第5章 计算机视觉技术

计算机视觉技术是一种利用计算机和数学算法模拟人类视觉系统对图像和视频进行识别、理解、分析和处理的前沿科技。该技术融合了图像处理、模式识别、计算机图形学以及深度学习等多个领域的知识，旨在使计算机能像人类一样感知和理解视觉信息。深度学习为计算机视觉提供了一种强大的学习和表示方法，是计算机视觉领域的关键驱动力和技术基础。随着大数据、计算能力的提升和深度学习算法的进步，计算机视觉技术正快速发展，并逐步实现更多曾被视为科幻的应用。本章将深入剖析视觉模型的核心机制，包括分类任务的精细解析、目标检测任务的精准实现、图像分割任务的细致划分，并探索视觉自监督预训练的前沿技术，最后通过实战案例展示这些技术在视觉领域的广泛应用与探索。

5.1 视觉模型

视觉模型指的是用于完成计算机视觉任务的神经网络模型，主要分为卷积神经网络(Convolutional Neural Network，CNN)模型与视觉 Transformer(ViT)模型。

5.1.1 CNN 模型

CNN 是计算机视觉领域的核心模型之一，广泛应用于图像识别、目标检测、图像分割等任务。CNN 的引入极大地推动了计算机视觉的发展，是现代深度学习的重要里程碑。卷积神经网络能提取图像中的不同特征，并将图像最终表示为一个特征向量，随后可以通过特征向量对图像进行分类，如图 5-1 所示。

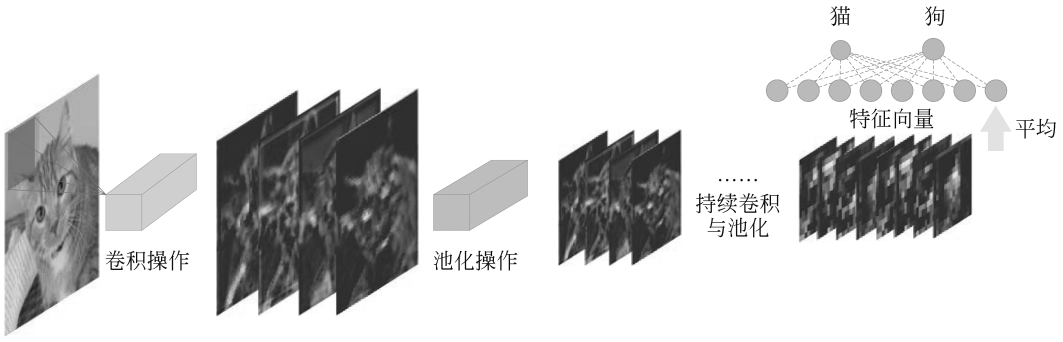

图 5-1 卷积神经网络的特征提取

卷积神经网络往往作为骨干网络(backbone network)。骨干网络又称为主干网络，在

计算机视觉领域，骨干网络通常指的是一个深度神经网络的主体部分，它负责从输入数据中提取特征。骨干网络是许多现代深度学习架构的核心，尤其在进行图像分类、目标检测和语义分割等时。

1）卷积神经网络的优势

CNN 通过卷积层能自动提取图像中的空间层次特征，从低级的边缘、纹理到高级的形状、语义信息，逐层构建对图像的理解。卷积操作利用了图像的局部相关性，通过权重共享的方式，大大减少了参数数量，使得网络更容易训练，也降低了模型的计算成本。

CNN 具有较强的平移不变性（translation invariance）。由于卷积操作在图像的每个位置都应用相同的滤波器，因此 CNN 能很好地识别出同一物体在不同位置的图像，从而提高了模型的鲁棒性。

此外，CNN 结构相对简单且易于扩展。通过堆叠更多的卷积层和池化层，CNN 可以设计出更深的网络，从而捕捉更复杂的图像特征。结合现代硬件的并行计算能力，CNN 可以在较短的时间内处理大规模图像数据集，训练速度快，应用范围广。

2）卷积神经网络的缺点

CNN 对旋转、缩放等变换的鲁棒性相对较差，虽然通过数据增强等方法可以部分缓解这一问题，但它仍然是 CNN 的一个局限性。此外，CNN 的卷积层虽然能提取局部特征，但难以捕捉长距离的依赖关系。因此，在处理具有复杂全局关系的任务时，CNN 的表现可能不如其他方法。

CNN 的训练需要大量的标注数据和计算资源。深度 CNN 模型通常包含数百万甚至数亿个参数，这使得训练变得非常耗时，并且对 GPU 等高性能硬件的依赖性较强。在没有大规模数据集的情况下，CNN 容易出现过拟合的问题，需要通过复杂的正则化方法和数据增强技巧缓解。

3）卷积神经网络的发展历程

CNN 的基础思想可以追溯到 20 世纪 80 年代[9]，当时 Yann LeCun 等在研究神经网络在图像识别中的应用。他们在 1989 年发表的论文中首次引入了 CNN，用于手写数字识别。这种早期的 CNN 模型被称为 LeNet-5，是为美国邮政服务开发的手写数字识别系统的一部分。LeNet-5 的成功证明了神经网络在图像处理上的潜力。Yann LeCun 也因此成为 2018 年图灵奖得主之一。

尽管 LeNet-5 在 20 世纪 90 年代取得了一些成功，CNN 在随后的十几年中并未得到广泛应用。直到 2012 年，随着计算能力的提升、大规模数据集的出现，以及新的训练技巧的引入，CNN 才迎来真正的突破。2012 年，Alex Krizhevsky、Ilya Sutskever 和 Geoffrey Hinton 提出的 AlexNet 模型在 ImageNet 图像识别竞赛中大获成功，标志着深度学习时代的到来。

AlexNet 与 LeNet-5 相比，显著增加了网络的深度和宽度，并引入了 ReLU（Rectified Linear Unit）激活函数，以加快训练速度。此外，AlexNet 还采用 Dropout 正则化方法来防止过拟合，并通过数据增强技术进一步提高模型的泛化能力。这些创新使得 AlexNet 在当时的数据集上大幅领先于其他方法，震惊了计算机视觉界。

自 AlexNet 之后，CNN 模型得到进一步的发展。2014 年，VGGNet 由牛津大学的研究团队提出，这一模型通过使用更深的网络（如 16 层或 19 层）进一步提高性能。VGGNet 采用了较小的 3×3 卷积核，并在网络末端添加了多个全连接层，进一步提高了图像分类的精

度。同年，Google 提出 GoogLeNet(Inception)，该模型引入了 Inception 模块，通过在同一层中使用多个不同大小的卷积核，捕捉多尺度的图像特征。

2015 年，ResNet(Residual Network)由微软研究院提出，再次刷新了图像分类的记录。ResNet 引入了残差块(Residual Block)的概念，通过跳过连接(Skip Connections)使得网络可以训练得更深(甚至达到 152 层)，同时避免了深度网络中常见的梯度消失问题。ResNet 的成功不仅奠定了其在计算机视觉中的地位，还启发了许多后续的网络设计。

随着移动设备的普及，人们对轻量化卷积神经网络的需求越来越高。MobileNet[10] 系列由 Google 于 2017 年提出，目标是设计一种计算量小但性能良好的模型，适用于移动和嵌入式设备。MobileNet 的核心创新在于深度可分离卷积(Depthwise Separable Convolution)，它将标准卷积分解为两个步骤：深度卷积(Depthwise Convolution)和逐点卷积(Pointwise Convolution)。这种方法大大减少了计算量和参数量，同时保留了较好的性能。MobileNet 的成功证明了轻量化卷积网络在资源受限的环境中依然可以达到较高的精度，并激发了后续(如 MobileNetV2、MobileNetV3)的进一步研究。这些模型继续优化了结构，提升了效率，并在图像分类、物体检测等任务中得到广泛应用。

EfficientNet[11] 于 2019 年提出，代表了一种系统化的网络架构搜索方法。EfficientNet 的核心思想是复合缩放(Compound Scaling)，即同时在网络的深度、宽度和分辨率三个维度进行缩放，以找到最优的模型规模。这种方法打破了以往依赖经验进行网络设计的局限，通过自动化搜索得到的模型在多个任务中表现出色。EfficientNet 的成功展示了自动化设计网络架构的潜力。通过复合缩放，EfficientNet 在减少参数量和计算量的同时，保持了甚至超越了更大模型的性能。这一设计思想不仅优化了 CNN 的效率，还推动了神经架构搜索(Neural Architecture Search，NAS)领域的研究。

ConvNeXt[12] 是 2022 年提出的一种改进的卷积神经网络架构，旨在结合 CNN 的优势和 Transformer 的先进理念。ConvNeXt 通过简化设计、增加层次以及引入一些从 Transformer 中借鉴的策略(如大尺寸卷积核)，在保留传统 CNN 优势的同时，实现了性能的提升。ConvNeXt 展示了现代卷积神经网络在与新兴架构的竞争中依然具有强大的生命力。

卷积神经网络是计算机视觉中最主流的模型，它的原理与实现已在第 2 章中介绍，这里不赘述。

5.1.2 ViT 模型

ViT(Vision Transformer)[13] 是一种基于 Transformer 的视觉分类模型，将自然语言处理领域的 Transformer 架构应用于计算机视觉任务。ViT 模型在 2020 年由谷歌研究人员提出，这一创新使传统的 CNN 在图像识别领域的主导地位受到挑战。

在 ViT 提出之前，CNN 一直是图像识别领域的主流模型。CNN 通过局部感受野捕捉图像的局部特征，并通过多层的卷积和池化操作逐步提取更高层次的特征。然而，CNN 在处理图像中的长距离依赖关系时存在局限性，因为卷积操作的范围有限。例如，如果图像的左上角区域与右下角区域的特征具有较强的相关性，CNN 仅凭局部的小窗口无法将其覆盖，只是卷积难以提取到这种长距离依赖的关系。而 Transformer 中的自注意力机制却能很好地解决此问题。

ViT 模型的设计思想来源于 NLP 领域中取得巨大成功的 Transformer 模型。Transformer 最初是为了解决序列到序列任务而提出的，它擅长处理包括翻译、文本生成和语言建模在内的各种 NLP 任务。其核心是自注意力机制（Self-Attention），可以高效地捕捉输入数据的全局依赖关系，而不依赖于传统的 RNN 或者 CNN。

ViT 的核心思想是利用 Transformer 处理图像数据，而不是传统的 CNN。在 ViT 中，图像首先被划分为固定大小的方块（patch），这些方块被展平成一维向量，然后被视作 Transformer 模型的输入“词嵌入”（类似于 NLP 中的词向量）。这些向量经过加权处理，作为输入被送入标准的 Transformer 架构中。ViT 的结构如图 5-2 所示。

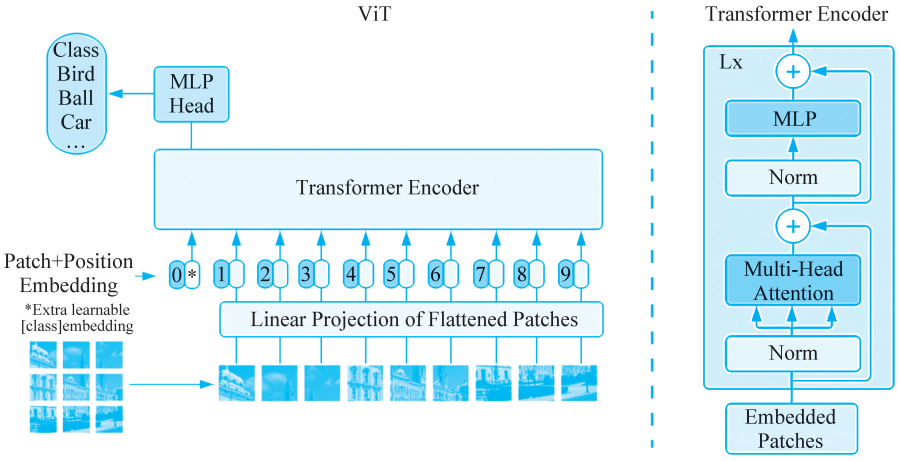

图 5-2　ViT 的结构

为了将 Transformer 应用于图像数据，ViT 模型做了几个关键修改。首先，它将输入的图像划分为固定大小的方块，比如 16×16 像素的方块。这些方块被展平成长度为 N 的向量，其中 N 取决于方块的大小和通道数。例如，对于一个 224×224 大小的 RGB 图像，每个方块包含 768 个特征值（16×16×3＝768）。这些方块向量被当作 Transformer 的输入，类似于文本中的单词。

然后，ViT 在每个方块向量前加上一个位置嵌入（Positional Embedding），用于保留小块在图像中的相对位置信息。这是因为 Transformer 自身不具备空间位置信息的感知能力，因此需要显式地添加这些信息。

ViT 能实现对图像的分类，还依靠一个重要的结构——分类标记（Class Token）。分类标记是一个人为定义的向量。若将输入 ViT 的图像序列视作时间序列，那么分类标记就是在时间序列前添加了一个新的时间点，这个新的时间点会通过 Transformer 中的自注意力机制与其他时间点（图像块）进行交互，捕获与其他时间点的相关性，以此进行特征提取。完成自注意力机制特征提取后，分类标记将被输入至全连接分类器中，得到最终的预测输出。

与传统的 CNN 相比，ViT 省略了诸如池化层、激活函数等部分，取而代之的是全连接的自注意力层。Transformer 的多头自注意力机制可以并行处理每个补丁的关系，从而在计算上具有优势。

为了简单演示 ViT 的具体实现，通过 PyTorch 自定义一个简易的 ViT 模型，代码如下。

```
from torch import nn
import torch
import math

class PositionalEncoding(nn.Module):
    def __init__(self, d_model, dropout, max_len=5000):
        super(PositionalEncoding, self).__init__()
        self.dropout = nn.Dropout(p=dropout)
        #位置编码矩阵,维度[max_len, embedding_dim]
        pe = torch.zeros(max_len, d_model)
        #单词位置
        position = torch.arange(0.0, max_len)
        position.unsqueeze_(1)
        #使用 exp 和 log 实现幂运算
        div_term = torch.exp(torch.arange(0.0, d_model, 2) * (- math.log(1e4) /
d_model))
        div_term.unsqueeze_(0)
        #计算单词位置沿词向量维度的纹理值
        pe[:, 0:: 2] = torch.sin(torch.mm(position, div_term))
        pe[:, 1:: 2] = torch.cos(torch.mm(position, div_term))
        #增加批次维度,[1, max_len, embedding_dim]
        pe.unsqueeze_(0)
        #将位置编码矩阵注册为 buffer(不参加训练)
        self.register_buffer('pe', pe)

    def forward(self, x):
        #将一个批次中语句的所有词向量与位置编码相加
        x += self.pe[:, : x.size(1), :]
        return self.dropout(x)

class ViT(nn.Module):
    def __init__(self, patch_size=16, num_classes=10, dim=512, depth=6, heads=
   8, mlp_dim=1024, dropout=0.1):
        super(ViT, self).__init__()
        self.dim = dim
        #利用卷积对图像进行分块
        self.patch_embedding = nn.Conv2d(in_channels=3, out_channels=dim,
        kernel_size=patch_size, stride=patch_size)
        #位置编码
        self.pos_embedding = PositionalEncoding(dim, dropout)
        #分类标记
        self.cls_token = nn.Parameter(torch.randn(1, 1, dim))

        #Transformer Encoder
        self.transformer = nn.TransformerEncoder(
             nn.TransformerEncoderLayer(dim, heads, mlp_dim, dropout, batch_
             first=True), num_layers=depth
        )
        self.to_cls_token = nn.Identity()

        #分类头
        self.mlp_head = nn.Sequential(
            nn.LayerNorm(dim),
```

```
            nn.Linear(dim, num_classes)
        )

    def forward(self, x):
        #对图像进行分块
        x = self.patch_embedding(x)
        #将分块后的图像拉平
        x = torch.flatten(x.permute(0, 2, 3, 1), 1, 2)
        print(x.shape)

        #将分类标记添加到序列中
        cls_tokens = self.cls_token.expand(x.size(0), -1, -1)
        x = torch.cat((cls_tokens, x), dim=1)

        #位置编码
        x = self.pos_embedding(x)
        #编码器
        x = self.transformer(x)
        #获取分类标记的计算结果
        x = self.to_cls_token(x[:, 0])
        #返回分类结果
        return self.mlp_head(x)
```

1）ViT 模型的优势

ViT 模型具有几个显著的优点。首先，由于它摒弃了 CNN 中局部感受野的限制，能在处理过程中捕获更全局的特征，这使得 ViT 在大规模数据集上表现出色。其次，ViT 模型结构更简单，没有复杂的卷积和池化操作，易于实现并行计算，这对于完成大规模训练任务尤其有利。

ViT 还展示了良好的扩展性，当训练数据足够大时，模型可以显著提升精度。研究表明，ViT 在 ImageNet-21k 等大规模数据集上的表现优于基于 CNN 的模型，尤其是在数据丰富的情况下，其优势更加明显。

2）ViT 模型的不足

ViT 也存在一些缺点。首先，ViT 对大规模数据集的依赖性较强。在较小的数据集上，ViT 的性能可能不如传统的 CNN 模型，因为缺乏足够的数据，可能导致 Transformer 难以学习到有用的特征。此外，ViT 对计算资源的要求较高。尽管其结构较为简洁，但 Transformer 的自注意力机制需要大量的计算资源，尤其是在高分辨率图像或大模型规模的情况下。

3）ViT 模型的意义

ViT 模型的提出标志着计算机视觉领域的一次重要变革。它展示了 Transformer 架构的广泛适用性，不仅在自然语言处理（NLP）领域中表现出色，也能成功应用于视觉任务。这为研究者提供了一种新的思路，即不必拘泥于传统的卷积操作，完全可以探索其他架构来处理图像数据。

从更广的视角看，ViT 推动了跨领域架构设计的趋势，即借鉴不同领域的成功模型解决新的问题。ViT 的成功也激发了大量后续研究，包括混合模型（如使用 CNN 与

Transformer 结合)和其他完全基于 Transformer 的视觉模型。这些研究不断推进着计算机视觉领域的发展,为更强大的视觉理解模型奠定了基础。

5.2 分类任务

在计算机视觉中,分类任务是指通过算法将输入图像归类到预定义的类别中,这是人工智能和机器学习中的一项基本任务。图像分类任务的核心目标是使模型能自动识别并分类图像中的内容,从而实现对视觉数据的智能处理。

图像分类任务通常分为两类:单标签分类(Single-label Classification)和多标签分类(Multi-label Classification)。单标签分类是最常见的一种形式,即每幅输入图像被分配到一个唯一的类别中。而多标签分类则更加复杂,允许每幅输入图像被分配到多个类别中,这种分类方式更贴近一些复杂的现实场景。

5.2.1 单标签分类

单标签分类是图像分类中最基础的一种任务形式。在这种任务中,模型需要从多个预定义类别中选择一个最符合输入图像的类别。

单标签分类任务的特点是每幅图像只能属于一个类别,这在许多场景中是合理且高效的。训练这样的分类模型通常需要一个带有明确标签的图像数据集,标签通常是图像所属类别的标记。模型通过对数据集的学习,能掌握各个类别的特征,并在预测阶段根据这些特征对新的图像进行分类。例如,如图 5-3 所示,在动物分类任务中,预先定义了四个类别,分别为“猫”“狗”“牛”“羊”,每幅图像只有一个标签,只能是预定义的四个类别中的一个,即图 5-3 中的图像标签只能有一个,并且为“猫”。图像分类任务无法识别预定义类别中不存在的类别。

图 5-3 单标签数据

在单标签分类任务中,损失函数是用来衡量模型预测结果与实际标签之间差距的一个函数。通过最小化损失函数的值,模型可以逐渐调整其参数,以提高分类精度。在单标签分类任务中,最常用的损失函数是交叉熵损失(Cross-Entropy Loss)。交叉熵损失函数用于比较模型预测的概率分布与实际类别的真实分布。当模型预测与真实标签完全一致时,交叉熵损失的值最小,表示模型的分类结果最准确。交叉熵损失函数的公式如下。

$$\text{CrossEntropy}(\hat{y},y)=\sum_{i=1}^{N}y_i\log(\hat{y}_i) \tag{5-1}$$

其中，y 是真实标签，$\hat{y}$是模型预测的概率值，N 是类别的总数。式(5-1)表明，模型的预测概率$\hat{y}_i$ 与真实类别 y_i 越接近，损失值越小。通过反向传播算法，模型会逐步调整其权重，使得损失函数值逐渐降低，进而提高分类的准确性。

5.2.2 多标签分类

多标签分类是一种更为复杂的分类任务。在这种任务中，一幅图像可以同时属于多个类别。这种分类方式更加贴近现实生活中的一些复杂场景。例如，在自动驾驶场景中，一幅道路图像可能同时包含车辆、行人、交通标志等多个物体，这些物体需要被同时识别和分类。再如，在医疗影像中，一幅图像可能同时显示多种病灶，这些病灶都需要被准确地标记出来。

假设有一幅小猫的图像，这幅图像可能需要被分类为多个类别，比如小猫的品种、毛色、瞳孔形状等，如图 5-4 所示。对于这幅图像，它可能被同时标记为“品种：橘猫”、“毛色：橘色”、“瞳孔形状：菱形”等类别。这意味着，模型需要同时预测多个属性，每个属性对应一个类别标签。常规的单标签分类方法无法处理这种情况，因为它们只能为每幅图像分配一个唯一的类别标签。因此，多标签分类模型必须能在多个类别之间进行独立的判断，并且对每个类别都给出一个相应的预测。

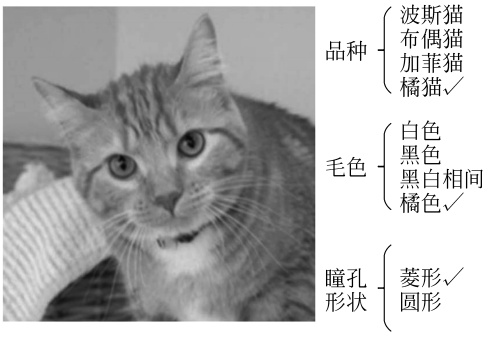

图 5-4 多标签数据

在多标签分类任务中，损失函数也要相应地进行修改。多标签分类任务中，模型具有多个预测输出，在图 5-4 展示的情形中，模型一共有三个输出，三个输出分别用于预测品种、毛色、瞳孔形状。记品种的预测为 $\hat{y}_1$，品种的标签为 y_1；毛色的预测为$\hat{y}_2$，毛色的标签为 y_2；瞳孔形状的预测为$\hat{y}_3$，瞳孔形状的标签为 y_3，则多标签分类任务的损失函数可记为

$$\begin{aligned}\text{Loss}=&\text{CrossEntorpy}(\hat{y}_1,y_1)+\text{CrossEntorpy}(\hat{y}_2,y_2)+\\&\text{CrossEntorpy}(\hat{y}_3,y_3)\end{aligned} \tag{5-2}$$

每种标签在计算交叉熵损失时相互独立，互不影响。每种标签的损失计算完成后，将三种损失相加到一起，形成总损失 Loss，再将总损失 Loss 进行方向传播，即可完成多标签任务的训练。

1. CNN 的多标签分类

使用 CNN 进行多标签分类，通常可以使用以下两种结构。

1）多流卷积神经网络

多流卷积神经网络由多个子卷积神经网络组成，每个子卷积神经网络负责一种标签的预测。每个子神经网络的输入都为待预测的图像，输出为各自负责预测的类别，如图 5-5 所示。

图 5-5 多流卷积神经网络

这种卷积神经网络通常可以获得较高的准确率，但神经网络的规模过于庞大，资源消耗过多，且多个标签之间不存在信息交互，忽略了标签之间的相关性。

2）多分类层

另一种思路是定义多个分类层进行分类。在单标签分类任务中，卷积层负责图像特征的提取，将图像转换为一个特征向量，分类层再依据这个特征向量进行分类。在多标签分类任务中，可以定义多个分类层用于输出不同标签，多个分类层使用相同的特征向量进行分类，如图 5-6 所示。

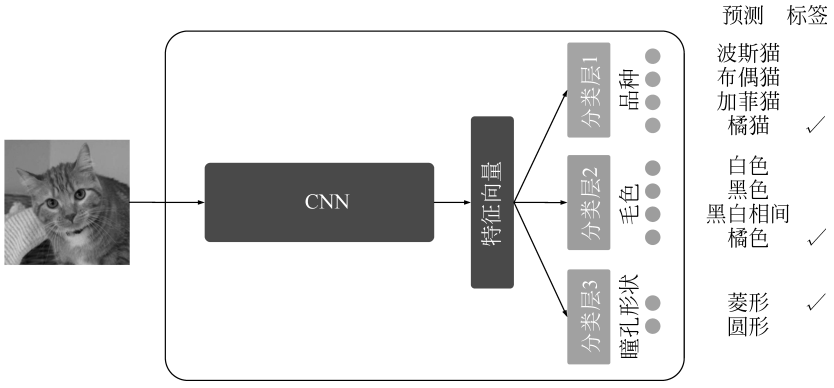

图 5-6 多分类层

这种方式可能在一定程度上降低准确率，但资源消耗少，计算量也相对较少。由于多个分类层共享一个特征向量，这种方式对特征向量的要求较高，因此需要设计功能强大的 CNN 模块来提取特征。

2. ViT 的多标签分类

ViT 的多标签分类也需要对模型进行修改，如图 5-7 所示。用 ViT 进行单标签分类时，

定义了一个分类标记，用于提取特征；相应地，进行多标签分类时，可以定义多个分类标记，再使用多个分类层分别对每个分类标记进行分类，即可得到多标签分类结果。

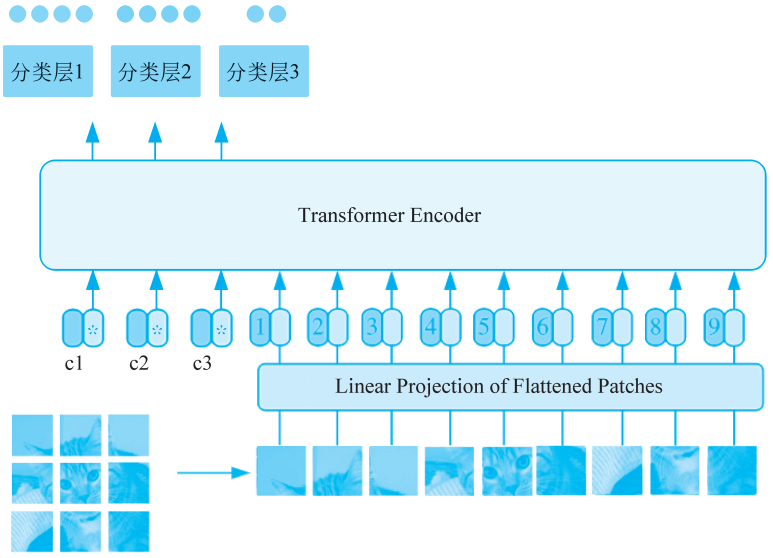

图 5-7　ViT 的多标签分类

ViT 的多标签分类优势在于，分类时不依靠共享的特征向量，而是在自注意力机制中，多个分类标记各自提取特征，同时，多个分类标记之间也可以通过自注意力机制提取特征，捕获分类标记之间的相关性。

5.2.3　分类任务的评估指标

在分类任务中，评估模型性能至关重要，而不同的评估指标则提供了不同的视角来衡量模型的表现。

1. 混淆矩阵

混淆矩阵是分类任务中用来评价分类模型性能的基础工具。它通过总结模型的预测结果与实际情况的对应关系，直观地展示了模型的分类能力。混淆矩阵通常用于二分类任务，但也可以扩展到多分类任务。

在二分类任务中，混淆矩阵由一个 2×2 的表格组成，表格的每个单元表示模型的预测结果与真实情况的组合，如表 5-1 所示。具体来说，混淆矩阵包括以下四个元素。

表 5-1　二分类中的混淆矩阵

真实样本	预测样本	
	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

- True Positive(TP)：模型正确预测为正类的样本数量。
- True Negative(TN)：模型正确预测为负类的样本数量。