鸟类识别与分类任务

刘东琛 2023011040

2025.11.12

Abstract

本次大作业尝试使用多种方法来对 CUB-200 鸟类数据集上的分类任务进行尝试，并根据验证集上的结果来评估方法效果。在传统模式识别方法部分，采用了支持向量机模型来实现这个分类任务，使用提供的特征向量训练分类器，在验证集上达到了 95% 以上的准确率。在深度神经网络部分，搭建 ResNet 架构，从头训练，在验证集上达到了 74.90% 的准确率。此外，我还使用了 ViT 方法，在预训练权重上进行微调，达到了 75.90% 的准确率。完整代码与使用方法保存在 github 链接：https://github.com/Li-Battery-dc/bird_classication

1 Dataloader 设计

为了支持灵活的数据输入和不同实验的需求，我实现了一个专门针对 CUB-200-2011 鸟类数据集的 DataLoader 类。

提供两种加载模式：

- 特征模式: 加载预先提取并存储的特征文件（.pt 格式）。这支持了基于特征向量的分类任务.
- 图像模式: 加载原始图像文件（.jpg 格式）。这支持了传统的图像分类任务，并允许在加载时应用用户定义的 transform

在数据获取接口上，除了一次性按特定规则加载指定子集的方法外，还使用了类似 PyTorch 的迭代器接口，支持 batch 数据的随机加载，方便后续训练。

2 SVM 方案

2.1 分类器设计

通过 one vs one 策略，将 svm 方法用于多分类。

首先实现了一个基础分类器 BinarySVM 实现了软间隔核化支持向量机的基础算法。采用 cvxopt 库中的二次规划求解器 solvers.qp 来解决 SVM 的对偶问题。支持线性、多项式和 rbf 函数三种常用核函数。

对于 K 类待预测标签，构建 $\frac{K(K-1)}{2}$ 个二分类器进行预测。通过 vote 机制综合每个分类器的预测结果，得到该样本的预测结果。

2.2 实验结果

使用 svm 分类器，使用 Dataloader 加载数据集中提供的特征文件，随机选取 200 类中的 10 类进行训练和验证。并测试了不同 C 值和不同核函数对于结果的影响。结果在 **Table1** 中展示：

Table 1: SVM results

核函数	C 值	训练集准确率 (%)	验证集准确率 (%)
linear	0.1	100.00	96.61
	1	100.00	96.61
	10	100.00	96.61
	100	100.00	96.61
rbf	0.1	86.29	81.36
	1	96.19	94.92
	10	100.00	96.61
	100	100.00	96.61
poly	0.1	92.00	89.83
	1	98.29	94.92
	10	100.00	96.61
	100	100.00	96.61

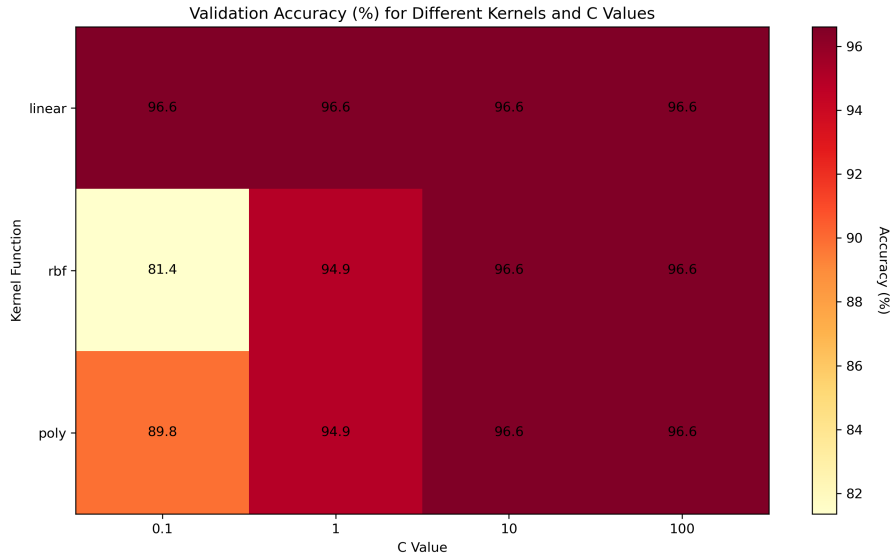

Figure 1: svm result heatmap

2.3 结果分析

从结果来看，使用已经提取好的特征进行分类任务的准确率很高，并且在线性核上表现最好。这表明预先提取的特征能很好地表征图像内容，且这些特征在类别间具有较好的线性可分性。

3 卷积神经网络方案

3.1 初期尝试

实验初期使用了简单的多层卷积->ReLU-> 池化->Flatten 的方式，使用 Adam 优化器和简单交叉熵损失进行实验，得到的结果如 **figure2** 所示

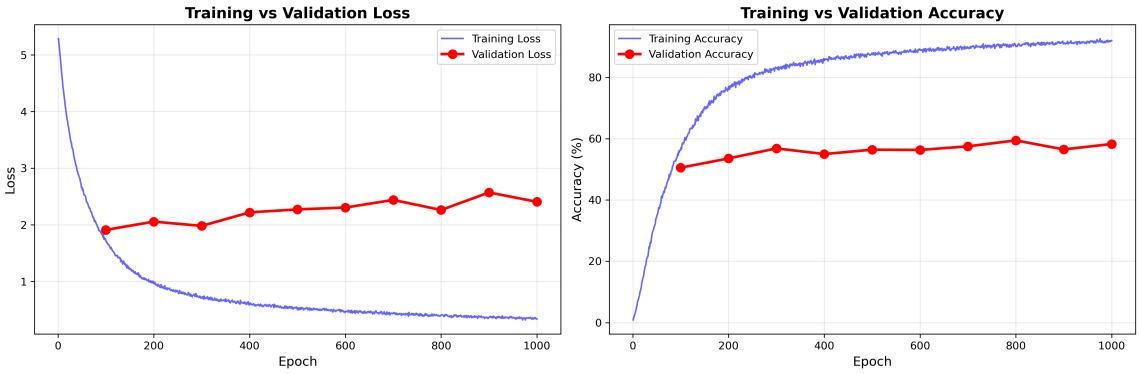

Figure 2: cnn result 1

从这次实验看出，在这个训练集上极易过拟合，并且训练后期 loss 降低较缓，于是决定采用 ResNet 架构，并设计了优化器和损失函数来解决这些问题。

3.2 网络架构

网络架构主要采用 ResNet 架构 (He et al. [2015])，网络架构分为三个主要部分：初始特征提取层、残差块堆叠层以及分类器。并根据遇到的过拟合问题，整体采用了较浅的网络层数和较少的通道数量。

初始层包含 7×7 卷积和最大池化，用于快速下采样。中间层由五个残差块组成。其中第 3,4,5 个残差块使用了Ghiasi et al. [2018] 使用的 DropBlock 方法，通过随机 mask 特征图的连续的一部分，替代简单 Dropout, 后续实验表明对提高网络的泛化能力很有作用。最终，特征图经过自适应平均池化和两个全连接层进行分类。

网络的详细结构和各层的输入输出尺寸如 **Table 2** 所示。

3.3 损失函数设计

为改善过拟合问题，我采用了一种改进的交叉熵损失函数。主要有三个机制：

1. 该损失函数主要使用了 FocalLoss 框架，通过引入动态缩放因子 $w = (1 - p_t)^\gamma$ 来调整损失贡献。加了难分类样本在损失函数中的权重

2. 加入了 label smoothing 操作，在计算交叉熵时，将传统的 One-Hot 真实标签替换为一个平滑的概率分布。防止模型对真实标签的预测概率过于自信，减少过拟合。
   对于平滑度 ϵ, one-hot 向量 Y 满足 $y_k = 0, k \neq true$ 进行平滑操作：

$$\tilde{y}_k = (1 - \epsilon) \cdot y_k + \frac{\epsilon}{K}$$

Table 2: 网络架构

层级	模块	操作	输入尺寸	输出尺寸	核大小	步长
初始卷积 (cov)	1	Conv+BatchNorm+ReLU	$3 \times H \times W$	$64 \times \frac{H}{2} \times \frac{W}{2}$	7×7	2
	2	MaxPool	$64 \times \frac{H}{2} \times \frac{W}{2}$	$64 \times \frac{H}{4} \times \frac{W}{4}$	3×3	2
残差块 (resblocks)	3	ResBlock	$64 \times \frac{H}{4} \times \frac{W}{4}$	$64 \times \frac{H}{4} \times \frac{W}{4}$	3×3	1
	4	ResBlock	$64 \times \frac{H}{4} \times \frac{W}{4}$	$128 \times \frac{H}{8} \times \frac{W}{8}$	3×3	2
	5	ResBlock+DropBlock	$128 \times \frac{H}{8} \times \frac{W}{8}$	$128 \times \frac{H}{8} \times \frac{W}{8}$	3×3	1
	6	ResBlock+DropBlock	$128 \times \frac{H}{8} \times \frac{W}{8}$	$256 \times \frac{H}{16} \times \frac{W}{16}$	3×3	2
	7	ResBlock+DropBlock	$256 \times \frac{H}{16} \times \frac{W}{16}$	$256 \times \frac{H}{16} \times \frac{W}{16}$	3×3	1
分类器 (classifier)	8	AvgPool	$256 \times \frac{H}{16} \times \frac{W}{16}$	$256 \times 1 \times 1$	Adaptive	-
	9	Linear+ReLU+Dropout	256	128	-	-
	10	Linear	128	N_{class}	-	-

$H \times W$ 为输入图像的高度和宽度；$N_{class} = 200$ 为分类类别数。*ResBlock* 采用两层 3×3 卷积结构。带有 *DropBlock* 的残差块 (*ResBlock+DropBlock*) 使用 0.1 的 *drop prosibility* 和 7×7 的 *DropBlock* 大小。

3. 加入了 Warmup 机制：在训练初期 γ 值通过指数增长的方式从 0 平滑上升至目标值 (target_gamma)，以避免 Focal Loss 在训练开始时对难样本的过度关注，导致收敛缓慢。并且支持对 γ 进行平滑的动态调整，用于不同阶段训练。

最终得到 Loss 的表达式：

$$L = (1 - p_t)^{\gamma} \cdot \left[\sum_{k=1}^{K} -\tilde{y}_k \log(p_k) \right]$$

其中 $p_k = \text{Softmax}(output)$，$p_t$ 为真实标签预测概率。

3.4 优化器设计

为了改善过拟合问题，行实现了带有动量和权重衰减的随机梯度下降（SGD）优化器。

加入 L2 正则化衰减项：$\mathbf{g}'_t = \mathbf{g}_t + \text{weight\_decay} * \mathbf{w}_t$，并维护动量缓冲区 $\mathbf{v}$，用于累积历史梯度信息，加速收敛并抑制振荡。$\mathbf{v}_t = \mu \mathbf{v}_{t-1} + \mathbf{g}'_t$ 最终的参数更新公式为：

$$\mathbf{w}_{t+1} = \mathbf{w}_t - lr * \mathbf{v}_t$$

并且使用了常用的余弦退火策略来动态调整学习率，学习率遵循余弦曲线衰减：

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\text{initial}} - \eta_{\min}) \left(1 + \cos \left(\pi \frac{epoch_{now}}{epoch_{total}} \right) \right)$$

并且添加了灵活的学习率调节方法用于不同训练阶段。

3.5 实验细节

3.5.1 Data augmentation

模型的抗过拟合能力和特征的鲁棒性，在 transform 中使用了以下数据增强技术：

Table 3: 训练阶段应用的数据增强策略

transform	参数	目的
`RandomResizedCrop`	224×224	增强特征尺度不变性。
`RandomHorizontalFlip`	$p = 0.5$	减少模型对物体方向的依赖。
`ColorJitter`	亮度/对比度/饱和度 ± 0.2	模拟不同光照条件和传感器变化。
`RandomRotation`	$\pm 20°$	引入角度变化，增强旋转鲁棒性。
`RandomErasing`	$p = 0.5$, 尺度/长宽比 ($\sim$ 0.02–0.2)	随机擦除，增强对遮挡的鲁棒性。
`Normalize`	ImageNet 常用的均值/方差	数据标准化。

3.5.2 训练流程

根据一些尝试结果观察，得到了最终的三阶段训练策略：

- warmup 阶段大学习率迅速收敛，FocalLoss 参数较小
- 主要训练阶段余弦退火学习率，FocalLoss 参数保持
- 冻结浅层深训练阶段，降低学习率，提高 FocalLoss 参数，专注难样本。

Table 4: 三阶段深度学习模型训练策略参数细节

参数 / 策略	阶段 I: Warmup	阶段 II: 主要训练	阶段 III: 冻结深训练
训练周期 (Epoch)	0 – 199	200 – 749	750 – 1000
学习率 (LR)	0.025 (保持不变)	从 0.025 开始余弦退火	冻结时 $\text{LR}_{\text{current}} \times 0.05$
Focal Loss γ	2.0	2.0	从 2.0 迅速平滑过渡到 3.0
模型可训练性	全部层可训练 (`Full`)	全部层可训练 (`Full`)	冻结 cov 层和前 4 层 resblock (Partial)

3.6 实验结果和分析

仅在每个 checkpoint 处验证准确率，可视化训练全流程结果如 **Figure 3**所示。最终最好的模型参数在验证集上的表现准确率为

74.90%

最佳模型参数也提交到了作业压缩包中。

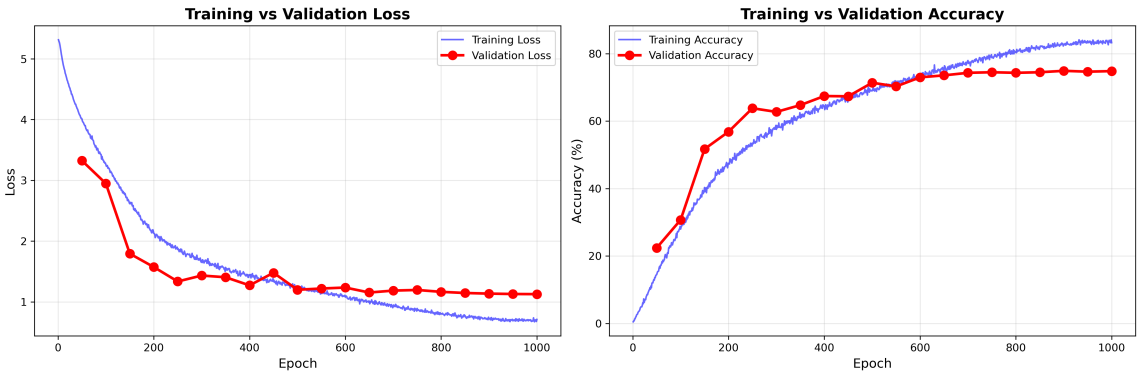

Figure 3: 全流程结果

从结果来看，冻结后训练的效果并不好。后续的一些消融实验表明，尽管在冻结启动时设置了平滑过渡策略，但是还是会遇到 loss 不稳定导致准确率骤降的问题。即使进行多次调参和冻结后训练，验证集准确率也并未突破 75%. 继续训练会慢慢出现过拟合现象。

后续可继续结合训练日志定位模型波动原因，进一步提高模型准确率。

4 Vision Transformer 方案

根据Dosovitskiy et al. [2021] 提出的 Vision Transformer(ViT) 架构，我们对图像进行不重叠的分块训练 transformer 得到较好的分类结果, 这里做了一些尝试。

4.1 说明

由于给定的任务数据集的大小不足以从头训练这样一个 ViT 网络，我不得不在预训练权重上进行微调，我对 ViT 的架构进行了复现，仅使用预训练权重。自己实现了损失函数设计和学习率调度器。并且为了适应复杂训练，我直接使用了 AdamW 优化器。

由于上面说明的问题，此部分仅作为大作业补充内容。

4.2 网络架构

由于需要使用预训练权重，网络架构参数不得不与预训练模型相同

4.2.1 Patch Embedding

ViT 架构的核心模块，对图像进行分块展平，并加入位置编码和分类 class token. 将图像鲁棒地转化为 Transformer 可用的输入。

4.2.2 Multihead Attention

此部分直接使用Vaswani et al. [2017] 提出的多头注意力机制

$$head_i(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

$$Attention = concat(head_i) \tag{2}$$

4.2.3 MLP

每个 Attention 模块后经过一个单隐藏层的 MLP, 这里的激活函数采用了 GeLU 激活函数而非常用的 ReLU 函数。其数学表达为:

$$GeLU(X) = X \cdot \phi(X) \tag{3}$$

$$= X \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{4}$$

4.2.4 transformer block

每个 block 包含包含多头自注意力层和 MLP 两个子层。每个子层均通过残差连接与层归一化稳定训练。并且为了加强正则化加入了 Stochastic drop path，缓解层数过深导致的过拟合。

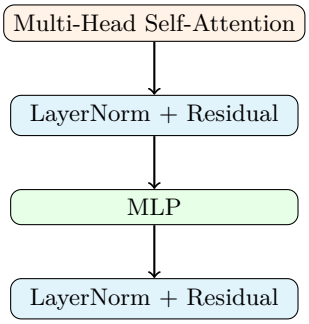

Figure 4: block 架构

4.2.5 完整架构

输入图像通过 patch embedding 划分为固定大小的图像块（patch），并将每个图像块线性映射到高维嵌入空间中。随后，在序列化的 patch 表示上应用多个 Transformer block 结构，通过多头自注意力机制捕捉全局依赖关系。最后，提取 class token 并通过分类头进行预测。

其中,分块大小,transformer block 数与使用的预训练模型"vit_base_patch16_224_in21k"保持一致。保证预训练参数加载正确。

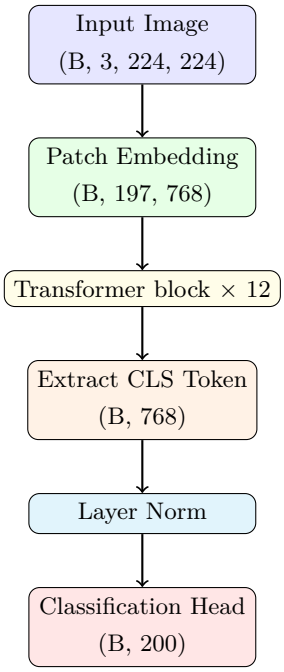

Figure 5: 完整架构

4.3 损失函数和优化器

损失函数为标签平滑交叉熵函数，与 CNN 中定义相同，此处不再赘述。

优化器直接使用了 torch 中带 weight decay 的 AdamW 优化器以适应复杂训练。

4.4 学习率调度

主要采用 warmup 和余弦退火策略，定义如下：

$$\eta_t = \begin{cases} \eta_{\text{start}} + (\eta_{\text{base}} - \eta_{\text{start}}) \dfrac{t}{T_{\text{warmup}}}, & t < T_{\text{warmup}} \\ \eta_{\min} + \dfrac{1}{2}(\eta_{\text{base}} - \eta_{\min}) \left[1 + \cos\left(\dfrac{\pi(t - T_{\text{warmup}})}{T_{\text{total}} - T_{\text{warmup}}} \right) \right], & t \geq T_{\text{warmup}} \end{cases}$$

此外，为了不损失预训练模型的通用参数，实验借鉴了 BERT 与 DeiT 中的逐层学习率衰减，对不同深度 Transformer 施加缩放系数以实现渐进式微调。设置：

$$\eta_i = \eta_t \cdot \lambda^{L-i}$$

靠近输入的低层保持较小学习率，从而保留预训练特征；而高层及分类头层具有较大学习率，以适应分类任务本身。

4.5 实验策略

4.5.1 Data augmentation

为防止过拟合，应用了多种正则化操作，具体参数如下：

Table 5: 训练阶段使用的数据增强策略配置

增强类型	参数配置
RandomResizedCrop	size=224
RandAugment	n=2, m=5
RandomHorizontalFlip	p=0.5
ColorJitter	brightness=0.2, contrast=0.2, saturation=0.2
RandomRotation	$\pm 20^\circ$
Normalize	mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]

除了对于训练图像的正则化之外，我还使用了 ViT 训练中常用的 Mixup 混合样本，通过对两个样本线性插值得到新样本来提高泛化性。定义如下：

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \tag{5}$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \tag{6}$$

$$\lambda \sim Beta(\alpha, \alpha) \tag{7}$$

对比实验发现 mixup 对于训练非常重要，实验参数如下：

Table 6: 各训练阶段的 MixUp 与 CutMix 参数设置

阶段	MixUp α	CutMix α	概率 (prob)	切换概率 (switch_prob)
Stage 1	未启用	–	–	–
Stage 2	0.2	0.0	0.25	0.5
Stage 3	0.2	0.0	0.50	0.5

4.5.2 不同阶段训练策略

为了充分利用预训练的 Vision Transformer（ViT）特征表示，同时适应鸟类分类新任务，本实验采用三阶段逐步解冻与差异化学习率调度的策略。

- 第一阶段仅训练分类头, 快速适应 CUB-200 鸟类分类任务，不启用 MixUp 增强。
- 第二阶段解冻最后 4 个 Transformer Block, 引入 mixup, 降低学习率
- 第三阶段解冻最后 8 层 Transformer Block，使模型具备更强的全局适应性。更强 MixUp 策略, 进一步降低学习率。

实验得到当前最优的不同阶段的训练目标与参数配置如下表所示。

Table 7: 三阶段训练策略与主要参数配置

阶段	训练部分	解冻层数	基础学习率	Warmup 轮数	Batch Size	epoch 数
Stage 1	仅分类头 (Head)	全部 backbone	1×10^{-3}	0	384	30
Stage 2	后 4 层	4 层	1×10^{-4}	10	256	80
Stage 3	后 8 层	8 层	3×10^{-5}	15	128	150

其中，由于实验 GPU 限制，第二阶段和第三阶段 batch size 降低，并且采用混合精度训练，且没有进行全模型微调。

4.6 实验结果和分析

使用上面的最佳参数进行训练，得到的结果如图：

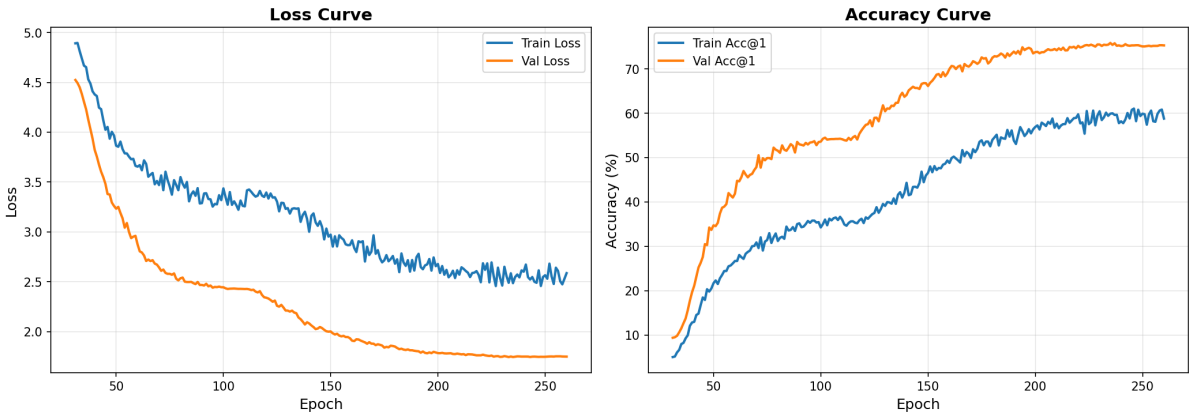

Figure 6: 训练日志曲线

其中，由于 mixup 正则的存在，训练集上的 loss 会出现较大的波动。由于验证集的准确率是基于 one-hot 编码结果，准确率比 mixup 下的训练集准确率高是正常的。

最终最好的模型参数在验证集上的表现准确率为

75.84%

最佳模型参数也提交到了作业压缩包中。

我们可视化最后一层的注意力图：

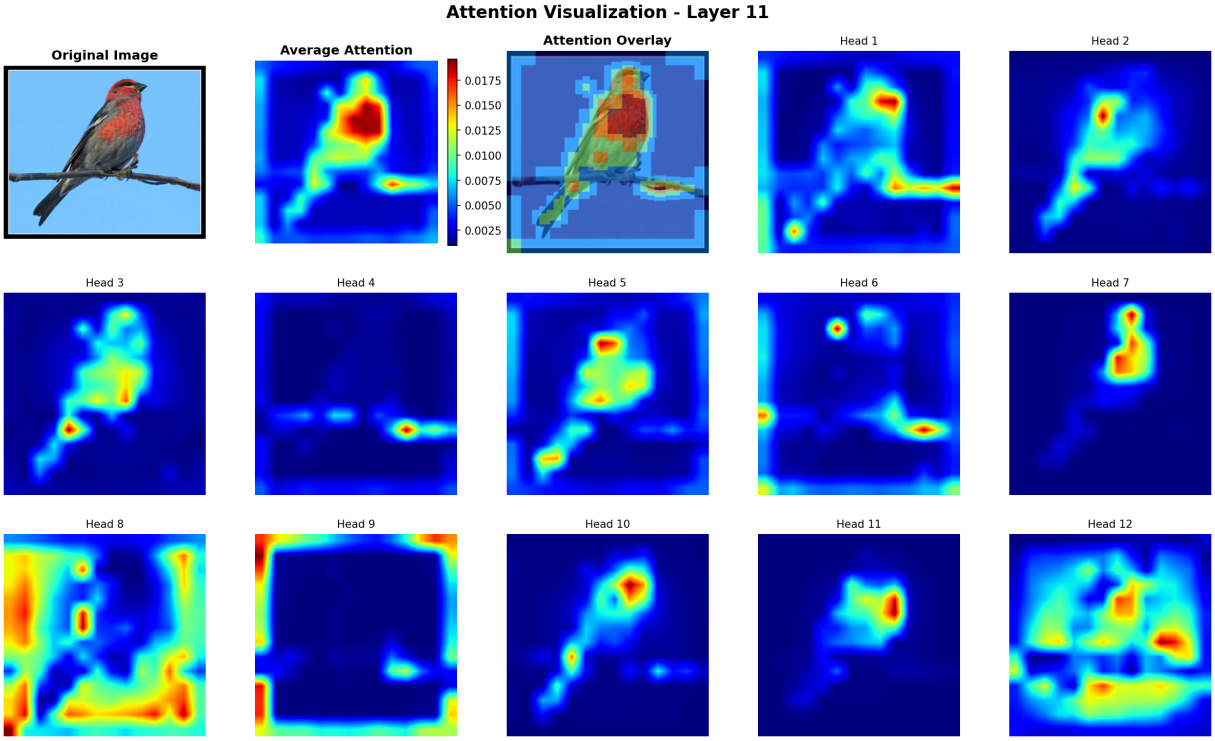

Figure 7: attentino 可视化

可以看出模型确实学到了重要的分类特征。

相比于 cnn 方法，ViT 方法明显训练难度更大，并且模型参数远远多于 CNN 方法，但使用纯粹的 attention 结构就能达到卷积一样的性能，再次证明了 Attention is all you need.

5 实验说明

实验的全部内容运行在 ubuntu 22.04 上，使用 RTX 3090 24G 进行训练。

References

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2021. URL https://arxiv.org/abs/2010.11929.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018. URL https://arxiv.org/abs/1810.12890.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. URL https://arxiv.org/abs/1512.03385.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez,

Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL https://arxiv.org/abs/1706.03762.