模式识别

大作业

（2024 - 2025 学年度　　秋季学期）

实验名称　分类

姓名　张博仕

学号　2022010153

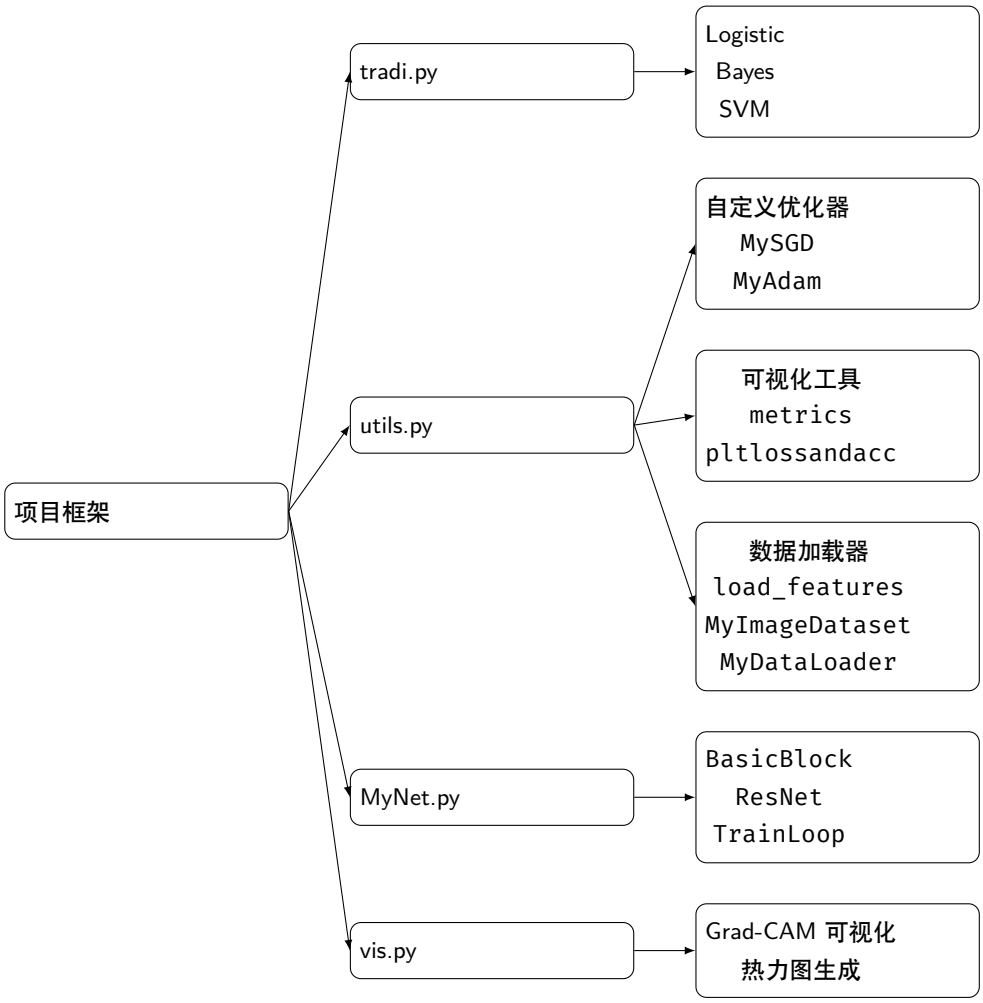
院系　自动化系

教师　鲁继文

时间　2024.11.19

目录

1 项目代码框架



代码主要分为以下几个模块，各模块之间通过清晰接口进行调用：

1. **utils.py**：工具函数和核心组件的实现，为任务 1 和任务 2 提供了基础支持
   - 数据加载器:实现任务 1 的特征加载函数 `load_features`,以及任务 2 的数据集类 `MyImageDataset` 和自定义数据加载器 `MyDataLoader`
   - 自定义优化器：实现了 `MySGD` 和 `MyAdam` 优化器；一些可视化工具包
2. **tradi.py**：
   - Logistic 回归：手写实现多分类 Logistic 回归器，包含训练逻辑（基于交叉熵损失的优化）和预测功能，并通过验证集评估性能。
   - Bayes 分类器：实现了基于均值和欧几里得距离的简单分类方法，作为传统概率方法的基线对比；SVM 分类器：设计了适用于多分类任务的 SVM 。
3. **MyNet.py**：
   - 深度网络设计：基于 `BasicBlock` 和 `ResNet` 构建了可配置的残差网络。训练与验证，支持自定义优化器和动态学习率调度功能。
4. **vis.py**：
   - Grad-CAM 热力图生成：实现 Grad-CAM 算法，直观展示网络在图像中的关注区域，帮助理解模型的决策依据。

2 传统方法

2.1 使用 Logistic 回归完成十分类任务

2.1.1 代码功能描述

手搓了一个 Logistic 线性回归器，在 500iters 时就基本已经收敛，acc 可以跑到 0.98。

Listing 1: 使用 Logistic 回归做分类

```
class MyLogistic():
    def __init__(self, x_train, y_train):
        self.x_train = x_train
        self.y_train = y_train
        self.k = y_train.shape[1]
        self.n = x_train.shape[1]
        self.wT = np.random.rand(x_train.shape[1],y_train.shape[1])*0.1
        self.b = np.zeros((1, self.k))

    def softmax(self, z):
        exp_z = np.exp(z - np.max(z,axis=1,keepdims=True))
        return exp_z/np.sum(exp_z,axis=1,keepdims=True)

    def cross_loss(self,W,x,Y,b):
        z = np.dot(x,W) + b
        eps = 1e-12
        soft_z = self.softmax(z) + eps
        loss = - np.sum(Y * np.log(soft_z)) / x.shape[0]
        return loss

    def fit(self,iters=3000, lr = 1e-1):
        for i in range(iters):
            z = np.dot(self.x_train,self.wT)+self.b
            p = self.softmax(z)
            g_wT = np.dot(self.x_train.T, (p-self.y_train))/self.x_train.
                shape[0]
            g_b = np.mean(p-self.y_train,axis=0,keepdims=True)
            self.wT -= lr * g_wT
            self.b -= lr * g_b
            loss = self.cross_loss(self.wT,self.x_train,self.y_train,self.b)
            if i % 100 == 0 :print(f"loss{loss:.4f},iters{i}")

    def predict(self,x_test):
        z = np.dot(x_test,self.wT)+self.b
        p = self.softmax(z)
        return np.argmax(p,axis=1)
```

2.1.2 结果分析

绘制出混淆矩阵以及训练函数如下，在多次随机抽样类别中，acc 均可达 0.97 及以上，分类性能良好。

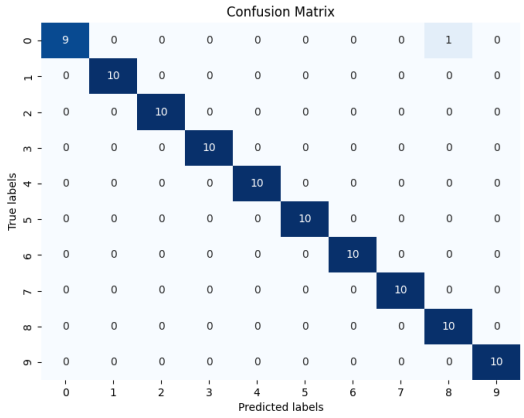


图 1: 使用 Logistic 回归对随机类别的分类效果

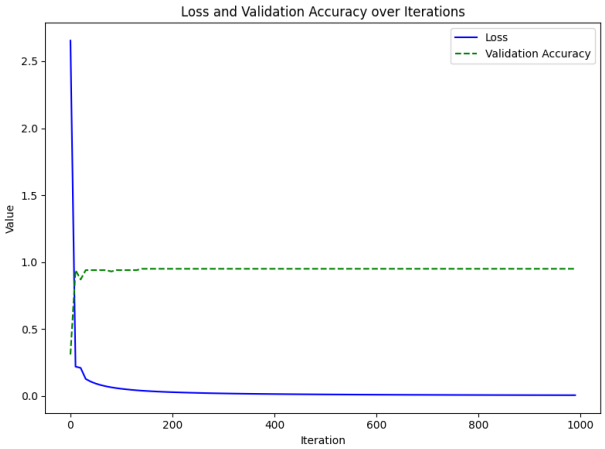


图 2: loss 和 acc 随时间变化的图像

2.2 Bayes 和 svm 方法的简单尝试

为了探究其余方法能否取得更好的效果，也为了弥补编程作业中没有手搓的遗憾，我写了简单的 Bayes 和 svm；

Listing 2: bayes 分类器

```
def MyBayes(X_train, y_train, X_test):
    classes = np.unique(y_train)
    means = np.array([np.mean(X_train[y_train == c], axis=0) for c in
        classes])
    dists = []
    for mean in means:
        diff = X_test - mean
        dis = np.sum(diff ** 2, axis=1)
        dists.append(dis)
    dists = np.array(dists)
    y_pred = classes[np.argmin(dists, axis=0)]
    return y_pred
```

svm 代码比较冗长，打包的代码里面有，不贴在这里了，以算法形式总结代码的实现功能：

Algorithm 1 十分类 SVM

输入: 训练数据集 X，标签 y，惩罚参数 C
输出: 多类分类器
procedure 训练 SVM(X, y, C)
    初始化 $w \leftarrow$ None，$b \leftarrow 0$
    计算核矩阵 K，其中 $K_{ij} =$ 核函数(X_i, X_j)
    调用 cvxopt 库解二次规划，得到拉格朗日乘子 α
    确定支持向量，计算 w 和 b
    return w, b, α
end procedure
procedure 多类 SVM(X, y, C)
    获取类别集合 $classes \leftarrow$ unique(y)
    for 每个类别 $c \in classes$ **do**
        生成二分类标签 $binary\_labels$
        调用训练 SVM(X, binary_labels, C) 得到 w, b, α
        存储模型 models$[c] \leftarrow (w, b, \alpha)$
    end for
    return models
end procedure
procedure 预测 (X, models)
    计算每个类别的决策函数值
    return 决策函数值最大的类别
end procedure

3 深度神经网络方法

3.1 网络的基本架构

在尝试了普通的 CNN 方法后，我遗憾的发现 val-acc 很难达到一个让人满意的 level，为了使得初始图像的信息尽可能的在网络中保留，我采样了 Resnet 的架构，具体的架构如图3所示：

具体的参数如下表所示：

表 1: 自定义 ResNet 的网络参数表

层次名称	输出尺寸	参数配置	备注
输入图像	$224 \times 224 \times 3$	-	RGB 图像输入
初始卷积层	$112 \times 112 \times 64$	7×7, Stride=2, Padding=3	BatchNorm + ReLU
最大池化层	$56 \times 56 \times 64$	3×3, Stride=2, Padding=1	-
残差块 1	$56 \times 56 \times 64$	$\begin{bmatrix} 3\times3, 64, Stride=1 \\ 3\times3, 64, Stride=1 \\ 3\times3, 64, Stride=1 \end{bmatrix} \times 3$	无下采样
残差块 2	$28 \times 28 \times 128$	$\begin{bmatrix} 3\times3, 128, Stride=2 \\ 3\times3, 128, Stride=1 \\ 3\times3, 128, Stride=1 \end{bmatrix} \times 3$	第一层 Stride=2 下采样
残差块 3	$14 \times 14 \times 256$	$\begin{bmatrix} 3\times3, 256, Stride=2 \\ 3\times3, 256, Stride=1 \\ 3\times3, 256, Stride=1 \end{bmatrix} \times 3$	第一层 Stride=2 下采样
平均池化层	$1 \times 1 \times 256$	自适应平均池化	-
全连接层	$1 \times 1 \times$ num_classes	FC 256 $\rightarrow$ num_classes	分类输出

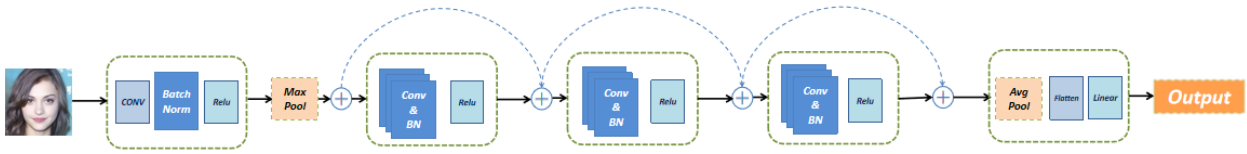


图 3: 本任务所采用的基于残差的网络架构

3.2 实验效果

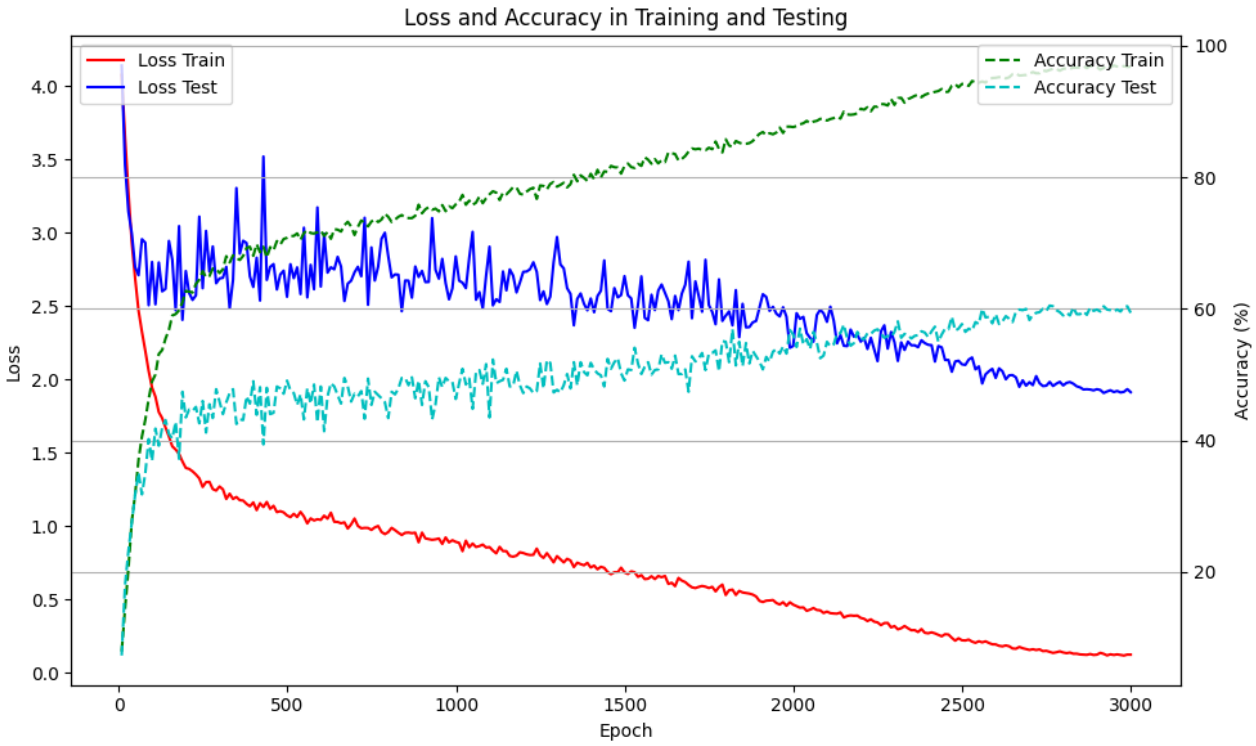


图 4: 3block-Resnet 模型的 loss 和 acc 随时间变化图

表 2: 图4所示实验的实验设置与最终效果

设置项	
输入大小	224×224
网络架构	3-block ResNet
残差块结构	[3, 3, 3]
Batch Size	64
Learning Rate	0.1
优化器	SGD (Momentum=0.9, Weight Decay=1e-4)
数据增强	随机裁剪、翻转、旋转、亮度调节
最终训练集损失	0.1248
最终训练集准确率	96.8%
最终测试集损失	1.9827
最终测试集准确率	60.5%

3.3 对比实验

3.3.1 是否进行数据增强

在实验中，为了提升模型的泛化能力和对不同场景的适应性，我们在训练阶段引入了多种数据增强策略。这些策略通过模拟可能的图像变换（例如裁剪、翻转、旋转等），扩展了训练数据的分布，减轻了过拟合问题。

训练集数据增强策略 训练集中采用了以下数据增强操作：

- **RandomResizedCrop**：从输入图像中随机裁剪一个区域，并将裁剪区域缩放至 224×224 的固定大小。
- **RandomHorizontalFlip 和 RandomVerticalFlip**：图像有一定概率（50%）进行水平或垂直翻转。这一操作扩展了数据分布，使模型能够适应不同方向的图像。
- **RandomRotation**：图像随机旋转一定角度（范围为 ±30°）。这一操作增强了模型对不同拍摄角度的适应性。
- **ColorJitter**：随机调整图像的亮度、对比度和饱和度，变化范围为 ±40%。通过模拟光照条件的变化，提升模型在多样化环境下的表现。
- **RandomGrayscale**：图像有 20% 的概率被转换为灰度图像，提升模型对黑白图像的适应能力，例如处理夜间图像或低光条件下的图像。
- **Normalize**：使用均值 $[0.485, 0.456, 0.406]$ 和标准差 $[0.229, 0.224, 0.225]$ 对像素值进行归一化，确保输入数据的分布一致。

验证集数据增强策略 为了确保验证集的稳定性，避免数据增强带来的额外随机性干扰评估结果，验证集中仅采用了以下两种预处理操作：

- **中心裁剪（CenterCrop）**：图像裁剪至 224×224 的大小，确保验证数据的尺寸一致。
- **标准化(Normalize)**:同训练集的标准化操作,使用均值 $[0.485, 0.456, 0.406]$ 和标准差 $[0.229, 0.224, 0.225]$ 进行归一化。

对比实验效果 数据增强策略显著提升了模型的泛化能力。以下是使用和未使用数据增强时模型在验证集上的表现对比：

图 5: 保留数据增强

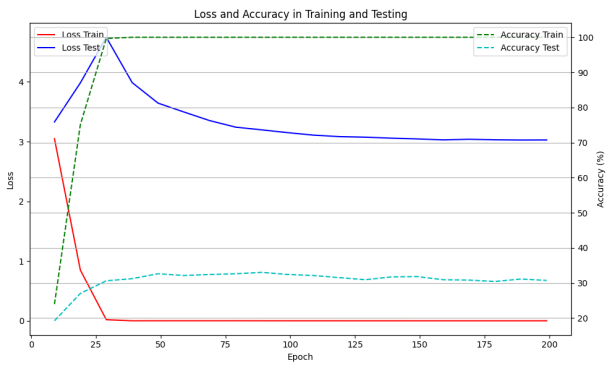


图 6: 去除所有的数据增强

表 3: 数据增强对比实验效果

数据增强	最终验证集损失	最终验证集准确率
未使用	3.027	30.7%
使用	1.983	60.5%

由表可见，引入数据增强后，验证集准确率显著提升，验证了数据增强策略在提升模型泛化能力方面的有效性。

3.3.2 改变 BatchSize 和 lr

由于我调用

Listing 3: 调用 cosine-sheduler 在训练的后期逐渐降低学习率

```
self.scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(self.optimizer
    ,T_max=self.num_epochs)
```

不同 lr 下，训练后期相同网络的分类效果基本完全一致，所以不再赘述了。

此外，我研究了 bc=64、128、256 下的训练分布图如下：

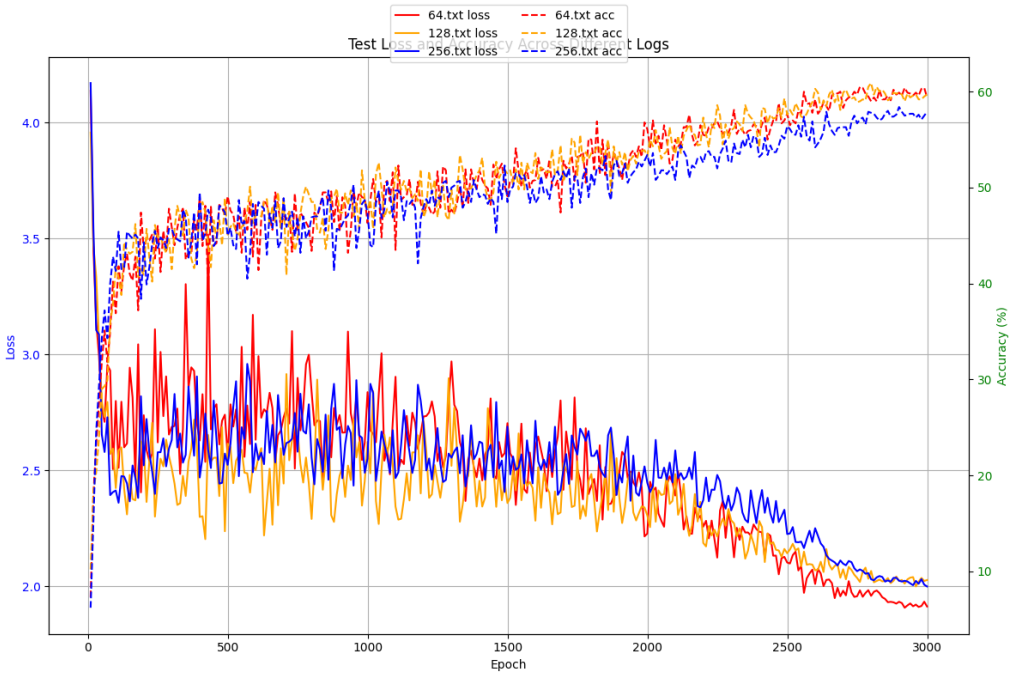


图 7: 不同 bc 下的对比图像

从图 7 中可以看出，在不同的批量大小下，模型的训练表现差异并不大。比较明显的是当批量大时，训练的波动较大，表明梯度更新更加随机化，这可能导致训练收敛速度较慢，但有助于跳出局部最优解。在实际训练中需要根据模型的规模和计算资源选择适当的批量大小，以在收敛速度和泛化能力之间取得平衡，这也是我选择 bc=64 做主要 exp 的原因。

3.3.3 增减网络架构

主要考虑对 block 内卷积数量的增减，如下图所示：

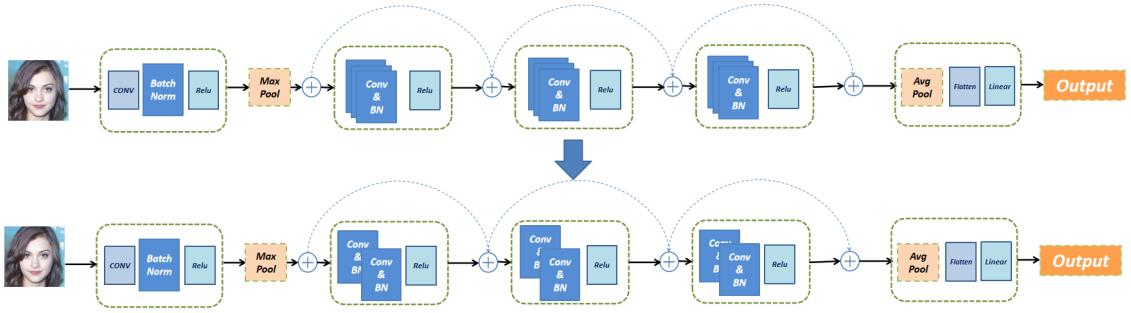


图 8: 改变 block 内卷积数量

或者是模型层数的增加（即 block 的个数），如下图所示：

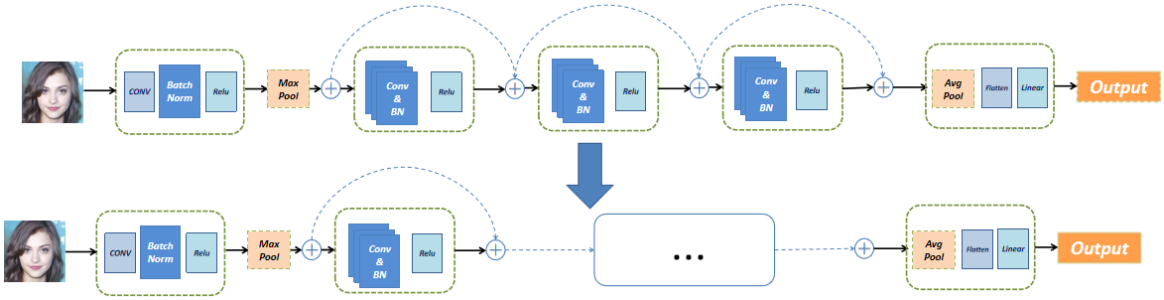


图 9: 增加 block 数量

由于计算资源的限制，我选取了一些较小的层数做训练：

表 4: 不同深度网络架构在验证集准确率对比

Block 数量	卷积层数量		
	2	3	4
2 Blocks	57.8%	59.2%	59.5%
3 Blocks	60.1%	60.5%	**61.1%**
4 Blocks	56.8%	/	/

在本任务的实验设置下，整体模型深度为 8-12 下表现较好，具体 block 对模型的表示能力影响并不显著，在实验的这些区间里对 block 的变动有一定的鲁棒性。

4 一些探索 and 可视化

4.1 手写优化器

我手写了 SGD 和 Adam 优化器，同时禁用了 lr-scheduler 进行实验。结果是：我手写的优化器与库类在效果上没有差别，但训练速度慢了至少 5 倍，因此我不得不放弃使用手写优化器做主要实验。

我认为可能是手写优化器在更新参数时，通常需要多次显式分配和释放内存，这不仅增加了时间开销，还可能导致显存使用效率下降。而且手写优化器中未显式调用 GPU 并行计算接口，则无法充分利用 GPU 的计算能力。

4.2 热力图可视化

Grad-CAM（Gradient-weighted Class Activation Mapping）是一种用于深度神经网络可视化的技术，能够生成热力图，显示模型对某张输入图像的哪些部分最敏感。这种方法主要用于理解和解释分类模型的

决策过程。具体的计算方法如下：

1. **计算类别的梯度**：

   对模型输出中某一特定类别的得分关于目标层特征图的梯度进行反向传播。这些梯度表示该类别对特征图每个通道的重要性。

2. **计算权重**：

   - 对目标层的每个通道，计算梯度值在空间维度（宽度和高度）上的全局平均：

   $$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

   其中：

   - α_k^c：通道 k 的权重。
   - $\frac{\partial y^c}{\partial A_{ij}^k}$：类别 c 的预测值对特征图 A 的第 (i, j) 位置梯度。
   - Z：特征图的总像素数。

3. **生成热力图**：

   - 将权重 α_k^c 和目标层的特征图按通道加权求和：

   $$L^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$$

   其中：

   - L^c：生成的热力图。
   - 使用 ReLU 函数确保热力图仅保留对该类别有正贡献的部分。

4. **插值热力图**：

   - 将生成的热力图上采样到输入图像的大小，便于叠加展示。

我选取了网络最后一个 block 的最后一个 conv 的输出，即第三个残差块的最后一个层（因为全连接层已经丢失了这些信息）。

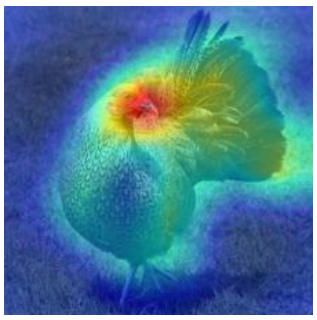


图 10: 鸡

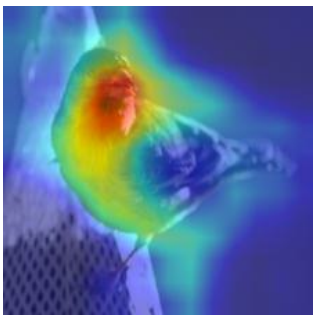


图 11: 鸟

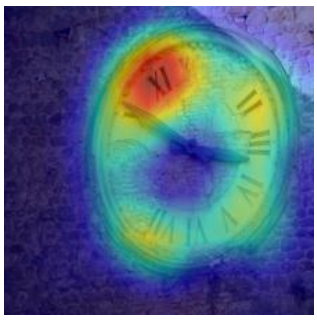


图 12: 时钟

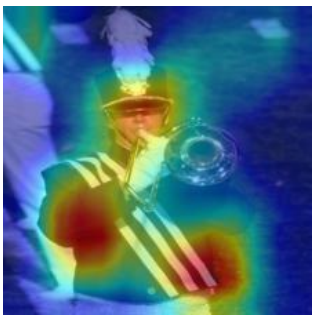


图 13: 军人

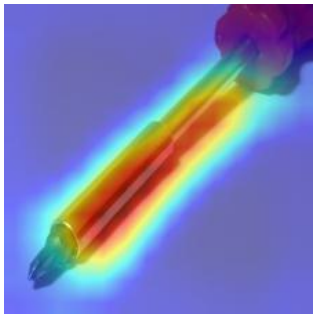


图 14: 螺丝刀

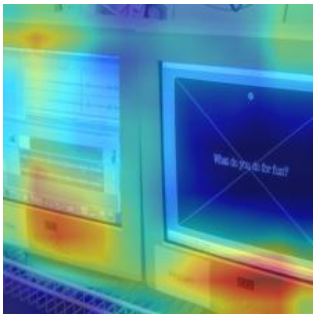


图 15: 电脑

图 16: 汽车

图 17: 餐厅

图 18: 训练完成的 ResNet 的热力图可视化

可以看到鸡冠部位、鸟喙部位、时钟的罗马数字部位、军人的礼帽、螺丝刀的长杆、电脑的按钮屏幕、车的前灯和轮子、餐厅的餐具等的热力图权重明显较高，这良好的解释的分类的信息依据，能够较为契合地体现本模型的可解释性。

5 总结

本实验完成度高，覆盖了任务要求的所有要点，并具有以下优越性：

- 在如此小的数据集上，通过设计**和反复实验**，val-acc 跑到了 0.60 以上，按鲁老师所说这是不容易的！
- 补充了多组参数对比实验，**解释了模型各参数对性能影响**。同时手动实现了绝大部分功能。
- 进行了较为完善的可视化，模型有良好的可解释性。

感谢老师和助教的指导和付出！通过本次实验我受益匪浅。

参考文献

[1] He, Kaiming, et al. ”Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] https://github.com/DocDode/6-clsfier（我自己的 github 账号，本实验使用了该仓库的可视化工具 cam.py）