第十三届中国软件杯

AppShark
涉诈 APP 智能分析系统

技术文档

AppShark

2024/8/20

王艺杰 苏弘康 王文瑞

摘要

随着数字化转型的加速发展，犯罪形态发生了显著变化，在诸多诈骗类型中，又以电信网络诈骗犯罪最为普遍。这类犯罪从传统的电话、短信诈骗，逐步转向利用APP 等网络工具进行，成为当前发展增速最快、涉及范围最广的刑事犯罪类型，各种诈骗 APP 层出不穷，严重威胁公民安全。本项目旨在开发一个“涉诈 APP 智能识别分析系统”，利用人工智能技术，提对电信网络诈骗软件的甄别能力。系统将通过智能分析，识别和预警潜在的涉诈 APP，为用户提供实时的保护。

具体的，本项目主要分成三大模块。首先是 APP 自动下载与获取，系统将分析用户提供的链接及二维码，递归爬取网页及其子页面，通过发起随即滚动与点击模拟用户真实浏览行为，确保获取到用户可能遇到的下载链接。其次，系统将根据链接自动获取并解析对应 APP 安装包。在这部分中，系统将通过特征提取与黑白名单对比，基于序列预测分类技术的的源代码分析，多模态的图像分类模型等技术对安装包开展详细的分析。最后，系统将根据上述分析结果与信息为用户生成分析报告，供用户查看。经过实验验证与调查，我们的软件具有良好的用户使用体验以及较高的识别准确率。

关键词：多模态，机器学习，涉诈分析，特征分析，指纹提取

目录

第一章 项目背景

1.1 社会背景

随着数字化转型的加速发展，犯罪形态也发生了显著变化。在诸多诈骗类型中，电信网络诈骗犯罪尤为普遍。这种犯罪从传统的电话、短信诈骗，逐步转向利用 APP 等网络工具进行，成为当前发展增速最快、涉及范围最广的刑事犯罪类型。各种诈骗 APP 层出不穷，严重威胁公民的安全和财产。

在这样的背景下，开发一个“涉诈 APP 智能识别分析系统”显得尤为重要。该系统旨在利用人工智能技术，提升对电信网络诈骗软件的甄别能力。通过智能分析，系统能够识别和预警潜在的涉诈 APP，为用户提供实时的保护。

1.2 技术背景

人工智能技术的快速发展为解决这一问题提供了新的思路。通过机器学习、深度学习等技术，可以训练模型识别恶意 APP 的特征，从而实现自动化的检测和预警。此外，多模态分析技术的应用，如图像识别和自然语言处理，可以进一步提升识别的准确性和全面性。

1.3 项目意义

我们认为：开发“涉诈 APP 智能识别分析系统”具有如下重要的社会和经济意义：

1) **保护用户安全**：通过提前识别和预警涉诈 APP，减少用户受骗的风险。

2）**维护社会稳定**：减少因诈骗引发的社会问题，提升公众对网络环境的信任度。

3）**促进经济发展**：保护用户财产安全，促进电子商务和在线支付等数字经济的健康运行。

第二章 需求分析

我们将题目要求中的基本功能需求与非功能性需求进行了归纳和整合，并将其划分为以下几个模块：apk 自动下载、apk 特征分析、apk 源码分析、apk 图像分析、apk 分析报告生成。接下来详细介绍各个部分的需求以及需要实现的功能。

2.1 自动下载

为了全面支持 apk 文件的多种下载方式，我们需要设计并实现一个具备多种功能的下载管理器。具体的，需要支持通过链接下载，二维码下载，本地上传 apk 三种方式。下载后的 apk 文件交由软件管理。下载过程需要可视化，并支持多线程下载，下载进度查看。

2.1.1 输入解析

系统首先识别并验证输入的 URL 是否为有效的 apk 文件下载链接，通过 HTTP 头信息检查（如 Content-Type）、文件扩展名（如.apk）以及可能的服务器重定向逻辑来实现。对于加密或短链接，系统需能够解析并还原至最终下载地址。

此外，系统应当支持爬取网页自动获取下载地址，例如输入某软件首页地址，系统应当能自动分析该页面的所有子页面，寻找是否存在下载链接。

系统还需要支持通过二维码下载，用户可将二维码截图或文件在软件中打开，又软件自动提取其中的下载地址。

2.1.2 下载管理

为了提高下载效率，采用多线程并发下载技术。根据网络状况和资源可用性，动态调整并发数，确保下载过程既快速又稳定。

支持断点续传功能，即在下载过程中若因网络问题或其他原因中断，能够在恢复连接后从上次中断的位置继续下载，而非重新开始。

实时显示下载进度、速度、剩余时间等信息，并提供下载完成的通知机制，如弹窗、声音提示或系统通知。

设计用户友好的界面，确保用户可以方便的输入下载链接、二维码，并实时查看下载进度和结果。

提供下载设置选项，允许用户自定义下载路径、并发数、安全扫描级别等，满足不同使用场景下的需求。

2.2 特征分析

apk 文件的安全分析流程中，特征分析是至关重要的一环，它直接关系到后续风险评估的准确性。这一环节主要聚焦于对 apk 文件的深入解包与特征提取，以及基于这些特征进行的初步过滤与风险评估。

2.2.1 解包与特征提取

首先，系统需要对 apk 文件进行解包处理。apk 文件本质上是一个 ZIP 压缩包，包含了应用程序的所有组件和资源。通过解压 apk 文件，系统需要访问其中的 Manifest 文件、DEX 代码、资源文件（如图片、布局文件等）以及 META-INF 目录下的签名和证书信息。

在解包过程中，系统需要关注并提取出多个维度的特征指纹，这些特征构成了 apk 文件的“身份标识”和“行为轮廓”。具体包括但不限于：

1） 标题（Label）：应用程序在用户设备上显示的名称，通常用于识别应用。

2） 包名（Package Name）：唯一标识一个 Android 应用的字符串，用于区分不同的应用程序。

3） 开发者信息：包括开发者的名称、网址等，有助于追溯应用的来源。

4） 签名与证书：用于验证 apk 文件完整性和来源的签名信息及其对应的证书，是确保应用未被篡改的重要手段。

2.2.2 黑白名单对比

提取出的特征指纹随后需要与系统中预先配置的黑白名单进行对比。黑白名单是基于历史数据和专家经验构建的，包含了已知安全或恶意的 apk 特征。通过这一对比过程，系统能够迅速识别出已知的安全应用或恶意软件，实现初步过滤。

2.2.3 权限分析

除了特征指纹对比外，系统还需要对 apk 文件申请使用的系统权限进行详细分析。Android 应用程序在 Manifest 文件中声明所需权限，以访问系统资源或执行特定操作。权限滥用是恶意软件常见的行为模式之一，因此权限分析是评估 apk 风险性的重要环节。

系统需要根据 apk 申请的权限列表，评估其是否合理且必要。对于请求了过多敏感权限（如读取短信、拨打电话、访问网络等）而实际功能与之不匹配的应用，系统应当发出警告，提示用户注意潜在的风险。此外，系统还需要将某些高风险权限组合视为特定恶意行为的标志，从而进一步提高风险识别的准确性。

2.3 源码分析

源码分析是 apk 文件安全审查中的核心环节之一，系统需要提取 apk 文件内编译过的字节码，通过分析来揭示潜在的恶意行为或安全漏洞。在这一阶段，我们采用多种技术方法综合分析，以确保分析的全面性和准确性。

2.3.1 URL 提取与涉诈网站检测

首先，系统会对解包后的 apk 文件中的代码进行解析，特别是关注那些可能涉及网络通信的部分。系统需要通过静态代码分析技术，自动提取出应用程序中所有硬编码或动态生成的 URL 地址。这些 URL 地址可能是应用访问的服务器、API 接口、广告服务或其他网络资源。

接下来，系统需要对这些 URL 地址进行涉诈网站的判断。这一判断过程使用黑名单机制。黑名单机制依赖于一个不断更新的数据库，该数据库包含了已知的诈骗网站、恶意软件分发点等不安全 URL。当系统检测到 apk 中的 URL 与黑名单中的条目匹配时，需要发出警告，提示用户注意潜在的风险。

2.3.2 涉诈代码检测

除了 URL 提取与涉诈网站检测外，源码分析还涉及到对 apk 文件中代码本身的审查。这部分功能采用机器学习技术与深度学习算法实现，系统需要使用预训练好的模型识别涉诈代码。该模型通过学习大量已知恶意软件样本的代码特征，如常见的诈骗手法、恶意行为模式、异常的系统调用等，自动判别 apk 是否涉诈。

在源码分析阶段，系统会将被分析的 apk 文件的代码转化为机器学习模型可以处理的格式，然后输入到训练好的模型中。模型推理并输出一个判断结果，即该 apk 文件是否包含涉诈的相关代码。如果判断结果为阳性，系统会立即发出警告，并提供详细的分析报告和可能的恶意行为描述，以帮助用户做出进一步的决策。

2.4 图像分析

此外，除了源码分析，我们还需要进一步分析出 apk 可能的涉诈类型，仅通过源码分析往往难以全面且精确地界定黑产、赌博、色情、诈骗等复杂多变的涉诈类型。这是因为，这些不法应用在底层权限请求和代码逻辑上可能存在一定的共性。但从用户交互界面的角度来看，尤其是图像内容方面，它们之间存在着显著的差异。这些差异体现在图像的题材、色彩、构图上，对于人类来说是直观且易于区分的。

因此，引入图像分析技术成为了解决这一难题的关键步骤。图像分析不仅能够捕捉 apk 文件中图片资源的直观特征，还能通过深度学习等技术提取出更高层次的语义信息，从而为涉诈类型的精准分类提供有力支持。

系统需要采用多模态模型对图像内容，图像中包含的文字，图像描述等信息进行综合分析。多模态模型能够融合不同来源的数据（如文本、图像、音频等），在跨模态的语境下理解和生成信息，在 apk 文件的图像分析场景中，多模态模型可以结合图片内容、图片描述（如果 apk 中包含的话）、以及从其他分析模块（如特征分析、源码分析）获取的信息，进行更加全面和深入的分析。

具体而言，在图像分析部分，系统需要实现包括以下几个功能：

1） 图片提取与预处理：首先，从 apk 文件中提取出所有图片资源，并进行必要的预处理，如尺寸调整、格式转换等，以确保后续分析的一致性和准确性。提取图像描述文字或图像内的文字。

2） 特征提取：利用预先训练的模型，对预处理后的图片进行特征提取。包括图像的纹理、形状、颜色等低层次特征，也包括图像中的对象、场景、情感等高层次语义特征。

3） 多模态融合：将图像分析得到的特征与从其他分析模块（如文字提取、源码分析）获取的信息进行融合。这一过程可以通过多模态融合层或注意力机制等实现，以确保不同来源的信息能够相互补充，共同作用于涉诈类型的分类决策。

4） 分类决策：基于融合后的多模态特征，利用分类器对 apk 文件的涉诈类型进行分类。分类结果将涵盖黑产、赌博、色情、诈骗等多种涉诈类型，为用户提供清晰明了的风险提示。

2.5 分析报告生成

这部分是对前面功能的分析结果的输出。生成的报告需包括：apk 特征信息（标题、包名、开发者、签名、证书等），apk 权限分析结果（哪些权限具有风险），apk 是否涉诈，apk 涉诈类型，apk 涉诈原因，apk 涉诈代码段，apk 访问的涉诈 url，apk 的涉诈图像。

第三章 现有技术调研

3.1 文件解包工具

3.1.1 apktool

apktool 是 APK 文件逆向工程领域的常见工具，以其强大的反编译能力而广受开发者、安全研究人员及逆向工程师的青睐。该工具通过解析 APK 文件的结构，能够将其“拆解”为一系列可读的源代码和资源文件，包括 XML 布局文件、Smali（Dalvik 字节码的一种表现形式，类似 Java 的字节码但专为 Android 设计）文件、资源图片、原始 AndroidManifest.xml 文件等。

apktool 的本体是一个 Java 编写的 jar 包，这意味着它依赖于 Java 运行环境（JRE）才能执行。用户需要事先安装 Java，并在命令行中通过 java -jar apktool.jar 命令（假设 apktool.jar 是下载的 jar 文件名）来启动工具。

3.1.2 androguard

该工具使用 Python 编写，可以方便地通过 pip（Python 的包管理工具）进行安装，无需额外配置复杂的 Java 环境。androguard 的强项在于它能够快速提取 APK 文件中的关键信息，包括但不限于 APP 的标题、包名、开发者信息、签名证书以及 API 调用详情等。这些信息对于评估 APP 的安全性、功能特性或是进行逆向工程分析至关重要。

相比 apktool，androguard 在解包速度上略有优势，但值得注意的是，androguard 的解包过程并不直接将内容保存为独立文件，用户需要通过其提供的接口进行查询，用户必须在程序运行时才能访问这些数据。

3.2 源码分析

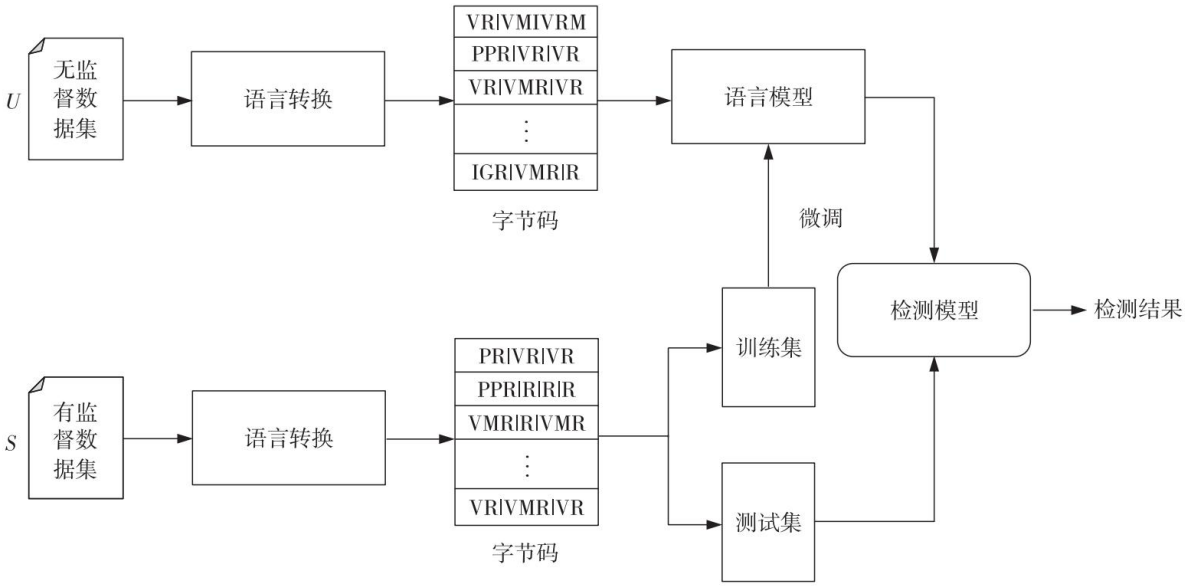
我们将涉诈 apk 的源码分析定义为一个二分类任务，即模型需要根据源码判断该 apk 是否涉诈。

这部分与恶意代码分析相类似。我们调研了用机器学习进行安卓恶意代码分类的相关论文，以下是我们的调研结果：

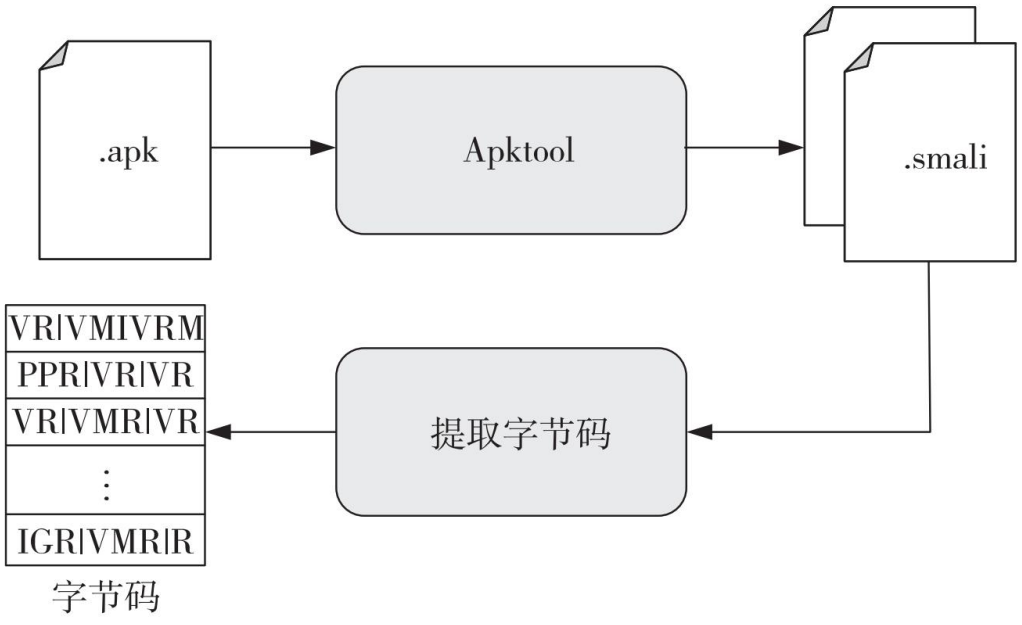
3.2.1 数据处理

印杰等人的《基于预训练语言模型的安卓恶意软件检测方法》，提出了基于预训练语言模型的安卓恶意软件检测方法。

论文中提到的模型训练方法主要包括语言转换、语言模型预训练及语言模型微调 3 个阶段。首先是语言转换阶段,利用反汇编软件对 APK 文件进行反汇编,从 smali 文件中提取出对应的字节码序列。然后是预训练阶段,以无监督数据集的字节码序列作为输入,基于无监督的语言模型训练方法训练语言模型;最后是微调阶段,基于有监督数据集在预训练语言模型上进行微调,得到最终的恶意软件检测模型。



在数据处理部分，论文采用 Apktool 实现应用程序的反汇编。使用 Apktool 工具对 APK 样本进行解包,解包完成后,会 将生成的结果输出到指定路径中。其中 smali 文件夹下就是最终生成的 DalvikVM 汇编代码,AndroidManifest.xml 文件以及 res 目录下的资源文件也已被解码。



Android 应用程序常以压缩文 Android 应用程序包 APK 的形式交付,包含清单文件、资源 文件和 Dalvik 可执行文件 Dex(Dalvik VM executes)。Dex 文件包含应用 程序源代码,可 采 用 baksmali 进行反汇编。作为反汇编的结果,baksmali 为 Dex 文件生成了一组 smali 文件,其中每个 smali 文件代表一个包含该类所有方法的单个类。每个方法都包含可读的 Dalvik 字节码 (简称指令),每个指令有一个操作码和多个操作数。由于 Dalvik 指令有两百多条,可对其进行分类与精简, 去除无关的指令参数,仅保留了 7 大类核心指令集。

APK 文件经反汇编后,经过上述处理被进一步转换为不定长的字节码序列。论文采用定长方式将字节码划分成句，输入模型进行训练。

3.2.2 训练方法

McLaughlin 等人的《Deep Android Malware Detection》使用基于带有 embedding 层的 CNN 模型进行安卓的恶意代码分析。该论文使用深度卷积神经网络（CNN）。恶意软件分类基于反汇编程序中的原始操作码序列的静态分析。网络自动从原始操作码序列中学习表示恶意软件的特征，从而消除了对手动设计恶意软件特征的需求。与现有的基于 n-gram 的恶意软件检测方法相比，该论文的训练流程更为简单，因为网络是端到端训练的，能够同时学习适当的特征和进行分类，从而省去了在训练过程中显式枚举数百万个 n-gram 的需要。

Saracino 等人的《Graph-Based Android Malware Detection and Categorization through BERT Transformer》则使用 BERT 对 apk 文件的 api 调用关系进行分析，起到检测恶意软件的作用。该论文使用 BERT 对从 Android API 调用图中生成的 API 调用序列进行分类。通过利用 API 调用图，捕获 API 调用之间复杂的关系和依赖，从而能够更深入地理解 Android 恶意软件的行为表现，从而达到更高的分析准确度。

3.2.3 模型对比

Esra Calik Bayazit 等人的《A Deep Learning Based Android Malware Detection System with Static Analysis》，对比了各类 RNN（循环神经网络）模型在安卓恶意代码分析任务中的效果。结果表明，BiLSTM 模型优于其他基于 RNN 的深度学习方法，准确率为 98.85%。

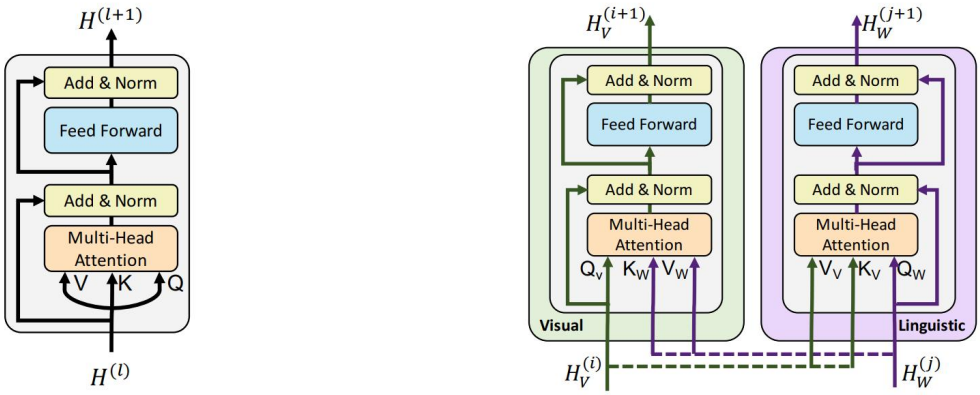
3.3 多模态图像分类

在处理 APK 文件分类的复杂任务时，尤其是当目标是将这些文件根据其潜在内容（如赌博、黑色内容、白色内容、色情、诈骗等）进行细致分类时，单纯依赖图像分析可能面临一定的局限性，因为 APK 文件本身不仅仅是图像数据的集合，还包含了代码、元数据等多种信息。然而，通过提取 APK 中的关键图像资源（如应用图标、截图等）并结合多模态图像分类技术，可以显著提升分类的准确性和效率。以下是对上述多模态模型在 APK 图像分类中潜在应用的详细探讨：

3.3.1 ViLBERT

ViLBERT 在预训练方面，主要面向 vision 做预训练。文本部分仍然是直接采用 BERT 的训练后的参数进行初始化，图像部分使用的是 Fast-CNN 模型进行大量数据的预训练。采用概念字幕数据集来进行训练。将 image-text 向量整合到一起并理解他们之间的关系。(通过多模态的掩码建模和多模态对齐预测两种方式来实现)

在推理过程中，每个流（视觉和语言）不仅处理自己模态的信息（通过自注意力机制），而且还能够参与到另一个模态的信息处理中。这是通过使用来自另一模态的键（K）和值（V）来实现的，而查询（Q）则来自于当前模态。例如，在图 (b) 中，视觉流使用语言流中的键和值进行注意力计算，反之亦然。这样，每个模态都能在自己的处理流程中集成另一模态的信息，从而使模型能够更好地理解和整合跨模态的数据。



(a) Standard encoder transformer block　　(b) Our co-attention transformer layer

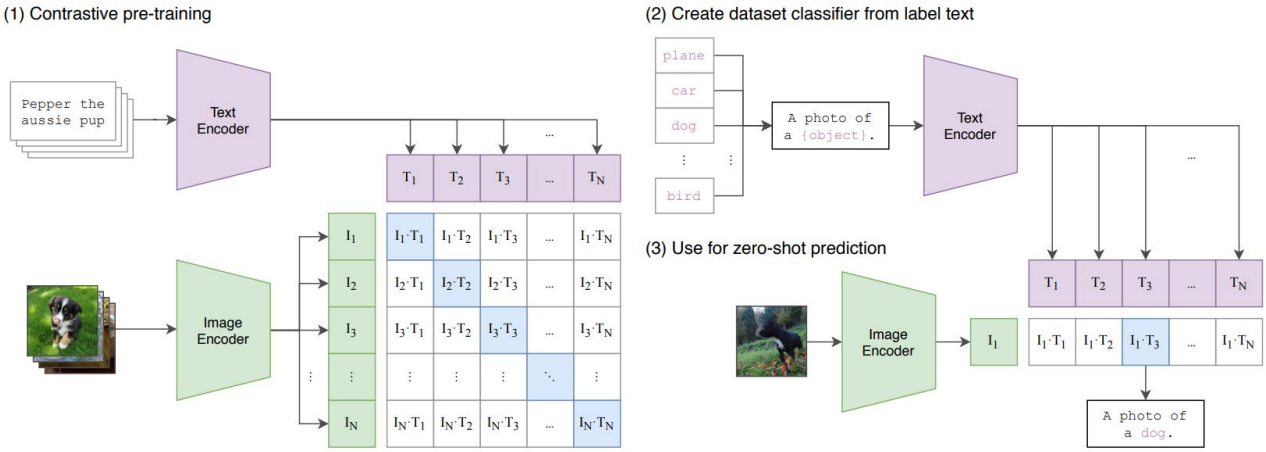
ViLBERT 是将 BERT 应用与多模态领域的代表性论文之一，具有强大的视觉与语言融合能力。

3.3.2 CLIP

CLIP（Contrastive Language-Image Pre-training）是一种由 OpenAI 开发的多模态预训

练模型，它主要学习了如何对文本和图像进行对比，从而实现跨模态的理解。CLIP 模型的核心思想是将文本和图像嵌入到一个共同的语义空间中，使得相关的文本描述和图像内容在这个空间中的表示彼此靠近，而不相关的则远离。这种设计使得 CLIP 模型能够在各种任务上表现出色，如图像分类、图像检索、文本分类等。

CLIP 在预训练阶段，对比学习十分灵活，只需要定义好正样本对和负样本对就行，其中能够配对的图片-文本对即为正样本。具体来说，先分别对图像和文本提特征，这时图像对应生成 I1、I2 ... In 的特征向量，文本对应生成 T1、T2 ... Tn 的特征向量，然后中间对角线为正样本，其余均为负样本。这样的话就形成了 n 个正样本，n^2 - n 个负样本。一旦有了正负样本，模型就可以通过对比学习的方式训练起来了，完全不需要手工的标注。



CLIP 模型的特点包括：

1） 统一的向量空间：CLIP 将图像和文本都映射到同一个向量空间中，这使得模型能够直接在向量空间中计算图像和文本之间的相似性，而无需额外的中间表示。

2） 对比学习：CLIP 使用对比学习的方式进行预训练，模型被要求将来自同一个样本的图像和文本嵌入映射到相近的位置，而将来自不同样本的嵌入映射到较远的位置。这使得模型能够学习到图像和文本之间的共同特征。

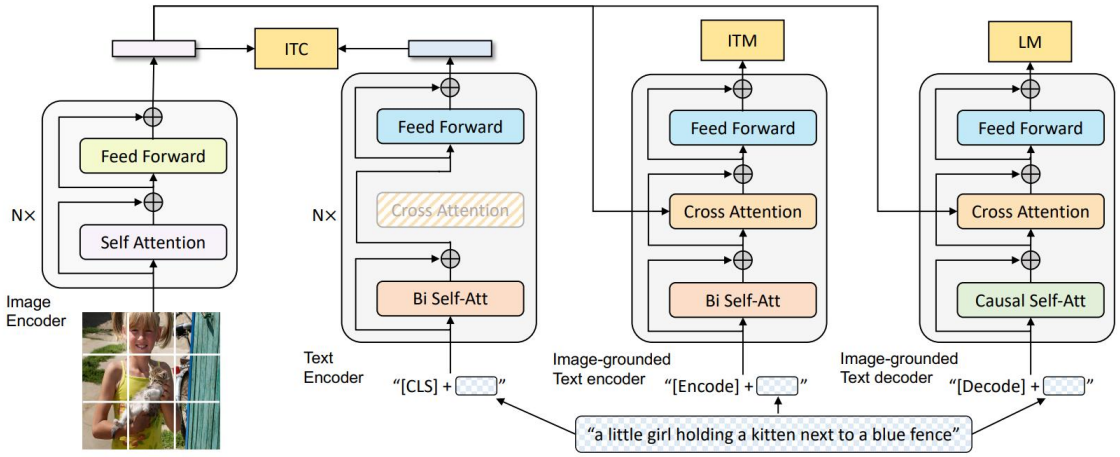
3） 无监督学习：CLIP 的预训练是无监督的，这意味着它不需要大量标注数据来指导训练。它从互联网上的文本和图像数据中学习，使得它在各种领域的任务上都能够表现出色。

3.3.3 BLIP

BLIP（Bootstrapping Language-Image Pre-training）是一个创新的视觉语言预训练框

架，旨在统一视觉语言理解和生成任务。它通过引导字幕有效地利用了嘈杂的网络数据，通过生成合成字幕和过滤嘈杂字幕的方法，优化了现有的视觉语言预训练（VLP）模型的缺陷。

BLIP 的设计主要解决了两个关键问题：一是大多数现有的 VLP 方法只能擅长于理解或生成任务，而 BLIP 通过统一的框架同时优化了这两方面；二是通过有效利用嘈杂的网络数据，BLIP 提高了模型的性能，尽管这些数据可能包含噪声。此外，BLIP 还引入了新的技术组件，如单模态编码器和以图像为基础的文本编码器/解码器，以及特定的损失函数，以实现其独特的功能和优化目标。

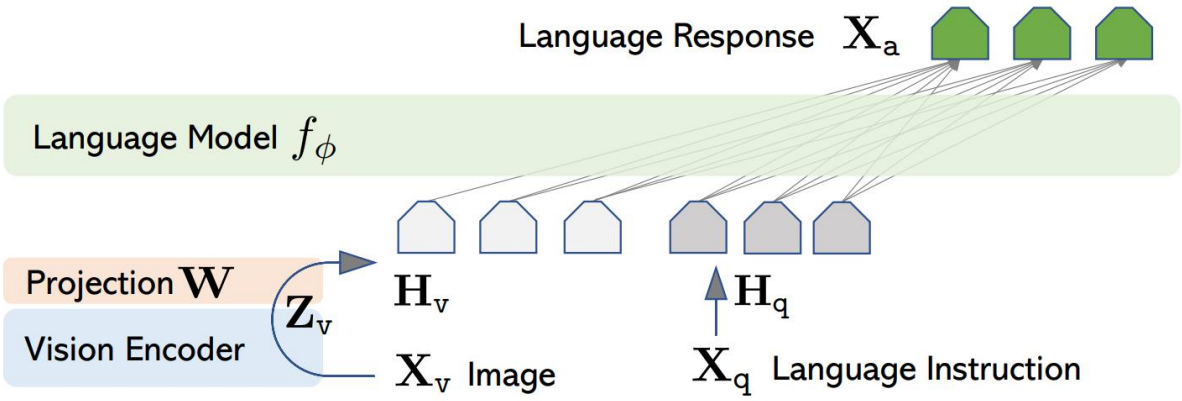


BLIP 模型是一个 Mixture of Encoder-Decoder（MED）结构。模型作为 Encoder 的时候，Transformer Layer 包含双向的 self-attention 和 FFN 层，作为 Decoder 的时候，双向的 self-attention 变为 Causal 的单向 self-attention，中间还会插入 Cross Attention 层用于将图片特征信息加入到文本特征中。

BLIP 同时兼顾了图文匹配和图文生成方面的性能。BLIP 在多种视觉语言任务上都取得了较好的效果，具有很强的迁移能力，性能甚至超越了 CLIP。

3.3.4 LLaVA

LLaVA(Large Language and Vision Assistant)，即大型语言和视觉助手，是一个端到端训练的大型多模态模型，将视觉编码器和大语言模型连接起来实现通用的视觉和语言理解。



LLaVA 是利用纯文本 GPT-4 将现有 COCO 边界框和图像描述数据集扩展为多模态指令跟随数据集的先驱，该数据集包含三种类型的指令跟随数据：对话式 QA、详细描述和复杂推理 。

LLaVA 的主要贡献：

1） 多模态指令跟随数据集：一项关键挑战是缺乏视觉语言指令跟随数据集，使用 ChatGPT/GPT-4 将图像文本对转换为适当的指令遵循数据格式，生成三种类型的指令跟随数据：对话、详细描述、复杂推理。总共收集了 158K 个语言图像指令跟随数据样本，其中对话样本 58K，详细描述样本 23K，复杂推理样本 77K。

2） 大型多模态模型：开发了一个大型多模态模型（LMM），通过将 CLIP 的开放集视觉编码器与 LLaMA 语言解码器连接起来，并在生成的视觉语言指令跟随数据集上对它们进行端到端的微调。

第四章 系统实现

4.1 自动下载

apk 自动下载方面，当用户提供 url 时，如果 url 本身就是某个 apk 的下载链接，则直接下载。如果不是，我们采用爬虫程序的写法，即根据输入的 url，递归爬取其子页面下的所有 apk 链接。考虑到多数网页的下载地址都是由 js 代码动态加载的，一般的爬虫抓取到的 html 文本不含下载链接，我们采用 chrome 浏览器内核驱动，模拟用户真实的浏览行为，获取动态渲染后的网页。同时，通过添加随机的点击和滚动操作，确保能够获取到隐藏的下载链接。搜索网页后即可获取到该网页下包含的所有 apk 下载链接。

将链接中的各个下载地址与预先设定的黑白名单进行比对，如果该 url 或者该 apk 的包名处在白名单中，则无需下载。

此外，对于用户提供的二维码，程序将提取二维码内包含的下载链接，自动获取相应的 apk。此外，用户也可直接提供 apk 文件，系统将统一管理上述三种途径获取到 apk 文件。

下载采用多线程下载下载，提供进度条实时反映下载进度，下载完成的 apk 文件存放在特定的文件夹中（文件夹路径可修改）。

4.2 特征分析

此功能具体分为两个部分：对 apk 文件的解包，以及对 apk 文件的特征进行分析。

4.2.1 文件解包

基于前文提到的两种 apk 解包工具，apktool 可以将解包后的文件保存起来，而 androguard 则可以直接提取 apk 文件中的特征信息。

因此，我们同时采用了 apktool 和 androguard。apktool 负责对 apk 文件进行解包，以供后面的 apk 源码分析和 apk 图像分析使用。而 androguard 则负责直接对 apk 文件的特征信息进行提取。

4.2.2 分析过程

对提取完的特征信息（标题、包名、开发者、签名、证书、权限），采取静态的方式进行分析。对于标题、包名、开发者等信息，通过与黑名单进行比较来进行初步的排查。对于签名、证书信息，通过 androguard 自带的工具检测其是否合法。对于权限，通

过静态设定高危权限的方式进行排查（例如读取文件列表、修改系统设置、浮于应用上层等）。

4.3 源码分析

apk 源码分析同样分成两部分：后台通联地址分析以及 apk 源代码分析。

4.3.1 后台通联地址分析

后台通联地址分析部分，通过正则表达式匹配 apk 源码中出现的后台通联地址，通过与黑名单中的 url 进行比较，判断是否涉诈。

4.3.2 源代码分析

apk 源代码分析部分，使用语言模型对 apk 的源代码进行二分类，判断其是否涉诈。对于题目提供的 apk 数据，我们将 batch1-4 作为训练集，其余作为测试集。

4.3.3 模型选择

基于前文对恶意代码分类模型的调研结果可知，在代码分类任务中，循环神经网络和 BERT 模型各有优劣。循环神经网络在训练数据较少的情况下，模型的准确率高于 BERT。而在训练数据较多时，BERT 的预训练效果更佳，优于循环神经网络。

但由于本题提供的数据集较小，一共仅 500 左右的 apk 文件，且在不同类别上存在数据不平衡的现象（有些 apk 无法解包）。因此，BERT 的训练效果不算好，而在循环神经网络中，又属 BiLSTM（双向长短期记忆网络）对代码分类任务的效果最佳，因此我们最终选择了 BiLSTM 模型。

4.3.4 模型训练

模型的训练方面，我们参考了《基于预训练语言模型的安卓恶意软件检测方法》。首先从解包后的 apk 文件中提取 smali 文件（apk 源码的反汇编代码），然后将该 apk 的所有 smali 文件转换为 Dalvik 字节码（具体转换规则见`DalvikOpcodes.txt`文件），并一起按行存入一个 txt 文件中。

在训练时，以长度为 512 字节的指令序列作为单个样本打上标签进行模型的训练。长，序列长度不足 512 时自动填充 0。BiLSTM 的隐藏层维度为 512，训练时 batch size 设置为 128，学习率设置为 0.002。

4.3.5 模型预测

在模型的预测任务中，同样将 apk 文件的源码分解为若干长度为 512 字节的指令序

列作为输入。将分割后的序列全部送入模型推理，统计所有被判定为涉诈代码段的序列个数，记录并等待后续使用。

4.4 图像分析

4.4.1 模型选择

由于题目给出的 apk 文件中，图片的数量较少，我们倾向于选择能具有零样本或者少样本迁移能力的模型。又由于本题实际上是图片的多分类任务，如果使用多模态大语言模型，可能对机器的配置有比较高的要求，并不是特别必要。

最终，经过调研，我们选择使用了 CLIP 模型的中文版本，其继承了 CLIP 的零样本学习能力，在图像分类任务上有比较好的表现，且由于其是在中文环境下进行预训练的，对本题数据中的 apk 图像适应性更强，效果更好。

4.4.2 分析过程

具体分析步骤则是从解包的文件中提取图像文件（png，jpg 等），向模型输入得到其与`[“赌博”，“色情”，“黑色产业”，“正常”，“诈骗”]`等词语之间的相似度，相似度最高且大于 0.9 的则被归类为那一分类。由此来判断 apk 图像乃至 apk 文件的涉诈类型。

4.5 报告生成

报告生成部分，主要是将前面提到的分析内容做一个整合，并按照给定的格式进行输出。报告内容共包含了前文所述的全部信息，并配有统计图。在分析的原因部分，我们调用了大语言模型对分析报告进行润色。

4.6 黑白名单

4.6.1 初始值

我们将使用题目提供的 excel 文件作为黑白名单的初始值。这个文件包含了已知的安全威胁（如恶意域名、病毒签名等）和可信来源（如官方应用市场的域名、知名开发者的签名等）。同时，我们还会在 url 部分添加一些常见的境外涉诈域名，以扩大黑名单的覆盖范围。

4.6.2 自定义设置

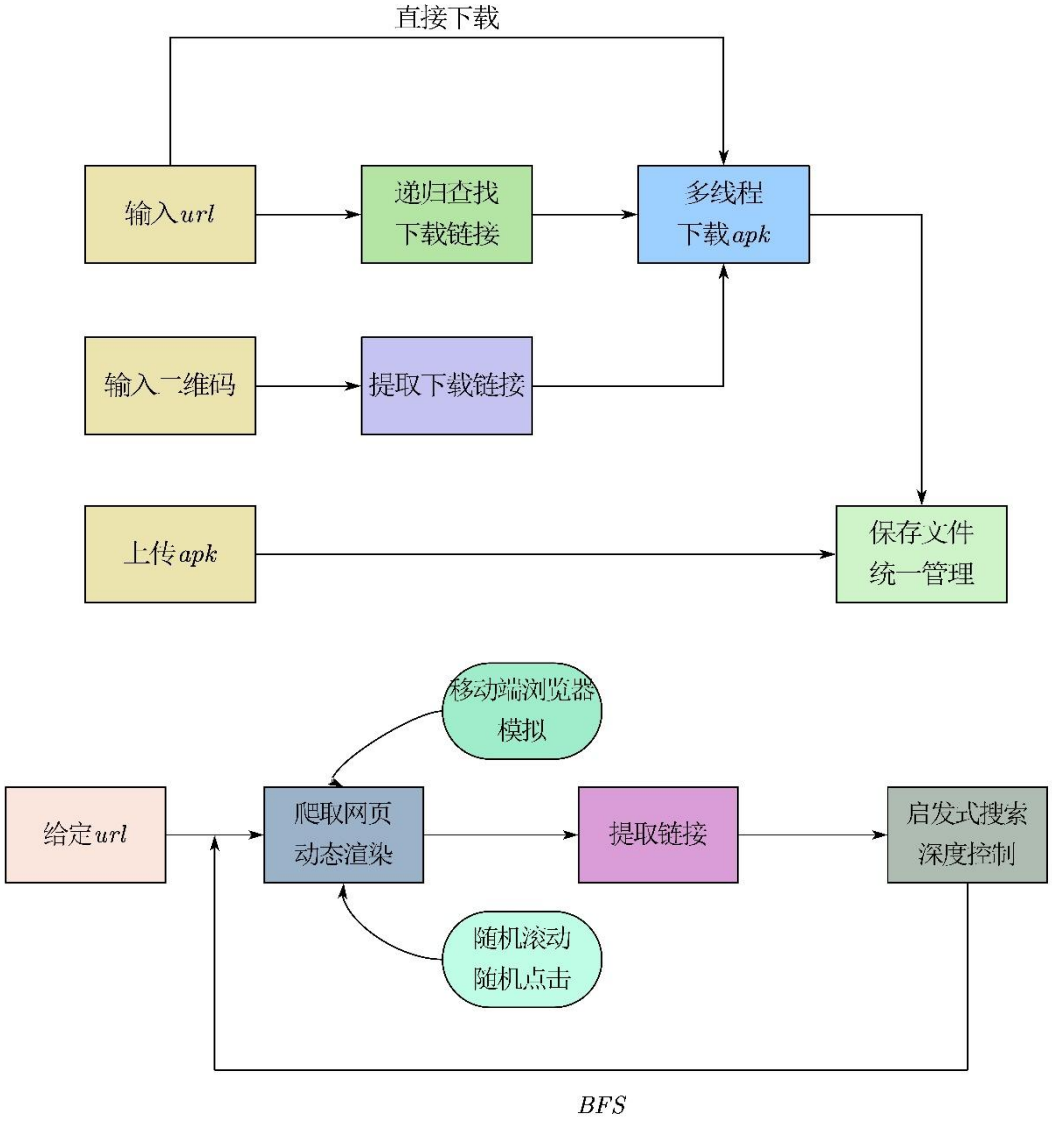
为了满足用户的个性化需求，我们提供了自定义设置功能。用户可以通过写入指定 txt 文件的方式，向黑白名单中添加或删除条目。此外，我们还在用户界面上提供了输入框，允许用户直接输入要添加或删除的条目信息，提高了操作的便捷性。

4.6.3 自动更新

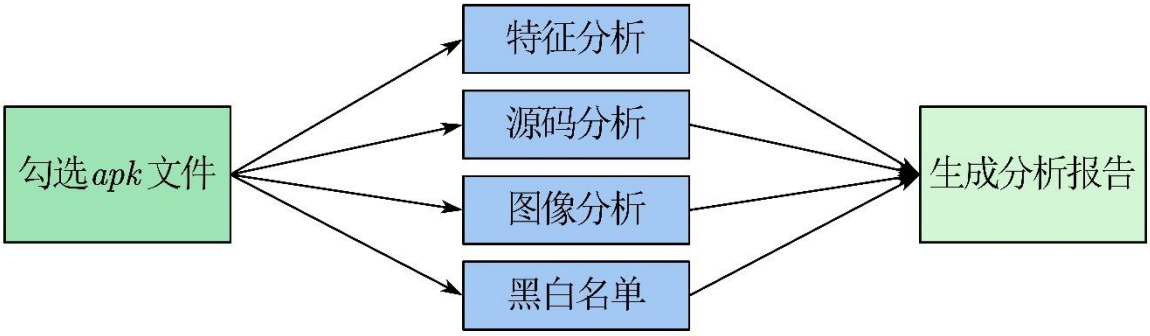
为了保持黑白名单的时效性和准确性，我们实现了自动更新机制。每当用户调用 apk 分析功能并得出涉诈结论时，系统会自动将该 apk 及其相关的后台通联地址添加到黑名单中。这样，随着系统的不断使用，黑白名单将逐渐积累更多的安全威胁信息，从而提高系统对未知安全威胁的识别能力。同时，我们也将定期审查黑白名单中的条目，确保其准确性和有效性。

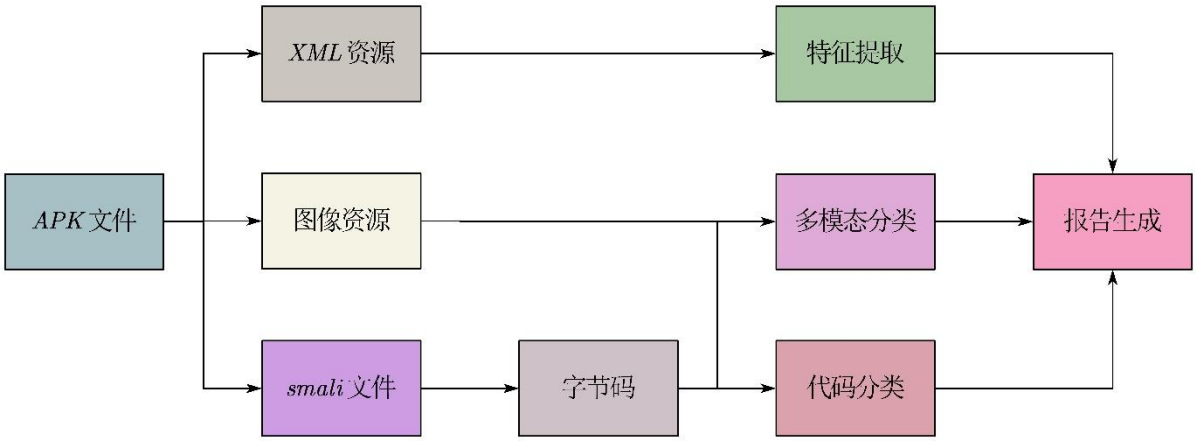
第五章 系统功能执行流程

5.1 apk 获取



5.2 静态分析

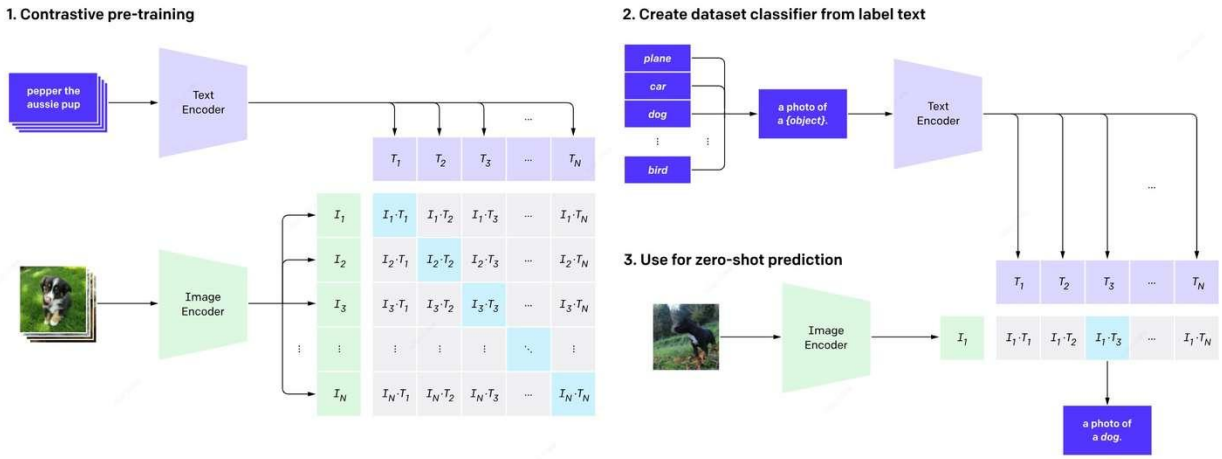



APK 文件
XML 资源
图像资源
smali 文件
字节码
特征提取
多模态分类
代码分类
报告生成

第六章 算法及模型说明

6.1 CLIP 模型说明

本项目采用了预训练的 CLIP 模型处理图像信息，该模型包括两个部分：文本编码器（Text Encoder）和图像编码器（Image Encoder)。Text Encoder 选择的是 Text Transformer 模型；Image Encoder 选择了两种模型，一是基于 CNN 的 ResNet（对比了不同层数的 ResNet），二是基于 Transformer 的 ViT。



6.1.1 预训练

对于一个包含 N 个<文本-图像>对的训练 batch，使用 Text Encoder 和 Image Encoder 提取 N 个文本特征和 N 个图像特征。这里共有 N 个正样本，即真正属于一对的文本和图像（矩阵中的对角线元素），而剩余的文本-图像对为负样本。

将 N 个文本特征和 N 个图像特征两两组合，CLIP 模型会预测出可能的文本-图像对的相似度，这里的相似度为文本特征和图像特征的余弦相似性（cosine similarity）。

CLIP 的训练目标就是最大化 N 个正样本的相似度，同时最小化负样本的相似度，即最大化对角线中蓝色的数值，最小化其它非对角线的数值。

6.1.2 zero-shot 图像分类

训练后的 CLIP 其实是两个模型：视觉模型+文本模型，与 CV 中常用的先预训练然后微调不同，CLIP 可以直接实现 zero-shot 的图像分类，即不需要任何训练数据，就能在某个具体下游任务上实现分类。

根据任务的分类标签构建每个类别的描述文本，然后将这些文本送入 Text Encoder 得到对应的文本特征。将要预测的图像送入 Image Encoder 得到图像特征，然后与 N 个文本特征计算缩放的余弦相似度（和训练过程一致），然后选择相似度最大的文本对应

的类别作为图像分类预测结果，进一步地，可以将这些相似度看成 logits，送入 softmax 后可以到每个类别的预测概率。预测概率最大的类别即为该图像所属的类别。

6.1.3 模型规模

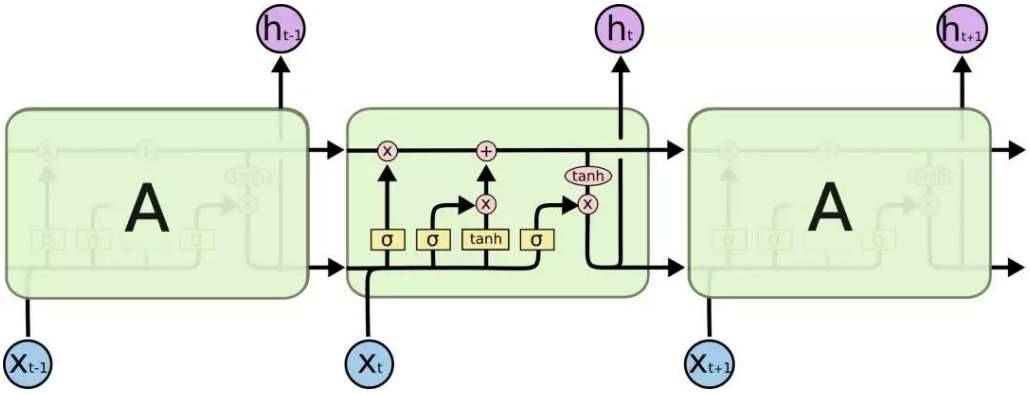
本文选取的 Chinese-CLIP 模型目前开源 5 个不同规模，其模型信息见下表：

模型规模	参数量	视觉侧骨架	视觉侧参数量	文本侧骨架	文本侧参数量	分辨率
$\text{CN-CLIP}_{\text{RN50}}$	77M	ResNet50	38M	RBT3	39M	224
$\text{CN-CLIP}_{\text{ViT-B/16}}$	188M	ViT-B/16	86M	RoBERTa-wwm-Base	102M	224
$\text{CN-CLIP}_{\text{ViT-L/14}}$	406M	ViT-L/14	304M	RoBERTa-wwm-Base	102M	224
$\text{CN-CLIP}_{\text{ViT-L/14@336px}}$	407M	ViT-L/14	304M	RoBERTa-wwm-Base	102M	336
$\text{CN-CLIP}_{\text{ViT-H/14}}$	958M	ViT-H/14	632M	RoBERTa-wwm-Large	326M	224

6.2 BiLSTM 模型说明

本项目采用 BiLSTM 模型对 APK 文件中提取出的字节码做二分类。双向长短期记忆网络（BiLSTM） 是一种特殊类型的循环神经网络（RNN），它在每个时间步同时处理过去和未来的信息。模型通过引入门控机制（包括输入门、遗忘门和输出门）解决了传统 RNN 的梯度消失问题。这些门控机制允许网络学习何时保留或遗忘信息，从而能够捕捉长期依赖关系。这种结构特别适合于需要同时考虑上下文信息的任务，如自然语言处理和时间序列分析。

6.2.1 模型结构



上图展示了一个 LSTM 单元的计算流程。LSTM 模型是由 t 时刻的输入词 Xt，细胞状态 Ct，隐层状态 h，遗忘门 f，记忆门 i，输出门 o 组成。LSTM 的计算过程可以概括为，通过对细胞状态中信息遗忘和记忆新的信息使得对后续时刻计算有用

的信息得以传递，而无用的信息被丢弃，并在每个时间步都会输出隐层状态 h，其中遗忘，记忆与输出由通过上个时刻的隐层状态和当前输入 Xt 计算出来的遗忘门，记忆门，输出门来控制。

6.2.2 模型规模

在本项目中，我们为 BiLSTM 模型设置了以下参数：

vocab_size: 词汇表大小，即模型能够识别的不同单词或标记的数量。在本项目中，我们将其设置为 256。

embedding_dim: 嵌入维度，即每个单词向量的维度。我们将其设置为 256，这允许模型学习丰富的单词表示。

hidden_dim: LSTM 层的隐藏状态的维度。我们将其设置为 512，这为模型提供了足够的容量来捕捉复杂的模式。

num_layers: LSTM 层的数量。我们使用 2 层 LSTM，这有助于模型学习更深层次的特征。

num_classes: 输出类别的数量。在本项目中，我们将其设置为 2。

6.2.3 超参数

除了模型的架构参数外，我们还设置了一些超参数来优化模型的训练过程：

batch_size: 每个训练批次的样本数量。我们将其设置为 128。

num_epochs: 训练的总轮数。我们将其设置为 15。

learning_rate: 优化算法（Adam）的学习率。我们将其设置为 0.002。

以上超参数的设置可以使得训练后模型的预测效果达到最优

6.3 爬虫算法说明

算法基于广度优先搜索 BFS，对于给定的初始链接 url，爬取页面的 html 源码，提取所有 a 标签中的链接，添加至 BFS 队列中。考虑到大多数下载链接都是写在 js 中，在浏览器加载网页时动态渲染的，因此，直接发送 http 请求得到的结果往往无法提取到有效的下载链接。为此，本算法采取三种额外措施：采用 selenium 库操作浏览器驱动程序，爬取 html 后本地渲染；动态调整请求头，模拟移动端和桌面端的不同情况，防止某些链接只能在移动端网页渲染；利用浏览器驱动程序发起随机滚动和点击，进一步确保链接能够渲染在 html 中。

BFS 搜索时，采取启发式搜索策略，并通过限制搜索深度确保能够在较短时间内爬

取到链接并结束搜索。具体的。爬取的深度不超过两层，当链接中带有下载关键字时，优先爬取，当搜索队列长度超过 30 时，将不含搜索关键字的队列中链接删除一般。以上两种措施保证了搜索的速度和质量。

第七章 创新点

7.1 高效涉诈代码检测模型

在代码检测机制中，我们使用了当前恶意代码分类领域的 SOTA 模型——基于双向长短期记忆网络（BiLSTM）。这一模型不仅深度挖掘了代码中的潜在欺诈模式，还通过其卓越的序列处理能力，对代码的逻辑流与结构进行了细致入微的分析。

我们充分利用了题目特供的 APK 数据集，对 BiLSTM 模型进行了预训练，使其能够精准地适应并识别本题特定场景下的欺诈行为，从而实现高准确度检测。

7.2 多模态分析技术

系统引入了 Chinese-CLIP 多模态模型，对 APK 文件中的图像资源进行深入分析。通过提取图像的纹理、形状、颜色等特征，输入 Chinese-CLIP 模型中计算标签相似度，结合文本描述和源码分析，系统能够更准确地识别涉诈类型。

系统利用多模态融合技术，系统能够整合来自不同模态的信息，提供更全面的分析结果。这种方法显著提高了对涉诈 APP 的识别准确性。

7.3 浏览行为仿真爬虫

系统采用 BFS 算法，从给定的初始 URL 开始，逐层爬取网页及其子页面，寻找下载链接。这种方法确保了爬虫能够高效地发现所有潜在的下载链接。且考虑到许多下载链接是通过 JavaScript 动态生成的，系统采用 Selenium 等工具模拟真实用户的浏览行为，获取动态渲染后的网页内容。这种方法提高了下载链接的发现率。

7.4 智能报告生成

在报告内容方面，智能报告生成系统展现了高度的专业性和全面性。首先，它详细列出了 APK 的基本特征信息，包括但不限于应用的名称、版本号、包名、签名证书、开发者信息等。接着，系统通过权限分析模块，全面解析 APK 请求的各类系统权限，并对比行业标准与最佳实践，评估这些权限请求是否合理，是否存在过度索权的风险，为用户揭示应用可能存在的隐私泄露或恶意行为风险。

在源码分析部分，系统首先提取 APK 文件中的后台通联地址，与黑名单进行比对，输出危险 url 信息。然后，系统调用 BiLSTM 模型，对 APK 文件的字节码展开涉诈代码检测，在报告中给出判断结果，以及涉诈代码的占比。（饼图的形式）

在图像分析部分，系统使用 Chinese-CLIP 对 APK 文件的图像进行分析，并在报告中给出涉诈图像的分类结果（“赌博”，“诈骗”，“色情”），并给出涉诈图像的占比（饼图）。在分析报告的最后，展示涉诈图像。

在总结部分，系统将综合特征分析，源码分析，图像分析三部分的分析结果，给出最终的 APK 分析结果，包括 APK 是否涉诈以及涉诈类型。并计算涉诈分析的置信度。

7.5 高并发

在单个 APK 的运行过程中，我们设计了分析流程，实现了高并发处理，使得特征分析，源码分析，图像分析三个模块可以并发执行。通过降低分析步骤间的耦合度，我们确保了各个模块之间的独立性与灵活性，使得它们能够并发执行，互不干扰地处理不同任务。

对于多个 APK 文件的分析，系统同样支持并发处理。这种设计不仅提升了处理速度，还确保了在高负载环境下的稳定性能，保证了系统的吞吐量。

总结与展望

本项目成功开发了一个涉诈 APP 智能识别分析系统，通过自动化的网络爬虫技术、文件下载、特征提取和机器学习模型，实现了对涉诈 APP 的智能识别和分析。项目圆满完成题目要求的同时，采用了多种前言技术，在实验验证中表现出良好的用户使用体验和较高的识别准确率。

在本项目开发的全流程中，采用 git 进行版本管理，飞书进行开发进度管理，实现全部资料云平台托管，团队成员整体协作顺畅，开发流程管理高效可控。

受限于时间与精力，本项目在如下方面仍存在改进空间，包括进一步优化深度学习算法，提高系统的识别准确率和响应速度。例如，通过引入更先进的深度学习模型和自然语言处理技术，提升对恶意代码和图像内容的分析能力；进一步挖掘 APK 文件中的数据并处理；构建更完善的数据集；进一步完善客户端用户使用体验；开发移动端应用等等。团队成员将在未来持续维护并开发本项目。

总的来说，完成本项目的过程中团队成员投入了大量时间，并学习了多种前沿技术，在开发过程中锻炼了自己的专业能力。再次对软件杯官方提供的平台，指导老师和团队成员表示感谢！

参考文献

[1] 印杰, et al. "基于预训练语言模型的安卓恶意软件检测方法." 计算机工程与科学 45.08 (2023): 1433.

[2] Saracino, Andrea, and Marco Simoni. "Graph-based android malware detection and categorization through bert transformer." Proceedings of the 18th International Conference on Availability, Reliability and Security. 2023.

[3] McLaughlin, Niall, et al. "Deep android malware detection." Proceedings of the seventh ACM on conference on data and application security and privacy. 2017.

[4] Bayazit, Esra Calik, Ozgur Koray Sahingoz, and Buket Dogan. "A deep learning based android malware detection system with static analysis." 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). IEEE, 2022.

[5] Calik Bayazit, Esra, Ozgur Koray Sahingoz, and Buket Dogan. "Deep learning based malware detection for android systems: A Comparative Analysis." Tehnički vjesnik 30.3 (2023): 787-796.

[6] Hafner, Markus, et al. "CLIP and complementary methods." Nature Reviews Methods Primers 1.1 (2021): 1-23.

[7] Lu, Jiasen, et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks." Advances in neural information processing systems 32 (2019).

[8] Li, Junnan, et al. "Blip: Bootstrap** language-image pre-training for unified vision-language understanding and generation." International conference on machine learning. PMLR, 2022.

[9] Li, Junnan, et al. "Blip-2: Bootstrap** language-image pre-training with frozen image encoders and large language models." International conference on machine learning. PMLR, 2023.

[10] Yang, An, et al. "Chinese clip: Contrastive vision-language pretraining in chinese." arxiv preprint arxiv:2211.01335 (2022).