

基于图像的垃圾分类

姓 名: 周 子 凯

学 号: 1120212546

班 号: 2148

深度学习与计算机视觉

北京理工大学

计算机学院

数据科学与大数据技术全英文

2023 年 11 月 28 日

目 录

1 概述	3
2 预备知识	3
2.1 特征提取方法	3
2.1.1 SIFT 算法	4
2.1.2 ORB 算法	5
2.1.3 HOG 算法	6
2.1.4 卷积核	7
2.2 传统机器学习方法	8
2.2.1 Logistic Regression	8
2.2.2 Support Vector Machine	8
2.2.3 K-Means	9
2.2.4 Random Forest	10
2.3 深度学习方法	10
2.3.1 Pyramidnet	11
2.3.2 Resnet	11
3 实验设计	13
3.1 数据处理	13
3.1.1 数据划分	13
3.1.2 数据增强	13
3.1.3 数据归一化	14
3.1.4 优化器和学习率调节器	15
3.2 传统机器学习	15
3.3 深度学习	15
4 总结	16

1 概述

垃圾分类作为一项重要的环保举措，是在日益增长的城市化和工业化过程中应对不断增加的废弃物问题的一种关键策略。一方面，垃圾分类能够解决废弃物管理问题，能够减缓环境压力；另一方面，垃圾中蕴含着丰富的可再生资源，其回收再利用能缓解资源紧张问题，促进可持续发展。然而，随着人口的增加和生活水平的提高，垃圾的产生量不断攀升，垃圾种类日益丰富，导致传统的垃圾分类方法难以应对这一挑战。在这一背景下，如何有效实施垃圾分类与回收，成为一个我们迈向绿色环保、可持续发展的社会所亟需解决的问题。

与此同时，机器学习技术在近些年中取得了显著的进展与发展，推动了人工智能在图像识别、自然语言处理、语音识别、医学诊断等领域的广泛应用。通过使用机器学习技术，能够大大实现工程任务自动化、准确化，极大提高人类生活的效率。

机器学习算法与垃圾分类任务相结合存在一定的必要性和重要性：

- 机器学习算法能够应对复杂多变的垃圾物品。
- 机器学习算法能够实现垃圾特征的自动提取。
- 机器学习算法能够大规模处理数据。
- 机器学习算法具有实时性和准确性。

在这个实验报告中，我们主要去探讨机器学习技术应用在垃圾分类任务上的可行性，并通过传统机器学习的算法以及深度学习算法之间的实验对比，去探索基于机器学习算法的垃圾分类器的前景。

2 预备知识

为了帮助读者理解实验的背景和相关理论知识，以便其更好地分析实验结果。我在这里将会介绍一些相关的理论知识与实验所用的算法模型的介绍。

2.1 特征提取方法

特征提取在机器学习和深度学习中起着至关重要的作用。原始数据往往包含着大量冗余信息或者噪声，而通过特征提取可以将数据转换为更为紧凑、高效的表现形式。此外，有效的特征提取有助于减少模型训练所需的样本数量，提高模型的泛化能力。由于提取的特征往往具有实际的意义，能

够更好地反映数据的本质特征，这样能够使得模型的输出更易于理解和解释，能够提升模型的可靠性。

2.1.1 SIFT 算法

尺度不变特征转换 (Scale-Invariant Feature Transform, SIFT) [1] 算法由 David G. Lowe 在 1999 年首次提出，并在 2004 年的论文《Distinctive Image Features from Scale-Invariant Keypoints》中进行了深入的阐述。SIFT 是一种用于图像处理和计算机视觉中的特征提取算法，其主要目的是在图像中寻找出具有显著性质的关键点，并生成这些关键点的描述子，以便进行图像匹配、目标识别等任务。

SIFT 算法的主要步骤如下：

尺度空间极值点检测：SIFT 算法首先通过高斯差分函数构建图像的尺度空间，然后在不同尺度上使用高斯拉普拉斯算子 (LoG) 来检测图像中的极值点，这些极值点通常对应于图像中的关键点。在不同尺度上检测极值点的目的是为了实现在尺度不变性，使得 SIFT 特征对于图像中不同尺度的目标都具有良好的描述能力。

关键点定位：在检测到极值点之后，SIFT 算法会对这些极值点进行精确定位，以获得更加准确的关键点位置。为了实现关键点的稳定性和鲁棒性，SIFT 算法会对检测到的极值点进行精细化的位置和尺度调整，同时通过去除低对比度的极值点和边缘响应较大的极值点来提高关键点的质量。

方向分配：对于每个关键点，SIFT 算法会计算其周围像素的梯度方向，并在周围区域内构建梯度方向直方图。然后，SIFT 算法会选择关键点周围梯度方向直方图中的主导方向作为关键点的方向，以实现旋转不变性。

关键点描述：最后，SIFT 算法会基于关键点周围的梯度信息构建关键点的描述符。这里使用了一种称为高斯加权直方图的方法来描述关键点周围的梯度分布情况，通过这种方式生成的描述符具有很好的不变性和区分性能。

SIFT 算法通过尺度空间极值点检测、关键点定位、方向分配和关键点描述这几个步骤，能够提取出具有尺度不变性和旋转不变性的图像特征，因此在图像匹配、目标识别等任务中具有很好的性能。下面是实验所用 SIFT 代码：

```
class Sift:
    def __init__(self, name='sift'):
        self.name = name

    def extract_sift(self, image):
```

```

        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        sift = cv2.SIFT_create()
        keypoints= sift.detect(gray, None)
        keypoints, descriptors = sift.compute(gray, keypoints)

        descriptors = descriptors.mean(0)
        return descriptors

    def extract(self, image):
        sift_features = self.extract_sift(image)

        return sift_features

```

2.1.2 ORB 算法

ORB (Oriented FAST and Rotated BRIEF) [2] 是一种结合了 FAST 关键点检测和 BRIEF 特征描述的算法，它能够快速且准确地提取图像特征。下面是 ORB 算法的主要原理和算法步骤：

FAST 关键点检测：ORB 算法首先使用 FAST (Features from Accelerated Segment Test) 算法来检测图像中的关键点。FAST 算法是一种快速的角点检测算法，它通过比较像素周围的强度值来识别角点，从而快速找到图像中的关键点。

关键点方向分配：与 SIFT 类似，ORB 算法也会对检测到的关键点进行方向分配，以实现旋转不变性。这一步通常使用一种基于图像梯度的方法来计算关键点的主导方向。

BRIEF 特征描述：在确定了关键点的位置和方向之后，ORB 算法使用 BRIEF (Binary Robust Independent Elementary Features) 描述符来描述关键点周围的图像信息。BRIEF 描述符是一种二进制的局部特征描述方法，它通过比较关键点周围的像素对的亮度信息，生成一个具有较小维度的二进制特征向量。

方向不变性：为了实现旋转不变性，ORB 算法通常会在关键点周围生成一组具有不同方向的 BRIEF 描述符，然后选择其中最具代表性的描述符作为最终的关键点描述符。

ORB 算法通过 FAST 关键点检测、关键点方向分配和 BRIEF 特征描述这几个步骤，能够快速而准确地提取图像特征，并具有较低的计算成本。由于 BRIEF 描述符是二进制的，因此 ORB 算法在匹配和识别任务中具有较高的效率和速度。下面是实验所用 ORB 代码：

```

class Orb:
    def __init__(self, name='orb'):
        self.name = name

```

```

def extract_orb(self, image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    orb = cv2.ORB_create()
    keypoints, descriptors = orb.detectAndCompute(gray, None)
    descriptors = descriptors.mean(0)
    return descriptors

def extract(self, image):
    orb_features = self.extract_orb(image)

    return orb_features

```

2.1.3 HOG 算法

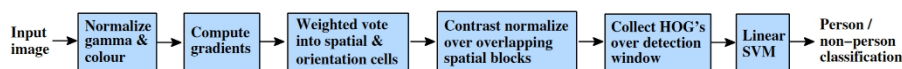


图 1: HOG 算法的流程图

HOG (Histogram of Oriented Gradients) [3] 是一种用于图像识别和目标检测的特征描述方法，它通过计算图像局部区域的梯度方向直方图来描述图像的特征。下面是 HOG 算法的主要原理和算法步骤, 同时图 1 也介绍了算法的基本流程:

图像预处理: 首先, 对输入的图像进行预处理, 通常包括图像的灰度化、归一化和去除噪声等操作。这些预处理步骤有助于提高 HOG 算法对图像的稳定性和鲁棒性。

计算图像梯度: HOG 算法通过计算图像中每个像素点的梯度信息来描述图像的局部特征。通常采用 Sobel 算子或 Prewitt 算子等方法来计算图像的水平 and 垂直方向的梯度。

图像分割: 将图像分割成小的局部区域 (cell), 每个局部区域通常包含多个像素。这些局部区域可以是正方形或矩形的, 通常情况下, 它们是相互重叠的。

计算梯度方向直方图: 对于每个局部区域, HOG 算法会计算其内部像素的梯度方向, 并将梯度方向分布在若干个方向区间内。然后, 将这些方向信息组合成一个梯度方向直方图, 用来描述该局部区域的梯度分布情况。

归一化: 为了提高 HOG 特征的鲁棒性, 通常会对每个局部区域的梯度方向直方图进行归一化处理, 例如通过对梯度方向直方图进行块归一化或整体归一化等方法。

特征向量构建：将所有局部区域的归一化梯度方向直方图组合成一个大的特征向量，该特征向量即为图像的 HOG 特征描述符。

HOG 算法通过计算图像梯度、分割图像、计算梯度方向直方图、归一化和构建特征向量这几个步骤，能够提取出具有较好鲁棒性的图像特征，适用于目标检测、行人检测等任务。下面是实验所用 HOG 代码：

```
class Hog:
    def __init__(self, name='hog'):
        self.name = name

    def extract_hog(self, image):
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        hog = cv2.HOGDescriptor()
        descriptors = hog.compute(gray)
        return descriptors

    def extract(self, image):
        hog_features = self.extract_hog(image)

        return hog_features
```

2.1.4 卷积核

卷积核是卷积神经网络（CNN）中的一个重要组成部分，它通过卷积操作来提取图像的特征。下面是卷积核提取特征的原理：

局部感受野：卷积核在输入图像上进行滑动卷积操作，每次卷积操作都只涉及输入图像的一个局部区域，这个局部区域称为卷积核的局部感受野。通过对局部感受野进行卷积操作，卷积核可以提取出该区域的特征。

特征提取：卷积核的数值权重决定了它所提取的特征类型。在卷积操作中，卷积核的权重与输入图像的像素值进行相乘累加，得到卷积结果。通过设计不同的卷积核，可以提取出不同的特征，例如边缘、纹理、颜色等。

参数共享：卷积核的参数是共享的，即在整个输入图像上，每个位置都使用相同的卷积核进行卷积操作。这样可以大大减少需要学习的参数数量，同时提高了模型的泛化能力。

多通道卷积：对于多通道的输入图像，卷积核也是多通道的，它可以同时提取不同通道上的特征，并将它们组合起来形成新的特征图。

卷积核通过局部感受野、特征提取、参数共享和多通道卷积等原理，能够高效地提取图像的局部特征，这些局部特征在卷积神经网络中逐渐组合、抽象，最终形成对图像整体特征的表示。因此，卷积核是 CNN 中非常重要的特征提取器，它的设计和学习对于网络的性能具有重要影响。

2.2 传统机器学习方法

传统机器学习是指在深度学习兴起之前的一类机器学习方法，它主要包括了一系列经典的监督学习、无监督学习和强化学习方法。传统机器学习方法通常依赖于手工设计的特征提取器，并且使用相对较浅的模型来进行学习和预测。

2.2.1 Logistic Regression

逻辑回归 (Logistic Regression) 是一种用于解决分类问题的线性模型。虽然名字中包含“回归”一词，但实际上逻辑回归是一种分类算法，用于预测输入变量与输出变量之间的关系，并将输入变量映射到离散的输出类别。

逻辑回归的原理和算法步骤如下：

原理：逻辑回归通过将线性模型的输出通过一个逻辑函数（也称为 Sigmoid 函数）进行映射，将连续的预测值转换为概率值，然后根据阈值将概率值转换为类别标签。

其中 Sigmoid 函数的数学表达式为： $g(z) = \frac{1}{1+e^{-z}}$ ，其中 $z = w^T x + b$ 为线性模型的输出， w 为权重向量， x 为输入特征向量， b 为偏置项。

算法步骤：

1. 输入数据：给定包含特征变量和对应类别标签的训练数据集。
2. 特征权重初始化：初始化特征权重向量 w 和偏置项 b 。
3. 损失函数：定义逻辑回归的损失函数，通常采用对数损失函数 (Log Loss) 来衡量预测概率与真实标签之间的差异。
4. 梯度下降优化：通过梯度下降等优化算法，最小化损失函数，更新权重 w 和偏置 b 。
5. 预测：使用训练好的模型对新的输入数据进行预测，通过计算 Sigmoid 函数得到概率值，再根据阈值将概率值转换为类别标签。

逻辑回归的优点包括模型简单、易于解释、计算效率高等，适用于二分类问题。然而，逻辑回归也有一些局限性，例如它只能解决线性可分问题，对于非线性数据需要进行特征工程或者使用多项式特征扩展。

2.2.2 Support Vector Machine

支持向量机 (Support Vector Machine, SVM) 是一种用于解决分类和回归问题的监督学习算法。SVM 的原理和算法步骤如下：

原理：SVM 的目标是找到一个最佳的超平面，将不同类别的样本分隔开，并且使得两个类别的样本间隔最大化。在线性可分的情况下，SVM 通过最大化间隔来找到最佳的超平面。在线性不可分的情况下，SVM 通过引

入核函数将数据映射到高维空间，从而在高维空间中找到一个最佳的超平面来分隔数据。

算法步骤：

1. 输入数据：给定包含特征变量和对应类别标签的训练数据集。
2. 特征标准化：对特征变量进行标准化处理，使得数据的均值为 0，方差为 1。
3. 选择核函数：根据数据特点选择合适的核函数，常用的核函数包括线性核、多项式核、高斯核等。
4. 计算间隔最大化：通过求解凸优化问题，找到最大间隔超平面，使得支持向量到超平面的距离最大化。
5. 核函数计算：对于非线性问题，使用核函数将数据映射到高维空间。
6. 模型训练：根据训练数据集，计算出最优的超平面参数，包括权重向量和偏置项。
7. 预测：使用训练好的模型对新的输入数据进行预测，根据超平面的位置将输入数据归为不同的类别。

SVM 的优点包括对于高维数据的处理能力强、泛化能力好、在小样本数据上表现良好等。然而，SVM 的缺点包括对大规模数据的处理能力较弱、对参数敏感、需要选择合适的核函数等。

2.2.3 K-Means

K-Means 是一种常见的聚类算法，用于将数据集中的样本分成 K 个不同的簇。其原理和算法步骤如下：

原理：K-Means 算法的目标是将数据集划分为 K 个簇，使得每个样本点都属于离它最近的簇的中心。算法通过不断迭代更新簇的中心和重新分配样本点的簇标签，直到满足停止条件为止。K-Means 算法基于样本之间的距离来进行簇的划分，通常采用欧氏距离来衡量样本之间的相似度。

算法步骤：

1. 初始化：随机选择 K 个样本作为初始的簇中心。
2. 分配样本到最近的簇：对于每个样本，计算其与各个簇中心的距离，将其分配到距离最近的簇中心所对应的簇。
3. 更新簇中心：对于每个簇，计算其所有样本的均值，将均值作为新的簇中心。
4. 重复迭代：重复进行分配样本和更新簇中心的步骤，直到簇中心不再发生变化或者达到预定的迭代次数。
5. 输出结果：最终得到 K 个簇，每个簇包含一组样本，以及对应的簇中心。

K-Means 算法的优点包括简单、易于理解和实现，对于大规模数据集也有较好的扩展性。然而，K-Means 算法也有一些缺点，例如对初始簇中心的选择敏感，对异常值敏感，对于非凸形状的簇划分效果不佳等。

2.2.4 Random Forest

Random Forest（随机森林）是一种集成学习方法，通过构建多棵决策树来进行分类或回归。它的原理和算法步骤如下：

原理：随机森林由多棵决策树组成，每棵决策树都是基于随机抽取的子样本和特征构建的。在构建每棵决策树时，随机森林通过自助采样(bootstrap sampling)从原始训练集中抽取多个不同的子样本，然后使用这些子样本来构建决策树。在每个节点处，随机森林会随机选择一部分特征来进行分裂，这有助于增加树之间的差异性。最终的分类（或回归）结果是通过所有决策树的投票（分类）或平均（回归）得到的。

算法步骤：

1. 构建子样本：从原始训练集中进行自助采样，生成多个不同的子样本。
2. 构建决策树：对于每个子样本，使用决策树算法（如 CART）构建一棵决策树，但在每个节点处只考虑部分特征。
3. 随机选择特征：在构建决策树的过程中，每次分裂节点时，随机选择一部分特征进行考虑。
4. 构建多棵决策树：重复上述步骤，构建多棵决策树。
5. 输出结果：对于分类问题，随机森林通过投票来确定最终的分类结果；对于回归问题，随机森林通过平均来确定最终的回归结果。

随机森林的优点包括对高维数据和大规模数据的处理能力强，对异常值不敏感，具有较好的泛化能力等。然而，随机森林也有一些缺点，如模型解释性较差，对噪声数据敏感等。

2.3 深度学习方法

深度学习是一种机器学习的子领域，它使用人工神经网络来模拟和学习复杂的数据表示。深度学习的核心思想是通过多层非线性变换来学习数据的高层抽象表示，从而实现对复杂模式和关系的学习。其相比传统的机器学习方法的优势在于，能够自动的去提取特征，不需要通过人为的先验知识来手工提取特征。深度学习已经在图像识别、语音识别、自然语言处理、推荐系统等领域取得了重大突破，并在许多领域取代了传统的机器学习方法。

2.3.1 Pyramidnet

PyramidNet 是由 Google Brain [4] 提出的一种深度神经网络架构,旨在解决深度神经网络训练中的梯度消失和训练速度慢的问题。PyramidNet 的核心思想是通过引入渐进式的深度结构,以及密集连接 (dense connection),来提高网络的训练速度和准确性。

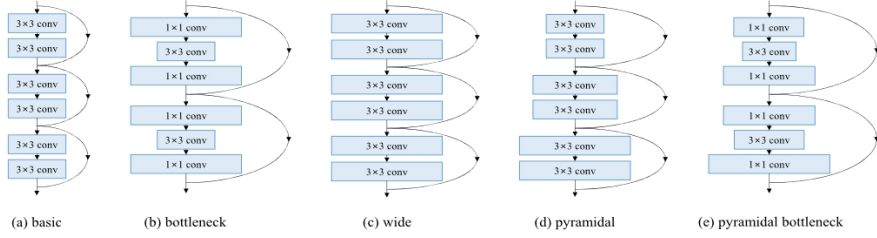


图 2: Pyramidnet 网络的基本组成单元

在 PyramidNet 中 (如图 2 所示), 网络的结构被设计成金字塔形状, 即网络的层数和通道数随着深度的增加而递增或递减。这种设计使得网络在不增加参数数量的情况下, 可以拥有更深的结构, 从而提高了网络的表征能力。此外, PyramidNet 还采用了密集连接 (Dense Connection) 的思想, 即每个层都与前面的所有层相连接, 这种结构有助于信息的传递和梯度的反向传播, 提高了网络的训练效率。

PyramidNet 的基本单元是由若干层卷积层和批量归一化层组成的“Pyramid Block”, 在每个 Pyramid Block 中, 通过增加通道数和减小分辨率的方式, 使得网络可以学习到不同尺度和抽象层次的特征表示。通过堆叠多个 Pyramid Block, 可以构建出深度和通道递增或递减的金字塔形网络结构。

PyramidNet 在图像分类、目标检测等领域取得了较好的性能, 尤其在大规模数据集进行训练时, PyramidNet 能够更快地收敛并达到更好的性能。同时, PyramidNet 也在一定程度上解决了深度神经网络训练中的梯度消失和训练速度慢的问题。

PyramidNet 通过引入金字塔形状的网络结构和密集连接的设计, 提高了网络的表征能力和训练效率, 取得了在图像分类和目标检测等任务上的良好表现。

2.3.2 Resnet

ResNet (Residual Network) [5] 是一种深度神经网络架构, 由微软亚洲研究院的 Kaiming He 等人提出。ResNet 的核心思想是通过引入残差连接

(residual connection), 解决了深度神经网络训练过程中的梯度消失和梯度爆炸问题, 使得可以训练非常深的网络。

在传统的深度神经网络中, 随着网络层数的增加, 网络性能通常会出现饱和甚至下降的情况。这是因为在反向传播过程中, 梯度的传播会受到网络深度的影响, 导致梯度消失或梯度爆炸的问题。ResNet 通过引入残差连接, 将网络的输入直接与输出相加, 从而使得网络可以学习残差映射 (即输入与输出之间的差异), 而不是直接学习原始的映射关系。这种设计可以使得网络更容易学习恒等映射 (即将输入直接映射到输出), 从而缓解了梯度消失和梯度爆炸的问题。

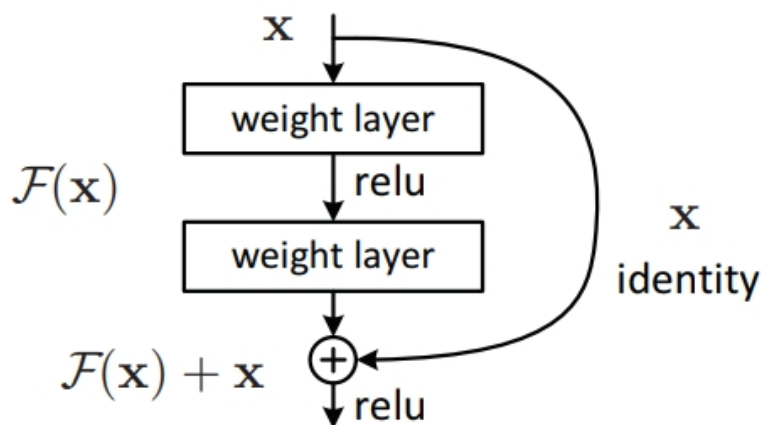


图 3: Resnet 残差块的基本组成

ResNet 的结构主要包括基本块 (building block) 和残差块 (residual block) (如图 3 所示)。基本块由若干层卷积层和批量归一化层组成, 而残差块则在基本块的基础上添加了残差连接。ResNet 通常采用堆叠多个残差块来构建深层网络, 通过堆叠不同深度的残差块, 可以构建出非常深的神经网络。

ResNet 在图像识别领域取得了巨大成功, 尤其是在 ImageNet 图像识别比赛中取得了优异的成绩。此外, ResNet 的残差连接思想也被广泛应用于其他深度学习模型的设计中, 成为了深度学习领域的重要突破之一。

3 实验设计

3.1 数据处理

3.1.1 数据划分

在深度学习中，通常需要将数据集划分为训练集、验证集和测试集，以便进行模型的训练、调参和评估。这种数据划分的目的是为了保证模型在训练过程中能够充分学习和泛化，同时能够对模型的性能进行准确的评估。

1. 训练集 (Training Set): 训练集是用来训练深度学习模型的数据集。模型通过训练集中的样本进行参数更新和学习，从而逐渐提高对数据的拟合能力。

2. 验证集 (Validation Set): 验证集是用来评估模型在训练过程中的性能表现的数据集。在训练过程中，可以使用验证集对模型进行调参和选择最佳的超参数，以提高模型的泛化能力。

3. 测试集 (Test Set): 测试集是用来评估训练好的模型在未见过的数据上的性能的数据集。测试集的作用是验证模型的泛化能力，以确保模型在实际应用中的有效性。

在这次实验中，我们采用了 7:3 的比例来划分训练集和测试集。这种划分方法保证了我们有足够多的数据来训练模型（占总数据的 70%），同时也留有足够的数据（占总数据的 30%）用来测试和验证模型的泛化能力。选择 7:3 的比例是一个常见的实践，因为它在确保模型有足够的数据来学习的同时，也保证了我们有一定量的数据来进行模型评估，从而达到平衡。

3.1.2 数据增强

在深度学习中，数据增强是一种通过对原始数据进行一系列变换，生成新的训练数据的技术。数据增强的目的是为了扩充数据集，增加数据的多样性和数量，从而提高模型的泛化能力和鲁棒性。

常见的数据增强技术包括：

1. 随机裁剪 (Random Cropping): 随机裁剪是一种随机剪裁原始图像的技术，以生成不同的图像。通过在原始图像的不同位置进行随机裁剪，可以生成不同大小和不同内容的图像，从而增加数据的多样性。

2. 随机旋转 (Random Rotation): 随机旋转是一种随机旋转原始图像的技术，以生成不同角度的图像。通过在一定范围内随机旋转原始图像，可以生成不同角度和不同方向的图像，从而增加数据的多样性。

3. 随机翻转 (Random Flipping): 随机翻转是一种随机翻转原始图像的技术，以生成左右或上下翻转的图像。通过随机翻转原始图像，可以生成

不同方向的图像，从而增加数据的多样性。

4. 随机亮度、对比度和色彩变换 (Random Brightness, Contrast and Color Variation)：随机亮度、对比度和色彩变换是一种随机改变原始图像的亮度、对比度和色彩的技术，以生成不同色彩和亮度的图像。通过随机变换原始图像的亮度、对比度和色彩，可以生成不同颜色和亮度的图像，从而增加数据的多样性。

5. 噪声添加 (Noise Addition)：噪声添加是一种在原始图像中添加随机噪声的技术，以生成与原始图像不同的图像。通过添加不同类型和强度的噪声，可以生成不同类型和质量的图像，从而增加数据的多样性。

在这次实验中,我们对于深度学习的方法采用了自动数据增强 (AutoAugment[6])，AutoAugment 是一种用于数据增强的自动化方法，旨在通过搜索算法自动确定适合特定任务的数据增强策略 (如图 4 所示)。数据增强是指通过对训练数据进行随机变换来生成新的训练样本，以增加数据的多样性和数量，从而改善深度学习模型的泛化能力和鲁棒性。



















	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						
		Equalize, 0.4, 4 Rotate, 0.8, 8	Solarize, 0.6, 3 Equalize, 0.6, 7	Posterize, 0.8, 5 Equalize, 1.0, 2	Rotate, 0.2, 3 Solarize, 0.6, 8	Equalize, 0.6, 8 Posterize, 0.4, 6

图 4: AutoAugment 的策略选择

3.1.3 数据归一化

图像归一化 (Image Normalization) 在计算机视觉的深度学习实践中，经常被视为一项基础而关键的预处理步骤。理论与实验均表明，输入数据的归一化可以帮助神经网络更为稳定和迅速地收敛。通过对输入图像进行归一化处理，我们可在一定程度上减轻初始化权重对模型训练的影响，同时也能够降低训练模型的难度，提高模型的泛化性能。

在图像的归一化过程中，我们通常会在通道 (channel) 维度上进行操作。在我们的工作中，基于训练集数据，计算得到各通道的均值为 $[0.5434, 0.5218, 0.4918]$ ，标准差为 $[0.2796, 0.2745, 0.2848]$ 。对每一张输入图像，我

们按通道分别减去其对应的均值并除以标准差，实现了归一化。

3.1.4 优化器和学习率调节器

在深度学习中，优化器和学习率调节器是用于训练神经网络模型的重要组件。

1. 优化器 (Optimizer): 优化器是用于更新神经网络模型参数的算法。在训练过程中，优化器根据损失函数计算的梯度信息，通过调整模型参数的方式来最小化损失函数，从而使模型更好地拟合训练数据。在本实验中，我们采用随机梯度下降 (Stochastic Gradient Descent, SGD)[7]，通过每次迭代随机选择一个样本来计算梯度并更新参数，用于加速训练过程。

2. 学习率调节器 (Learning Rate Scheduler): 学习率是优化器中控制参数更新步长的重要超参数，学习率调节器用于动态调整学习率以提高模型的训练效果。在本次实验中，我采用了由 Chen et al. 提出的 Adaptive Learning Rate Scheduler(ALRS)[8]。

3.2 传统机器学习

本实验中，我们分别测试了 SIFT, ORB 以及 HOG 特征提取器与传统的机器学习算法相结合的效果，如表 1 所示。

表 1: 特征提取器和传统机器学习算法在验证集上的准确率

	Sift	Hog	Orb
Logistic Regression	0.3694	0.3170	0.2373
SVM	0.3649	0.3599	0.2613
K-Means	0.0451	0.0543	0.0512
Random Forest	0.4256	0.3159	0.2919

值得注意的是，几乎所有的传统机器学习方法在该数据集上的表现都比较差。在我看来原因主要在于，传统的特征提取器不能提取到很好的特征，对某些图片形式特别敏感，甚至不能够提取到特征 (如图 5 所示)。在这种复杂多样的图片类型以及纹理条件下，很难得到一个很好的模型。

3.3 深度学习

对于深度学习算法，我们采用了 AutoAugment 进行自动数据增强，以及 ALRS 作为学习率调制器。实验结果如表 2 所示。我们发现深度学习算法的表现要远远优于传统机器学习算法的表现。我觉得原因主要在于，深度学习网络能够自主地提取高维特征。深度学习相对于机器学习具有更强大的特征学习和表征学习能力，适用于处理大规模、高维度、复杂的数据，但

也需要更多的数据和计算资源。机器学习则更适用于小规模数据和相对简单的任务。我们的实验证明，深度学习方法在这个数据集上的表现更优。

表 2: 深度学习算法分别在训练集和测试集的准确率

	训练集	测试集
pyramidnet272	97.65	81.74
resnet	98.56	87.69

4 总结

在这个作业中，我分别用深度学习方法和传统计算机视觉的方法实现了垃圾分类。并通过实验证明，在这个数据集上，深度学习的效果要优于机器学习。垃圾分类与深度学习的结合具有巨大的前景，通过机器学习算法和深度学习算法的结合，可以实现对垃圾图像的自动分类识别，智能垃圾桶的设计，以及提高垃圾分类的准确性和效率。这不仅有助于减少环境污染，提高资源的再利用率，更能够推动垃圾分类工作的智能化和自动化，为建设美丽中国和可持续发展做出重要贡献。因此，垃圾分类与机器学习的结合具有重要的意义，为环境保护和可持续发展开辟了新的道路。

参考文献

- [1] Lowe, David G.. “*Distinctive Image Features from Scale-Invariant Keypoints.*” International Journal of Computer Vision 60 (2004): 91-110.
- [2] Rublee, Ethan et al. “*ORB: An efficient alternative to SIFT or SURF.*” 2011 International Conference on Computer Vision (2011): 2564-2571.
- [3] Dalal, Navneet and Bill Triggs. “*Histograms of oriented gradients for human detection.*” 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05) 1 (2005): 886-893 vol. 1.
- [4] Han, Dongyoon et al. “*Deep Pyramidal Residual Networks.*” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 6307-6315.
- [5] He, Kaiming et al. “*Deep Residual Learning for Image Recognition.*” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 770-778.
- [6] Cubuk, Ekin Dogus et al. “*AutoAugment: Learning Augmentation Strategies From Data.*” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019): 113-123.
- [7] Goyal, Priya et al. “*Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour.*” ArXiv abs/1706.02677 (2017): n. pag.
- [8] Chen, Huanran et al. “*Bootstrap Generalization Ability from Loss Landscape Perspective.*” ECCV Workshops (2022).

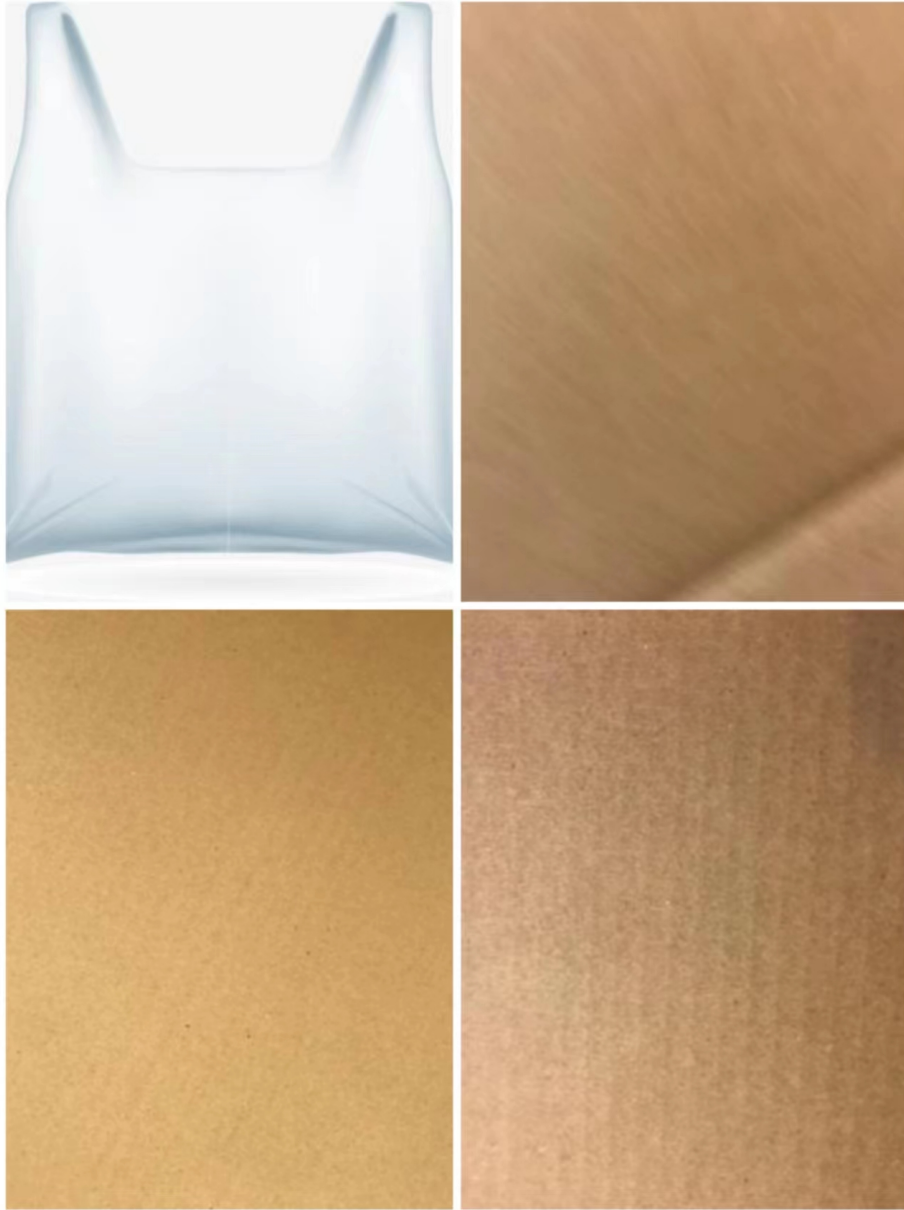


图 5: Orb 和 Sift 提取错误的图片