大理大学本科毕业设计

人工智能技术在 Android 恶意软件检测的研究

RESEARCH ON ARTIFICIAL INTELLIGENCE TECHNOLOGY IN ANDROID MALWARE DETECTION

学　　院：数学与计算机学院

学生　姓名：刘凯锋

学　　号：20201108900348

指导　教师：何远

专　　业：信息安全

年级（班级）：2020 级信息安全班

起止　日期：2023 年 12 月——2024 年 5 月

摘 要

随着 Android 设备的普及，Android 恶意软件的数量也在不断增加，对用户的隐私和信息安全构成了严重威胁。因此，开发高效的 Android 恶意软件检测方法对于保护用户权益和信息安全具有重要意义。

本研究旨在通过人工智能技术探索高效的 Android 恶意软件检测方法。具体目标包括：1）评估不同机器学习模型在恶意软件检测任务中的性能；2）探索基于卷积神经网络的图像检测方法在 APK 文件检测中的应用；3）构建基于动态软投票的集成学习模型，以提高恶意软件检测的准确性和稳定性。

本研究首先使用数据集中的 APK 文件转换为灰度图，并应用卷积神经网络（CNN）进行恶意软件检测。同时，针对 APK 数据的特征向量数据集，本研究应用了 K 近邻（KNN）、支持向量机（SVM）、随机森林（RF）、Adaboost、GBDT（梯度提升决策树）和 XGBoost 等多种机器学习模型进行检测。随后，将上述六个模型作为基模型，通过动态软投票（Dynamic Soft Voting）策略构建了一个集成学习模型，以进一步提高特征向量数据集的检测性能。此外，本研究还使用神经网络对数据特征进行了检测，以全面评估不同方法在恶意软件检测任务中的有效性。

通过实验对比和分析，本研究得出以下结论：1）在 Adware、Banking 和 SMS 类恶意软件的检测中，卷积神经网络和动态软投票集成学习模型均展现出了较高的性能，AUC、准确率和 F1 分数均优于其他单一模型；2）动态软投票集成学习模型通过整合多个基模型的预测结果，提高了恶意软件检测的准确性和稳定性，尤其在 SMS 类恶意软件的检测中表现尤为突出；3）神经网络在恶意软件检测任务中也具有一定的应用价值，但需要根据具体任务和数据集进行调整和优化。本研究为 Android 恶意软件的检测提供了一种新的思路和方法，对于提高 Android 设备的安全性具有一定的实践意义。

关键字：Android 恶意软件；机器学习；灰度图片检测； Android 安全

Abstract

With the increasing popularity of Android devices, the number of Android malwa re is also growing rapidly, posing a significant threat to users' privacy and information security. Therefore, developing efficient Android malware detection methods is crucial for protecting users' rights and ensuring information security.

This study aims to explore efficient Android malware detection methods using arti ficial intelligence techniques. The specific objectives include: 1) evaluating the perform ance of different machine learning models in malware detection tasks; 2) exploring the application of image-based detection methods using convolutional neural networks (C NNs) for APK file detection; 3) constructing an ensemble learning model based on dy namic soft voting to improve the accuracy and stability of malware detection.

In this study, APK files from the dataset were first converted into grayscale imag es and then fed into a convolutional neural network (CNN) for malware detection. Me anwhile, for the feature vector dataset of APKs, various machine learning models such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forests (RF), Adaboost, Gradient Boosting Decision Trees (GBDT), and XGBoost were applie d for detection. Subsequently, these six models were used as base models to construct an ensemble learning model through the dynamic soft voting strategy, aiming to furth er enhance the detection performance on the feature vector dataset. Additionally, neura l networks were also employed to detect malware based on data features to comprehe nsively evaluate the effectiveness of different methods in malware detection tasks.

Through experimental comparison and analysis, this study draws the following co nclusions: 1) In the detection of Adware, Banking, and SMS malware, CNNs and the ensemble learning model based on dynamic soft voting exhibited superior performanc e, with higher AUC, accuracy, and F1 scores than other single models; 2) The ensem ble learning model based on dynamic soft voting improved the accuracy and stability of malware detection by integrating the predictions of multiple base models, particularl y outstanding in the detection of SMS malware; 3) Neural networks also have certain application value in malware detection tasks, but require adjustment and optimization

based on specific tasks and datasets. This study provides a new approach and method for Android malware detection, which has practical significance for improving the security of Android devices.

Keywords: Android Malware; Machine Learning; Grayscale Image Detection; Android Security

目　录

第一章 绪论

1.1 研究背景及意义

Android 是一种基于开放源码的操作系统，其核心为 Linux，自 2008 年发布以来，在智能手机和平板电脑领域得到了极其广泛的应用[1]。其强大的自定义能力使得 Android 能够支持海量应用和丰富特性，为用户带来无限可能。伴随着安卓手机的广泛应用，恶意程序日益增多，这给网络环境下的用户隐私和信息安全带来了巨大的风险。

目前的 Android 平台的恶意软件检测方法大多需要云技术服务以及权威的机构提供的庞大的病毒库的支持[2]。依靠散列和云技术的指纹数据库在应对每天产生的数量庞大的新应用时效率低下，就像基于签名的恶意软件检测等传统方法一样[3]，恶意软件可以通过轻易改变其指纹来绕过这种类型的检测，如果仅仅使用传统的检测方法将带给 Android 应用安全服务商很大的压力。

针对安卓系统中存在的问题，本项目提出了一种基于人工智能的恶意程序检测新方法，该方法不依赖于一个固定的指纹，而是利用机器学习对其进行特征抽取。然后，在此基础上，利用所得到的特性信息，对所得到的模型进行训练，并依据所得到的特征信息，对所得到的程序进行检测。

总体而言，面对层出不穷的恶意软件以及日益增长的病毒防护压力和防护成本，迫切需要新的检测方法。因此基于人工智能技术的 Android 恶意软件检测方法将不断被探索研究。

1.2 国内外研究现状

1.2.1 国内研究现状

近年来，我国学者对安卓系统的恶意程序进行了大量的研究。在此基础上，本项目拟采用机器学习，深度学习等人工智能技术，并结合安卓系统的特点，设计出一系列高效的针对安卓系统的恶意代码检测方法。这些方法主要包括静态分析、动态分析、混合分析以及基于行为分析的检测等。

对于静态的分析，国内的研究人员主要是通过对 Android 软件的字节码或者源码进行分析，提取出各种静态特征，如 API 调用、权限申请、代码结构等，然后根据上述特性对分类器进行了训练使其能够区分恶意软件和正常软件。此外，他们还研究了如何有效处理 Android 应用程序的混淆和加密技术，以提高静态分析的准确性。比如姚斌荣等人：针对目前 Android 恶意软件的静态检测方法在特征选取上类型单一、同类型数量较

多以及得到的检测模型效率不高等问题，提出一种基于组合特征的安卓恶意软件静态检测方法，组合特征集包含权限、组件和预见性 3 个方面[4]。

在动态分析上，国内学者通过仿真安卓应用的运行环境，采集系统调用、网络请求、文件处理等多种动态信息，并据此进行恶意攻击的检测。动态分析能够发现静态分析难以发现的隐藏行为，但也可能面临性能开销大、难以处理加密通信等问题。张雪芹等人为提高 Android 恶意软件的检测精度，提出一种基于改进 DenseNet 网络的 Android 恶意软件动态检测方法[5]。

对于混合型的分析，我国学者们将静态与动态的优点相融合，将两者有机地运用起来，从而提升了检验的精度与效率。此外，他们还研究了如何结合应用程序的上下文信息（如用户行为、设备状态等）来进一步提高检测的准确性。湛忠祥针对传统检测方案的局限性,本文提出了一个结合静态结构分析和动态检测恶意软件的混合方案[6]。

在我国，以行为分析为基础的目标识别是当前的研究热点。该方法是通过对安卓系统应用的行为模式进行分析，从而识别出恶意软件。研究人员搜集并分析了从普通软件到恶意软件的海量行为数据，建立了行为模型库，并利用这些模型库来检测未知的应用程序是否具有恶意行为。余宇劲等人,对每一类别下的恶意应用进行行为分析,研究入侵方式根据其实施的恶意行为进行分类,结合 Android 新版本特性如权限机制中的运行时研究了 Android 恶意软件运行期间的行为特征,对目前流行的恶意软件进行检测[7]。

1.2.2 国外研究现状

国外在 Android 恶意软件检测方面的研究也非常活跃。他们同样利用机器学习、深度学习等人工智能技术来提出各种检测方法，并且在技术上取得了一些领先的成果。

在静态分析方面，国外研究者通过深入研究 Android 应用程序的字节码和源代码结构，提出了更加精确的静态特征提取方法和分类算法。他们还研究了如何结合应用程序的元数据和上下文信息来提高静态分析的准确性。Ou Fan 等人提出了一种新的基于静态敏感子图的 Android 恶意软件检测特征 S3Feaure[8]。

在动态分析方面，国外研究者开发了更加高效和灵活的动态分析平台，能够支持更大规模的应用程序分析和更复杂的恶意行为检测。他们还研究了如何结合应用程序的上下文信息和用户行为来进一步提高动态分析的准确性。Alegre Sanahuja Juan 等人提出了一个基于代理的模型来量化 Android 恶意软件的感染演化[9]。

在混合分析方面，国外研究者通过结合静态分析和动态分析的优势，提出了更加全面和高效的混合分析方法。他们通过深入分析 Android 应用程序的运行机制和恶意行为模式，提出了更加有效的检测策略和技术。Ding Chao 等人提出了一种检测 Android 恶意软件和分类恶意软件家族的混合分析方法,并对多特征数据进行了部分优化[10]。

此外，国外研究者还积极探索了基于人工智能技术的自动化恶意软件检测和分析方法。他们通过利用自然语言处理、图像识别等人工智能技术来自动化地分析应用程序的文本、图标、布局等特征，以发现其中的异常和恶意行为。这些方法具有高度的灵活性和可扩展性，能够为 Android 恶意软件检测提供有力的技术支持。

第二章 相关基础知识

本章节将对 Android 软件的组成结构进行阐述并描述在进行 Android 恶意软件检测过程中所使用到的相关机器学习库和使用到的机器学习算法以及模型评估指标，主要内容有： Sklearn 类库、 PyTorch 类库、激活功能、 Android 恶意程序检测算法（线性回归算法、 SVM 算法、 K 近邻算法等）；针对 Android 系统中存在的问题，以深度学习（卷积神经网络，集成学习等）为核心，开展针对 Android 系统的恶意代码的检测与分析研究，以及 ACC、 AUC、F1_score 等模型评价指标。

2.1 Android 软件的组成结构

Android 系统的软件（一般称作“应用”或者“APP”）一般与 Android 系统的组件（APK）一起绑定。APK 实际上是一种压缩格式的压缩文件，它的基本架构见图 2-1。由下面 7 个部分组成：

1）AndroidManifest.xml：它记载了 App 的名称、权限声明、所包含的组件等一系列信息。

2）classes.dex：这个类是从程序源代码中产生的. class 文件，再进行转化，形成 Android 可读的 DalvikByte 代码。并且，因为 Android 和 JVM 中的 bytecode 之间存在差异，因此，当应用程序涉及到第三方 jar 程序时，一般都会将其包括在内。

3）resources.arsc：这是一个编译后的二进制资源档案，存储了资源的索引。当在 res 文件夹添加任何文件时，aapt 工具会自动生成相应的标识符并置入 .R 文件。尽管 .R 文件的作用在于确保编译过程无误，但在应用程序实际运行时，系统依据这些 ID 查找资源的实际位置。resources.arsc 文件扮演关键角色，它记载了每个 ID 与资源文件路径的映射关系。

4）res 目录：尚未被编译的资料文件.。

5）resources：包括所有已编译的应用程序资源的信息。

6）libs 文件夹：这是专门用来存放由 ndk 开发的.so 库的地方。

7）META-INF 文件夹：它的主要功能是保存应用的特征及认证等，以保证软件的完整与安全。当生成 APK 时，该软件包中的一切都被检验，并且在这里保存结果。设备在安装应用时，会再次验证内容并与 META-INF 中的信息对比，防止 APK 被恶意修改。该目录通常包含以下三个文件：

（1）MANIFEST.MF 文档：所有的资料文档均与唯一的 SHA-256-Digest （SHA1）签名保持一致。这个文件的 SHA256 哈希（以 SHA1 为基础）是通过 base64 进行编码的。在 CERT. SF 中构成 SHA256 (SHA1)-Digest-Manifest。

（2）CERT.SF：文件开始部分设定了 SHA256（基于 SHA1）-Digest-Manifest 的值，随后的部分，是对 MANIFEST.MF 各条目的 SHA256（基于 SHA1）哈希值进行 base64 编码的结果。

（3）CERT.RSA：这个文件包含公钥信息和加密算法描述。首先，采用 SHA256（SHA1）对 CERT. SF 进行了数值运算，采用 RSA 算法对 CERT. SF 进行了加密。然后，开发人员用自己的私有密钥签署这个文件。当您安装时，系统将使用公开密钥对其进行加密。

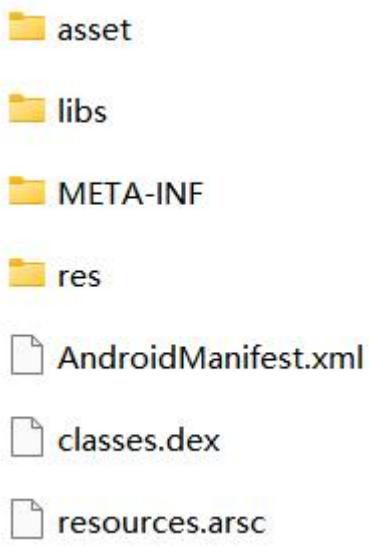


图 2-1　apk 主要结构

2.2 Android 恶意软件检测相关机器学习库

在实验中利用到了机器学习库 Sklearn 和 PyTorch，前者提供了丰富便捷的机器学习算法工具，后者提供的灵活高效的张量计算以及易于使用的神经网络模块。两者结合使用提高了实验过程的效率[11]。

2.2.1 Sklearn 库

Sklearn 是一种以 NumPy, Scipy, Matplotlib 为核心的，以 Python 为核心的开放源码的机器学习类库[12]。该软件为机器学习提供了大量的运算法则和工具包括分类、回归、聚类、降维和特征选择等。Sklearn 类库的结构简洁，易于使用，而且易于扩充，使得构建的机器学习模式变得方便快捷。

本课题中主要使用到的 Sklearn 库中的类方法和函数如表 2-1 所示。

表 2-1 Sklearn 库中的类方法和函数

名称	作用
train_test_split 函数	用于划分数据集为训练集和测试集。
StandardScaler 类	用于特征缩放的类，通常用于数据预处理步骤中。
KNeighborsClassifier 类	实现 k-近邻算法。
DecisionTreeClassifier 类	实现决策树算法。
SVC 类	实现支持向量机算法。
RandomForestClassifier 类	实现随机森林分类算法。
AdaBoostClassifier 类	实现 AdaBoost 算法
GradientBoostingClassifier 类	实现梯度提升 Gradient Boosting 算法
XGBClassifier 类	实现 XGBoost 算法。
StackingClassifier 类	实现模型堆叠。
accuracy_score 函数	用于计算分类任务中模型预测的准确率。
roc_auc_score 函数	用于计算分类任务中模型预测结果的 ROC 曲线下面积。
f1_score 函数	用于计算二分类或多分类问题中 F1 分数的函数。
shuffle 函数	shuffle 函数用于随机打乱数组、列表或类似可迭代对象的顺序。

2.2.2 PyTorch 库

PyTorch 是一款以 Python 为基础的开放源码的机器学习类库，该类类库被 Facebook 的 AI 研究小组用来构造深度学习模式。该方法具有结构简单、结构简单等特点，便于开发人员对深度学习进行建模与培训。PyTorch 的主要能力就是对张量进行运算，而张量则是一个与 NumPy 阵列相似但却拥有更多能力的多维数组。PyTorch 为 Tensor 计算提供了类似于 NumPy 的 Tensor 计算语法，并且能够在 GPU 平台上高效地进行张量计算，因此其在处理大型和复杂的数学问题时具有良好的性能。

本课题中主要使用到的 PyTorch 库中的类方法和函数如表 2-2 所示。

表 2-2　PyTorch 库中的类方法和函数

名称	作用
transforms 类	提供了一系列常用的图像预处理和增强方法。
DataLoader 类	提供了一种方便、高效的方式来加载数据，并且可以很容易地进行批量处理、打乱数据以及使用多线程进行数据加载。
from_numpy 函数	用于将 NumPy 数组转换为 PyTorch 张量（tensor）。
torch.nn.Module 基类	是所有神经网络模块的基类。当定义一个神经网络结构时，通常会创建一个继承自 torch.nn.Module 的类，并在其中定义网络的前向传播（forward pass）和其他可能的操作。
torch.nn.Linear 类	用于实现线性变换。
torch.nn.Sigmoid 函数	用于将任意实数值映射到 0 到 1 之间的范围内。
torch.optim.SGD 类	用于创建一个随机梯度下降优化器。
torch.nn.BCELoss 类	用于计算二元交叉熵（Binary Cross Entropy）损失

2.3 Android 恶意软件检测相关算法

机器学习算法能够从大量数据中提取特征，建立模型，并通过不断地调整模型参数来提高检测的准确性。这种能力使得机器学习算法能够识别复杂的恶意软件模式，并准确地检测出恶意软件。机器学习算法的泛化能力是其在实际应用中可靠和鲁棒的关键。

2.3.1 激活函数

通常来说，实践中大部分的分类和回归问题，线性模型很难拟合。为了采用线性模型解决更为复杂的问题，模仿人类神经元组织结构增加了非线性的阈值输出。

文中采用了 Sigmoid 和 ReLU 两种活化功能。在学习过程中，激励功能是一个非常关键的问题，通过将激励功能的引入引入到一个新的模型中，使得它能够更好地处理更加复杂的问题。

1）Sigmoid 函数

sigmoid 函数是机器学习中最经典的且最早被使用的激活函数之一[13]，公式如下所示：

$$\rho = \frac{1}{1+e^{-x}} \tag{2-1}$$

其函数图像如图 2-2 所示。

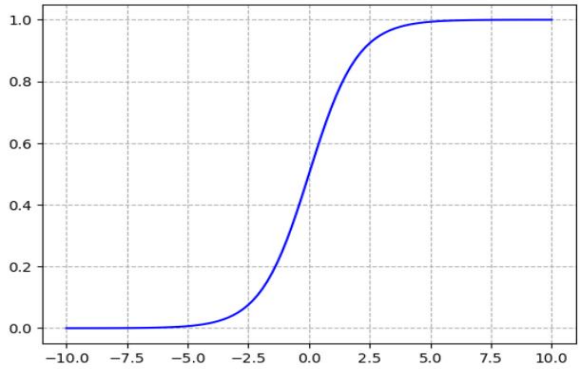


图 2-2　Sigmoid 函数图像

如图可知，Sigmoid 在其定义区域内都有可导性。如果神经元网络中的层次太多，则在经过多个层次的训练后，其权值将趋于 0。这就导致了该模式的不收敛。这也就是 sigmoid 函数曾经被广泛使用，但现在却很少使用的原因。

2）ReLU 函数

Sigmoid 具有梯度消失的问题，并且因为它包含了幂操作，所以计算耗时较长。于是，一种新的线性修正单元函数（ReLU）应运而生[14]。，并成为目前最常用的激活函数之一。公式如下：

$$f(x)=\begin{cases}0, & x<0\\ x, & x\geq 0\end{cases} \tag{2-2}$$

其函数图像如图 2-3 所示。

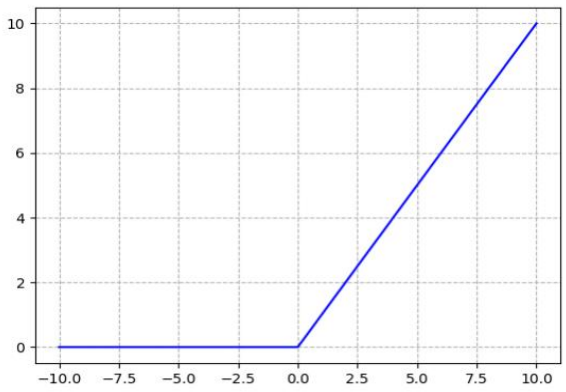


图 2-3　ReLU 函数图像示例

其中，当 x<0 时， ReLU 函数取 0，这时梯度值也等于 0，这样就降低了梯度操作的代价。在时 x> =0 时， ReLU 的取值是 x，所以梯度总是恒定不变，从而解决了梯度不存在的难题。

2.3.2 决策树

决策树（Decision Tree)，也叫判决树（decision Tree)，是一种基于树形（二叉树、多树）的预测分析模型，是一种基于树形（二叉树、多树）的预测分析模型[15]。它的每一个非叶子节点都是对一个特性的检验，每一个分枝都是对该特性的一个取值，而叶子则是一个类。通常，一种决策树由一个根结点、几个内结点和几个叶子结点组成。叶结点与判定的结果相关联，而其它节点则与一种特性的检验相关联。将各节点所构成的采样集按其特性检验的结果分为若干子节点，其中一根节点含有采样集，其从根节点到各叶子节点的路线就是判断试验序列的一部分。该算法旨在生成一种具有较高的推广性能，也就是对未知样本具有很好的适应性。

2.3.3 K 近邻

K 近邻算法（KNN）是一种适用于分类与回归问题的基于示例学习的机器学习算法[16]。针对新的样本，K-近邻算法的目的是寻找“离它最近”的一个样本，如果大多数样本都是一个类别，则这个样本也可以被归入一个类别 K。

在 KNN 方法中，最常见的度量是欧几里德与曼哈顿距。

在平面几何学中，欧几里德距离是最常见的一种，也就是两个点到一条线的间距。公式如下：

$$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2-3}$$

式中：

d —— 两点间欧几里得距离；

n —— n 维空间中。

曼哈顿距离是计算两点在一个网格上的路径距离，与上述的直线距离不同，它只允许沿着网格的水平和垂直方向移动。公式如下：

$$d_{12} = \sum_{\mathrm{k}=1}^{\mathrm{n}} |x_{1\mathrm{k}} - x_{2\mathrm{k}}| \tag{2-4}$$

式中：

d —— 两点间曼哈顿距离；

n —— n 维空间中。

2.3.4 支持向量机

支撑向量机（SVM）是一种有指导的学习方式，可以在数据分类和回归分析中得到广泛的使用[17]。这种方法是把矢量投影到高维空间中，其中包含了极大区间的超面。在该超面的两侧分别建立了两个相互平行的超面，该超面的划分使得该超面之间的间距最大。在此基础上，提出了多个并行面之间的间距或间距增大时，分类器的整体错误率降低。相对于 Logistic 回归及神经网路，支援矢量机对于处理较复杂的非线性方程式，具有更清楚且功能更强的方法。

2.3.5 卷积神经网络

卷积神经网络（CNN）是一类基于深度网络的新型深度网络，可广泛应用于图像、视频等数据进行分析[18]。卷积神经网络通过模仿人类大脑的加工模式，具有自主学习与抽取的能力，已在计算机视觉研究中获得了明显的效果。卷积神经网络采用卷积层、池

化等多个层次对输入进行深度学习，把底层的特征进行融合，形成较高层的特征表达，并在极大程度上实现对数据的降维，并将其作为一个完整的连通层，得到最后的识别效果。

典型的 CNN 由 3 个部分构成：

1)卷积层（提取图像中的局部特征）

卷积层在卷积网络中占有非常关键的地位。。该算法由若干个过滤器（亦被称作卷积核）组成，通过对输入的数据执行卷积运算，产生一个特性图[19]。各滤波算法分别针对图像中的边缘、纹理等特性进行了分析。通过数个卷积核，卷积网络可以学习更为复杂与抽象的特性。在卷积层中，一个权值的矩阵，也就是一个卷积核。卷积核在“滑动”过程中，先求权阵与已搜索到的资料阵之积，再将运算结果以一幅特征图形式显示出来。以 5*5 的二维图片为例，卷积过程如图 2-4 所示。

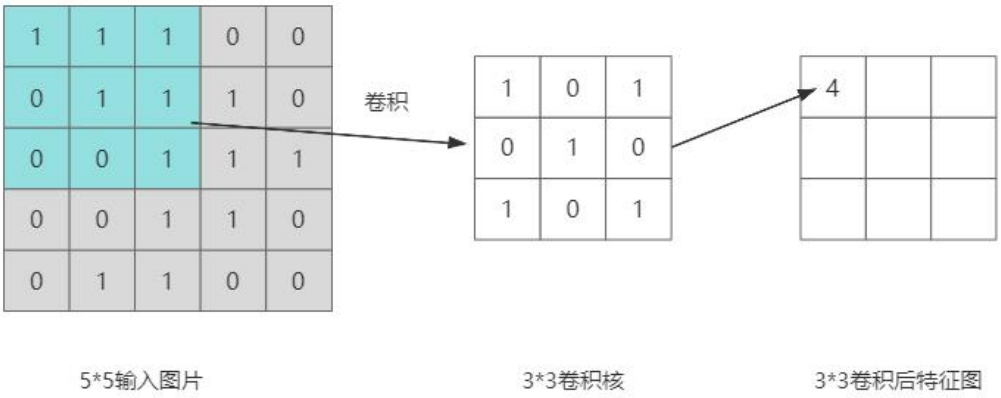


图 2-4　卷积过程示例图

因为在卷积后，图像的维数将随卷积核维数的变化而变化，因此，为了保证图像的特性尺寸，一般采用 padding 进行填充，padding 的取值通常根据 stride 的取值和卷积核的大小而定。以上述 5*5 维度图片为例，将 padding 设置为 1 时，将用 0 把输入图片扩展为 7*7 维度。再将扩展后的图片和卷积核进行卷积得到 5*5 维度特征图像。如图 2-5 所示。

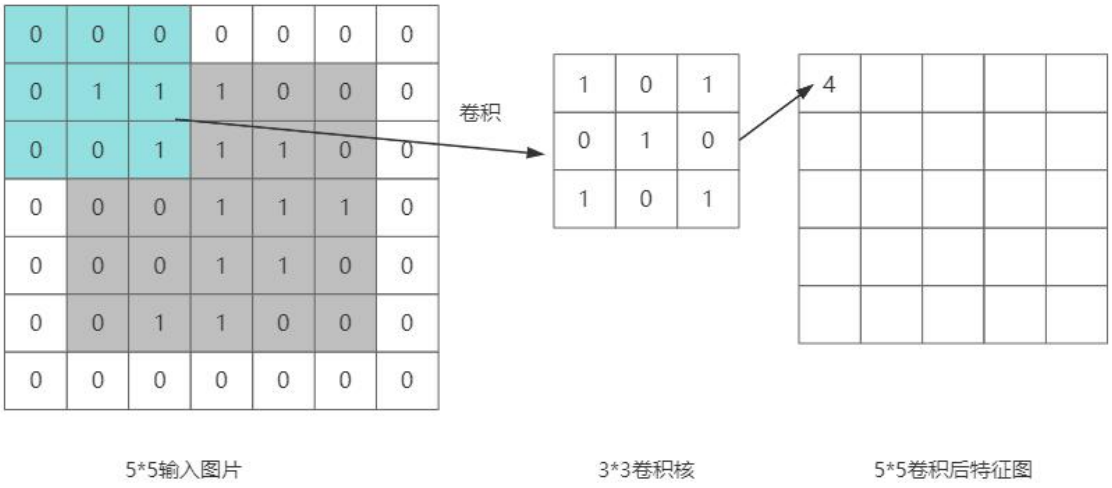


图 2-5　padding 演示

2)池化层（大幅降低数据量级，降维，或称为“降采样”）

在图像处理中，由于图像中存在较多冗余信息，可用某一区域子块的统计信息（如最大值或均值等）来刻画该区域中所有像素点呈现的空间分布模式，以替代区域子块中所有像素点取值，这就是卷积神经网络中池化（pooling）操作[20]。

该方法在保持特征图谱的重要信息的前提下，将所抽取的特征数据降维，降低其运算复杂性。同时对图像进行了图像的特征压缩，以提高图像的平移不变性，降低了图像的过拟程度。但实际上，池化更多的是在牺牲一些数据的情况下，强行将功能进行了压缩。

在广义卷积处理之后进行了池化处理。池化的实质，就是抽样。为了加速计算，Pooling 会对一个特征映射进行降维和压缩。

池化的几种常见方法包括：平均池化 与 最大池化。池化示例如图 2-6 所示。

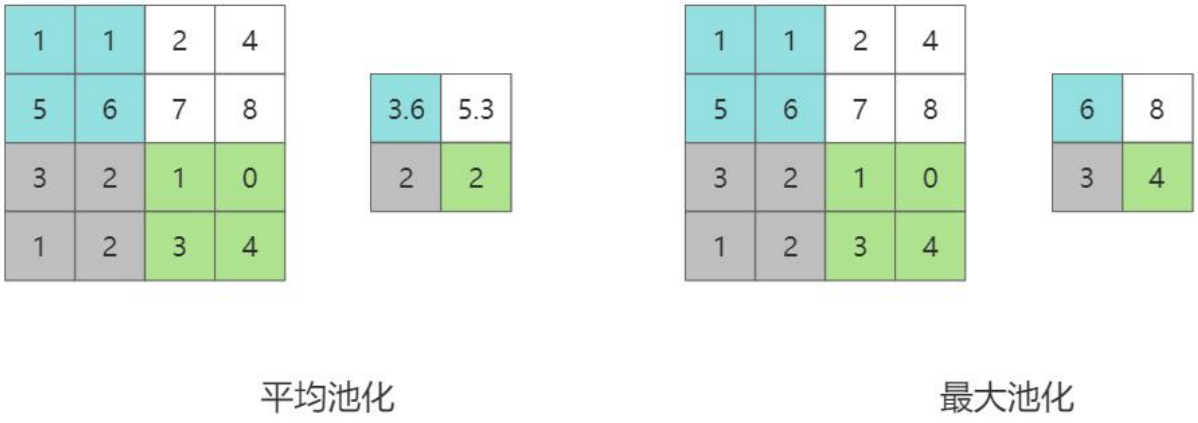


图 2-6　池化示例图

3)全连接层（传统神经网络，输出）

全连接层把从卷积和池化中抽取出来的特征联系在一起，并输出最终的分类或回归结果[21]。它使用了传统神经网路中普遍使用的完全连结架构，即每一个神经元都连结在上面一层。其中，完整连通层的功能就是将这些特性结合起来，形成最后的预测结果。在图 2-7 中显示了一个完整的连接层结构。

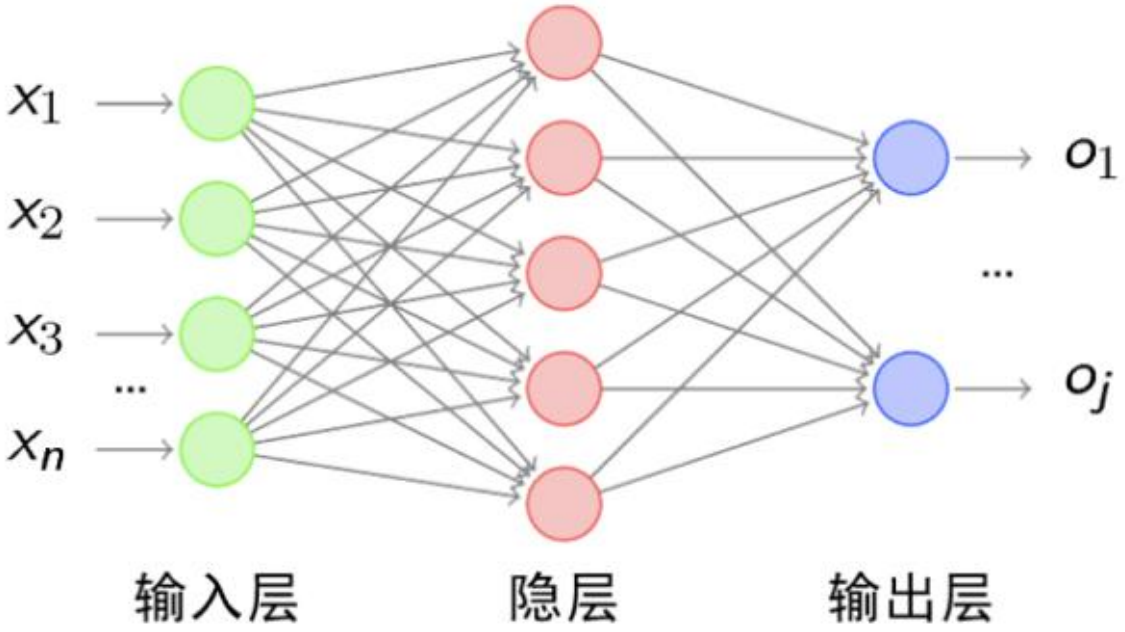


图 2-7　全连接层结构图

总结起来，CNN 通过卷积层逐步提取局部特征，使用池化层减小数据维度，而后在全连接层中对特征进行组合和分类。这种架构使得 CNN 能够有效地处理具有空间结构的输入数据。CNN 可以根据具体的目标，通过对不同的目标进行特征表达，达到更好的识别效果。图像识别，物体检测，人脸识别等。

2.3.6 集成学习

集成学习（Ensemble Learning）指建立和整合多个学习主体（也被称作基本学习主体或个人学习主体）来实现学习的过程。[22]。其基本思想是通过对多个基本模型的预测值进行融合，以提高整体预测性能。该方法能够有效地改善建模精度与稳定性，特别是对于复杂的数据与任务。采用融合学习方法可以有效地降低过度学习、欠拟合等问题，从而改善模型的推广性能。将多个模式的预报值结合起来，可以减少错误比率，增强模型的稳健性。综合算法可以分为 Bagging、Boosting、Stacking 三大类。。

1）Bagging

Bagging 为 Bootstrap aggregating 简写，即套袋法[23]。选取若干个训练集合，将每一个集合从原来的样本集合上经过反复的循环，组合成一个训练集合（在一个训练集合中，有的抽样可以重复出现，有的抽样一次都没有）。经过 m 次的学习，形成了 m 个彼此独立的学习集合。在此基础上，通过对不同的样本进行分类，得出不同的分类结果。通过对前一阶段获得的 m 个模型进行表决，得出最终的分类效果。对以上各模式进行了平均处理，得到最终结果。流程如图 2-8 所示。

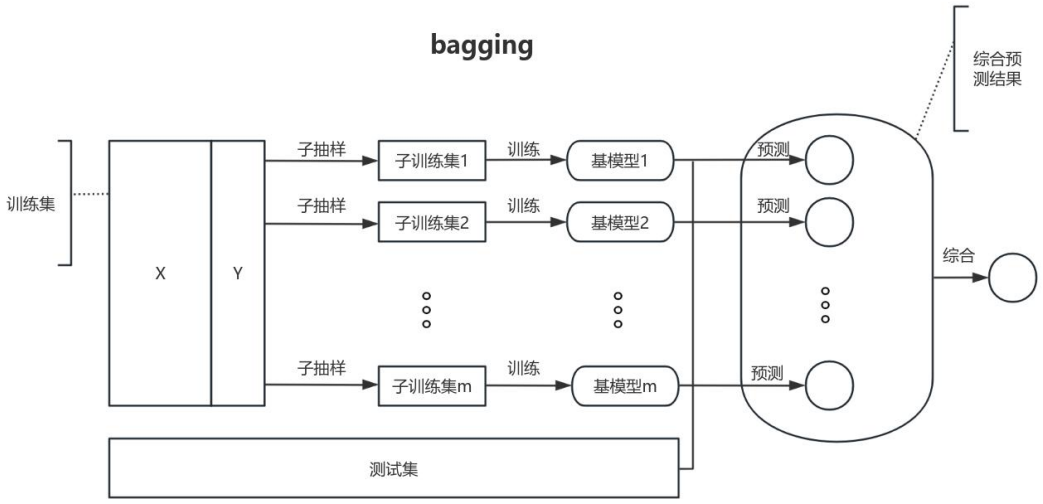


图 2-8　bagging 算法流程图

（1）随机森林 Random Forest

随机森林是一种由决策树构成的 Bagging 集成算法[24]。随机森林方法是一种对干扰信号有很强的自适应能力的方法。另外，由于该方法能有效降低对非关键属性的依赖性，因此能有效地解决高维数据的问题。

随机森林是按照“随机约束”原则来设计的，也就是说，一个或者更多的决策变数都是随机的。采用反向抽样的方式，对原始样本进行抽样，形成带有一定随机性的样本集合。抽样是可以重复的，这是由于在一个随机选取的进程中，每一个抽样都会被反复地挑选出来，因此，它更容易被包括在一个独立的子集里。

2）Boosting

Boosting 算法的工作机理为：通过对训练集权值进行训练，建立一个基类（弱）模型 1，并根据其错误率率对训练样本权值进行更新，从而使前一类（弱）模型 1 的错误率较高的训练样本点赋予较大的权值，从而使后继的基类（弱类）模型 2 更加关注错误率高的数据。然后，基于权重调整后的训练样本，训练基（弱）2 直到弱学习子数目达到规定数目，再采用合并策略融合 m 个基（弱）模型，得到最终的强学习模型，进而得到最终的强学习模型，进而得到完整的预测结果[25]。流程如图 2-9 所示。

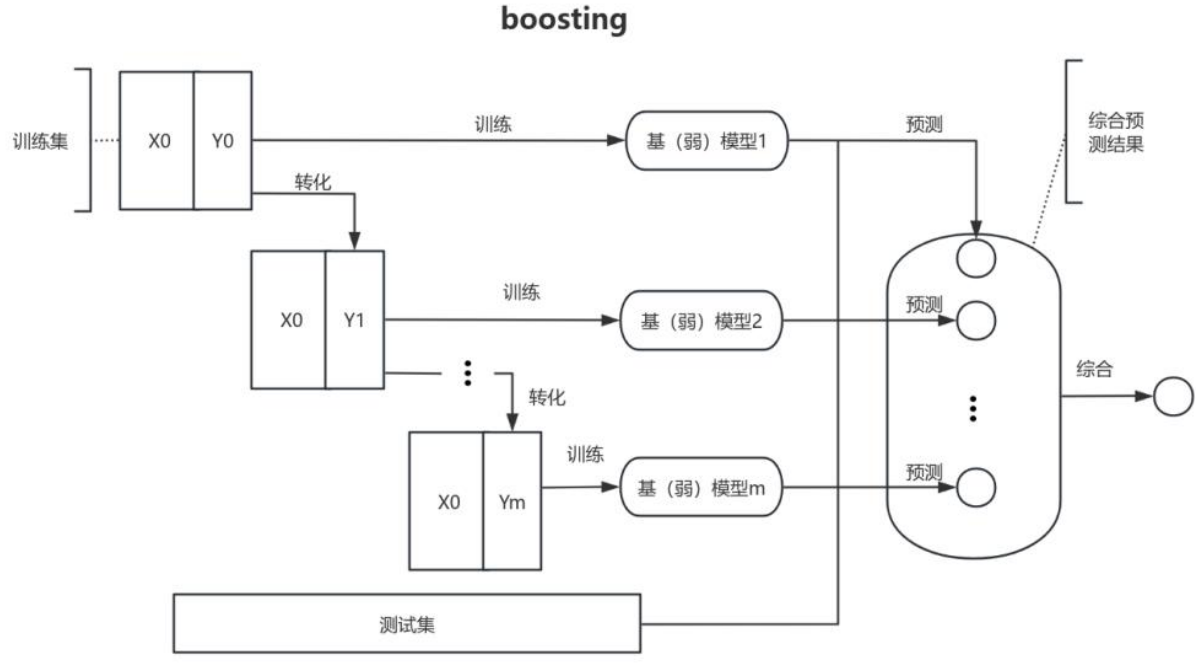


图 2-9　boosting 算法流程图

（1）AdaBoost

Adaboost（Adaptive Boosting 自适应提升算法）算法是一种常用的集成学习算法，解决的是二分类问题，该算法能把多种不同的弱分类器结合在一起，形成一个强大的分类器[26]。其基本思想是通过对样本集进行加权重和重复的训练来提高分类器的准确率。Adaboost 的工作流程如表 2-3 所示。

表 2-3　Adaboost 工作流程

步骤	具体实现
权重初始化	对于一个训练样本，在初始阶段，将各训练样本赋以权值。
训练弱学习器	在每个迭代中，使用权重调整的数据集来训练一个弱学习器。
计算错误率	使用当前的弱学习器对数据集进行预测，然后计算加权的错误率。
计算学习器权重	根据目前弱学习算法的出错率来确定权值，出错率越小的弱学习算法得到的权值越大。
更新样本权重	增加分类错误的样本的权重，并减少分类准确的样本的权重。
迭代	重复上述步骤，直到满足迭代次数或错误率达到预定的阈值。
组合弱学习器	所有的弱学习器以其权重为基础进行组合，形成一个强学习器。

（2）GBDT

GBDT(GradientBoosting Decision Tree),全名叫梯度提升决策树[27]。该方法和 Ada Boost 不同之处是： AdaBoost 通过对不同类别的类别进行权值调节，不断地进行迭代，从而得到一个很好的分类器。在此基础上， GBDT 以其负值为残差逼近，通过反复迭代及拟合回归树，得到一个较好的学习模型。

（3）XGBoost

XGBoost(Extreme Gradient Boosting),即一种高效的梯度提升决策树算法[28]。他对原来的 GBDT 作了一定的改良，极大地提高了建模的效率。其中心思想是利用“Boosting”的集成概念，把若干个弱学习器通过某种方式组合在一起，形成一个强大的学习系

统。也就是采用多颗树联合决定，将各颗树的结果作为目标和前面几颗树的预测值之间的差值，再把这些结果相加，就可以获得最后的结论，从而提高了整体模型的效能。XGBoost 是由多棵 CART(Classification And Regression Tree)，即分类回归树组成，所以它能解决诸如分类和回归之类的问题。

2.4 模型相关评估指标

2.4.1 ACC

ACC，也就是正确率，是一种最简便和应用最广泛的方法。该函数表示对模型进行适当的分类，从而直观地反映了模型的分类性能。

在模型评估中，ACC 的计算基于四个基本概念：真实样本（TP）、真实负样本（TN）、伪正样本（FP）、伪负样本（FN）。

ACC 的计算公式为：

$$ACC = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{2-5}$$

2.4.2 AUC

模型评估指标 AUC（Area Under the Curve）是一种常用于评估分类模型性能的指标，特别是当数据集中存在不平衡类别时。AUC 的核心思想是通过计算 ROC（Receiver Operating Characteristic）曲线下的面积来评估模型的分类性能[29]。

ROC 曲线是真实比率（TPR）与错误比率（FPR）间的权衡曲线。真实比率代表由该模型预测为真实例子的样品中作为积极例子的比率，并且错误比率代表该真实负面例子的样品中由该模型作为积极例子的比率。AUC 值越大，表示模型的分类性能越好，因为它能够更准确地识别正例并减少误报。

AUC 不需要设定特定的分类阈值，因此可以全面评估模型的分类性能。此外，AUC 对正负样本的均衡性不敏感，因此在处理不平衡数据集时具有更好的性能。

2.4.3 F1-score

模型评估指标 F1-score 是一种常用的综合评估分类模型性能的指标，它结合了精确率（Precision）和召回率（Recall）的特点，通过计算这两个指标的调和平均数来评估模型的性能[30]。

F1-score 的计算公式为：

$$F1score = \frac{2TP}{2TP+FP+NP} \tag{2-6}$$

F1-score 能在正确率与召回率之间取得平衡，克服单一评估方法的片面。在类别识别问题中，正确率与召回率常常是相互对立的，提升一项就有可能导致另外一项的下降。而 F1-score 则是通过对二者进行协调均值的测算，使两者达到某种均衡，从而得出更为完整、精确的评价结论。

第三章 基于特征的 Android 恶意代码检测设计与实现

在 Android 恶意代码检测领域，基于特征的方法因其直观性和有效性而广受欢迎。通过提取和分析 Android 应用程序（APK 文件）中的特征，可以实现对恶意代码的识别和分类。然而，单一的特征或分类器往往难以应对复杂多变的恶意代码。因此，本章旨在通过集成多种特征和分类器，以提高 Android 恶意代码检测的准确性和鲁棒性。我们利用现有的特征数据集进行恶意代码检测。这些特征数据集包含了从 Android 应用程序中提取的静态和动态特征，如权限使用、API 调用、系统调用等，这些特性可以充分地体现出一个软件的行为和属性。为后续的模型训练和分类提供有力的数据支持。设计并实现一系列基分类器，包括 K 近邻（KNN）、支持向量机（SVM）、随机森林（RF）、梯度提升决策树（GBDT）、AdaBoost 以及 XGBoost 等。这些基分类器能够从不同角度捕捉恶意代码的特征，为后续的集成学习提供丰富的基础。

为了进一步提高检测性能，我们将采用动态软投票的策略来集成这些基分类器。动态软投票是一种根据基分类器的性能动态调整其权重的方法，可以在保证准确性的同时降低过拟合风险。具体而言，我们将使用 AUC 值作为衡量基分类器性能的指标，并根据 AUC 值的大小动态调整其权重。通过这种方式，我们可以充分利用各个基分类器的优势，实现更准确的恶意代码检测。

除了基分类器的集成外，我们还设计了一个全连接神经网络模型作为独立的检测工具。该模型通过多层的神经元连接和激活函数，能够自动提取和学习输入特征中的关键信息，实现对恶意代码的准确分类。全连接神经网络模型学习性能好，泛化性好，对复杂数据集也能获得良好的检测结果。

3.1 数据预处理

3.1.1 特征提取

为了从 Android APK 中提取特征向量，首先要解析与每个 Android APK 相关的 JSON 文件，以提取 APK 在 CopperDroid（基于 VMI 的动态分析系统）中运行时调用的系统调用、绑定和复合行为。之后，我们通过将所有 APK 文件的所有不同的低级行为放在一起并将每个特征设置为特定行为的调用频率，为每个 APK 文件创建大小为 470 的特征向量。然后使用归一化方法将特征向量归一化为[0,1]之间的值[31]。

3.1.2 无效特征去除

由于所提取特征数量较大，为防止无效特征影响实验结果，本实验认为如果某特征列所有数值相加值为 0 则该特征为无效特征将其剔除。放大上述 0 值为某一阈值，如果某特征列所有数值相加小于等于该阈值，则认为该特征为无效特征。基于这一思想，经测试选择阈值为 63，将无效特征剔除后剩余 290 个特征，除去最后一个类别列共 289 个特征。

3.1.3 训练集和测试集划分

利用 sklearn 中的 model_selection 函数，将训练样本分为比例为 8:2 的两组，分别是训练样本和测试样本。

训练集：其核心功能在于培育机器学习模型的智能。模型借助大量带有标记的信息（表现为输入特性和相应的输出标识）来领悟数据间的深层联系和趋势，从而提升对未知数据预测的精确性。

测试集：此集合的作用是对已完成训练的模型在未知数据上的效能进行评估。通过对比模型在测试集预测的结果与实际的标签，我们可以量化地判断模型的准确性，有效验证其表现。

3.1.4 数据标准化及张量封装

为了加快模型的收敛速度,本实验将使用 sklearn.preprocessing 来对原始数据进行标准化。

为了加快模型训练速度，将数据封装到 pytorch 的张量对象中。

3.2 集成学习模型检测设计与实现

3.2.1 设计思路

在本小节中希望通过搭建多个基分类器，对这些基分类器进行训练对测试集进行预测并得出相应的模型评估指标。再将各个基分类器在测试集上得到的 AUC 分数作为动态权重。软投票过程中，不是简单地取所有分类器的预测概率的平均值，而是根据分类器的性能（即权重）对预测概率进行加权。这允许性能更好的分类器在最终预测中拥有更大的影响力。我们将详细描述我们所使用的多种机器学习算法的训练过程、评估方法，以及如何通过动态软投票策略将它们集成起来，以进一步提升分类性能。

3.2.2 算法选择与训练

针对上述问题，本项目拟选取 6 种典型的机器学习方法，即 K 最近邻（KNN）、极限梯度提升（XGBoost）、自适应提升（AdaBoost）、梯度提升决策树（GBDT）、

SVM （SVM）和随机森林（RCF）。这些算法在分类问题中广泛应用，且各具特点，适合用于本研究的比较。

KNN 算法：是一个以实例为基础的学习方法，该方法通过对与被试样本最近的 K 个抽样进行分类，从而对待测样本进行分类。在本研究中，我们选择了 K=4 作为邻居数量，并使用欧氏距离作为相似度度量。通过训练数据集进行训练后，我们得到了 KNN 模型，并使用测试数据集进行性能评估。

XGBoost 算法：在一个梯度上升的基础上，通过连续增加新的决策树对目标函数进行了优化。我们使用了 XGBoost 库进行模型训练，并调整了学习率、树的数量、最大深度等超参数以优化模型性能。

AdaBoost 算法：一种通过调节各采样点加权的迭代方法，对有误分类的采样点进行重点处理，然后根据修正后的权值对新的分类器进行学习。本文采用了基于自适应神经网络的基分类器——决策树，采用 AdaBoost 方法对其进行融合。

GBDT 算法：同样是一种与 XGBoost 相似的、以梯度为基础，决策树为基分类器的集成学习方法。然而，GBDT 使用更传统的梯度提升方法，而不是 XGBoost 中的近似算法。我们同样使用 Scikit-learn 库中的 GradientBoostingClassifier 类实现 GBDT 算法，并调整了相关超参数。

SVM 算法：根据最大间距原则，在一个高维度上搜索一个可以划分出各种类型的样本的超面，从而完成对目标的分类。在本研究中，我们选择了径向基函数（RBF）作为核函数，并使用 Scikit-learn 库中的 SVC 类实现 SVM 算法。

RCF 算法：是一种基于集成学习的分类算法，它通过构建多个决策树并进行投票来决定最终分类。在本研究中，我们使用 Scikit-learn 库中的 RandomForestClassifier 类实现 RCF 算法，并调整了树的数量、最大深度等参数。

3.2.3 评估方法

为了评估各个模型的性能，使用了准确率（acc）、AUC 值和 F1 分数作为评价指标。这些指标在分类问题中广泛应用，能够全面反映模型的分类性能。我们使用了 Scikit-learn 库中的相关函数来计算这些指标，并在测试数据集上进行了评估。

3.2.4 动态软投票集成策略实现

在得到各个模型的预测结果后，我们采用了动态软投票集成策略将它们集成起来。这种方法结合了各个模型的预测概率，并根据它们在测试集上 AUC 值动态地分配权重。具体而言，我们首先使用用交互验证方法把原数据集分成训练集，测试集。然后，在每

个模型上进行交叉验证，以评估其在训练集上的性能。根据 AUC 值，为每个模型分配了不同的权重。在测试集上，将各模型的预报概率与相应的权值相加，得出预测效果。

3.3 神经网络模型检测设计与实现

在当今日益增长的数据处理需求中，神经网络模型因其强大的特征学习和表示能力而备受关注。特别是在二分类问题中，神经网络模型能够捕捉数据中的复杂模式，从而实现高精度的分类预测。本小节旨在构建一个基于简单神经网络结构的二分类检测模型，通过对模型的设计、训练以及评估过程的详细阐述，探讨该模型在特定数据集上的性能表现。

3.3.1 模型设计思路

首先，明确特征数据集的维度和待检测的类别数，据此设置 in_features 和 out_classes 参数。这些参数将决定输入层和输出层的大小。

在模型的初始化阶段，我们设置了第二个全连接层 self.linear2，它负责将隐藏层提取的特征映射到最终的类别预测空间。在模型的前向传播（forward 方法）过程中，输入特征 x 首先经过第一个全连接层 self.linear1 和 ReLU 激活函数 self.relu 进行转换和非线性处理，以提取高级特征。随后，这些高级特征再被送入第二个全连接层 self.linear2 进行进一步变换，得出每个类别的分数。为了得到输入样本属于各个类别的概率分布，这些分数可以通过 softmax 函数进行归一化处理。softmax 函数将分数转换为概率值，使得所有类别的概率之和为 1，从而方便我们根据概率大小判断输入样本最可能的类别归属。

3.3.2 模型定义

在构建神经网络模型时，我们首先需要确定模型的结构和参数。针对二分类任务，我们构建了一个简单的神经网络模型 HWR_Model，它包含两个线性层。模型首先通过输入层接收特征向量，然后这些特征经过隐藏层进行非线性变换，最终由输出层给出分类结果。这个设计使得模型能够处理二分类问题。

具体地，HWR_Model 模型在初始化时定义了输入层、隐藏层和输出层的参数。输入层接收具有 in_features 数量的特征向量，隐藏层包含 hidden_size 数量的神经元，并应用 ReLU 激活函数增加非线性表达能力，输出层则包含 out_classes 数量的神经元，用于输出最终的分类结果。在本例中，由于我们处理的是二分类问题，因此 out_classes 的值为 1。

3.3.3 模型训练

在模型训练阶段，我们采用了双变量交互熵损函数（BCEWithLogitsLoss）作为损失函数，将 sigmoid 运算和双变量交互熵损相融合，适用于二分类问题的损失计算。优化器方面，我们选择了随机梯度下降（SGD）算法，并设置了学习率为 0.1。在此基础上，通过迭代更新模型参数，使损失函数最小，优化器对模型进行持续优化。

在训练神经网络模型 HWR_Model 时，我们将数据集划分为训练集和测试集。首先，通过前向传播，模型对训练集中的样本进行预测，并生成预测输出。接着，这些预测输出与真实标签一起用于损失函数中计算损失值。这个损失值反映了模型预测与真实情况之间的差异。然后，利用反向传播，求出了损失值与模型参数之间的梯度关系，并利用优化器对模型的参数进行更新。每次循环完成后，为了防止下次循环的积累，我们会将先前计算好的梯度值删除。

为了监控训练过程，我们设定了 2000 次迭代作为训练的总次数。同时，我们每 500 个迭代输出一次当前的 epoch 数和损失值，以便观察模型的训练进度和收敛情况。通过不断迭代优化，模型逐渐学习到数据中的规律，并在训练集上达到较好的性能表现。

3.3.4 模型评估

为了全面评估训练好的神经网络模型的性能，我们在测试集上进行了测试。首先，我们将测试集数据输入到模型中，获得模型对每个样本的预测输出。由于模型的输出是 logits（即未经过激活函数的原始输出），我们需要将其转换为二进制预测标签。在本例中，我们设置了一个阈值（通常为 0.5）来进行转换。

在评估模型性能时，我们使用了准确率（ACC）、AUC 值和 F1 分数等多个指标。准确率反映了模型分类正确的比例，AUC 值衡量了模型在不同阈值下的综合性能，而 F1 分数则考虑了精确率和召回率，适用于不平衡数据集。这些指标为我们提供了模型在测试集上性能的全面评估。

第四章 基于深度学习的 Android 恶意代码检测设计与实现

在 Android 恶意代码检测领域，基于特征的检测方法通过提取和分析 APK 的静态或动态特征来识别潜在的恶意行为。然而对特征的提取分析时长繁琐且提取的特征检测效果难以保证。深度学习技术的快速发展为恶意代码检测提供了新的思路。特别是卷积神经网络（CNN），以其强大的特征提取和模式识别能力，在图像识别、自然语言处理等领域取得了显著成果。传统的基于特征的方法需要手动设计并 Android 恶意软件的特征，而卷积神经网络则通过自动学习数据的内在特征，大大提升了检测精度和效率。

为此，本项目拟研究一种新的基于深度学习的安卓恶意代码检测算法，其中核心思想是将 APK 文件转化为灰度图，并利用卷积神经网络进行检测。

4.1 数据预处理

4.1.1 APK 转灰度图

在将 APK 文件转为灰度图之前先判断 APK 的大小，超过 10MB 将不进行转换。将 APK 的二进制序列中每 8 位二进制数化为一个 0 至 255 之间的数（对应像素值），将这些数输入到设定长度的一维数组中，再将该数组重塑为 n*n 的二维数组，最后使用 PIL 将其转为对应的 n*n 大小的灰度图片。根据文件大小对数组进行适当调整使得所转灰度图片保留尽可能多的特征。所转的 256*256 大小灰度图如图 4-1 所示：

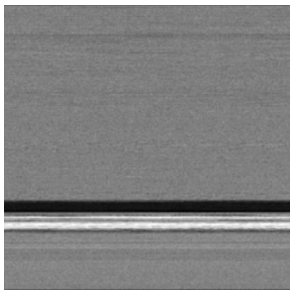


图 4-1　APK 灰度图

4.1.2 设置 Datasets

在机器学习、深度学习等领域，对样本进行加载与预处理是建模过程中的重要环节。为了有效地处理图像数据和对应的标签，设计了一个自定义数据集，称为 Mdataset，该类从标准的 Dataset 类中继承。该类的主要目的是从生成的灰度图片文件夹中读取图像路径和标签，将图片数据以 8:2 划分为训练数据、训练标签和测试数据、测试标签。

4.2 卷积神经网络模型检测设计与实现

4.2.1 模型设计思路

使用 APK 所转灰度图片尽可能保留其特征数量同时便于检测，将其大小定为 256*256。定义第一个卷积层将输入的 1 张灰度图片进行第一次卷积操作输出 6 张大小不改变的特征图片，使用 3*3 的卷积核且设置填充为 1 来保持图片的大小，使用 MaxPool2d(2, 2)最大池化层进行特征压缩并提取主要特征使得图片大小降维到 128*128，再进行一次卷积操作同样设置卷积核为 3*3 大小填充为 1 不改变输入特征图大小，输出进一步卷积后的 16 张特征图。将输出的 128*128*16 的特征图片进行池化操作后得到 64*64*16 的特征图片，此时的特征数量较大影响模型训练效率，再添加一个池化层得到 32*32*16 的特征图，将 32*32*16 的特征传入全连接层进行组合和整合，最后输出二分类结果。

4.2.2 模型定义

定义了一个卷积神经网络模型 CNN，用于处理图像数据。该模型基于 PyTorch 框架构建，并继承自 torch.nn.Module 基类，使得我们能够灵活地定义网络结构。该网络架构包括两个卷积层、三个最大池化层以及三个全连接层。卷积神经网络检测流程图如图 4-2 所示下面详细描述了该模型的各个组成部分。

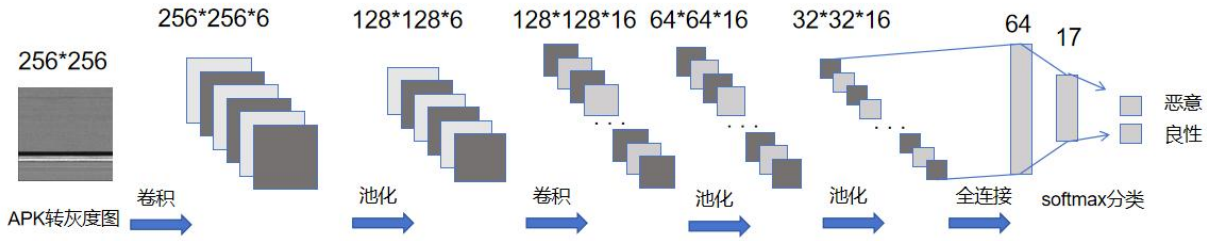


图 4-2　卷积神经网络检测 APK 灰度图流程

1）卷积层

第一个卷积层（conv1）：输入通道数为 1（对应于灰度图像），输出通道数为 6。采用 3*3 的卷积核，步长（stride）为 1，并在图像边缘填充 1 个像素的零（padding 为 1）。该层旨在从输入图像中提取初步的特征映射。

第二个卷积层（conv2）：输入通道数为 6（对应于上一个卷积层的输出通道数），输出通道数为 16。采用 3x3 的卷积核，步长（stride）为 1。与第一个卷积层相比，卷积核数量的增加反映了特征分类的逐渐细化，使模型能够学习到更复杂的特征表示。

2）池化层

在两个卷积层之后及第二次池化操作之后，分别应用了一个最大池化层（pool）。这三个池化层均采用 2x2 的池化窗口，步长也为 2。最大池化操作能够减少特征映射的空间尺寸，从而降低计算复杂度，并增强模型的平移不变性。

3）全连接层

第一个全连接层（fc1）：输入节点数为 32x32x16（对应于第三个池化层的输出特征映射的大小），输出节点数为 64。该层进一步整合了从卷积层和池化层提取的特征，并准备将其传递给后续的分类层。

第二个全连接层（fc2）：输入节点数为 64，输出节点数为 17。这一层继续对特征进行抽象和转换，以准备进行分类任务。

第三个全连接层（fc3）：输入节点数为 17，输出节点数为 2。这一层是模型的输出层，用于输出两个类别的预测概率。

在模型的训练过程中，利用 ReLU 激活函数，对各卷积网络及完整连通网络进行了非线性转换，提高了学习效率。最后，该算法将得到一幅具有不同分类特征的两维矢量。通过这种设计，CNN 模型能够有效地从输入图像中提取有用的特征，并通过多个层次的特征变换和整合，实现准确的图像分类。

4.2.3 模型训练

在本节中，将详细阐述模型的训练过程。模型训练包括数据加载、前向传播、计算损失、反向传播和优化器更新权重等步骤。

1）数据准备

在进行模型学习前，首先要建立一种能够有效地对样本进行重复处理的方法。我们将利用 PyTorch 中的 DataLoader 的思想，为训练样本和测试样本建立一个单独的数据装载器。

对于训练数据，我们设定了一个加载机制，即 train_loader，其中我们决定每次从数据集中抽取 100 个样本及其对应的标签作为一个批次进行处理。同时，我们确保在每个完整的训练周期（epoch）开始时，训练集中的样本顺序会被重新随机排列，以增强模型的泛化能力。

类似地，对于测试数据，我们也构建了一个名为 test_loader 的加载机制。虽然在实际操作中，测试集通常不会被打乱，但为了代码展示的一致性，我们这里也设定了打乱操作。

2）模型构建

我们创建了一个名为 modelcnn 的卷积神经网络（CNN）实例，这个实例基于 PyTorch 的神经网络基础模块 nn.Module 进行扩展。接着，我们为模型指定了损失计算的方

法，即交叉熵损失，并选用了 Adam 优化算法来更新模型的参数，其中学习率被设定为 0.01。

3）训练过程

模型的训练是通过一系列连续的周期（epoch）来完成的。在每个周期中，模型会遍历整个训练数据集。对于数据集中的每个批次，我们执行以下步骤：

前向传递：我们将输入图像传递给 modelcnn 模型，模型基于这些图像进行预测并产生输出。

损失计算：我们根据模型的预测输出和真实的标签，使用交叉熵损失计算方法来确定预测与真实值之间的差异。

梯度计算：我们基于损失值反向传播至模型的各个层，以计算损失对于模型参数的梯度。

权重更新：在更新模型参数之前，我们首先清除之前计算得到的梯度信息，然后基于新计算出的梯度来更新模型的权重。

监控性能：在每个批次结束时，我们记录下当前的损失值和训练准确率，以便后续分析。

模型保存：在整个训练过程中，我们持续监控模型的性能，并在模型性能有所提升时（如损失值降低或准确率提高）保存其权重，以便后续使用。

4.2.4 模型评估

在训练结束后，使用独立的测试集来评估模型的性能。首先，加载在训练过程中保存的最佳模型权重。然后，关闭梯度计算，以确保在评估过程中不会更新模型的权重。

接下来，遍历测试数据加载器 test_loader，将每个批次的数据输入到模型中，并收集模型的预测结果和对应的真实标签。为了评估模型的性能，我们计算了准确率 ACC、AUC 和 F1 分数。这些指标提供了关于模型分类性能的全面视图。在代码中，我们使用了 scikit-learn 库中的 accuracy_score、roc_auc_score 和 f1_score 函数来计算这些指标。

第五章 实验与分析

5.1 实验数据集、

5.1.1 数据集介绍

本文所使用的数据集来自于 CICMalDroid 2020，包含 11,598 个 APK 文件及相应的 470 个 APK 提取特征，包括系统调用、绑定程序和复合行为的频率。

该数据集中有四个不同的类别：Adware, Banking malware, SMS malware,和 Benign。如表 5-1 所示。每个恶意软件类别的简要描述如下：

1.Adware 是指通常隐藏在被恶意软件感染的合法应用程序（可在第三方市场上获得）中的广告材料（即广告）。

2.Banking malware 是一种专门的恶意软件，它可以模拟原来的银行系统和网络接口，从而入侵到使用者的网络银行帐户。

3.SMS malware 借助 SMS 服务作为攻击途径，拦截并操作 SMS 消息中的有效载荷以实施恶意行为。

4.Benign 不属于上述类别的所有其他应用程序都被视为良性应用程序。

表 5-1 数据集各分类样本

样本类别名称	中文名称	样本数量
Adware	恶意广告插件	1253
Banking malware	银行恶意软件	2100
SMS malware	短信恶意软件	3904
Benign	良性程序	1796

5.1.2 数据集分类

本课题将对恶意程序进行二分类的检测。相比于多分类，二分类大大简化了分类的复杂性和计算量，使得分类过程更加快速和高效。在二分类的情况下，分类器只需要关注两个类别的差异，因此可以更集中地学习和识别这两个类别的特征。这有助于提高分类的精度和准确性，降低误报率和漏报率。虽然二分类只关注两个类别，但它可以很容易地扩展到多分类。例如，可以构建多个二分类器，每个分类器负责识别不同的恶意软件类别。这种方法结合了二分类和多分类的优点，既保持了分类的准确性和效率，同时又具备了处理多种恶意软件的能力。采用二分类检测方法可以对不同类别的 Android 恶意软件检测的 ACC、AUC、F1_score 值进行对比分析，通过分析可以得出哪类程序基于何种算法模型的检测效果更好。

在第三章中使用的是现有特征数据集，将数据集按照 APK 类别分类如表 5-2 所示。

在第四章中使用的数据集为第三章中用于提取特征数据集的相对应 APK 文件，数据集划分比例同表 5-2

表 5-2　将数据集按照 APK 类别进行划分

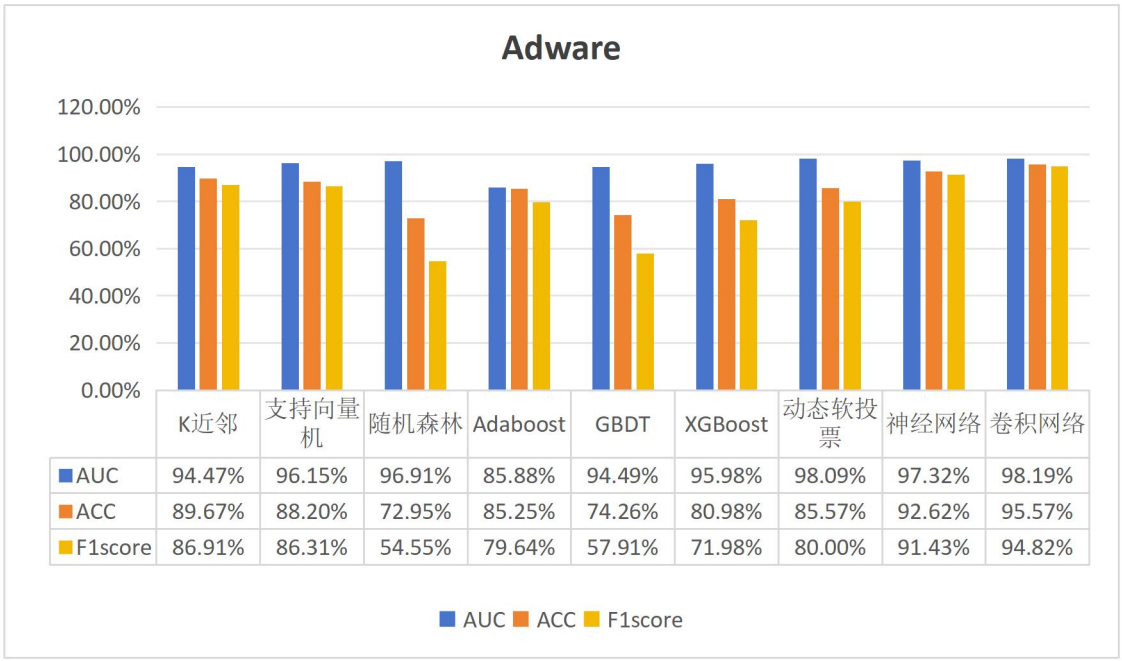
数据集	APK 类别	数量比
数据集一	Adware：Benign	1253:1795
数据集二	Banking：Benign	2100:1795
数据集三	SMS ：Benign	3904:1795

5.2 实验设置

本课题实验环境基于 Windows 11 操作系统，使用 Jupyter Notebook 7.0.8 作为交互式编程界面，结合 Python 3.11.7 作为编程语言，利用 CUDA 11.1 支持 GPU 加速，安装了 PyTorch 2.2.2 深度学习框架和 Sklearn 1.2.2 机器学习库，以支持数据分析和模型训练。

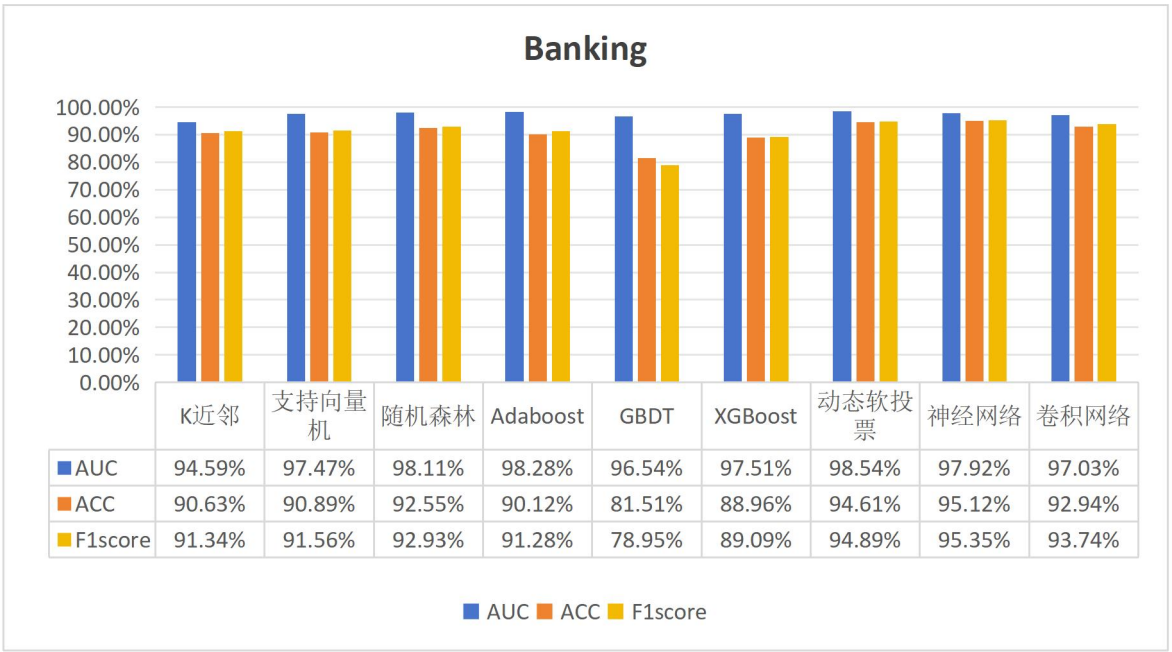
5.3 实验结果

Adware、Banking、SMS 三类恶意程序在集成学习各分类器模型、使用动态软投票方法集成的强模型、神经网络和卷积网络中的各模型指标（ACC、AUC、F1 分数）如图 5-1 至 5-3 所示。



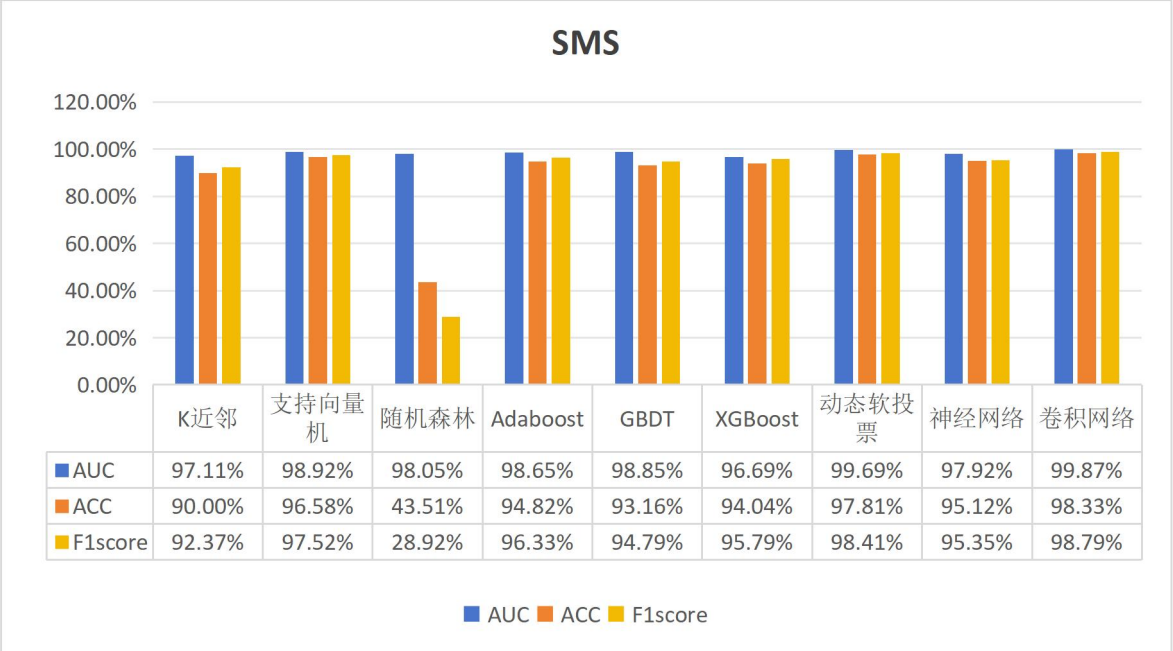
	K近邻	支持向量机	随机森林	Adaboost	GBDT	XGBoost	动态软投票	神经网络	卷积网络
AUC	94.47%	96.15%	96.91%	85.88%	94.49%	95.98%	98.09%	97.32%	98.19%
ACC	89.67%	88.20%	72.95%	85.25%	74.26%	80.98%	85.57%	92.62%	95.57%
F1score	86.91%	86.31%	54.55%	79.64%	57.91%	71.98%	80.00%	91.43%	94.82%

图 5-1　Adware 类恶意软件不同模型性能指标



	K近邻	支持向量机	随机森林	Adaboost	GBDT	XGBoost	动态软投票	神经网络	卷积网络
AUC	94.59%	97.47%	98.11%	98.28%	96.54%	97.51%	98.54%	97.92%	97.03%
ACC	90.63%	90.89%	92.55%	90.12%	81.51%	88.96%	94.61%	95.12%	92.94%
F1score	91.34%	91.56%	92.93%	91.28%	78.95%	89.09%	94.89%	95.35%	93.74%

图 5-2　Banking 类恶意软件不同模型性能指标



	K近邻	支持向量机	随机森林	Adaboost	GBDT	XGBoost	动态软投票	神经网络	卷积网络
AUC	97.11%	98.92%	98.05%	98.65%	98.85%	96.69%	99.69%	97.92%	99.87%
ACC	90.00%	96.58%	43.51%	94.82%	93.16%	94.04%	97.81%	95.12%	98.33%
F1score	92.37%	97.52%	28.92%	96.33%	94.79%	95.79%	98.41%	95.35%	98.79%

图 5-3　SMS 类恶意软件不同模型性能指标

5.4 结果分析

5.4.1 总体性能

1）AUC（Area Under the Curve，曲线下面积）：在所有类别的恶意软件检测中，大多数模型的 AUC 值都很高，尤其是动态软投票和卷积网络模型，显示了它们良好的分类能力。

2）ACC（Accuracy，准确率）：准确率在不同类别和模型间有所差异。一些模型在某些类别上表现良好（如卷积网络在 SMS 类上达到 98.33%的准确率），而在其他类别上则表现一般（如随机森林在 SMS 类上仅有 43.51%的准确率）。

3）F1score：F1 分数是精确率和召回率的调和平均，适用于评估不平衡数据集上模型的性能。在大多数类别中，动态软投票和卷积网络模型的 F1 分数都较高，表明它们在处理不平衡数据集时表现良好。

5.4.2 模型比较

1）动态软投票：在所有类别的恶意软件检测中，动态软投票模型都取得了很高的 AUC、准确率和 F1 分数，证明了集成多个模型进行投票是一种有效的策略，可以提高整体分类性能。

2）卷积网络：在 Adware 和 SMS 类恶意软件检测中，卷积网络模型表现出色，尤其是 SMS 类，其准确率和 F1 分数均接近或超过 98%。这可能是因为卷积网络在处理具有复杂特征和模式的数据时具有优势。

3）随机森林：随机森林在 Adware 类恶意软件检测中表现较差，准确率仅为 72.95%，远低于其他模型。这可能是由于 Adware 类数据集中正负样本分布的不平衡或随机森林对该类数据的特征表示不敏感所致。

4）支持向量机（SVM）和 K 近邻（KNN）：这两个模型在大多数类别中都取得了中等的性能，表明它们在恶意软件检测任务中具有一定的适用性，但可能不是最优选择。

5）Adaboost 和 GBDT：Adaboost 和 GBDT 在某些类别中表现较好，但在其他类别中则相对较差。这可能是由于它们对特定数据集的适应性有限。

6）神经网络：神经网络在 Banking 和 SMS 类恶意软件检测中取得了良好的性能，但在 Adware 类中表现一般。这可能与神经网络对数据特征的敏感性和学习能力有关。

第六章 总结与展望

6.1 总结

本研究专注于开发高效的 Android 恶意软件检测方法，旨在应对 Android 设备普及背景下日益增长的恶意软件威胁。通过深入探索人工智能技术在恶意软件检测中的应用，本研究不仅评估了不同机器学习模型的性能，还探索了基于卷积神经网络的图像检测方法在 APK 文件检测中的潜力，并成功构建了一个基于动态软投票的集成学习模型，以提升恶意软件检测的准确性和稳定性。

在研究方法上，本研究首先通过数据预处理，将 APK 文件转换为灰度图，并利用卷积神经网络（CNN）进行了图像识别与分类。同时，针对 APK 文件提取的特征向量，本研究对比了包括 K 近邻（KNN）、支持向量机（SVM）、随机森林（RF）、Adaboost、梯度提升决策树（GBDT）和 XGBoost 在内的多种机器学习模型的性能。为了进一步提升检测效果，本研究采用动态软投票策略，将上述模型作为基模型构建了一个集成学习模型。

实验结果显示，卷积神经网络和基于动态软投票的集成学习模型在检测 Adware、Banking 和 SMS 类恶意软件时均表现出色，其 AUC、准确率和 F1 分数均优于其他单一模型。这一成果不仅验证了人工智能技术在恶意软件检测中的有效性，也展示了集成学习在提升模型性能方面的巨大潜力。

此外，本研究还探讨了神经网络在恶意软件检测任务中的应用，尽管其性能依赖于具体任务和数据集，但其在特征提取和分类方面的能力仍然值得进一步研究和优化。

总之，本研究为 Android 恶意软件检测提出了新颖的思路和方法，通过整合人工智能技术和集成学习策略，有效提高了恶意软件检测的准确性和稳定性。这不仅有助于保护 Android 设备的安全，也为未来的恶意软件检测研究提供了有价值的参考。随着技术的不断进步和恶意软件的不断演变，我们将继续探索更加高效和智能的恶意软件检测方法，以应对日益严峻的信息安全挑战。

6.2 展望

基于本文所设计的检测实验还存在以下两个改进方向：

1）数据集的拓展

数据集的规模和质量对机器学习和深度学习模型的性能至关重要。对于 Android 恶意软件检测这一领域，数据集的拓展可以从以下几个方面进行：

（1）增加样本数量

通过收集更多的 Android 恶意软件样本，可以增加数据集的规模，使模型能够学习到更多的恶意行为模式。这不仅可以提高模型的泛化能力，还可以使模型在面对新的未知恶意软件时具有更好的检测效果。

（2）扩大样本多样性

在增加样本数量的同时，还需要确保样本的多样性。这包括收集来自不同来源、不同攻击方式、不同版本的恶意软件样本。通过扩大样本的多样性，可以使模型更加全面地学习到恶意软件的特征，提高检测的准确性和鲁棒性。

（3）引入良性样本

除了恶意软件样本外，还可以引入大量的良性样本（即正常软件）来构建更加均衡的数据集。这有助于模型更好地区分恶意软件和良性软件，减少误报率。

（4）数据预处理和增强

对收集到的样本进行适当的数据预处理和增强也是非常重要的。

2）模型优化

（1）超参数调优

机器学习与深度学习中存在大量的超参量（如学习速率、批处理规模、迭代次数等），它们的大小直接关系到模型的性能。为此，必须进行试验研究，寻找最佳的超参量组合。

（2）特征选择和降维

对于高维特征数据，可以通过特征选择和降维技术来降低模型的复杂度，提高计算效率。

（3）迁移学习和预训练模型

迁移学习和预训练模型是深度学习中常用的技术，可以利用在其他任务上训练好的模型来加速当前任务的训练过程，并提高模型的性能。在 Android 恶意软件检测中，可以尝试使用在图像识别、自然语言处理等领域训练好的预训练模型来提取更加有效的特征，提高检测的准确性。

参考文献

[1] 老顽童聊科技.Android 操作系统介绍及市场占有率分析.2024-02-16.https://baijiahao.baidu.com/s?id=1791005246720393497&wfr=spider&for=pc.

[2] 张恪易.基于云服务技术的数据共享交换集成应用探究[J].网络安全技术与应用,2023,(09):67-69.

[3] 宁卓,邵达成,陈勇,等.基于签名与数据流模式挖掘的 Android 恶意软件检测系统[J].计算机科学,2017,44(S2):317-321.

[4] 姚斌荣,张娜.基于组合特征的安卓恶意软件静态检测方法研究[J].软件导刊,2024,23(02):129-134.

[5] 张雪芹,王逸璇,赵敏.基于深度学习的 Android 恶意软件动态检测[J].计算机工程与设计,2024,45(01):10-16.DOI:10.16208/j.issn1000-7024.2024.01.002.

[6] 湛忠祥.基于集成学习的安卓恶意软件混合检测方案研究[D].南京信息工程大学,2023.DOI:10.27248/d.cnki.gnjqc.2023.000225.

[7] 余宇劲.基于行为分析的 Android 恶意软件检测技术研究[D].广东工业大学,2020.DOI:10.27029/d.cnki.ggdgu.2020.000474.

[8] Fan O ,Jian X .S3Feature: A static sensitive subgraph-based feature for android malware detection[J].Computers Security,2022,112

[9] Juan S A ,Javier C ,Carlos J L C , et al.Agent-Based Model to Study and Quantify the Evolution Dynamics of Android Malware Infection[J].Abstract and Applied Analysis,2014,20141-10.

[10] Chao D ,Nurbol L ,Bei L , et al.A Hybrid Analysis-Based Approach to Android Malware Family Classification[J].Entropy,2021,23(8):1009-1009.

[11] 菠萝爱吃肉. 深入解读：Scikit-learn、Python 版本与 PyTorch 的区别与联系. 2024.04.09. https://developer.baidu.com/article/details/3309384.

[12] Charles J .Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib[J].SIAM REVIEW,2020,62(2):515-517.

[13] Huyen H D ,Sven N ,Chee P Y , et al.Optimized Sigmoid Functions for Speech Presence Probability and Gain Function in Speech Enhancement[J].Circuits, Systems, and Signal Processing,2024,43(5):2891-2908.

[14] Zhiqiang C ,Jingshuang C ,Min L .Least-Squares ReLU Neural Network (LSNN) Method for Scalar Nonlinear Hyperbolic Conservation Law[J].Applied Numerical Mathematics,2022,(prepublish):

[15] Rahman M ,Kamal N ,Abdullah F N .EDT-STACK: A stacking ensemble-based decision trees algorithm for tire tread depth condition classification[J].Results in Engineering,2024,22102218-.

[16] Kumar ,Pravin,Jain , et al.Surface roughness prediction in micro-plasma transferred arc metal additive manufacturing process using K-nearest neighbors algorithm[J].The International Journal of Advanced Manufacturing Technology,2022,(prepublish):1-13.

[17] Jianfei W .Optimizing support vector machine (SVM) by social spider optimization (SSO) for edge detection in colored images[J].Scientific Reports,2024,14(1):9136-9136.

[18] Yan T ,Chen G ,Zhang H , et al.Convolutional neural network with parallel convolution scale attention module and ResCBAM for breast histology image classification [J].Heliyon,2024,10(10):e30889-.

[19] Chagnon J ,Hagenbuchner M ,Tsoi C A , et al.On the effects of recursive convolutional layers in convolutional neural networks[J].Neurocomputing,2024,591127767-.

[20] Xie Z ,Xu X ,Li L , et al.Residual networks without pooling layers improve the accuracy of genomic predictions.[J].TAG. Theoretical and applied genetics. Theoretische und angewandte Genetik,2024,137(6):138-138.

[21] Wang L ,Liu B ,Goeckel D , et al.Connectivity in cooperative wireless ad hoc networks[C]//University of Massachusetts, Amherst, Amherst, MA, USA;;University of Massachusetts, Lowell, Lowell, MA, USA;;University of Massachusetts, Amherst, Amherst, MA, USA;;University of Massachusetts, Amherst, Amherst, MA, USA;;DoCoMo Labs USA, Palo Alto, CA, USA,2008:

[22] Jia Z ,Yao G ,Zhao K , et al.A fault diagnosis framework based on heterogeneous ensemble learning for air conditioning chiller with unbalanced samples[J].Measurement Science and Technology,2024,35(8):

[23] Wang Y ,Hu Y ,Ren H , et al.Integrated transcriptomic, metabolomic, and functional analyses unravel the mechanism of bagging delaying fruit cracking of pomegranate (Punica granatum L.)[J].Food Chemistry,2024,451139384-.

[24] Zhang Q ,Wang Y ,Feng L , et al.Predicting leak aperture in the pipeline of ultra-long coal mine working faces under strong noise interference based on joint denoising and random forest[J].Flow Measurement and Instrumentation,2024,97102609-.

[25] Zexiao D ,Luhua L .State-Space Perturbation Analytical Solution for the Dynamics of Launch-Vehicle Boost Phase[J].Journal of Aerospace Engineering,2024,37(5):

[26] Paula A J ,Diogo S C ,Inácio I O D , et al.Optimal trade-off between boosted tolerance and growth fitness during adaptive evolution of yeast to ethanol shocks[J].Biotechnology for Biofuels and Bioproducts,2024,17(1):

[27] Han Y ,Huang J ,Ma Z , et al.GBDT Method Integrating Feature-Enhancement and Active-Learning Strategies—Sea Ice Thickness Inversion in Beaufort Sea[J].Sensors,2024,24(9):

[28] Ayyoub F ,Mohamed M ,Mehdi M , et al.Harnessing LSTM and XGBoost algorithms for storm prediction[J].Scientific Reports,2024,14(1):11381-11381.

[29] Arnold S ,Orvin D ,Patel M , et al.Methicillin-Resistant Staphylococcus aureus Bacteremia Treated With Vancomycin Calculated by Area-Under-the-Curve in Patients With Elevated Vancomycin Minimum Inhibitory Concentrations[J].Hospital Pharmacy,2024,59(3):329-333.

[30] Disha P ,Vishal D .Predicting breast cancer using machine learning classifiers and enhancing the output by combining the predictions to generate optimal F1-score[J].Biomedical and Biotechnology Research Journal (BBRJ),2021,5(3):331-334.

[31] Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, Ali A. Ghorbani; Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning, The 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), Aug. 17-24, 2020.

致谢

在此，我首先要向我的导师何远表示最诚挚的感谢。在论文的选题、研究思路、实验设计和论文撰写等各个环节，老师都给予了我悉心的指导和无私的帮助。X 老师严谨的学术态度、深厚的学术造诣，使我受益匪浅。他的言传身教不仅让我在人工智能领域有所收获，更让我在人生道路上受益良多。

感谢我的家人，他们一直是我坚强的后盾。在我面临困难和挑战时，他们总是给予我无私的支持和鼓励，让我有勇气面对一切困难。他们的爱和关怀是我不断前进的动力。

同时，我也要感谢我的同学们和朋友们。在论文的写作过程中，我们相互学习、相互鼓励、共同进步。他们的陪伴和支持让我感受到了友谊的温暖和力量。

此外，我还要感谢学校和学院为我们提供的良好的学习环境和资源。图书馆丰富的藏书、实验室先进的设备以及老师们精彩的授课都为我的论文写作提供了极大的帮助。

最后，我要向所有参与论文评审和答辩的老师们表示衷心的感谢。他们的宝贵意见和建议让我的论文更加完善。

再次感谢所有在我论文写作过程中给予帮助和支持的人们，是你们的关心和帮助让我能够顺利完成这篇论文。我将永远铭记你们的恩情，并以此为动力，继续努力前行。