}}\right) + \frac{\partial L}{\partial \sigma^2}[c] \cdot \frac{-2\sum_{k=0}^{N\cdot H\cdot W-1} (\mathbf{x}_{\text{flat}}[k,c] - \mu[c])}{N\cdot H\cdot W}, \\
\frac{\partial L}{\partial \mathbf{x}_{\text{flat}}}[k,c] &= \frac{\frac{\partial L}{\partial \hat{\mathbf{x}}}[k,c]}{\sqrt{\sigma^2[c]+\epsilon}} + \frac{\partial L}{\partial \sigma^2}[c] \cdot \frac{2\,(\mathbf{x}_{\text{flat}}[k,c] - \mu[c])}{N\cdot H\cdot W} + \frac{\frac{\partial L}{\partial \mu}[c]}{N\cdot H\cdot W}.
\end{aligned}$$

重塑 $\frac{\partial L}{\partial \mathbf{x}_{\text{flat}}}$ 为 $\frac{\partial L}{\partial \mathbf{x}} \in \mathbb{R}^{N\times C\times H\times W}$，输出梯度 $\frac{\partial L}{\partial \mathbf{x}}$、$\frac{\partial L}{\partial \gamma}$ 和 $\frac{\partial L}{\partial \beta}$。

3.6 全连接层 (`fc_forward`, `fc_backward`)

全连接层将卷积和池化层的特征展平后映射到类别空间，生成分类任务的 logits。其通过矩阵乘法和偏置加法实现高维特征到低维类别的转换，适用于 CIFAR-10 的 10 类分类任务。全连接层整合全局特征，与 Softmax 结合输出分类概率。

- **前向传播**：

  给定展平特征 $\mathbf{x} \in \mathbb{R}^{N\times D}$、权重矩阵 $\mathbf{w} \in \mathbb{R}^{D\times C}$ 和偏置 $\mathbf{b} \in \mathbb{R}^{C}$，其中 D 为展平维度，C 为类别数。计算：

$$\text{out} = \mathbf{x}\mathbf{w} + \mathbf{b},$$

  输出 logits $\text{out} \in \mathbb{R}^{N\times C}$。

- **反向传播**：

  给定上层梯度 $\frac{\partial L}{\partial \text{out}} \in \mathbb{R}^{N\times C}$、原始输入 $\mathbf{x} \in \mathbb{R}^{N\times D}$ 和权重 $\mathbf{w} \in \mathbb{R}^{D\times C}$，计算：

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{x}} &= \frac{\partial L}{\partial \text{out}} \mathbf{w}^\top, \\
\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{x}^\top \frac{\partial L}{\partial \text{out}}, \\
\frac{\partial L}{\partial \mathbf{b}}[c] &= \sum_{n=0}^{N-1} \frac{\partial L}{\partial \text{out}}[n,c].
\end{aligned}$$

  输出梯度 $\frac{\partial L}{\partial \mathbf{x}}$、$\frac{\partial L}{\partial \mathbf{w}}$ 和 $\frac{\partial L}{\partial \mathbf{b}}$。

3.7 初始化

权重初始化是深度神经网络训练的重要环节，直接影响模型的收敛速度、稳定性和最终性能。以下是一些常见的初始化方法。

表 1: 初始化方法对比

初始化方法	激活函数	优点	缺点
He 初始化	ReLU 类	针对 ReLU 激活函数设计，能够缓解梯度消失问题；保持输入和输出的方差一致，加速网络收敛；适用于深度网络，尤其在 ReLU 或其变体（如 Leaky ReLU）中表现良好。	不适合 Sigmoid 或 Tanh 激活函数，可能导致梯度消失或梯度过大；需要正确设置权重的方差，否则可能影响训练效果。
无初始化（随机初始化）	几乎所有	简单易实现，无需额外计算；在某些情况下可能足够有效，尤其是在浅层网络中。	容易导致梯度消失或梯度爆炸，尤其是在深度网络中；训练过程可能不稳定，收敛速度慢；随机性可能导致不同初始化结果差异较大。
其他初始化（如 Xavier）	Sigmoid、Tanh	针对 Sigmoid 和 Tanh 激活函数设计，能够缓解梯度消失问题；保持输入和输出的方差一致，适合浅层网络；在 Sigmoid 和 Tanh 激活函数中表现良好。	不适合 ReLU 激活函数，可能导致神经元死亡问题；在深度网络中可能效果不如 He 初始化。

在我的模型中，初始化使用的方法是专为 ReLU 激活函数设计的，基于 He 初始化的变种。He 初始化由 He 等人于 2015 年提出，针对 ReLU 激活函数的特点优化了权重的方差 [4]。He 初始化将 ReLU 函数的负值截断为零，导致大约一半的神经元输出为零。为了维持信号在网络传播中的稳定性，权重的初始化需要确保每一层的输出方差接近输入方差。

从图 3 可见，进行了 He 初始化的卷积层的激活方差不会像没有初始化的那样过度趋近于 0 （左图），且不会发生突跃（右图）。由此可见，He 初始化方法能够起到加速收敛的目的，避免了信号过大（导致梯度爆炸）或过小（导致梯度消失）的问题，能减少模型对超参数的敏感性。

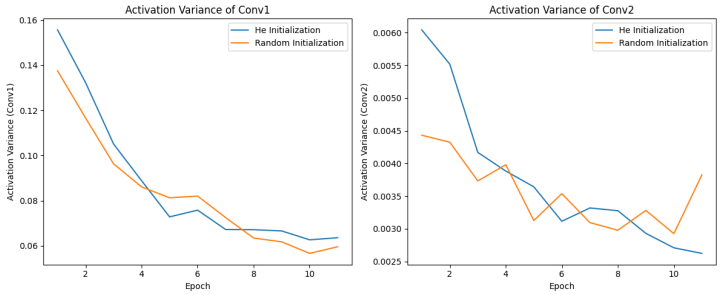

图 3: He 初始化于无初始化的比较

4 训练过程

训练过程旨在优化卷积神经网络的参数，使其在 CIFAR-10 数据集上实现高分类准确率。通过选择合适的优化器和损失函数，使 CNN 能够在训练中有效收敛并泛化到验证集和测试集。调试超参数，选择最好的模型保存。

4.1 优化器

采用随机梯度下降（Stochastic Gradient Descent, SGD）作为优化器，以最小化损失函数。SGD 的更新规则基于梯度下降，每次迭代利用小批量数据估计梯度，平衡计算效率与优化稳定性。

4.2 损失函数

在计算效率、准确性的标注下筛选不同策略后（如表 2 所示），我选择引入指数学习率衰减策略，衰减因子为 decay_rate（$0 \leqslant \text{decay\_rate} \leqslant 1$，我取了常用的 $decay\_rate = 0.995$），初始学习率为 $\eta_0 = \text{lr}$ 。具体而言，每轮（epoch）后，学习率按以下公式更新：

$$\eta_t = \eta_0 \cdot \text{decay\_rate}^t,$$

其中 t 为当前轮数。这种衰减策略使学习率随训练逐渐减小，有助于在训练后期精细调整参数，避免震荡并提升收敛质量。在我的任务中，SGD 能够有效优化网络参数，提升分类性能。

采用交叉熵损失（Cross-Entropy Loss）作为分类任务的主损失函数，结合 L2 正则化以防止过拟合。对于 N 个样本的批量，真实标签为 $\mathbf{y} \in \{0, 1, \ldots, C-1\}^N$（$C = 10$ 为 CIFAR-10 的类别数），模型输出的 logits 为

表 2: 不同学习率更新策略及其好处

策略	**公式**	**好处**
固定学习率	$\eta_t = \eta_0$	1. 简单易实现，无需动态调整。2. 适合初期快速实验或简单任务。3. 早期梯度下降稳定，适合平滑损失函数。
步长衰减	$\eta_t = \eta_0 \cdot \gamma^{\lfloor t/s \rfloor}$	1. 阶段性优化，早期快速逼近低谷，后期精细调整。2. 提高收敛稳定性，避免震荡。3. 适合长时间训练或复杂模型。
指数学习率衰减	$\eta_t = \eta_0 \cdot e^{-kt}$	1. 平滑衰减，避免突变，适合连续优化。2. 早期快速学习，后期稳定收敛，平衡速度与精度。3. 适应动态任务，易于调参。
余弦退火	$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{t}{T}\pi))$	1. 周期性调整，探索多种学习率，防止陷入局部最优。2. 适合复杂模型，提升泛化能力。3. 后期低学习率确保精细优化。
自适应学习率（如 Adam）	$\eta_t = \frac{\eta_0}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t$	1. 根据梯度自适应调整，适合稀疏或噪声数据。2. 加速收敛，减少调参工作。3. 对初始学习率不敏感，鲁棒性强。

$\mathbf{z} \in \mathbb{R}^{N \times C}$，交叉熵损失定义为：

$$L_{\text{CE}} = -\frac{1}{N} \sum_{n=0}^{N-1} \log \left(\frac{\exp(\mathbf{z}[n, \mathbf{y}[n]])}{\sum_{c=0}^{C-1} \exp(\mathbf{z}[n, c])} \right).$$

L2 正则化项对模型权重 $\mathbf{w}$（包含所有卷积核和全连接层权重）施加惩罚，系数为 $\lambda = \text{reg}$，定义为：

$$L_{\text{reg}} = \frac{\lambda}{2} \sum_{\mathbf{w}} \|\mathbf{w}\|_2^2.$$

总损失函数为：

$$L = L_{\text{CE}} + L_{\text{reg}}.$$

交叉熵损失引导模型学习正确的类别分布，而 L2 正则化限制权重幅度，能惩罚较大的权重值并鼓励较小的权重，从而防止任何一种权重主导模型增强模型泛化能力。适当的正则化系数 λ 可有效平衡训练准确率与验证集性能。

4.3 训练技巧

4.3.1 Dropout

Dropout 是一种正则化技术，通过在训练时以概率 p 随机丢弃全连接层的神经元输出，减少神经元之间的共适应属性，从而提升模型泛化能力 [5]。在全连接层前向传播中，Dropout 操作为：

$$\mathbf{x}_{\text{drop}} = \mathbf{x} \cdot \mathbf{m} / (1 - p),$$

其中 $\mathbf{x}$ 为输入特征，$\mathbf{m}$ 为 Bernoulli 分布的掩码（以概率 $1 - p$ 保留神经元），p 为 Dropout 率。实验中以 p 的概率丢弃神经元输出，并通过缩放因子 $1/(1 - p)$ 补偿输出幅度。

4.3.2 Early Exiting

Early exiting（早停）是一种防止过拟合的机制。它通过在训练过程中监控验证集的性能（如验证准确率或验证损失），如果验证性能在一定数量的训练轮次内没有改善，就提前停止训练。这种机制可以避免模型在训练集上过拟合，同时节省计算资源 [6]。

在我的代码中，`patience`（表示在验证性能没有改善的情况下，允许的最大连续轮数）被设置为 5（默认值）。即，如果验证准确率在连续 5 个 epoch 中没有提高，训练就会停止。

4.3.3 梯度裁剪

梯度爆炸会导致模型的训练过程剧烈波动，甚至发散。通过梯度裁剪，可以稳定训练过程，使模型更快地收敛到最优解。

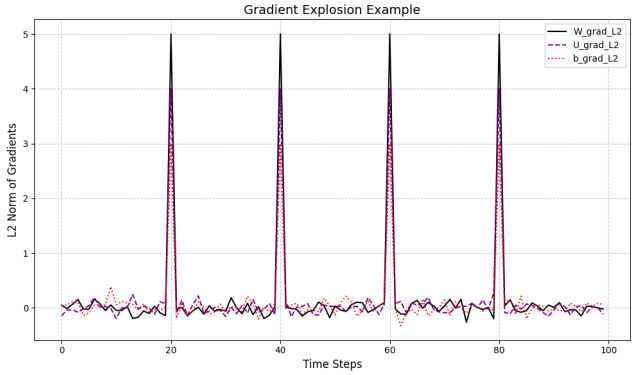

图 4: 梯度爆炸示意图

权重更新时引入梯度裁剪（`max_grad_norm=5.0`），通过检查梯度范数并缩放，防止梯度爆炸。公式如下：

$$\mathbf{g}_{\text{clipped}} = \begin{cases} \mathbf{g} \cdot \frac{\theta}{\|\mathbf{g}\|_2}, & \text{if } \|\mathbf{g}\|_2 > \theta, \\ \mathbf{g}, & \text{otherwise}, \end{cases}$$

我将裁剪应用于所有权重和批归一化参数（`dw1`, `dgamma1` 等），增强深层网络训练的稳定性。

4.4 模型保存

保留训练过程中的最佳模型的方法是比较验证集准确率。每轮训练结束后，计算模型在验证集上的分类准确率，记录当前最高准确率及对应的权重参数。若当前 epoch 的验证准确率高于历史最高值，则更新最佳权重，不同参数组合的模型比较历史最高验证准确率，选出最佳模型并存储为 `.npz` 文件，文件包含所有可训练参数（如卷积核 $\mathbf{w}$、偏置 $\mathbf{b}$、批量归一化的 γ 和 β 等）。如此可以确保模型在泛化性能最佳时被记录，适用于后续权重矩阵可视化和测试。

5 参数查找与调优

为提升卷积神经网络在 CIFAR-10 数据集上的分类性能，系统性地进行了超参数查找与调优。超参数的选择直接影响模型的收敛速度、泛化能力和最终准确率。通过网格搜索（Grid Search）方法，探索了学习率、正则化强度和隐藏层大小的最优组合，并检查了 Dropout 率的合理取值范围，以优化模型性能并避免过拟合。

在实验过程中，过拟合（Overfitting）是一个非常显著的问题。对于图像分类器，过拟合会降低分类的效果，如图所示。一般来说，过拟合表现在模型在训练集和验证集上表现很好，在测试集上表现很差，就像模型“记住”了训练标签一样。我们调参很大程度上就是为了减少过拟合，以提升模型的泛化能力。

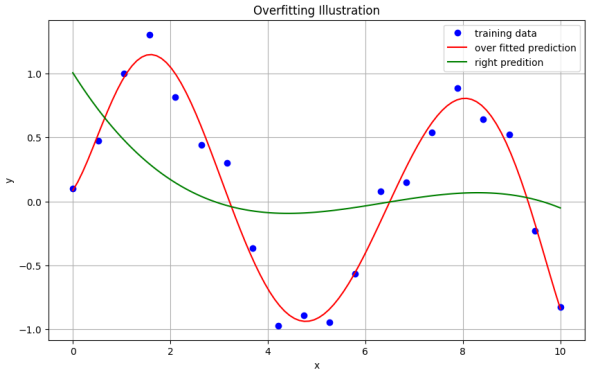

图 5: 过拟合示意图

5.1 超参数

以下为调优的主要超参数及其设置：

5.1.1 学习率

学习率 η 控制优化器（SGD）在梯度下降中的步长，影响模型参数更新的幅度。过高的学习率可能导致训练不稳定或发散，而过低的学习率会减慢收敛速度。在实验中，初始学习率 η_0 选取值为 $\{0.01, 0.05, 0.1\}$，并采用指数衰减策略：

$$\eta_t = \eta_0 \cdot \text{decay\_rate}^t,$$

其中 t 为当前 epoch 数，衰减因子为 decay_rate 。这种策略使学习率

随训练进程逐渐减小，有助于在训练后期精细调整参数。学习率的选择基于初步实验，旨在平衡收敛速度与稳定性。

5.1.2 正则化强度

正则化强度 λ 控制 L2 正则化项对损失函数的贡献，用于限制模型权重幅度以增强泛化能力。L2 正则化项定义为：

$$L_{\text{reg}} = \frac{\lambda}{2} \sum_{\mathbf{w}} \|\mathbf{w}\|_2^2,$$

其中 $\mathbf{w}$ 包括所有卷积核和全连接层权重。实验中搜索了正则化强度的以下取值：$\{0.001, 0.005, 0.01\}$。这些值旨在探索正则化对模型过拟合的影响，较小的 λ 允许更大权重自由度，而较大的 λ 增强权重约束。

5.1.3 隐藏层尺寸

隐藏层尺寸通过卷积层滤波器的数量（`num_filters1` 和 `num_filters2`）衡量，分别对应模型中第一和第二卷积层的输出通道数。滤波器数量直接影响模型的特征提取能力和计算复杂度。较大的滤波器数量可以捕获更丰富的特征，但可能增加过拟合风险和计算成本。在实验中，我主要依据 GPU 的算力搜索了以下滤波器数量组合：

$$(\texttt{num\_filters1}, \texttt{num\_filters2}) \in \{(16, 32), (32, 32), (32, 64)\},$$

其中 `num_filters1` 为第一卷积层的滤波器数量，`num_filters2` 为第二卷积层的滤波器数量。这些组合的选择旨在平衡模型容量与计算效率，通过验证集性能评估不同组合对模型表达能力的影响。

5.1.4 Dropout 率

全连接层前向传播中的 Dropout 为：

$$\mathbf{x}_{\text{drop}} = \mathbf{x} \cdot \mathbf{m} / (1 - p),$$

其中 $\mathbf{x}$ 为输入特征，p 为 Dropout 率。实验中先固定 Dropout 率为 $p = 0.5$，即以 50% 的概率丢弃神经元输出。完成关键超参的搜索后，再对 p 进行搜索。

5.2 搜索策略

为追求超参数搜索高效性和准确性的并存，我采取了分阶段超参数搜索的策略。首先确定出对模型影响最大的超参组合，即学习率、正则化强度和隐藏层大小，对他们进行网格搜索；然后对于相对模型影响较小的参数，在固定关键超参数组合的情况下随机搜索。

5.2.1 学习率、正则化强度和隐藏层大小的网格搜索

采用网格搜索方法对学习率、正则化强度和隐藏层大小的超参数组合进行评估，具体为学习率 $\eta_0 \in \{0.01, 0.05, 0.1\}$ ，正则化强度 $\lambda \in \{0.001, 0.005, 0.01\}$ 和隐藏层大小（两个卷积核个数的组合）$\{(32, 64), (32, 32)\ (16, 32)\}$ 的全组合，Dropout 率固定为 $p = 0.5$。对于每组超参数，训练模型 20 个 epoch，使用批量大小 `batch_size = 128`，并通过验证集准确率选择最佳组合。最佳模型权重基于验证集最高准确率保存为 `.npz` 文件，包含所有可训练参数（如卷积核 $\mathbf{w}$、偏置 $\mathbf{b}$、批量归一化参数 γ 和 β）。

训练过程中记录了每个 epoch 的训练损失、训练准确率、验证损失和验证准确率，并生成了可视化结果（损失和准确率曲线）。测试阶段加载最佳权重，计算测试集准确率作为最终性能指标。

5.2.2 Dropout 率的随机搜索

在确定了学习率和正则化强度这两个关键超参数后，我对 Dropout 率进行随机搜索。Dropout 率的取值范围在 0, 1 之间。为了找到最优的 Dropout 率，我们选择在 0, 1 之间进行随机搜索。

对于每个随机选择的 Dropout 率，重新训练模型 20 个 epoch ，使用批量大小 128 ，并通过验证集准确率监控模型性能。最后生成可视化结果。

5.3 调优结果与可视化

表 3 展示了不同超参数组合对最大验证准确率的影响的具体数据。我将其可视化为图 6 的 3D 图像如下。由此看出，参数变化和验证准确率的趋势如下：

- **学习率（lr）**：lr $= 0.01$ 时表现最佳（如 0.7002 和 0.7048），而 lr $=$ 0.1 时准确率显著下降（如 0.5156），表明高学习率易导致梯度震荡和训练不稳定。
- **正则化强度（reg）**：reg $= 0.005$ 在多数组合中表现较优（如 0.7002、0.7025），相比 reg $= 0.001$（0.6854）和 reg $= 0.01$（0.6943），适度正

则化更能平衡模型复杂度和泛化能力。

- **隐藏层大小**：滤波器数量为 (32, 32) 时准确率普遍高于 (32, 64) 和 (16, 32)（如 0.7048 vs. 0.7002），表明减小容量可缓解过拟合，容量太小容易导致特征提取效果差。
- **最优组合**：学习率 lr = 0.01、正则化强度 reg = 0.005、隐藏层大小 (32, 32) 取得了最高验证准确率 0.7048。

表 3: 学习率和正则化强度组合的最大验证准确率

lr	**reg**	隐藏层大小	**最大验证准确率**
0.01	0.001	(32, 64)	0.6854
0.01	0.005	(32, 64)	0.7002
0.01	0.01	(32, 64)	0.6918
0.1	0.001	(32, 64)	0.6856
0.1	0.005	(32, 64)	0.6930
0.1	0.01	(32, 64)	0.5993
0.05	0.001	(32, 64)	0.6569
0.05	0.005	(32, 64)	0.7025
0.05	0.01	(32, 64)	0.6754
0.01	0.001	(32, 32)	0.7048
0.01	0.005	(32, 32)	0.6959
0.01	0.01	(32, 32)	0.7058
0.1	0.001	(32, 32)	0.6053
0.1	0.005	(32, 32)	0.5323
0.1	0.01	(32, 32)	0.5156
0.05	0.001	(32, 32)	0.6204
0.05	0.005	(32, 32)	0.6509
0.05	0.01	(32, 32)	0.6943
0.01	0.001	(16, 32)	0.3296
0.01	0.005	(16, 32)	0.3047
0.01	0.01	(16, 32)	0.5534
0.1	0.001	(16, 32)	0.5241
0.1	0.005	(16, 32)	0.1051
0.1	0.01	(16, 32)	0.2324
0.05	0.001	(16, 32)	0.4578
0.05	0.01	(16, 32)	0.5292

绘制为 3D 图像如图 5 所示。

表 3 展示了不同 Dropout Rate 对最大验证准确率的影响；将之可视化为柱状图如图 6 所示。分析如下：

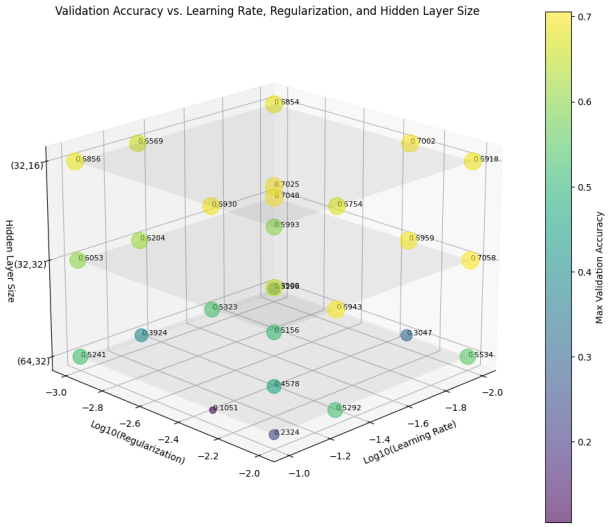

图 6: 学习率、正则化强度和隐藏层尺寸与验证准确率的关系

- **趋势分析**：$Dropout\_Rate = 0.3$ 时，最大验证准确率最高，达到 0.6966；相比之下，$Dropout\_Rate = 0$ 时准确率最低(0.6429)，$Dropout\_Rate = 0.5$ 和 0.7 时分别为 0.6762 和 0.6840，均低于 0.3 的表现。
- **原因推测**：$Dropout\_Rate = 0.3$ 可能在当前模型中取得了正则化与模型容量之间的最佳平衡，避免了过拟合（$Dropout\_Rate = 0$ 时易发生）的同时保留了足够的特征表达能力。Dropout Rate 过高（如 0.7）可能导致信息丢失过多，削弱模型的拟合能力，而 $Dropout\_Rate = 0.5$ 可能正则化强度稍显不足。
- **最优选择**：$Dropout\_Rate = 0.3$ 是当前实验中的最优值，显著提升了模型的泛化性能。

表 4: 不同 Dropout Rate 的最大验证准确率

Dropout Rate	**最大验证准确率**
0	0.6429
0.3	0.6966
0.5	0.6762
0.7	0.6840

综上可知：最佳超参数组合是 $\eta_0 = 0.01, \lambda = 0.005$，隐藏层大小为 $(32, 32), p = 0.3$.

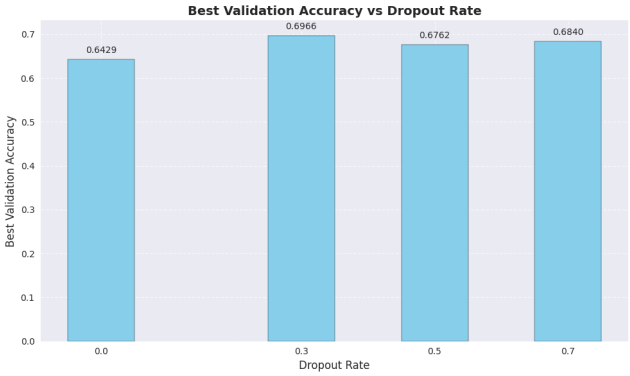

图 7: Dropout Rate 与验证准确率的关系

6 实验结果

6.1 训练过程可视化

在这里，我以最佳超参数组合为例，展示训练过程。

6.1.1 Loss 曲线

Loss 图像如图 8 所示，分析可得：

- **总体趋势**：随着 epoch 的增加，训练损失逐渐下降，表明模型在训练集上的表现逐渐改善。验证损失在前几个 epoch 中也逐渐下降，但在某些 epoch 后出现了波动（如第 5、14、16 和 18 个 epoch），表明模型在验证集上的表现不够稳定。
- **两者对比**：在前 5 个 epoch 中，训练损失和验证损失都呈下降趋势，且两者的差距较小，说明模型在训练集和验证集上的表现较为一致。从第 5 个 epoch 开始，验证损失的波动性增加，而训练损失持续下降。这表明模型可能开始过拟合（overfitting），即模型在训练集上表现良好，但在验证集上泛化能力还可以进一步提升。

6.1.2 Accuracy 曲线

Accuracy 图像如图 9 所示，结合验证损失函数的趋势，可得以下分析：

- **测试准确率趋势**：随着 epoch 的增加，验证准确率总体呈上升趋势，表明模型在验证集上的表现逐渐改善。表明模型在验证集上的表现虽然在准确率上有所提高，但损失的波动可能反映了模型在某些 epoch 上对验证集的特征捕捉不够准确。

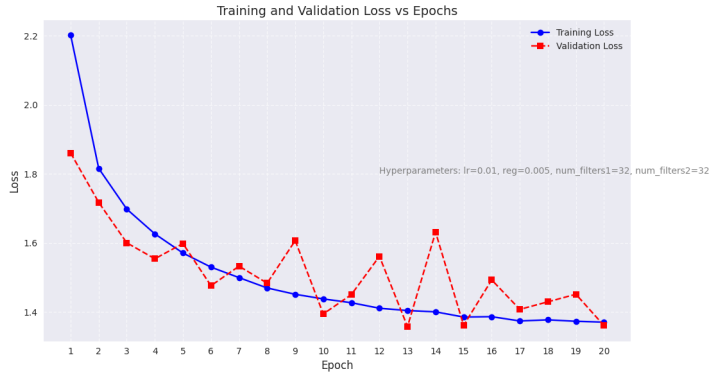

图 8: 损失函数曲线

- **两者趋势结合**：验证损失在第 4 个 epoch 之前陡降，在第 5、14、16 和 18 个 epoch 出现了明显的上升，这与验证准确率的初始下降和后期波动相对应。

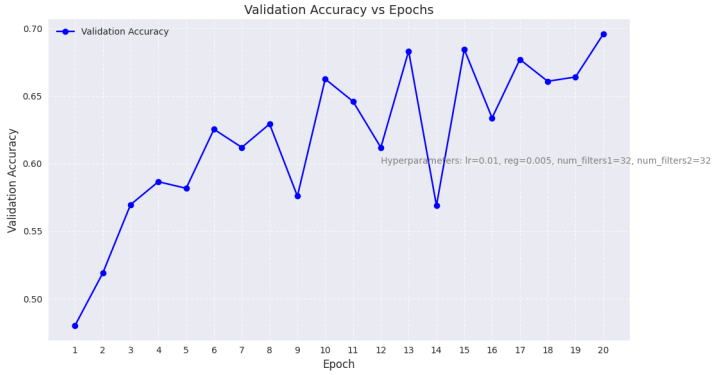

图 9: 验证准确率曲线

6.2 测试集性能

使用最佳超参数（lr = 0.01, reg = 0.005, 隐藏层大小 (32, 32)）和 Dropout Rate = 0.3 在测试集上进行评估，测试结果分析如下：

- **测试集性能**：最佳模型在测试集上的准确率为 0.6943，表明模型在未见过的数据上具有较好的泛化能力。
- **目标达成情况**：实验目标为从零开始实现一个三层 CNN 图像分类器，测试准确率达到 65%–70%。当前测试准确率 0.6943 已成功达到目标

区间上限，相比 PyTorch 实现的预期三层 CNN 准确率 72.84%，性能差距仅约 3.41%，验证了从零实现的有效性。

6.3 模型最优网格参数可视化

为了深入理解训练后的卷积神经网络（ConvNet）在 CIFAR-10 分类任务中的表现，我对模型的最优网格参数（包括权重和偏置）进行了可视化分析。这些参数包括第一卷积层（`w1`, `b1`）、第二卷积层（`w2`, `b2`）、全连接层（`w3`, `b3`）以及批归一化层的参数（`gamma1`, `beta1`, `gamma2`, `beta2`）。通过可视化这些参数，我们可以观察其分布规律和模式，从而揭示模型在特征提取和分类决策中的行为。

首先，第一卷积层的权重 `w1`（形状为 $32 \times 3 \times 3 \times 3$），包含了 32 个 3×3 的 RGB 滤波器。每个滤波器被可视化为 RGB 图像，经过归一化以增强显示效果。图 10展示了这些滤波器的分布情况。从图中可以观察到，部分滤波器呈现出明显的颜色偏好（如偏向红色或蓝色），表明它们专注于提取特定的颜色特征；其他滤波器则显示出边缘或纹理模式，反映了模型对低级空间结构的敏感性。这种多样性表明第一卷积层有效捕获了 CIFAR-10 图像中的多种低级特征。

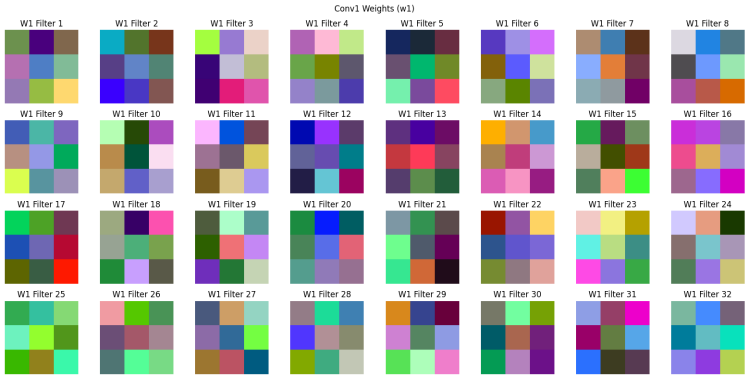

图 10: 第一卷积层权重（`w1`）的 32 个滤波器

接着，我可视化了第一卷积层的偏置 `b1`（形状为 32），如图 11所示。偏置值的条形图显示，大多数偏置值接近零，但少数偏置呈现较大的正值或负值。这种分布可能与批归一化层的存在有关，因为批归一化通过调整激活值的均值和方差减少了偏置的直接影响。较大的偏置值可能对应于某些特定滤波器的激活倾向。

对于第二卷积层的权重 `w2`（形状为 $32 \times 32 \times 3 \times 3$），由于其输入通道数较多（32 个），我们选择可视化每个滤波器的第一个输入通道（3×3），如

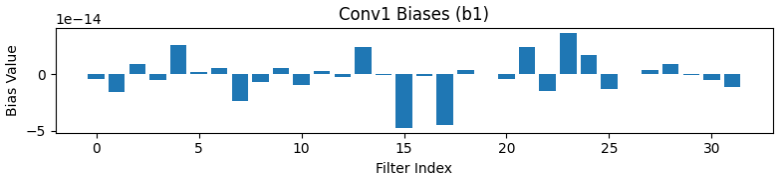

图 11: 第一卷积层偏置（b1）的分布

图 12所示。这些滤波器呈现出更加复杂的模式，部分滤波器显示出网格状或点状结构，表明第二卷积层倾向于捕获更高级的特征组合（如纹理或局部形状）。与第一卷积层相比，w2 的权重值分布更为分散，反映了模型在更深层网络中对抽象特征的建模。

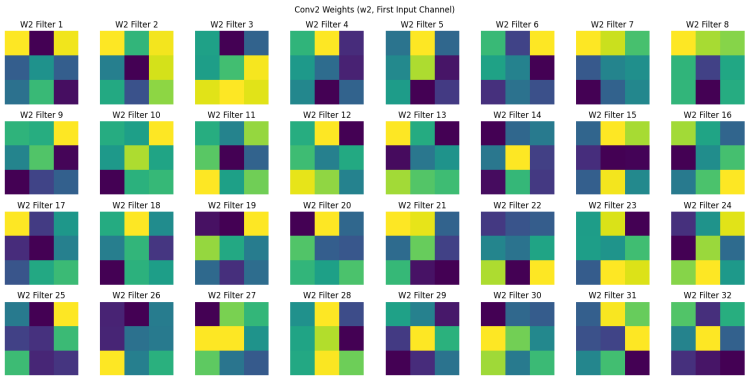

图 12: 第二卷积层权重（w2）的 32 个滤波器（首个输入通道）

第二卷积层的偏置 b2（形状为 32）如图 13所示。与 b1 类似，偏置值大多接近零，但波动幅度略大。这可能表明第二卷积层在特征提取中需要更强的偏移调整，以适应池化层后的特征空间压缩。

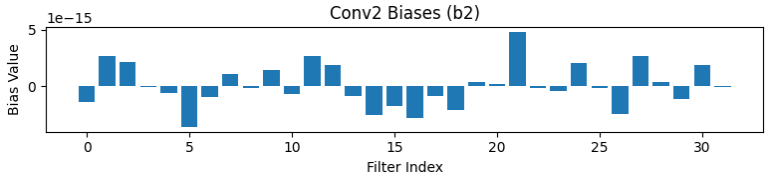

图 13: 第二卷积层偏置（b2）的分布

全连接层的权重 w3（形状为 $32\times9\times9\times10$）被可视化为热图，如图 14所示。热图的横轴表示输入特征（$32\times9\times9$），纵轴表示 10 个 CIFAR-10 类别。权重值呈现出稀疏且不均匀的分布，某些类别的权重值变化较大（如“飞机”和“卡车”），而其他类别（如“鸟”）的权重值较为平稳。这种模式

表明模型对某些类别的区分依赖于特定的特征组合，而对其他类别则依赖更广的特征分布。

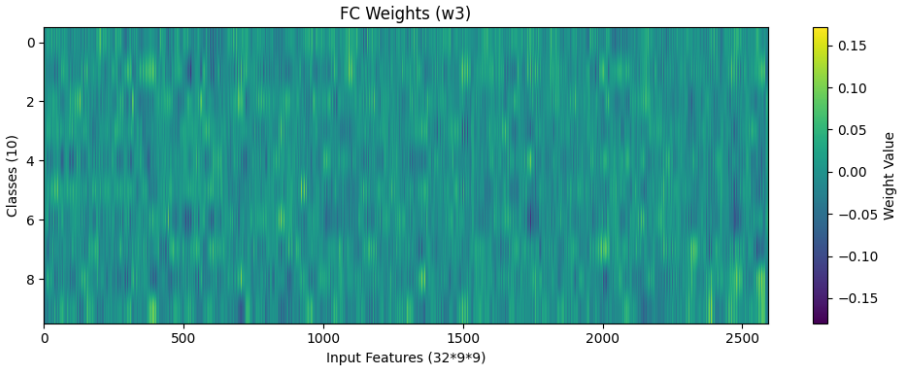

图 14: 全连接层权重（w3）的热图

全连接层的偏置 b3（形状为 10）如图 15所示。偏置值在不同类别间差异显著，例如“船”和“卡车”具有较大的正偏置，而“猫”和“狗”偏置较小。这种分布可能反映了模型对某些类别的初始倾向性，影响了 softmax 层的概率输出。

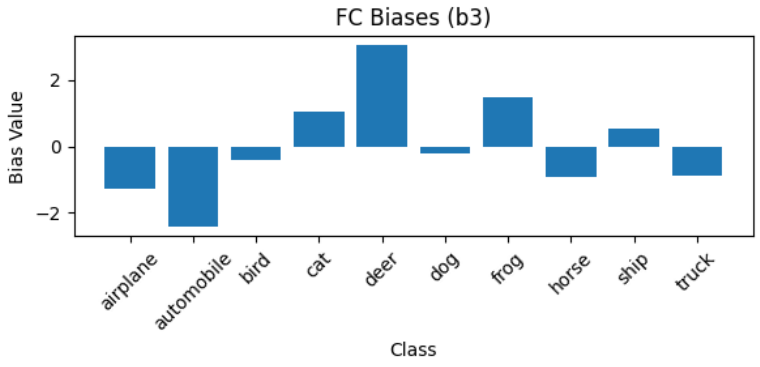

图 15: 全连接层偏置（b3）的分布

最后，我分析了批归一化层的参数。第一卷积层的 gamma1 和 beta1（均为形状 32）如图 16所示。gamma1 的值大多接近 1 ，但部分通道的值略高，表明这些通道的激活被放大；beta1 的值分布较为均匀，中心接近零。第二卷积层的 gamma2 和 beta2（图 17）显示出相似的模式，但 gamma2 的波动更大，反映了更深层网络中激活值的多样性。这些参数的规律性表明批归一化层有效稳定了特征分布，同时保留了通道间的差异。

综合来看，模型参数的分布呈现出以下规律：

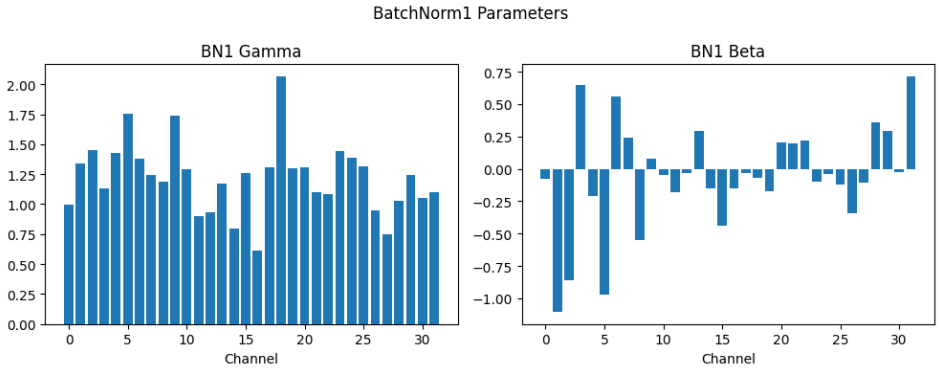

图 16: 第一批归一化层参数（gamma1 和 beta1）

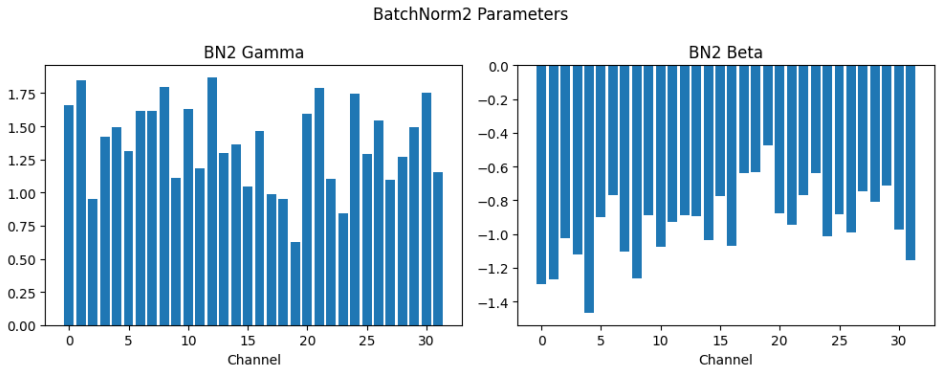

图 17: 第二批归一化层参数（gamma2 和 beta2）

- 卷积层权重从低级特征（颜色、边缘）到高级特征（纹理、形状）逐渐复杂；
- 偏置值总体接近零，受批归一化影响较小，但部分较大偏置反映了特定滤波器的倾向性；
- 全连接层权重和偏置显示出类别特异性，某些类别依赖更强的特征区分；
- 批归一化参数保持了激活值的稳定性，同时允许一定程度的通道差异。这些模式共同支撑了模型在 CIFAR-10 任务中的分类性能。

6.4 模型输出特征可视化

我通过随机加载了一个图像作为输入，使用最佳模型权重，分别了得到每层神经网络的可视化特征。

- **加载图像**：

图 18: 输入图像

- **第一层特征提取结果**：特征可视化存在一定规律，说明模型学到了低级特征。

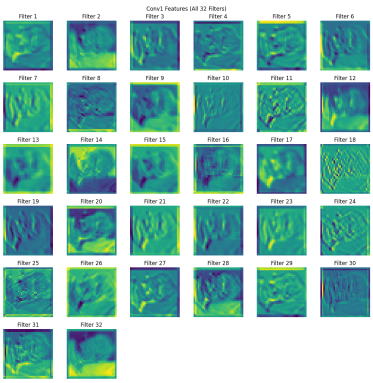

图 19: Conv 1 特征可视化

接着是 Pool 1 ，它相比 Conv 1，池化降低分辨率的效果显著。

- **第二层特征提取结果**：再一次卷积模型学到了更高级的特征。

  接着是 Pool 2 ，它降低分辨率的效果也很明显，肉眼基本读不出原图的形态。

- **输出结果**：以直方图显示模型对输入图像类别的预测，如下图所示。

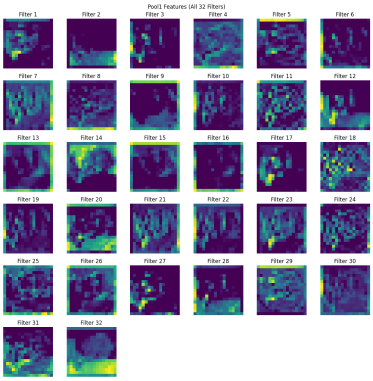

图 20: Pool 1 特征可视化

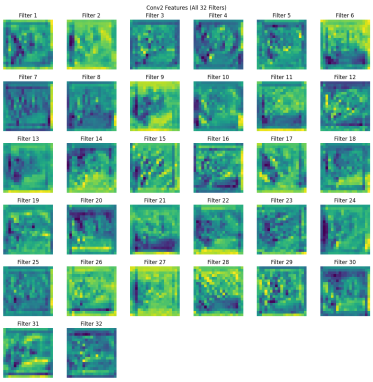

图 21: Conv 2 特征可视化

7 讨论与分析

7.1 模型优势

7.1.1 高效的 GPU 加速计算

模型基于 CuPy 库实现，充分利用 GPU 并行计算能力。卷积、池化和全连接层等操作均在 GPU 上高效执行，显著加速 CIFAR-10 数据集的训练与推理。超参数搜索因 GPU 支持得以快速完成，缩短实验周期。

7.1.2 模块化与灵活的网络架构

模型采用经典卷积神经网络结构，包含两个卷积层、批归一化、ReLU 激活、最大池化和全连接层。滤波器数量, 和 Dropout 率可调，支持灵活实验。这种模块化设计便于扩展和调试，适合快速原型开发。

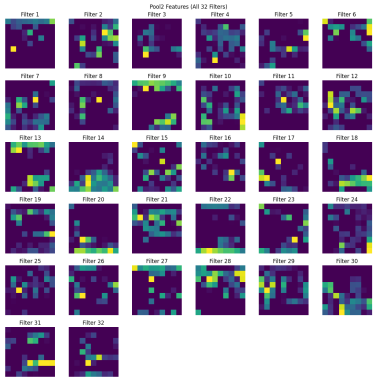

图 22: Pool 2 特征可视化

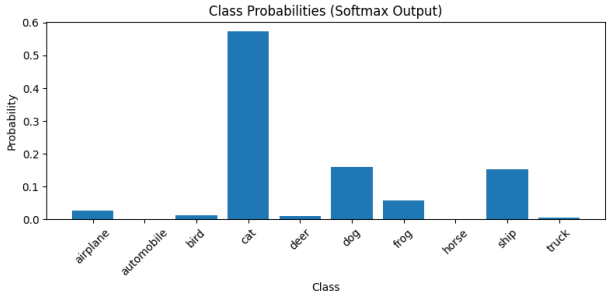

图 23: 输出预测类别

7.1.3 批归一化提升训练稳定性

每个卷积层后引入批归一化，通过标准化特征减少内部协变量偏移。批归一化维护运行均值和方差，支持高效推理。此设计加速收敛，降低对学习率的敏感性，提升模型对复杂数据分布的适应性。

7.1.4 鲁棒的正则化机制

模型通过 Dropout 和 L2 正则化防止过拟合。Dropout 在全连接层随机丢弃输出，L2 正则化约束权重。超参数搜索系统优化正则化效果，确保验证集和测试集性能稳定。

7.1.5 指数学习率与早停策略

模型采用指数衰减学习率和早停机制。学习率随训练进程减小，精细调整权重；早停根据验证准确率提前终止，防止过拟合。此策略平衡收敛速度与性能，节省计算资源并提升泛化能力。

7.1.6 全面的超参数调优支持

超参数搜索系统优化学习率（`lrs`）、正则化强度（`regs`）、隐藏层尺寸（`filter_sizes`）和 Dropout 率（`dropout_rates`）。保存搜索结果，便于分析；最佳模型权重自动存储，确保性能可复现。

7.1.7 精确的梯度裁剪

权重更新时引入梯度裁剪，通过检查梯度范数并缩放，防止梯度爆炸。裁剪应用于所有权重和批归一化参数，增强深层网络训练的稳定性。

7.2 不足

7.2.1 计算效率较低

本模型采用从零实现的卷积神经网络，相比主流深度学习框架（如 PyTorch 或 TensorFlow）的优化实现，计算效率较低。特别是在前向传播和反向传播过程中，矩阵运算和梯度计算未充分利用框架提供的自动优化和并行计算能力，导致训练时间较长，尤其在大数据量场景下表现尤为明显。

7.2.2 模型容量有限

当前模型仅包含两层卷积层，网络深度较浅，限制了其对复杂特征的提取能力。对于 CIFAR-10 中具有较高模式多样性的图像，浅层网络难以充分捕捉高层次的语义信息，可能导致模型在某些类别的分类任务上性能不足。

7.2.3 缺乏数据增强

现有实现未引入数据增强技术，导致模型在训练过程中对数据的多样性感知不足，容易出现过拟合现象，尤其在验证集和测试集上的泛化能力较弱。

7.2.4 优化器局限性

模型当前采用随机梯度下降（SGD）作为优化器，尽管通过学习率衰减一定程度上缓解了收敛问题，但与自适应优化器（如 Adam）相比，SGD 在复杂损失函数上的收敛速度较慢，且对超参数（如学习率）较为敏感。

7.3 改进方向

7.3.1 增加卷积层深度

为增强模型对复杂特征的提取能力，可以尝试加深卷积神经网络的层次结构。通过增加卷积层的数量，模型能够捕捉到更高层次的语义信息和更细粒度的局部特征，从而提升对复杂模式的学习能力。为了缓解深层网络可能带来的梯度消失问题，也可以选择引入了残差连接（Residual Connections）技术，以确保训练过程的稳定性并加速收敛。

7.3.2 采用 Adam 优化器

在优化策略方面，可以原始的优化器替换为 Adam（Adaptive Moment Estimation）优化器。Adam 结合了动量法和自适应学习率的优点，能够在训练初期快速收敛，同时在后期通过自适应调整学习率避免陷入局部最优解。相比传统的随机梯度下降（SGD），Adam 在非凸优化问题上表现更为鲁棒，尤其适合处理具有噪声的梯度更新场景。为进一步优化性能，可以再对 Adam 的超参数（如学习率、动量参数等）进行网格搜索，以找到最适合当前任务的配置。

7.3.3 引入数据增强技术

为提升模型的泛化能力和鲁棒性，可以数据预处理阶段引入了数据增强技术。具体而言，这些数据增强包括随机翻转、旋转、裁剪、颜色抖动和添加噪声等，能模拟数据分布中的多样性。这些操作可以有效扩充训练数据集的规模，缓解过拟合风险，并增强模型对未知数据的适应能力。此外，在每个训练批次中选择动态生成增强样本，从而进一步提高模型对数据变化的感知能力。

8 结论

为确保模型在 CIFAR-10 数据集上的性能，本研究精心设计了卷积神经网络（CNN）的结构与训练流程，并通过实验验证了超参数与优化策略的有效性。以下从模型结构和参数、超参数选择、优化策略和训练流程四个方面总结参数选择的关键点。

8.1 模型结构和参数

- **网络架构**：模型包含两层卷积层（Conv1 和 Conv2），滤波器数量分别为 32 和 64，均采用 3×3 卷积核，步幅为 1，填充为 2，以保持空间维度。每层卷积后接批归一化（Batch Normalization）、ReLU 激活函数和 2×2 最大池化（步幅为 2）。池化后的特征通过全连接层映射至 10 类输出。
- **权重初始化**：卷积层权重采用 He 初始化（方差为 $\sqrt{2/\text{fan\_in}}$），以稳定梯度传播；偏置初始化为 0；批归一化的缩放因子 γ 初始化为 1，平移因子 β 初始化为 0。
- **正则化**：全连接层前引入 Dropout（丢弃概率 0.5），仅在训练阶段启用，以增强泛化能力。
- **模型容量**：总计约 64×9×9×10（全连接层）+ 32×3×3×3（Conv1）+ 64×32×3×3（Conv2）个可训练参数，结构轻量但特征提取能力受限。

8.2 超参数选择

- **学习率**：初始学习率为 0.05，结合指数衰减策略（每轮乘以 0.995），以兼顾训练初期的快速收敛和后期的稳定优化。
- **正则化强度**：L2 正则化系数设为 0.005，平衡了模型复杂度与过拟合风险。
- **批次大小**：训练和验证批次大小为 128，确保梯度估计的稳定性和计算效率。
- **隐藏层大小**：Conv1 和 Conv2 的滤波器数量分别为 32 和 64，权衡了性能与计算复杂度。
- **搜索策略**：通过有限网格搜索（学习率 {0.05}，正则化 {0.005}）确定当前参数，未来可扩展至更广泛的超参数组合。

8.3 优化策略

- **优化器**：采用随机梯度下降（SGD），通过学习率衰减模拟自适应优化效果。梯度裁剪（最大范数 5.0）用于防止梯度爆炸。
- **损失函数**：使用交叉熵损失，结合 L2 正则化项，综合优化分类性能与模型复杂度。
- **批归一化**：在卷积层后引入批归一化（动量 0.9，$\epsilon = 10^{-5}$），加速收敛并增强训练稳定性，测试时使用运行均值和方差。

8.4 训练流程其他参数

- **数据集划分**：CIFAR-10 训练集（50,000 张图像）随机划分为 80% 训练集（40,000 张）和 20% 验证集（10,000 张），测试集为 10,000 张图像，均进行随机打乱以确保分布一致性。
- **训练轮数**：最多训练 20 轮，结合早停机制（耐心值 5），当验证准确率连续 5 轮未提升时终止训练，以防止过拟合。
- **模型保存**：当验证准确率提升时，保存模型权重（包括卷积层、全连接层和批归一化参数），确保最佳模型用于测试。
- **评估指标**：训练和验证阶段监控损失与准确率，测试阶段以准确率为最终指标。

9 代码与资源

9.1 Github 链接

github 仓库链接如下：https://github.com/RunRiotComeOn/first_cv_assignment20250413

9.2 模型权重

模型权重已上传至百度网盘，model_best.npz，链接为：https://pan.baidu.com/s/1p48X5sZQNKtiQRHpkxOAgQ?pwd=best （提取码：best）

参考文献

[1] data availability: https://www.cs.toronto.edu/~kriz/cifar.html

[2] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. Proceedings of the 14th International Conference on *Artificial Intelligence and Statistics (AISTATS)*, 315–323. Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

[3] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Journal of Machine Learning Research*, 15, 1929-1958.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929-1958.

[6] Teerapittayanon, S., McDanel, B., & Kung, H. T. (2017). BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. *arXiv preprint* arXiv:1709.01686.