ارائهٔ یک راه‌کار مبتنی‌بر یادگیری عمیق برای تشخیص بدافزارهای اندرویدی

علی علیائی طرقبه[۱]، عباس رسول‌زادگان[۲*]

دانشجوی دکترای مهندسی کامپیوتر – نرم‌افزار، گروه مهندسی کامپیوتر، دانشکدهٔ مهندسی، دانشگاه فردوسی مشهد، مشهد، ایران[۱]

دانشیار گروه مهندسی کامپیوتر، دانشکدهٔ مهندسی، دانشگاه فردوسی مشهد، مشهد، ایران[۲*]

چکیده

اندروید، سیستم عاملی متن‌باز با سهم بازار گسترده، یکی از اهداف اصلی توسعه‌دهندگان بدافزار است. افزایش تنوع دستگاه‌های مبتنی‌بر اندروید و پیچیدگی روزافزون بدافزارها، شناسایی آن‌ها را به چالشی جدی تبدیل کرده‌است. در این پژوهش، روشی مبتنی‌بر یادگیری عمیق برای تشخیص بدافزارها ارائه شده‌است که از تحلیل ایستا بهره می‌برد؛ این روش با تبدیل بایت‌کدهای خام برنامه‌ها به سیگنال‌های صوتی، استخراج اطلاعات مهم از این سیگنال‌ها و آموزش یک مدل با کمک یادگیری انتقالی، به صحت ۹۹.۳درصد و دقت ۹۹.۸درصد رسید. با تکیه‌بر نگاشت بایت‌کدهای خام به حوزهٔ صوت، علاوه‌بر ایجاد دیدگاه جعبه‌سیاه‌گونه، پیچیدگی محاسباتی کاهش و دقت شناسایی افزایش پیدا کرد. یکی از نوآوری‌های این روش که آن را برای استفاده در کاربردهای صنعتی و بازارهای اندرویدی مناسب می‌سازد، قابلیت آن در تشخیص بدافزارهای مبهم‌شده و عملکرد موفق آن در این زمینه است؛ موضوعی که یکی از چالش‌های اساسی در حوزهٔ تشخیص بدافزارهای اندرویدی به‌شمار می‌رود.

واژگان کلیدی: بدافزارهای اندروید، یادگیری عمیق، تحلیل ایستا، پردازش صوت، پردازش تصویر.

A Deep Learning Based Method for Android Malware Detection

Ali Olyaei Torqabeh[1], Abbas Rasoolzadegan*[2]

PhD Student of Software Engineering, Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran[1]

Associate Professor of Software Engineering, Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran[2*]

Abstract

Android malware remains a significant cybersecurity concern, with McAfee's 2022 report calling it "The Year of Stealthy Mobile Attacks." The open-source nature of Android and the widespread use of third-party app sources make it a prime target for cybercriminals. Malware variants such as Trojans, spyware, and worms often disguise themselves as legitimate applications while secretly executing malicious activities. Attackers employ techniques like repackaging apps, injecting harmful updates, and distributing deceptive downloads to infiltrate devices. In 2023 alone, over 600 million malware-infected applications were downloaded from Google Play, underscoring the urgent need for more sophisticated detection strategies. Traditional signature-based methods struggle against rapidly evolving threats, necessitating the adoption of more adaptive and intelligent approaches such as deep learning.

This research presents an innovative Android malware detection framework utilizing static analysis. Instead of relying on conventional feature extraction techniques, this approach transforms raw APK byte sequences into audio signals, allowing the application of spectral analysis methods. The conversion process begins by interpreting the raw byte sequence of an APK file as a one-dimensional numerical signal, where each byte value (0–255) is mapped to an amplitude level. To avoid loss of structural information, the sequence is reshaped and passed through a waveform generation process, effectively converting the byte stream into an audio signal. This transformation preserves the sequential dependencies inherent in the bytecode, allowing for effective feature extraction in the frequency domain. Once the APK file is represented as an audio waveform, cepstral analysis techniques such as Mel-Frequency Cepstral Coefficients (MFCCs), Bark-Frequency Cepstral Coefficients (BFCCs), and Gammatone-Frequency Cepstral Coefficients (GFCCs) are applied. These spectral features capture

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

• تاریخ ارسال مقاله: ۱۴۰۳/۱۱/۱۳ • تاریخ پذیرش: ۱۴۰۴/۱۱/۱۴ • تاریخ انتشار: ۱۴۰۴/۱۲/۲۹ • نوع مطالعه: پژوهشی

unique patterns embedded in the bytecode, revealing underlying structures that distinguish malicious applications from benign ones. The extracted features are then formatted as three-channel spectrogram-like images, forming the input to a deep learning model. By leveraging transfer learning on these spectral representations, the model can efficiently classify malware while remaining robust against code obfuscation and adversarial attacks.

To optimize malware classification, this research fine-tunes EfficientNetV2B0, a state-of-the-art convolutional neural network, using transfer learning. The pre-trained model, originally designed for large-scale image classification, is adapted to process spectral representations of APK files. The final layers are retrained on the malware dataset while earlier layers retain their learned feature extraction capabilities. This approach enables efficient learning with limited data while improving generalization, ensuring high detection accuracy even on previously unseen malware samples.

The proposed model achieves an impressive 99.3% accuracy and 99.1% recall, demonstrating its effectiveness in detecting malware, including obfuscated and evasive variants. By preserving the sequential structure of APK bytecode in an audio format, this method enhances detection robustness without requiring complex reverse engineering techniques. The model's performance was evaluated using a diverse dataset of over 11,000 applications, including obfuscated malware samples designed to bypass traditional security measures. The results highlight the effectiveness of the audio-based approach in distinguishing malicious patterns, achieving a detection rate significantly higher than conventional static analysis techniques. Unlike traditional machine learning models that require extensive feature engineering, this black-box method leverages deep learning to automatically extract relevant patterns from spectral features, improving both efficiency and scalability. The high accuracy and recall scores achieved in this study mark a significant advancement in Android malware detection, proving that audio-based representations can effectively capture the structural and behavioral properties of malicious software. This research not only provides a novel direction for malware classification but also introduces a scalable and adaptable framework capable of addressing modern threats with unprecedented accuracy and resilience.

Keywords: Android Malware Detection, Deep Learning, Static Analysis, Image Processing, Audio Signal Processing.

۱-مقدمه

با افزایش تولید و انتشار بدافزارهای اندرویدی، نگرانی‌ها دربارهٔ امنیت کاربران این اکوسیستم به شکل چشم‌گیری افزایش یافته است. گزارش اخیر McAfee سال ۲۰۲۲ را «سال حملات مخفیانهٔ تلفن همراه» معرفی کرده‌است؛ ماهیت متن‌باز سیستم عامل اندروید، نصب آسان از منابع ناشناس و نبود نظارت کامل بر فروشگاه‌های برنامه، این سکو را به هدفی ایدئال برای مهاجمان سایبری تبدیل کرده‌است [۱].

بدافزارهای اندرویدی با استفاده از روش‌هایی مانند بسته‌بندی مجدد، ارتقای حمله و تغییر ظاهر برنامه، سعی در دورزدن سامانه‌های امنیتی دارند. این بدافزارها با اهدافی نظیر سرقت اطلاعات، شنود، اجرای مخفی فرامین و ایجاد اختلال در عملکرد سامانه طراحی می‌شوند و در قالب انواعی مانند ویروس‌ها، تروجان‌ها، نرم‌افزارهای جاسوسی و کرم‌ها ظاهر می‌شوند [۲, ۵].

آمارها[1] نشان می‌دهند که حدود هر هفت‌ثانیه یک بدافزار جدید منتشر می‌شود. طبق گزارش‌های McAfee و Kaspersky، کشورهایی مانند ایران، الجزایر و نیجریه جزو آلوده‌ترین کشورها هستند؛ برای مثال، ۳۵ درصد از تلفن‌های همراه ایرانی، بیشترین میزان آلودگی به بدافزار را در بین تلفن‌های همراه کاربران دیگر کشورها دارند[2].

همچنین، در سال ۲۰۲۳ بیش از ششصد میلیون بار بدافزارها از طریق گوگل‌پلی بارگیری شده‌اند[3]. این آمارها نشان‌دهندهٔ نیاز فوری به روش‌های کارآمد برای شناسایی و مقابله با بدافزارها است.

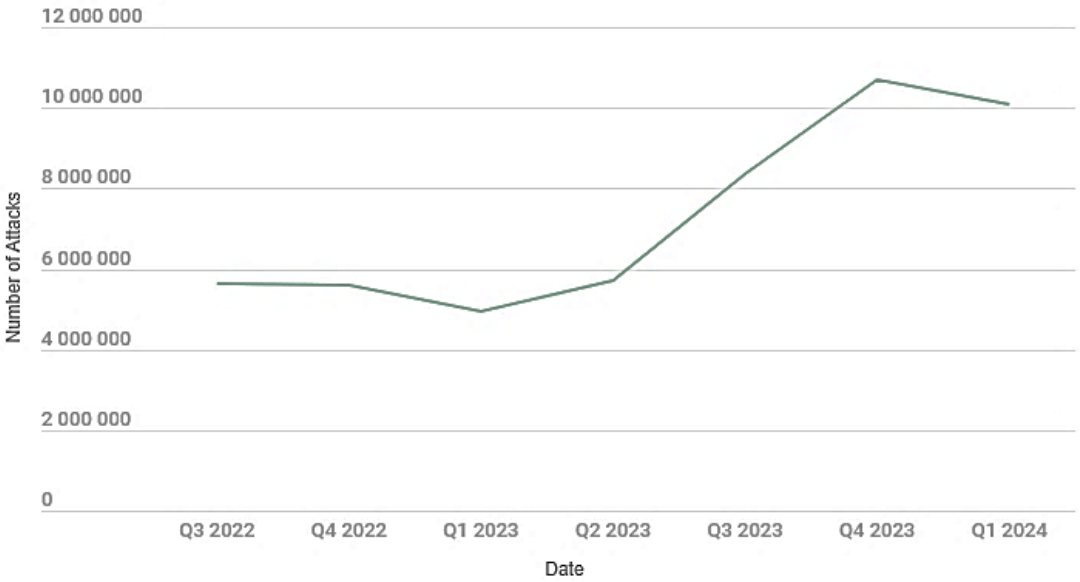
در سال‌های اخیر، روش‌های تشخیص مبتنی‌بر امضا به‌دلیل ناتوانی در شناسایی بدافزارهای جدید یا مبهم‌سازی‌شده، ناکارآمد شده‌اند [۶]. در پاسخ به این چالش، رویکردهای نوین مبتنی‌بر یادگیری عمیق و تحلیل‌های غیرسنتی مانند تبدیل کدهای اجرایی به سیگنال‌های صوتی یا تصاویر، محبوبیت یافته‌اند. این روش‌ها بدون نیاز به مهندسی معکوس و تجزیهٔ کد قادرند الگوهای پنهان را از طریق ویژگی‌های استخراج‌شده شناسایی کنند.

انگیزهٔ اصلی این پژوهش، ارائهٔ راه‌کاری نوآورانه برای تشخیص بدافزارهای اندرویدی با استفاده از یادگیری عمیق و بهره‌گیری از اطلاعات ساختاری بایت‌کدها بدون نیاز به تجزیهٔ برنامه‌ها است؛ بدین منظور، بایت‌کد خام برنامه‌ها به سیگنال‌های صوتی تبدیل و سه نوع طیف MFCC، BFCC و GFCC از آن استخراج شده‌اند. این طیف‌ها به‌صورت تصویر رنگی سه‌کاناله تبدیل شده‌اند تا با کمک یادگیری انتقالی، مدل ازپیش آموزش‌دیدهٔ EfficientNetV2B0 آموزش ببیند تا بتواند طبقه‌بندی نهایی را انجام دهد. از مزایای مهم این روش می‌توان به مقاومت بالا در برابر مبهم‌سازی کد، کاهش نیاز به مهندسی معکوس، کارایی بالا در محیط‌های واقعی و صنعتی و قابلیت اجرا

[1] https://www.gdatasoftware.com/news/g-data-mobile-malware-report-2019-new-high-for-malicious-android-apps
[2] https://financialtribune.com/articles/sci-tech/114469/phone-malware-widespread-in-iran
[3] https://www.kaspersky.com/blog/malware-in-google-play-2023/49579

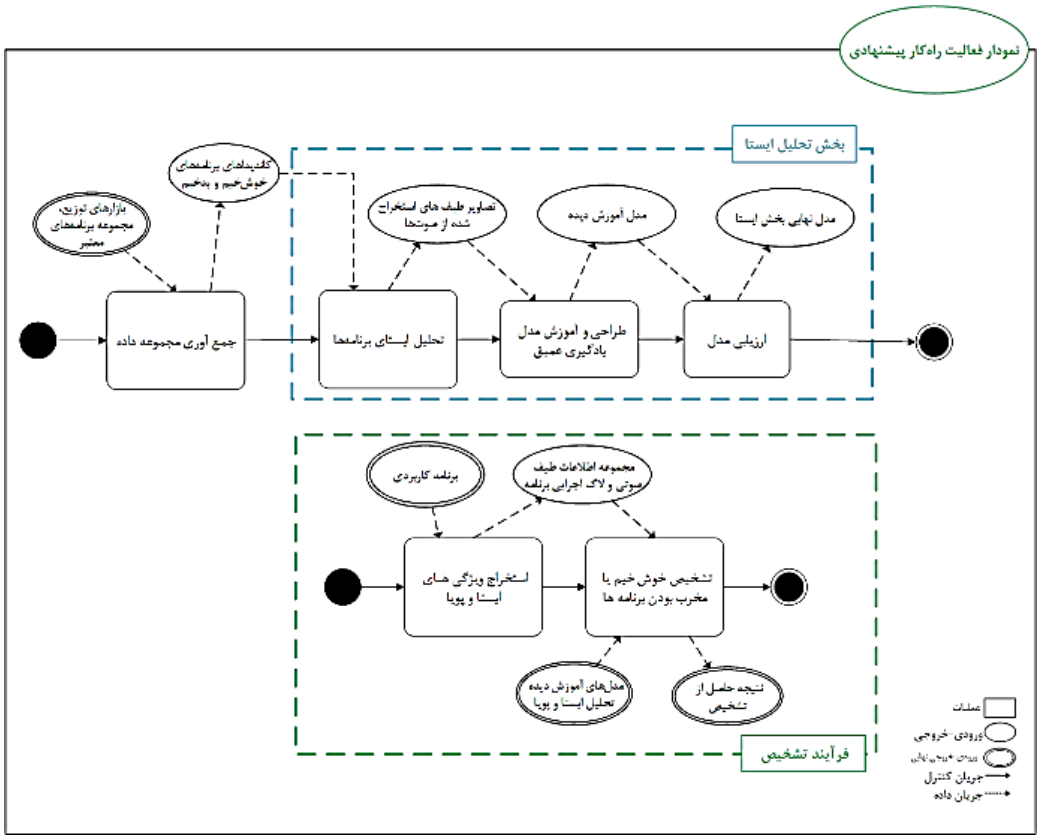
در ابزارهای امنیتی قابل حمل اشاره کرد. این ویژگی‌ها، روش پیشنهادی را به گزینه‌ای مناسب برای استفاده در بازارهای اندرویدی داخلی و محیط‌های توسعهٔ امن نرم‌افزار تبدیل می‌کند.

در ادامه و در بخش دو، به پیشینهٔ پژوهش و بخش سه به توضیح روش پیشنهادی مورد استفاده پرداخته می‌شود. در بخش چهار و پنج به‌ترتیب ارزیابی روش ارائه شده و نتیجه‌گیری نیز مورد بررسی قرار گرفته است.



(شکل-۱): نمودار حملات سایبری با کمک بدافزارهای موبایلی[1]

(Figure-1): Chart of cyberattacks carried out with the help of mobile malware



(شکل-۱): نمودار فعالیت راه‌کار پیشنهادی

(Figure-2): Activity diagram of the proposed solution

[1] https://securelist.com/it-threat-evolution-q1-2024-mobile-statistics/112750/

۲- پیشینهٔ پژوهش

در بررسی و تحلیل برنامه‌های اندرویدی، سه رویکرد اصلی وجود دارد: تحلیل ایستا، تحلیل پویا و تحلیل ترکیبی. در تحلیل ایستا، کد برنامه‌ها بدون اجرای آن‌ها بررسی می‌شود. این روش به‌دلیل امکان دسترسی مستقیم به کد منبع یا محتویات باینری فایل برنامه‌ها و استخراج ویژگی‌هایی مانند رشته‌ها، دسترسی‌ها و برخی توابع، بسیار سریع است؛ بااین‌حال، تحلیل ایستا در مواجهه با روش‌های مبهم‌سازی[1]، مانند تغییر نام متغیرها، کدگذاری رشته‌ها و درج کدهای بی‌استفاده دچار چالش می‌شود؛ از سوی دیگر، تحلیل پویا با اجرای برنامه در محیط کنترل‌شده، رفتار برنامه را در زمان اجرا بررسی می‌کند. این روش قادر است، فعالیت‌های مخرب پنهان‌شده در زمان اجرا، مانند دسترسی غیرمجاز به داده‌ها یا ارتباطات مشکوک شبکه‌ای را شناسایی کند؛ بااین‌حال، تحلیل پویا زمان‌بر بوده و نیازمند منابع بیشتری است. تحلیل ترکیبی با ترکیب مزایای دو روش پیشین، رویکرد جامعی برای بررسی برنامه‌ها ارائه می‌دهد.

یکی از چالش‌های اساسی در این حوزه، استفادهٔ گسترده از روش‌های مبهم‌سازی توسط طراحان بدافزار است که فرایند تشخیص را پیچیده‌تر می‌کند. این روش‌ها به بدافزارها کمک می‌کنند تا از شناسایی به‌وسیلهٔ ابزارهای مبتنی‌بر امضا، جلوگیری کنند؛ درنتیجه، روش‌های سنتی مبتنی‌بر امضا که بر شناسایی الگوهای ثابت در کدها متکی هستند، دیگر پاسخ‌گوی پیچیدگی بدافزارهای جدید نیستند. در سال‌های اخیر، روش‌های مبتنی‌بر یادگیری ماشین و یادگیری عمیق به‌دلیل توانایی بالای آن‌ها در شناسایی الگوهای پنهان و تحلیل داده‌های پیچیده، جایگزین روش‌های سنتی شده‌اند. این رویکردها با استخراج خودکار ویژگی‌ها و ارائهٔ مدل‌های انعطاف‌پذیرتر، نقش مهمی در بهبود دقت و سرعت تشخیص بدافزارها ایفا می‌کنند.

کانفورا و همکاران در [۷] روشی برای شناسایی بدافزارهای اندرویدی ارائه کردند که مبتنی‌بر محاسبهٔ بافت‌نگاشت فرکانس دستورهای اجرایی کد[2] و تحلیل آن‌ها است. در این روش، با بررسی فراوانی دستورهای عملیاتی خاص استخراج‌شده از کدهای Dalvik، تشخیص بدافزار انجام می‌شود. این دستورات بر حسب معمول برای تغییر جریان کنترل برنامه استفاده می‌شوند. روش پیشنهادی روی مجموعه‌دادهٔ Drebin و با استفاده از چندین طبقه‌بند مبتنی‌بر ساختار درخت ارزیابی شده و به دقت ۹۵ درصد دست یافته است.

الزیلایی و همکاران در [۸]، روش DL-Droid را معرفی کردند که یک رویکرد مبتنی‌بر یادگیری عمیق برای شناسایی بدافزارهای اندرویدی با استفاده از تحلیل پویا و تولید ورودی حالت است. در این پژوهش، بیش از ۳۱۰۰۰ برنامه جمع‌آوری شد که بیش از یازده‌هزار مورد آن‌ها بدافزار بودند. DL-Droid روی یک سکوی خودکار اجرا می‌شود که قادر به انجام تحلیل ایستا و پویا است. برای ارزیابی، از یک دستگاه اندروید واقعی استفاده شد که زمان تحلیل هر برنامه حدود ۱۹۰ ثانیه است. این روش با استفاده از ویژگی‌های تحلیل پویا به نرخ تشخیص ۹۷.۸ درصد و با افزودن ویژگی‌های تحلیل ایستا به ۹۹.۶ درصد دست یافت.

قاسمی و همکاران در پژوهش [۳۳]، ویژگی‌های برنامه‌های اندروید را هنگام نصب و اجرا از سمت تلفن همراه استخراج و برای پردازش و تحلیل به سرورهای ابری ارسال کردند؛ سپس با کمک الگوریتم‌های پایهٔ یادگیری ماشین، تشخیص بدافزار را انجام می‌دهند. این روش به‌دلیل استفاده از محاسبات ابری، سرعت بالا، دقت مناسب و مصرف منابع کمتر را درپی دارد که بر روی مجموعه‌دادهٔ Drebin، به‌دقت ۹۶.۴۴درصد دست یافتند.

دی پیر و همکاران در پژوهش [۳۴]، روشی برای ارزیابی ریسک امنیتی برنامه‌های موبایلی ارائه کرده‌اند که بر مبنای تحلیل مجوزهای درخواستی به‌وسیلهٔ برنامه‌ها عمل می‌کند. آن‌ها با بررسی مجوزهای ده‌ها بدافزار و صدها برنامهٔ قانونی، معیاری جدید برای سنجش ریسک امنیتی طراحی کردند و یک ابزار نرم‌افزاری برای پیاده‌سازی آن توسعه داده‌اند.

پژوهش‌گران در پژوهش [۹] با استخراج گراف فراخوانی‌های APIهای سامانه‌ای و استفاده از معماری مبتنی‌بر شبکهٔ CNN، تشخیص بدافزار را انجام داده‌اند. دقت تشخیص بدافزار در این پژوهش ۹۹ درصد و برای بدافزارهای جدید ۹۱ درصد گزارش شد.

در پژوهش [۱۰]، *شی-شیائو و همکاران* با استفاده از توالی فراخوانی‌های سامانه‌ای در حین اجرای برنامه، دو شبکهٔ حافظهٔ کوتاه‌مدت ماندگار[3] را آموزش دادند؛ یکی با برنامه‌های خوش‌خیم و دیگری با بدافزارها. تشخیص بدافزار با مقایسهٔ امتیازهای این دو شبکه و به‌وسیلهٔ یک طبقه‌بند انجام می‌شود. این پژوهش روی مجموعه‌دادهٔ Derbin و حدود هفت‌هزار برنامه ارزیابی شد و به دقت ۹۶ درصد دست یافت.

در [۱۱]، روش DeepAMD معرفی شد که ترکیبی از تحلیل ایستا و پویا است. در تحلیل ایستا، دقت تشخیص بدافزار ۹۳ درصد، تشخیص دسته بدافزار ۹۲ درصد و تشخیص خانواده بدافزار نود درصد بود. در تحلیل پویا، دقت تشخیص دسته بدافزار هشتاد درصد و خانواده بدافزار ۵۹ درصد بود. این پژوهش از مجموعه‌دادهٔ CICInvesAndMal2019 استفاده کرد و از طبقه‌بندهایی مانند بیز ساده[4]، شبکه عصبی چندلایه[5] و درخت تصمیم[6] بهره برد.

[1] Obfuscation
[2] Opcode

[3] Long Short-Term Memory (LSTM)
[4] Naive Bayes
[5] Multi-Layer Perceptron
[6] Decision Tree

طهرانی و همکاران در [۱۲] با استفاده از الگوریتم‌های بردار پشتیبان ماشین[1] و نزدیک‌ترین همسایه[2]، روشی برای تشخیص بدافزار ارائه کردند که بر بهبود دقت و کاهش نرخ خطا متمرکز است؛ این پژوهش همچنین به جمع‌آوری و متعادل‌سازی مجموعه‌داده پرداخت. با استفاده از ویژگی‌هایی مانند فهرست دسترسی‌ها و سرویس‌ها، دقت نهایی ۹۹ درصد گزارش شد.

لیمین شن در [۱۳] از شبکهٔ حافظهٔ کوتاه‌مدت ماندگار پیچشی مبتنی‌بر ویژگی خودآگاهی[3] برای تشخیص بدافزارهای اندرویدی استفاده کرد. در این پژوهش، داده‌های شبکه‌ای برنامه‌ها به تصاویر سطوح خاکستری تبدیل شدند و به‌عنوان ورودی شبکهٔ عصبی عمیق استفاده شدند. دقت تشخیص بدافزار ۹۵ درصد و تخمین خانوادهٔ بدافزار ۷۲ درصد بود.

باکور در [۱۴] روش DeepVisDroid را معرفی کرد که با تبدیل سورس‌کد نرم‌افزار کاربردی به تصاویر سطوح خاکستری[4]، چهار مجموعه‌دادهٔ تصویری تولید کرده و با کمک الگوریتم‌های SIFT، SURF، ORB و KAZE استخراج ویژگی‌ها انجام شده‌است. این روش روی مجموعه‌داده‌های Derbin و MalGenome ارزیابی شد و به دقت ۹۸ درصد دست یافت.

یوژین دینگ و همکاران در [۱۵] با تبدیل بایت‌کدهای فایل APK به تصاویر سطوح خاکستری، تشخیص بدافزار را انجام دادند. این پژوهش از مجموعه‌دادهٔ Derbin استفاده کرد و با معماری CNN به دقت ۹۴ درصد رسید؛ همچنین *دنیش وسن و همکاران* در [۱۷] از تصاویر رنگی RGB برای تشخیص بدافزار استفاده کردند. در روش IMCFN، از تنظیم مجدد[5] شبکه‌های معروفی مانند InceptionNet استفاده شد. این پژوهش روی مجموعه‌داده‌هایIoT-android mobile و Malimg malware ارزیابی شد. در [۱۸] نیز با تبدیل بایت‌کدها به تصاویر خاکستری و ارزیابی پنجاه‌هزار برنامه، دقت ۹۴ درصد به‌دست آمد. در پژوهش دیگری *پوجایاداو و همکاران* در [۱۹] با تبدیل فایل‌های دودویی به تصاویر رنگی، روشی برای تشخیص بدافزار ارائه کردند. این پژوهش از معماری EfficientNet استفاده کرد و به دقت ۹۵ درصد دست یافت.

فرخ‌منش و همکاران در [۲۱] با استفاده از روش‌های استخراج اطلاعات از موسیقی[6]، فایل‌های APK را به فایل‌های MIDIو سپس WAV تبدیل کردند و ویژگی‌های متنوعی را که از یک فایل صوتی قابل دریافت است، استخراج کردند؛ همچنین *فرانچسکو مرکالدو و همکاران* در [۲۲] با تبدیل بایت‌کدهای فایل Classes.dex به فایل‌های صوتی WAV، بررسی کردند که آیا بدافزارهای مشابه نمونه‌های صوتی مشابهی دارند یا خیر. این پژوهش از ویژگی‌هایی مانند کروماگرام و MFCC استفاده کرد و به دقت ۹۴ درصد رسید. در پژوهش دیگری *کاسولار و همکاران* در [۲۳] با تبدیل بایت‌کدها به فایل‌های صوتی، بر تشخیص خانوادهٔ بدافزار متمرکز شدند. این پژوهش از ویژگی‌هایی مانند نرخ عبور از صفر[7] و MFCC استفاده کرد و به دقت ۹۸ درصد دست یافت.

پژوهش‌گران در [۲۴] با ترکیب پردازش تصویر و صوت، بایت‌کدهای APK را به فایل‌های صوتی WAV تبدیل کردند و با استفاده از شبکه‌های CNN به دقت ۹۶ درصد در تشخیص بدافزار و نود درصد در تشخیص خانوادهٔ بدافزار رسیدند. *پاول و همکاران* در [۲۵] با استخراج بیش از صد ویژگی از تبدیل بایت‌کدهای APK به WAV، روشی برای تشخیص بدافزار ارائه کردند. این پژوهش روی مجموعه‌دادهٔ CICMalDroid 2020 انجام شد و به دقت ۹۸.۹ درصد دست یافت.

با بررسی پژوهش‌های مرتبط مشخص می‌شود که تاکنون روش‌های متنوعی برای نگاشت بایت‌کدهای برنامه‌های اندرویدی به حوزه‌های تصویری یا صوتی پیشنهاد شده‌اند. برخی از این روش‌ها تنها از یک نوع طیف صوتی مانند MFCC یا ویژگی‌های تصویری خاکستری استفاده کرده‌اند که اطلاعات ساختاری محدودی ارائه می‌دهند؛ همچنین بسیاری از این پژوهش‌ها نیازمند طراحی و آموزش مدل‌های اختصاصی از ابتدا بوده‌اند که فرآیند یادگیری را زمان‌بر و وابسته به منابع محاسباتی بالا کرده‌است.

در این میان، شکاف مهمی در استفادهٔ ترکیبی از چندین طیف از داده‌های صوتی و استفاده از آن‌ها به‌صورت تصویر رنگی سه‌کاناله وجود دارد؛ همچنین، استفاده از مدل‌های پیش‌آموزش‌دیدهٔ بهینه مانند EfficientNetV2B0، که علاوه‌بر سبک‌بودن، دقت بالایی نیز دارند و روش‌هایی مانند یادگیری انتقالی برای آموزش مدل‌ها، کمتر مورد توجه قرار گرفته است؛

بنابراین نوآوری اصلی پژوهش حاضر را می‌توان در سه بُعد تشریح کرد:

۱. ترکیب سه طیف صوتی MFCC، BFCC وGFCC برای ایجاد تصویری غنی‌تر و سه‌کاناله از برنامه‌ها؛
۲. استفاده از این تصویر رنگی به‌عنوان ورودی مدل ازپیش آموزش‌دیده EfficientNetV2B0 در قالب یادگیری انتقالی؛
۳. ارزیابی دقیق مدل روی داده‌های واقعی شامل نمونه‌های مبهم‌سازی‌شده، بدون نیاز به استخراج دستی ویژگی یا مهندسی معکوس.

این ترکیب از نگاشت بایت‌کد به حوزهٔ صوت، ایجاد تصویر رنگی و استفاده از یادگیری انتقالی روی مدلی سبک و بهینه، تاکنون در پژوهش‌های موجود به‌صورت یک‌پارچه انجام نشده و پژوهش حاضر با هدف پرکردن این شکاف و ارائهٔ روشی دقیق، سریع و مقاوم در برابر مبهم‌سازی طراحی شده‌است.

[1] Support Vector Machine (SVM)
[2] K-Nearest Neighbors (KNN)
[3] Self-Attention
[4] Grayscale
[5] Fine-tune
[6] Music Information Retrival (MIR)
[7] Zero-Crrosing Rate (ZCR)

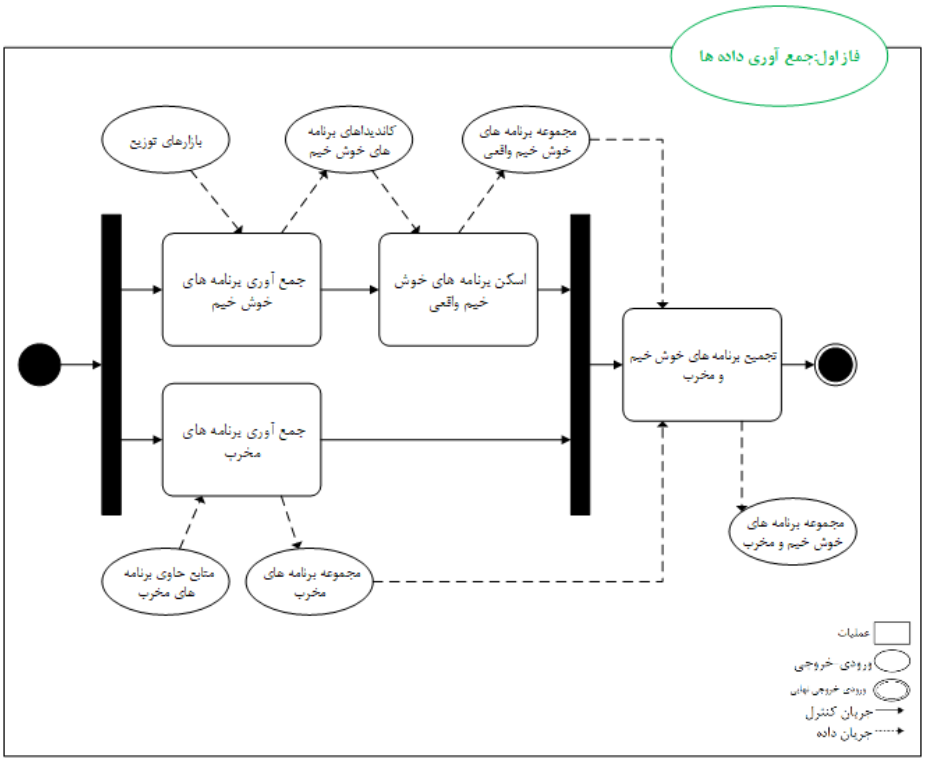
۳- روش پیشنهادی

پژوهش حاضر با استفاده از رویکرد تحلیل ایستا برای تشخیص بدافزارها انجام شده‌است. در تحلیل ایستا، داده‌های خام بایت‌کدها به حوزهٔ صوت نگاشت می‌شوند و سپس ویژگی‌های مورد نیاز ازجمله طیف‌های موردنظر استخراج می‌شوند. این طیف‌های دوبعدی همانند تصاویر عمل می‌کنند. این تصاویر با استفاده از تنظیم مجدد مدل‌های ازپیش آموزش‌دیده و با کمک یادگیری انتقالی، طبقه‌بندی می‌شوند. این روش به‌دلیل نیازنداشتن به مهندسی معکوس برنامه‌ها و داشتن دیدگاه جعبه سیاه‌گونه، دقت و کارایی بالایی در تشخیص بدافزارها ارائه می‌دهد.

راه‌کار پیشنهادی شامل چهار مرحلهٔ اصلی است: (۱) آماده‌سازی داده‌ها، (۲) انجام تحلیل ایستا، (۳) طراحی، آموزش و آزمایش مدل و (۴) ارزیابی نهایی. در کل، این روش با استفاده از داده‌های صوتی در تحلیل ایستا، دقت تشخیص را افزایش داده و چالش‌های ناشی از روش‌های مبهم‌سازی در فرایند مهندسی معکوس را کاهش می‌دهد.

۳-۱-فاز نخست: جمع‌آوری داده‌ها

فاز نخست مربوط به جمع‌آوری داده‌ها است. یک رویکرد مبتنی‌بر یادگیری عمیق، در وهلهٔ نخست نیاز به حجم زیادی از داده‌های ورودی دارد. مدلی با معماری شبکهٔ عصبی عمیق که شامل میلیون‌ها مؤلفه برای آموزش است؛ درصورتی می‌تواند به‌خوبی آموزش ببیند که با حجم زیادی از داده‌های مناسب، تغذیه شود؛ به همین دلیل در فاز نخست، نیاز به جمع‌آوری مجموعه‌ای بزرگ و به اندازهٔ کافی متنوع از داده‌ها است. نمودار شکل (۳)، فرایند جمع‌آوری داده‌های مورد نیاز را نشان می‌دهد.



(شکل-۲): نمودار فعالیت فاز نخست، جمع‌آوری داده‌ها

(Figure-3): Activity diagram for the first phase, data collection

در این پژوهش، داده‌های موردنیاز شامل فایل‌های APK برنامه‌های اندرویدی است که برای افزایش دقت و جامعیت داده‌ها، از مجموعه‌داده‌های معتبر CIC استفاده شد. این مجموعه‌داده‌ها که بیش از دویست گیگابایت حجم دارند و شامل بیش از نوزده‌هزار برنامه است، اطلاعات ایستا و پویای کاملی از بدافزارهای اندرویدی با تنوع بالا ارائه می‌دهند. دسترسی آسان و قابلیت مقایسهٔ نتایج پژوهش با سایر مطالعات مشابه از مزایای این مجموعه‌داده‌ها محسوب می‌شود. این مجموعه‌داده، پس از حذف فایل‌های ناقص حدود ۱۳۰ گیگابایت حجم دارد که جزئیات تعداد برنامه‌ها در جدول (۱) ارائه شده‌است.

(جدول-۱): تعداد داده‌ها در بخش تحلیل ایستا

(Table-1): Number of data in the static analysis section

	تعداد برنامه‌ها
خوش‌خیم	۵۶۸۷
بدخیم	۵۶۸۷
مجموع	**۱۱۳۷۴**

از آن‌جا که یکی از اهداف مدنظر ما در این پژوهش، بررسی عملکرد نهایی مدل بر روی برنامه‌های مبهم‌شده بود، این

مجموعه‌داده از جهت دیگر که وجود برنامه‌های مبهم‌شدهٔ واقعی در هر دو دسته برنامه‌های خوش‌خیم و بدخیم است، برای ما اهمیت دارد. با کمک ابزار ApkId که برای اهداف مختلفی در تحلیل فایل‌های APK استفاده می‌شود، مجموعه‌دادهٔ حاضر مورد بررسی قرار گرفت. در جدول (۲)، تعداد برنامه‌های مبهم‌شدهٔ موجود در مجموعه‌دادهٔ مورد استفاده ارائه شده‌است.

(جدول-۲): تعداد برنامه‌های مبهم‌شده
(Table-2): Number of obfuscated applications

	تعداد برنامه‌های مبهم
خوش‌خیم	۳۲۲
بدخیم	۵۷۰
مجموع	**۸۹۲**

با توجه به جدول (۲)، حدود ۷.۸ درصد از کل مجموعه‌داده‌ها را برنامه‌های مبهم‌شده شامل می‌شود که این مورد بسیار مناسبی برای استفاده در پژوهش جاری است. نگاه جعبه‌سیاه‌گونه روش مورد استفاده در راه‌کار پیشنهادی ما که توضیحات آن در بخش‌های آینده ارائه شده‌است، در تشخیص برنامه‌های مبهم‌شده نیز بسیار مناسب عمل کرده‌است؛ از طرف دیگر، وجود این تعداد برنامهٔ واقعی که از سناریوهای واقعی مبهم‌سازی استفاده می‌کنند، می‌تواند نتایج ارزیابی راه‌کار پیشنهادی را بسیار ارزشمندتر کند؛ چراکه با کمک داده‌های مبهم‌شدهٔ واقعی و نه فرایندهای تغییر بسته و یا مبهم‌سازی مجدد، انجام شده‌است.

مجموعه‌دادهٔ مورد استفاده در این پژوهش، بر اساس روش ارزیابی بیرون نگهدارنده، به سه قسمت برای آموزش، ارزیابی و آزمون درنظر گرفته می‌شود. برای این منظور، هشتاد درصد از داده‌ها به‌عنوان دادهٔ آموزشی، ده درصد از داده‌ها به‌عنوان دادهٔ ارزیابی و ده درصد نهایی نیز به‌عنوان دادهٔ آزمون درنظر گرفته می‌شود. داده‌های آموزشی در فرایند آموزش به مدل وارد می‌شوند. از داده‌های ارزیابی در طی فرایند آموزش و برای تنظیم بهینهٔ وزن‌ها و مؤلفه‌های مدل استفاده می‌شود. از داده‌های آزمون نیز برای ارزیابی نهایی مدل آموزش‌دیده استفاده می‌شود.

۲-۳-فاز دوم: تحلیل ایستای برنامه‌ها

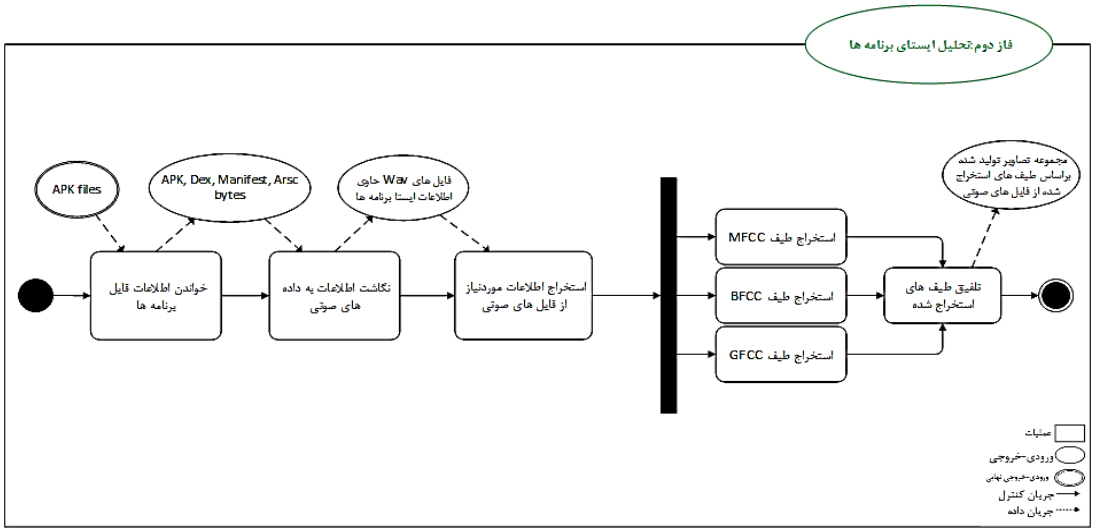
این فاز شامل انجام فرایندهای تحلیلی بر روی برنامه‌های جمع‌آوری‌شده در فاز نخست است. در این فاز برای استخراج ویژگی‌ها و اطلاعات مورد نیاز از هر برنامه، از تحلیل ایستا استفاده شد. شکل (۴)، نمودار فعالیت تحلیل ایستای برنامه‌ها را نشان می‌دهد.

این فاز شامل انجام تحلیل ایستای برنامه‌ها است؛ بنابر پژوهش‌های بررسی‌شده، استفاده از اطلاعات موجود در بایت‌کدهای خام فایل‌هایی مانند فایل APK و فایل Classes.dex، بسیار مؤثر خواهد بود. بایت‌کدهای یک فایل اجرایی درواقع یک نوع بازنمایی از رفتار برنامه است [۲۴, ۲۵]؛ بنابراین با تحلیل درست بایت‌کدهای فایل اجرایی برنامه‌ها، می‌توان دید خوبی از خوش‌خیم‌بودن یا نبودن آن به‌دست آورد.

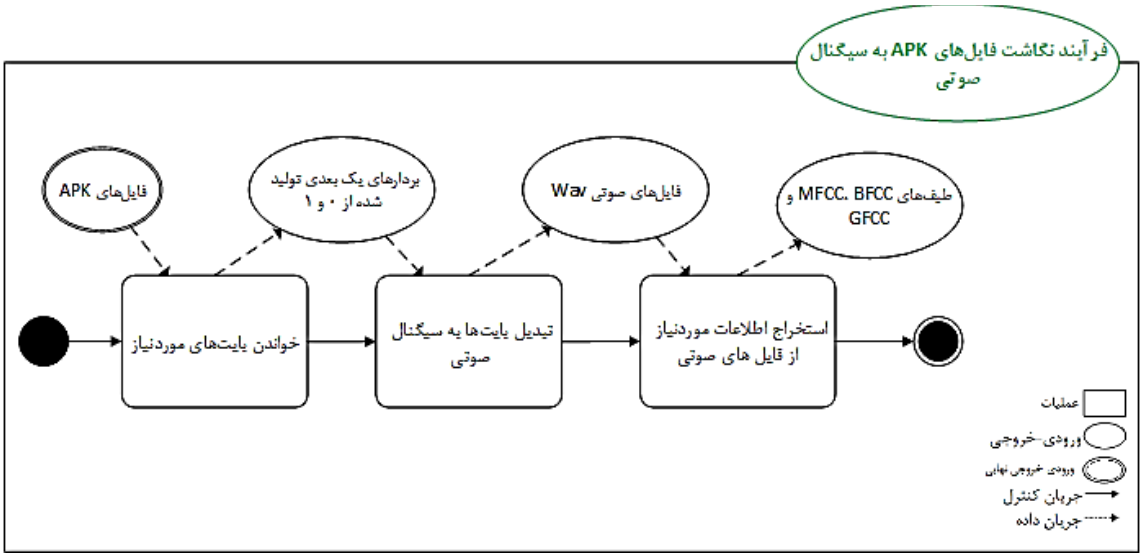
در پژوهش‌هایی مانند [۱۳, ۱۴, ۲۴]، با استفاده از یک نگاشت بین مقدار بایت‌کدها و پیکسل‌های تصویر، یک بازنمایی تصویری از فایل‌های اجرایی به‌دست می‌آورند؛ از آنجا که بایت‌کدهای یک فایل اجرایی به‌صورت پشت سرهم معنا دارند، تبدیل فهرستی از داده‌های پشت سرهم به ساختار داده‌ای مانند تصویر که دارای سطرها و ستون‌های مختلف است، باعث می‌شود اطلاعات نهفته در توالی این داده‌ها به‌هم ریخته و در گام‌های بعدی، خطاهای احتمالی را به‌وجود آورد.

یک ساختار دادهٔ بسیار عالی و قابل تحلیل که از پشت سرهم قرارگرفتن داده‌های مختلف ایجاد می‌شود، فایل‌های صوتی است. سیگنال‌های صوتی با داشتن داده‌های مختلف در زمان‌های مختلف، می‌توانند بهترین بازنمایی برای بایت‌کدهای خام باشند؛ بنابراین در این مرحله نیاز است تا برنامه‌ها به فایل‌های صوتی مختلف، نگاشت شوند.

برای انجام این تبدیل، یک فرایند نگاشت سه مرحله‌ای درنظر گرفته شده‌است که در شکل (۵) نمودار فعالیت آن ارائه شده‌است.



(شکل-۳): نمودار فعالیت فاز دوم از راه‌کار پیشنهادی
(Figure-4): Activity diagram for the second phase



(شکل-۴): نمودار فعالیت فرایند نگاشت فایل‌های APK به سیگنال‌های صوتی

(Figure-5): Activity diagram of the process of mapping APK files to audio signals

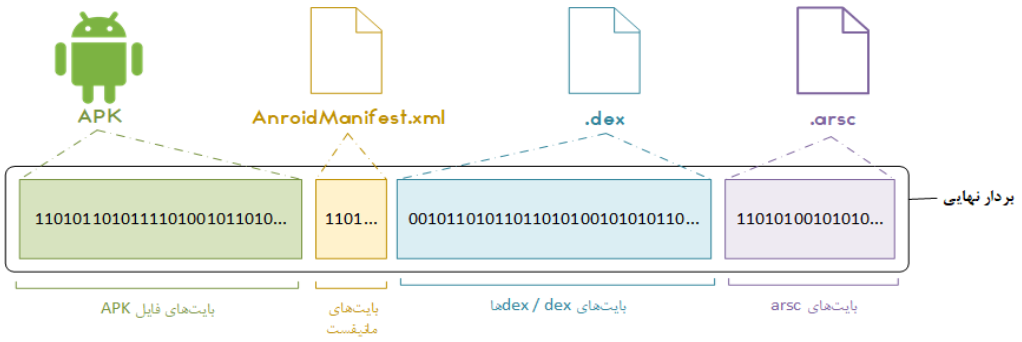
با توجه به شکل (۵)، فرایند سه مرحله‌ای نگاشت با خواندن بایت‌های مورد نیاز از فایل‌ها آغاز می‌شود. در ادامه بایت‌های خوانده‌شده با یک سازوکار خاص به سیگنال‌های صوتی تبدیل شده و درانتها، اطلاعات مورد نیاز از این سیگنال‌های صوتی استخراج می‌شود. در ادامه توضیحات این مراحل ارائه شده‌است.

مرحلهٔ نخست: استخراج بایت‌های فایل APK

فایل‌های APK که به‌عنوان فایل‌های فشردهٔ حاوی کدهای اجرایی، فایل‌های Manifest و منابع دیگر هستند، از حالت فشرده خارج شده و بایت‌های مربوط به فایل apk، Classes.dex، AndoridManifest.xml و resources.arsc پشت سرهم قرار می‌گیرند. خروجی این مرحله یک بردار یک‌بعدی از بایت‌ها با مقادیر عددی بین ۰ تا ۲۵۵ است که هر بایت با هشت بیت نمایش داده می‌شود که در شکل (۶)، شمای کلی از این فرایند ارائه شده‌است.

مرحلهٔ دوم: تبدیل بایت‌ها به سیگنال صوتی

بردار عددی به‌دست‌آمده از مرحلهٔ پیشین باید به یک سیگنال صوتی مناسب تبدیل شود که برای این منظور ابتدا مشخصات سیگنال تعیین می‌شود؛ از آنجا که سیگنال صوتی نهایی تنها برای آموزش مدل یادگیری عمیق استفاده می‌شود و نیازی به شنوندهٔ انسانی ندارد، افزایش نرخ دادهٔ آن تنها باعث افزایش حجم داده‌ها خواهد شد [۲۵]؛ همچنین، استفاده از یک کانال صوتی به‌جای دو کانال به کاهش حجم کمک می‌کند. نرخ نمونه‌برداری سیگنال، که تعداد نمونه‌ها در هر ثانیه را مشخص می‌کند، در این پژوهش شانزده کیلوهرتز درنظر گرفته شده‌است [۲۵] که علاوه‌بر حفظ داده‌های اصلی، نیاز به پهنای باند کمتری دارد. عمق بیت سیگنال نیز شانزده بیت تعیین شده‌است که امکان پشتیبانی از ۶۵۵۳۶ مقدار مختلف را فراهم کرده و بازنمایی دقیق‌تری ارائه می‌دهد؛ درنهایت، سیگنال به‌صورت غیرفشرده و خام با فرمت WAV و نوع uPCM ذخیره می‌شود؛ زیرا استفاده از فشرده‌سازی‌هایی مانند MP3 ممکن است داده‌های مرحله پیشین را تغییر دهد که مطلوب ما نیست.



(شکل-۵): کنار هم قرارگرفتن بایت‌های فایلAPK، Manifest، dex و arsc و تولید بردار نهایی

(Figure-6): Putting together the bytes of the APK, Manifest, dex, and arsc files and generating the final vector

سیگنال صوتی نهایی با نرخ نمونه‌برداری شانزده کیلوهرتز، عمق بیت برابر با شانزده بیت و نوع فشرده‌سازی PCM خام در قالب صوت تک‌کاناله و با فرمت WAV ایجاد خواهد شد؛ بنابراین در این مرحله، هر شانزده بیت از بردار ورودی به این مرحله،

به‌عنوان یک نمونهٔ صوتی درنظر گرفته می‌شود و هر شانزده‌هزار نمونه در یک ثانیه از سیگنال صوتی نهایی قرار خواهند گرفت. این فرایند با کمک بستهٔ نرم‌افزاری رسمی Wav در Python انجام می‌شود.

مرحلهٔ سوم: استخراج ویژگی‌های سیگنال صوتی

سیگنال‌های صوتی به‌دست‌آمده برای استخراج ویژگی‌های دوبعدی مانند ضرایب Cepstral پردازش می‌شوند. این ویژگی‌ها، سیگنال را از حوزهٔ زمان-دامنه به حوزهٔ زمان-فرکانس منتقل می‌کنند و به‌عنوان تصاویر دوبعدی برای استفاده در مدل‌های یادگیری عمیق ذخیره می‌شوند. این تصاویر مبنای تحلیل در مراحل بعدی پژوهش‌اند.

در این پژوهش، سه طیف اصلی از ضرایب صوتی شامل MFCC[1]، BFCC[2] و GFCC[3] برای استخراج ویژگی‌های دوبعدی سیگنال صوتی مورد استفاده قرار گرفته‌اند. ضرایب MFCC بر اساس دستگاه شنوایی و ادراک صوتی انسان طراحی شده‌اند [۲۶] و به‌دلیل تطابق با ویژگی‌های شنیداری، کاربرد گسترده‌ای در تشخیص و شناسایی گفتار دارند. ضرایب BFCC که مشابه MFCC استخراج می‌شوند، از بانک پالایه‌های Bark به‌جای پالایه‌های Mel استفاده می‌کنند. این ضرایب به‌تازگی در تشخیص بدافزارهای اندروید به‌کار گرفته شده‌اند و توانایی بالایی در تمایز بدافزارها نشان داده‌اند [۲۵, ۲۷]. ضرایب GFCC نیز برای شبیه‌سازی دقیق‌تر پردازش صوتی حلزون گوش انسان طراحی شده‌اند و با توجه به شباهت زیاد به نحوهٔ پردازش صوتی انسان، دقت و مقاومت بیشتری در برابر نوفه دارند. این ضرایب نیز به‌تازگی در شناسایی بدافزارها استفاده شده و عملکرد قابل توجهی از خود نشان داده‌اند [۲۵, ۲۸].

تا این مرحله، به‌ازای هر فایل برنامه در مجموعه‌داده‌ها، سه بردار دوبعدی MFCC، BFCC و GFCC که از سیگنال صوتی استخراج شده‌اند، در اختیار داریم. ابعاد این بردارها با یکدیگر برابر بوده و هرکدام را می‌توان به‌عنوان یک تصویر تک‌رنگ درنظر گرفت. با توجه به عملکرد بهتر مدل‌های یادگیری عمیق در طبقه‌بندی تصاویر رنگی به‌دلیل وجود اطلاعات بیشتر در این تصاویر نسبت به تصاویر سیاه‌وسفید [۱۷]، این سه بردار برای ایجاد یک تصویر رنگی سه‌کاناله مورد استفاده قرار می‌گیرند؛ بدین ترتیب، تصویر MFCC در کانال قرمز (Red)، تصویر BFCC در کانال سبز (Green) و تصویر GFCC در کانال آبی (Blue) قرار داده می‌شود؛ نتیجهٔ نهایی، یک تصویر رنگی سه‌کاناله است که در شکل (۷) به‌صورت گرافیکی نمایش داده شده‌است.

با استفاده از این روش، علاوه‌بر ایجاد یک تصویر رنگی و تلفیق اطلاعات سه ویژگی مختلف در یک دادهٔ غنی‌شده، نیاز به آموزش سه مدل مختلف برای طبقه‌بندی هرکدام از این ویژگی‌ها حذف شد؛ درنتیجه، به ازای هر برنامه در مجموعه‌داده، یک تصویر رنگی با برچسب برنامهٔ خوش‌خیم یا بدخیم وجود دارد. این فرایند با تحلیل ایستای برنامه‌ها آغاز شد و با توجه به شباهت توالی بایت‌ها در برنامه‌ها به توالی اطلاعات در سیگنال‌های صوتی، یک فرایند نگاشتی طراحی شد که با نگاهی جعبه‌سیاه‌گونه و حذف پیچیدگی‌های متداول در تجزیهٔ بسته‌های برنامه، هر برنامه را به یک سیگنال صوتی تبدیل کرد؛ سپس طیف‌های MFCC، BFCC و GFCC از این سیگنال‌ها استخراج شد و با تلفیق این سه طیف، به‌ازای هر برنامه، یک تصویر رنگی ایجاد شد. در ادامه، از ظرفیت بالای مدل‌های یادگیری عمیق در حوزهٔ پردازش و طبقه‌بندی تصویر برای انجام فرایند تشخیص استفاده خواهد شد.

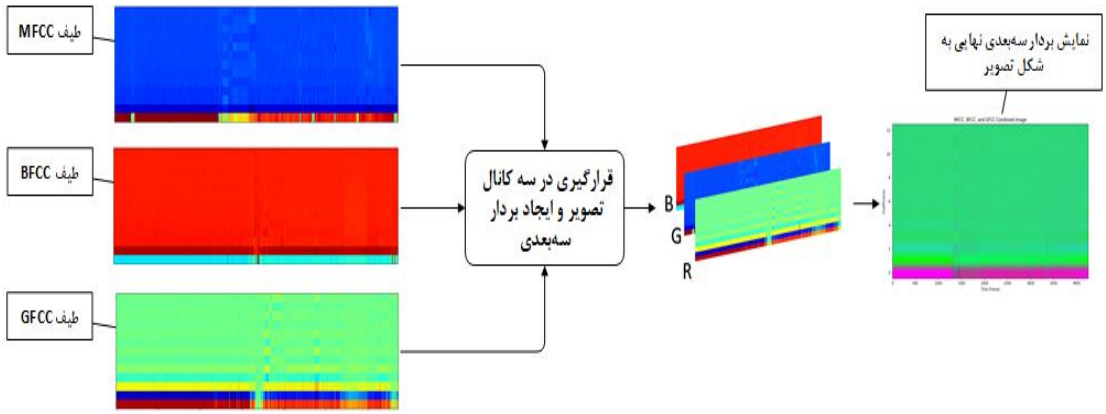
۳-۳- فاز سوم: طراحی، آموزش و آزمایش مدل‌ها

پس از جمع‌آوری داده‌ها در فاز نخست و انجام تحلیل‌های موردنیاز در فاز دوم، مجموعه‌ای از اطلاعات مربوط به داده‌های بخش تحلیل ایستا وجود دارد. سناریوهای مختلفی برای طراحی و آموزش یک مدل یادگیری عمیق وجود دارد ازجمله طراحی و آموزش یک مدل یادگیری عمیق از ابتدا و یا استفاده از مدل‌های ازپیش آموزش‌دیده که باید برای تنظیم مجدد آن‌ها اقدام کرد؛ از آنجا که پژوهش‌های مختلف [۱۴, ۱۶, ۲۰] نتایج بسیار مناسبی با استفاده از مدل‌های ازپیش آموزش‌دیده به‌دست آورده‌اند، پژوهش‌گران مقالهٔ حاضر نیز تصمیم بر استفاده از این نوع از مدل‌ها و سپس تنظیم مجددکردن آن گرفتند؛ همان‌طور که در بخش پیشین مشاهده شد، به‌ازای هر برنامهٔ اندرویدی موجود در مجموعه‌دادهٔ اولیه، یک تصویر رنگی در اختیار است که از ترکیب سه ویژگی MFCC، BFCC و GFCC به‌وجود آمده‌است. این سه طیف ویژگی برمبنای سیگنال صوتی تولیدی از بایت‌کدهای برنامه به‌وجود آمده‌اند؛ بنابراین در این قسمت پژوهش‌گران با یک مسئلهٔ طبقه‌بندی تصاویر رنگی مواجه‌اند. طبقه‌بندی تصاویر از مسائل بسیار معروف حوزهٔ پردازش تصویر است. در این مسائل، یک مدل یادگیری عمیق با کمک مجموعه‌ای از داده‌های تصویری، آموزش دیده و سپس طبقه‌بندی تصاویر را بر اساس برچسب‌های ورودی انجام می‌دهد. مدل‌های یادگیری عمیق زیادی وجود دارند که با معماری‌های بهینه، روی حجم بزرگی از داده‌های تصویری آموزش دیده‌اند تا بتوانند فرایند طبقه‌بندی تصاویر را با دقت بسیار بالایی انجام دهند [۲۹-۳۲].

از مدل‌های معروف می‌توان به MobileNetV2 با حجم چهارده مگابایت و ۳.۵ میلیون مؤلفه اشاره کرد که یکی از سبک‌ترین مدل‌هاییست که روی مجموعه‌دادهٔ ImageNet شامل ۱.۲ میلیون تصویر آموزش دیده و می‌تواند هزار کلاس مختلف را تشخیص دهد؛ درمقابل، مدل‌های پیچیده‌تری مانند VGG16 و NASNetLarge با حجم‌های بزرگ‌تر (به‌ترتیب ۵۲۸ و ۳۴۳ مگابایت) و تعداد مؤلفه‌های بیشتر (۱۳۸.۴ و ۸۸.۹ میلیون) طراحی شده‌اند که نشان‌دهندهٔ توانایی بالاتر آن‌ها در پردازش داده‌های پیچیده است. سایر مدل‌ها مانند Xception، InceptionV3 و EfficientNetB7 نیز با حجم‌ها و تعداد

[1] Mel Frequency Cepstral Coefficients
[2] Bark Frequency Cepstral Coefficients
[3] Gammatone Frequency Cepstral Coefficients

مؤلفه‌های متفاوت، گزینه‌های متنوعی برای کاربردهای مختلف ارائه می‌دهند.



و ایجاد یک بردار سه‌بعدیGFCC و BFCC، MFCC(شکل-۶): ترکیب سه طیف

(Figure-7): Combining the three spectra MFCC, BFCC, and GFCC and creating a 3D vector vector

مدل‌هایی که بر روی مجموعه‌داده‌های بزرگ و عمومی مانند ImageNet آموزش دیده‌اند، توانایی بالایی در درک بصری دارند. این مدل‌ها می‌توانند ویژگی‌های مؤثر بصری را از تصاویر استخراج و با استفاده از یادگیری انتقالی، بدون نیاز به آموزش از ابتدا، در مجموعه‌داده‌های کوچک‌تر یا خاص‌تر نیز به‌خوبی عمل کنند. این رویکرد باعث تسریع و افزایش دقت فرایند یادگیری می‌شود.

EfficientNetV2، خانواده‌ای از مدل‌های شبکهٔ عصبی پیچشی است که توسط پژوهش‌گران Google به‌عنوان نسخهٔ بهبودیافتهٔ مدل EfficientNet معرفی شده‌است. این مدل‌ها که برای طبقه‌بندی تصاویر طراحی شده‌اند، از روش آموزش تدریجی استفاده می‌کنند؛ در این روش، مدل با تصاویر ساده‌تر شروع به یادگیری می‌کند و سپس با تصاویر پیچیده‌تر ادامه می‌دهد [۳۲]. این استراتژی نه‌تنها زمان آموزش را کاهش می‌دهد، بلکه مصرف منابع محاسباتی را نیز بهینه می‌کند.

خانوادهٔ EfficientNetV2 شامل مدل‌هایی با اندازه‌ها و پیچیدگی‌های مختلف مانند EfficientNetV2-S، EfficientNetV2-M و EfficientNetV2-L است. این مدل‌ها برای کاربردهایی با محدودیت منابع یا وظایف پیچیده‌تر طراحی شده‌اند. EfficientNetV2-S برای محیط‌هایی با محدودیت منابع مناسب است؛ در حالی‌که EfficientNetV2-L برای دقت بالاتر در وظایف پیچیده‌تر استفاده می‌شود.

مدل EfficientNetV2 از چندین مزیت کلیدی برخوردار است: ۱) زمان آموزش کمتر و مصرف حافظهٔ پایین‌تر به‌دلیل یادگیری تدریجی، ۲) دقت بالاتر در طبقه‌بندی تصاویر نسبت به مدل‌های قدیمی‌تر مانند ResNet و MobileNet، ۳) قابلیت پیاده‌سازی بهینه در دستگاه‌های با محدودیت سخت‌افزاری مانند تلفن همراه‌ها و سامانه‌های نهفته. این ویژگی‌ها آن را گزینه‌ای ایدئال برای پژوهش حاضر می‌کند.

در این پژوهش، مدل EfficientNetV2B0 به‌عنوان مدل پایه انتخاب شد. تصاویر ورودی به ابعاد (۳، ۲۲۴، ۲۲۴) تغییر اندازه داده شدند. بخش انتهایی مدل پایه با شبکهٔ تمام متصل جایگزین شد تا تصاویر برنامه‌های خوش‌خیم و بدخیم را طبقه‌بندی کند. لایهٔ نهایی شامل یک نرون با تابع فعال‌سازی Sigmoid است که خروجی صفر یا یک را ارائه می‌دهد.

برای جلوگیری از بیش‌برازش[1] مدل، از لایهٔ Dropout استفاده شد. این لایه به‌طور تصادفی برخی از نرون‌ها را در زمان آموزش غیرفعال می‌کند. نرخ Dropout بین ۰.۲ تا ۰.۵ تنظیم شد تا از وابستگی بیش‌ازحد مدل به داده‌های خاص جلوگیری شود. تابع فعال‌سازی ReLU در لایه‌های Dense انتخاب شد؛ زیرا علاوه‌بر کارایی بالا، ویژگی‌های غیرخطی پیچیده‌تری را نیز می‌آموزد.

برای اتصال معماری جدید به مدل پایه، از لایهٔ GlobalAveragePooling2D استفاده شد. این لایه تعداد مؤلفه‌ها را کاهش داده و از آموزش بیش‌برازش جلوگیری می‌کند؛ همچنین، اطلاعات توزیعی در نقشهٔ ویژگی‌ها را حفظ می‌کند و دقت نهایی مدل را افزایش می‌دهد.

برای آموزش مدل، از بهینه‌ساز Adam استفاده شد که نرخ یادگیری متغیر و سرعت هم‌گرایی بالایی دارد. این بهینه‌ساز برای وظایف طبقه‌بندی دودویی به‌دلیل توانایی مقابله با داده‌های پیچیده و نوفه‌ها، بسیار مناسب است. تابع هزینهٔ انتخاب‌شده، آنتروپی متقابل دودویی بود که به‌طور مؤثر اختلاف بین پیش‌بینی و برچسب واقعی را محاسبه می‌کند.

[1] Overfit

(جدول-۳): معماری و مشخصات مدل

(Table-۳): Model architecture and specifications

	نام لایه	تعداد کل مؤلفه‌ها	ابعاد خروجی	تابع فعال‌ساز	توضیحات
۱	مدل پایهٔ EfficientNetV2B0	حدود ۶ میلیون	—	—	—
۲	GlobalAveragePooling2D	—	اندازهٔ خروجی: ۱۲۸۰	—	—
۳	Dense	۵۱۲	اندازهٔ خروجی: ۵۱۲	ReLU	—
۴	Dropout	۵۱۲	اندازهٔ خروجی: ۵۱۲	–	نرخ: ۰.۵
۵	Dense	۲۵۶	اندازهٔ خروجی: ۲۵۶	ReLU	—
۶	Dropout	۲۵۶	اندازهٔ خروجی: ۲۵۶	–	نرخ: ۰.۵
۷	Dense	۱	اندازهٔ خروجی: ۱	Sigmoid	—

(جدول-۴): تعداد مؤلفه‌های مدل نهایی در حالات مختلف

(Table-۴): Number of parameters of the final model in different cases

مشخصات مدل در زمان قابل آموزش‌بودن مدل پایه		مشخصات مدل در زمان غیرقابل آموزش‌بودن مدل پایه	
تعداد کل مؤلفه‌های مدل	۶۷۰۶۷۶۹	تعداد کل مؤلفه‌های مدل	۶۷۰۶۷۶۹
مؤلفه‌های قابل آموزش	۶۶۴۶۱۶۱	مؤلفه‌های قابل آموزش	۷۸۷۴۵۷
مؤلفه‌های غیرقابل آموزش	۶۰۶۰۸	مؤلفه‌های غیرقابل آموزش	۵۹۱۹۳۱۲

همان‌طور که از جدول (۳) نیز مشخص است، در مدل شبکهٔ عصبی عمیق این پژوهش، ابتدا مدل پایه قرار داشته و سپس لایه‌های جدید قرار گرفته‌اند. جدول (۴)، جزئیات مدل مثل تعداد مجموع مؤلفه‌های آن و تعداد مؤلفه‌های غیرقابل آموزش آن را نشان می‌دهد.

فرایند یادگیری انتقالی[1] یکی از روش‌های پرکاربرد و کارآمد در یادگیری عمیق است که به‌جای آموزش مدل از ابتدا، از دانش مدل‌های ازپیش آموزش‌دیده برای وظایف جدید استفاده می‌کند [۱۴، ۱۶]. این روش به‌ویژه در مواردی که داده‌های آموزشی محدودند، مؤثر است. مدل پایه که بر روی مجموعه‌داده‌های بزرگ و متنوع آموزش‌دیده، به‌عنوان منبع استخراج ویژگی‌های عمومی استفاده شده‌است و سپس این ویژگی‌ها برای آموزش مدل جدید و خاص‌منظوره به‌کار می‌روند؛ این روش علاوه‌بر کاهش زمان و منابع محاسباتی، دقت و کارایی مدل را بهبود می‌بخشد [۱۹].

در پژوهش حاضر، فرایند یادگیری انتقالی با استفاده از معماری پایهٔ EfficientNetV2B0 و افزودن لایه‌های جدید تمام‌متصل انجام شده‌است. استراتژی آموزش به‌صورت تدریجی و با فعال و غیرفعال‌کردن قابلیت یادگیری مدل پایه طراحی شده‌است تا از بیش‌برازش جلوگیری شود. در ابتدا، مدل پایه غیرقابل آموزش‌شده و لایه‌های جدید بر روی ویژگی‌های استخراج‌شده تنظیم می‌شوند و آموزش می‌بینند؛ سپس، با فعال‌سازی مجدد مدل پایه، ویژگی‌های آن برای تطبیق با داده‌های خاص بهینه‌سازی می‌شوند. این روش تعادلی میان استفاده از دانش عمومی و بهینه‌سازی تخصصی‌تر مدل برای وظیفهٔ جدید، ایجاد می‌کند.

این چرخهٔ آموزش تدریجی، موجب می‌شود، مدل از ویژگی‌های کلیدی مدل پایه بهره ببرد و درعین‌حال به تطبیق بهتر با داده‌های جدید دست یابد. این رویکرد نه‌تنها عملکرد مدل را در وظایف طبقه‌بندی دودویی بهبود می‌بخشد، بلکه تعمیم‌پذیری آن را نیز افزایش می‌دهد؛ همچنین، با مدیریت تدریجی داده‌ها و کاهش وابستگی سریع مدل به داده‌های جدید، مشکل آموزش بیش‌ازحد کاهش قابل‌توجهی می‌یابد. چالش مدیریت حجم بزرگ داده‌ها به کمک طراحی یک خط لولهٔ داده‌ای حل شد که با استفاده از TensorFlow پیاده‌سازی شده‌است؛ این خط لوله، داده‌ها را به‌صورت پویا در دسته‌های کوچک بارگذاری کرده و مراحل پردازش مانند استخراج طیف‌ها را هم‌زمان و در هنگام بارگذاری انجام می‌دهد. با این روش، مشکل محدودیت حافظه در حین آموزش مدل برطرف و فرایند آموزش بهینه‌تر شده‌است.

تمامی مراحل پیاده‌سازی و آموزش مدل در ویندوز یازده و با زبان پایتون نسخه ۳.۱۱.۱۰ انجام شده‌است؛ همچنین از کتابخانه‌های Tensorflow نسخه ۲.۱۷.۰ و NumPy نسخه ۱.۲۶.۴ برای آموزش مدل، پردازش داده‌ها و تحلیل نتایج استفاده شده‌است؛ همچین فرایند آموزش مدل با اندازه دستهٔ برابر با هشت و نرخ یادگیری برابر با ۰.۰۰۱ انجام شده‌است.

در این پژوهش، مدل جدید مبتنی‌بر یادگیری انتقالی طراحی و آموزش داده شده‌است. با مدیریت کارآمد داده‌ها و استفاده از معماری پیشرفتهٔ EfficientNet، پژوهش‌گران موفق شدند بر محدودیت‌های موجود غلبه کرده و عملکرد مدل را در طبقه‌بندی تصاویر بهبود دهند. در بخش‌های بعدی، نتایج و ارزیابی عملکرد مدل به‌صورت کامل ارائه شده‌است.

۳- فاز چهارم: ارزیابی

برای ارزیابی عملکرد مدل آموزش‌دیده در شناسایی بدافزارهای اندرویدی، از معیارهای مختلفی استفاده می‌شود تا دقت و کارایی آن مشخص شود. صحت نشان‌دهندهٔ درصد کلی نمونه‌هایی است که به‌درستی طبقه‌بندی شده‌اند. فراخوانی به کارایی مدل در تشخیص نمونه‌های بدافزار واقعی اشاره دارد؛ درحالی‌که دقت

[1] Transfer Learning

میزان صحت تشخیص‌های بدافزار را ارزیابی می‌کند. معیار F1 با محاسبهٔ میانگین هارمونیک دقت و فراخوانی، تعادل بین این دو را نشان می‌دهد. روابط (۱ تا ۴)، نحوهٔ محاسبهٔ این معیارها را نشان می‌دهد.

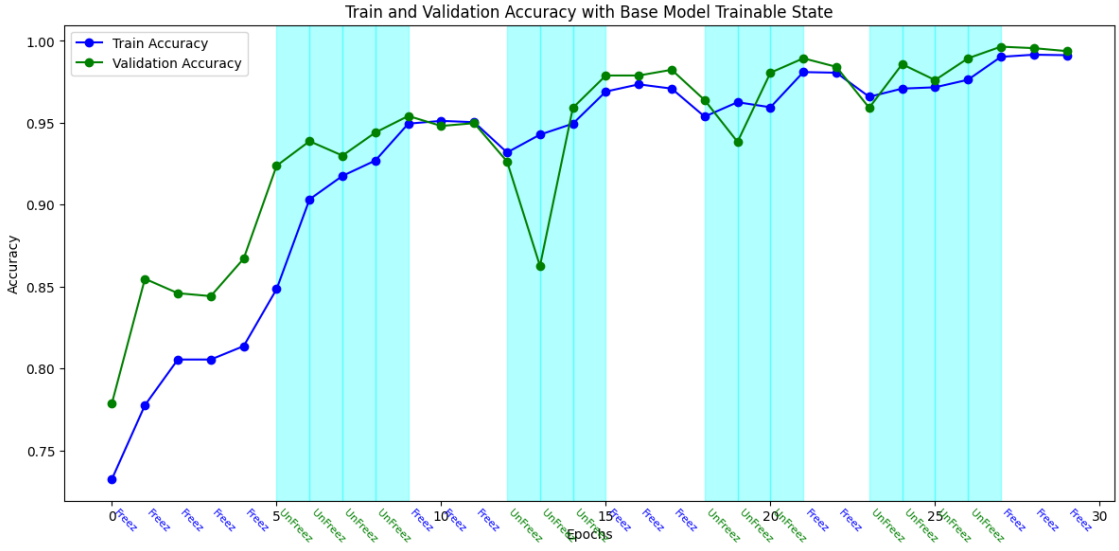
$$\text{Accuracy} = (TP+TN)/(TP+FP+TN+FN) \quad (۱)$$

$$\text{Recall} = TP/(TP+FN) \quad (۲)$$

$$\text{Precision} = TP/(TP+FP) \quad (۳)$$

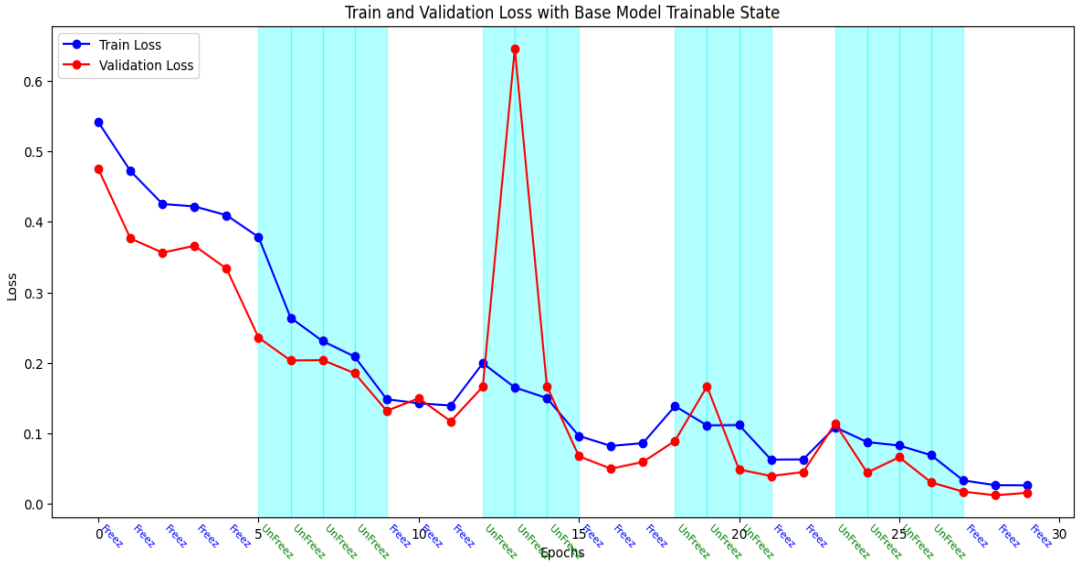
$$\text{F-measure} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision}+\text{Recall}) \quad (۴)$$

برای تحلیل عملکرد مدل در تشخیص نمونه‌های مثبت و منفی، از ماتریس اغتشاش استفاده می‌شود که تعداد نمونه‌های مثبت و منفی حقیقی و کاذب را مشخص می‌کند؛ همچنین، برای ارزیابی جامع‌تر، از منحنی ROC و معیار AUC بهره گرفته می‌شود؛ منحنی ROC نرخ تشخیص مثبت حقیقی در برابر مثبت کاذب را نمایش می‌دهد و مقدار AUC سطح زیر این منحنی را اندازه‌گیری می‌کند. مقدار بالاتر AUC نشان‌دهندهٔ کارایی بهتر مدل و استقلال آن از آستانهٔ تصمیم‌گیری است. این معیارها با نتایج پژوهش‌های معتبر مقایسه شده و نقاط قوت و ضعف مدل در ادامه تحلیل خواهند شد.



(شکل-۷): نمودار صحت مدل در train و validation. قابل آموزش‌بودن یا نبودن مدل پایه با زمینهٔ آبی رنگ در نمودار مشخص شده‌است

(Figure-8): Model accuracy chart in train and validation. The freezing state of the base model is indicated by the blue background in the chart.



(شکل-۸): نمودار هزینهٔ مدل در train و validation. قابل آموزش‌بودن یا نبودن مدل پایه با زمینهٔ آبی رنگ در نمودار مشخص شده‌است

(Figure-9): Model loss chart in train and validation. The freezing state of the base model is indicated by the blue background in the chart.

در شکل (۸)، نمودار صحت مدل در طول فرایند آموزش نمایش داده شده‌است. در بخش‌هایی با زمینهٔ سفید، وزن‌های مدل پایه غیرقابل آموزش بوده و در بخش‌هایی با زمینهٔ آبی، وزن‌های مدل پایه در حال بهینه‌سازی هستند؛ همان‌طور که مشاهده می‌شود، با پیشرفت آموزش، صحت مدل در مجموعهٔ اعتبارسنجی افزایش یافته که نشان‌دهندهٔ کارایی مطلوب فرایند آموزش است.

در شکل (۹)، نمودار هزینهٔ مدل در طول فرایند آموزش ارائه شده‌است. مشابه نمودار صحت، بخش‌های سفید و آبی به‌ترتیب نشان‌دهندهٔ غیرفعال‌بودن و فعال‌بودن وزن‌های مدل پایه برای آموزش هستند. کاهش مداوم هزینهٔ مدل در طول دوره‌های آموزش، عملکرد مناسب و اثربخشی فرایند آموزش را تأیید می‌کند.

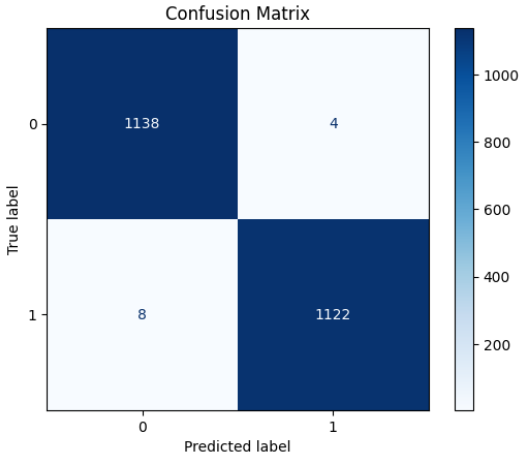
جزئیات فرایند آموزش مدل در جدول (۵) ارائه شده‌است.

(جدول-۵): مشخصات فرایند آموزش مدل

(Table-۵): Model Training Process Specifications

مجموع تعداد epochها		۳۰
تعداد epochها با غیرفعال‌بودن آموزش مدل پایه		۱۶
تعداد epochها با فعال‌بودن آموزش مدل پایه		۱۴
Train	میزان صحت در انتها	۰.۹۹۱
	میزان تابع هزینه در انتها	۰.۰۲
Validation	میزان صحت در انتها	۰.۹۹۳
	میزان تابع هزینه در انتها	۰.۰۱

نتایج به‌دست‌آمده نشان‌دهندهٔ عملکرد مطلوب مدل پس از آموزش است. صحت مدل در داده‌های آموزشی به ۹۹.۱۳ درصد و در داده‌های اعتبارسنجی به ۹۹.۳۸ درصد رسیده که بیان‌کنندهٔ توانایی بالای مدل در یادگیری الگوها و تعمیم‌دهی به داده‌های دیده‌نشده و مقادیر پایین تابع هزینه نیز نشان‌دهندهٔ خطای کم در پیش‌بینی‌ها است. این نتایج حاکی از یک مدل با کارایی بالا و تعمیم‌پذیری مناسب برای وظیفهٔ طبقه‌بندی دودویی است؛ همچنین، برای ارزیابی دقیق‌تر، عملکرد مدل‌ها با استفاده از مجموعه‌دادهٔ آزمون که به‌طور مجزا بر اساس روش HoldOut تهیه و محاسبه شده‌است. نتیجهٔ ارزیابی شامل صحت ۹۹.۳ درصد، دقت ۹۹.۸ درصد، فراخوانی ۹۹.۱ درصد و F-Measure برابر با ۹۹.۴ درصد را نشان می‌دهد. در شکل (۱۰) نیز ماتریس درهم‌ریختگی[1] عملکرد مدل بر روی مجموعه‌دادهٔ آزمون ارائه شده‌است.



(شکل-۹): ماتریس درهم‌ریختگی مدل

(Figure-10): Model confusion matrix

بررسی ماتریس درهم‌ریختگی عملکرد بسیار مناسب مدل را بر روی داده‌های آزمون تأیید می‌کند. این داده‌ها که پیش‌تر به‌وسیلهٔ مدل دیده نشده بودند، با دقت بالایی طبقه‌بندی شده‌اند. ماتریس نشان می‌دهد که مدل ۱۱۳۸ نمونه را به‌درستی به‌عنوان برنامه‌های خوش‌خیم و ۱۱۲۲ نمونه را به‌درستی به‌عنوان بدافزار شناسایی کرده‌است. خطاهای مدل بسیار محدود بوده و به‌ترتیب تنها چهار نمونه خوش‌خیم و هشت نمونه بدخیم به‌اشتباه برچسب خورده‌اند. این دقت بالا بیان‌کنندهٔ توانایی چشم‌گیر مدل در تفکیک داده‌هاست؛ هرچند باید به تهدید بالقوهٔ ناشی از شناسایی اشتباه هشت بدافزار به‌عنوان برنامهٔ خوش‌خیم توجه داشت. با افزایش حجم و تنوع داده‌ها در آموزش‌های بعدی می‌توان این خطاها را کاهش داد.

تحلیل شاخص‌های ارزیابی نیز این نتایج را تقویت می‌کند. صحت ۹۹.۳ درصد نشان‌دهندهٔ توانایی کلی مدل در تشخیص درست داده‌ها است. دقت ۹۹.۸ درصد تأیید می‌کند که نمونه‌های مثبت طبقه‌بندی‌شده بیشتر متعلق به برچسب صحیح بوده‌اند و فراخوانی ۹۹.۱ درصد بیان‌کنندهٔ آن است که بخش عمده‌ای از برنامه‌های بدخیم به‌درستی شناسایی شده‌اند. مقدار بالای F-measure تعادل میان دقت و فراخوانی را نشان می‌دهد.

علاوه‌براین، مدل توانسته است تمامی برنامه‌های مبهم‌شدهٔ موجود در داده‌های آزمون را بدون هیچ خطایی شناسایی کند و نرخ تشخیص برنامه‌های مبهم‌شدهٔ روش پیشنهادی به صددرصد در این مجموعه‌داده به ثبت برسد. این ترکیب از شاخص‌ها و نتایج، بیان‌کنندهٔ طراحی و آموزش دقیق مدل با کارایی بالا و خطای اندک است.

نتایج به‌دست‌آمده از مدل پیشنهادی، نه‌تنها در معیارهای استاندارد ارزیابی مانند دقت، صحت و فراخوانی عملکرد بسیار خوبی از خود نشان داده‌است، بلکه در مواجهه با یکی از چالش‌برانگیزترین مسائل حوزهٔ تشخیص بدافزار یعنی شناسایی برنامه‌های مبهم‌شده نیز به موفقیت دست یافته است. تمام نمونه‌های مبهم‌شدهٔ موجود در مجموعه‌دادهٔ آزمون، بدون خطا به‌وسیلهٔ مدل شناسایی شدند. این موضوع، یکی از برجسته‌ترین مزایای رویکرد پیشنهادی به‌شمار می‌رود؛ زیرا بسیاری از مدل‌های سنتی یا حتی یادگیری عمیق در مواجهه با این‌گونه نمونه‌ها، دچار خطا یا کاهش دقت می‌شوند.

همچنین، برای تحلیل بهتر عملکرد مدل در مواجهه با داده‌های نادیده، از منحنی ROC و مقدار AUC استفاده شد. مقدار AUC معادل ۰.۹۹۳۸ بر روی داده‌های آزمون و تفاوت ناچیز آن با داده‌های آموزش، نشان می‌دهد که مدل دچار بیش‌برازش نشده و دارای توان تعمیم بالا است؛ بنابراین باوجود دقت بالا، ساختار آموزش تدریجی و استفاده از یادگیری انتقالی، از بروز بیش‌برازش نیز جلوگیری شده‌است.

به‌منظور تحلیل عمیق‌تر خطاهای مدل، بررسی ماتریس درهم‌ریختگی نشان داد که درمجموع تنها دوازده نمونه از بیش از ۲۲۶۰ نمونه آزمون به‌اشتباه طبقه‌بندی شده‌اند و از میان آن‌ها، هیچ‌کدام مربوط به نمونه‌های مبهم‌شده نبوده و بیشتر

[1] Confusion Matrix

مربوط به نمونه‌هایی با ساختارهای مرزی و یا رفتارهای مخلوط بودند. این تحلیل نشان می‌دهد که بیشتر خطاهای مدل مربوط به نمونه‌های مرزی است که در بسیاری از مدل‌های طبقه‌بندی نیز به‌صورت ذاتی رخ می‌دهد.

برای روشن‌ترشدن جایگاه مدل پیشنهادی نسبت به مطالعات پیشین، جدول (۶) مقایسه‌ای بین عملکرد پژوهش حاضر و مطالعات کلیدی مرتبط ارائه می‌دهد. دقت ۹۹.۳ درصد مدل پیشنهادی، از پژوهش‌های مشابه که با رویکردهای صوتی یا تصویری کار کرده‌اند (نظیر پژوهش [۲۵]) بالاتر است.

(جدول-۶): مقایسهٔ عملکرد مدل پژوهش جاری با پژوهش‌های مرتبط

(Table-8): Comparing the performance of the current research model with related research

ردیف	پژوهش	نوع رویکرد		صحت	دقت	فراخوانی	امتیاز F	استفاده از مجموعه‌دادهٔ یکسان
		I*	A*					
۱	[۱۳]	*		۹۳	۹۲	۹۳	۹۲	بله
۲	[۲۰]	*		۹۵	۹۳	۸۹	۸۱	بله
۳	[۲۵]		*	۹۸.۹	۹۹.۰	۹۹.۶	۹۹.۳	بله
۴	پژوهش جاری	*	*	۹۹.۳	۹۹.۸	۹۹.۱	۹۹.۴	--

*A: رویکرد مبتنی‌بر داده‌های صوتی *I: رویکرد مبتنی‌بر داده‌های تصویری

نتایج این پژوهش نشان داد که راه‌کار پیشنهادی در تشخیص بدافزارهای اندرویدی عملکرد بسیار مناسبی دارد. نگاشت بایت‌های فایل‌های برنامه به حوزهٔ صوت و استخراج طیف‌های مختلف از آن، امکان شناسایی ویژگی‌های متنوع به‌وسیلهٔ مدل یادگیری عمیق را فراهم کرده و دقت تشخیص را به‌میزان قابل توجهی افزایش داده‌است.

با توجه به این نتایج، می‌توان گفت که رویکرد ارائه‌شده در این پژوهش نه‌تنها از نظر معماری مدل و ساختار داده نوآورانه است، بلکه در عمل نیز توانسته نسبت به روش‌های پیشین عملکرد بهتری ثبت کند و به‌دلیل ماهیت جعبه‌سیاه‌گونهٔ آن، برای شناسایی تهدیدات نوظهور، بسیار مناسب‌تر است.

۵- محدودیت و تهدیدهای وارد بر پژوهش

پژوهش حاضر، مانند هر پژوهش دیگری، با محدودیت‌ها و تهدیدهایی روبه‌رو بوده‌است؛ یکی از محدودیت‌های اصلی مربوط به فایل‌های APK ورودی بود؛ زیرا برخی از آن‌ها در مجموعه‌داده دچار خرابی بودند و ناچار باید حذف می‌شدند؛ به‌منظور حفظ اصالت داده‌ها و جلوگیری از استفاده از روش‌های نمونه‌سازی، تلاش شد با تعمیر فایل‌ها، بیشترین میزان ممکن از داده‌ها بازیابی و استفاده شود؛ البته افزایش حجم و تنوع مجموعه‌داده می‌تواند به بهبود عملکرد مدل کمک کند؛ همچنین محدودیت منابع پردازشی، استفاده از معماری‌های پیچیده‌تر شبکه‌های عصبی را زمان‌بر و پرهزینه می‌کرد؛ بنابراین تعادلی منطقی میان زمان آموزش و عملکرد مدل ایجاد شد تا هم کارایی حفظ شود و هم مدل نهایی در کاربردهای واقعی، به منابع پردازشی کمتری نیاز داشته باشد.

این افزایش دقت نشان‌دهندهٔ توانایی بهتر مدل پیشنهادی در کاهش خطاها و دسته‌بندی صحیح نمونه‌ها است. این برتری عملکردی، مدل پیشنهادی را به گزینه‌ای مناسب‌تر برای طبقه‌بندی دقیق داده‌ها تبدیل کرده‌است؛ همچنین، پژوهش فعلی تنها مدلی است که توانسته‌است در شرایط واقعی و با داده‌های مبهم‌شده، به دقت صددرصد در شناسایی آن‌ها دست یابد.

از سوی دیگر، بدافزارهای امروزی با استفاده از روش‌های پیشرفتهٔ مبهم‌سازی، شناسایی را دشوارتر می‌کنند؛ هرچند مدل پیشنهادی توانست نمونه‌های موجود در مجموعه‌داده را با دقت کامل تشخیص دهد، اما احتمال ناکارآمدی مدل در برابر روش‌های جدیدتر وجود دارد و نیاز به بهبود مدل برای مقابله با این روش‌ها احساس می‌شود؛ همچنین، مجموعه‌دادهٔ تحلیل پویا با ابزار CopperDroid در شرایط خاصی جمع‌آوری شده و ممکن است استفاده از ابزارها و شرایط دیگر به جمع‌آوری داده‌های غنی‌تر و دقیق‌تر منجر شود؛ درنهایت، ارزیابی راه‌کار پیشنهادی با فرض صحت نتایج پژوهش‌های معتبر صورت گرفت و مقایسه‌ها با بهترین پژوهش‌هایی انجام شد که از مجموعه‌داده‌های مشترک استفاده کرده بودند، تا هم ارزش علمی مقایسه افزایش یابد و هم در زمان و هزینه صرفه‌جویی شود.

یکی دیگر از چالش‌های موجود در این پژوهش، خطر بیش‌برازش به‌دلیل آموزش و ارزیابی مدل تنها بر روی مجموعه‌دادهٔ CIC است؛ هرچند این مجموعه‌داده شامل بدافزارهای متنوع، نمونه‌های واقعی و مبهم‌شده است، اما تکیه‌بر تنها یک منبع داده می‌تواند دقت مدل را در شرایط واقعی و داده‌های ناشناخته کاهش دهد. برای کاهش این خطر، از تقسیم دقیق داده‌ها به آموزش، اعتبارسنجی و آزمون استفاده و همچنین آموزش مدل به‌صورت تدریجی[1]، با کمک فرایند یادگیری انتقالی انجام شد تا توان تعمیم مدل افزایش یابد؛ بااین‌حال، برای اعتباربخشی بیشتر، در مراحل بعدی پژوهش، برنامه‌ریزی شده‌است تا مدل ارائه‌شده بر روی

[1] Freezing & Unfreezing

مجموعه‌داده‌های دیگر ازجمله Drebin، MalGenome و Androzoo نیز ارزیابی شود. این اقدام می‌تواند دید روشن‌تری از عملکرد مدل در مواجهه با تهدیدات متنوع‌تر فراهم کرده و پایداری آن را اثبات کند.

۶- نتیجه‌گیری و کارهای آینده

در این پژوهش، یک چهارچوب نوآورانه برای تشخیص بدافزارهای اندرویدی با تکیه‌بر تحلیل ایستا و یادگیری عمیق ارائه شد. رویکرد پیشنهادی با نگاشت بایت‌کد خام برنامه‌های اندرویدی به سیگنال‌های صوتی و استخراج سه طیف MFCC، BFCC و GFCC، داده‌ها را به‌صورت تصاویر رنگی سه‌کاناله بازنمایی کرد. این تصاویر به مدل ازپیش آموزش‌دیدهٔ EfficientNetV2B0 به‌عنوان ورودی داده شدند و با بهره‌گیری از روش یادگیری انتقالی، فرایند آموزش مدل با دقت و کارایی بالا انجام شد. دقت نهایی ۹۹.۳ درصد، همراه با توانایی در تشخیص به‌طور کامل موفق برنامه‌های مبهم‌شدهٔ واقعی، نشان‌دهندهٔ اثربخشی رویکرد پیشنهادی در شناسایی دقیق تهدیدهای اندرویدی است.

مزایای کلیدی روش پیشنهادی شامل مقاومت دربرابر مبهم‌سازی، بی‌نیازی به مهندسی معکوس، کاهش پیچیدگی محاسباتی و قابلیت اجرا در محیط‌های عملیاتی واقعی است. استفاده از مدل ازپیش آموزش‌دیده نیز زمان آموزش را کاهش داده و تعمیم‌پذیری مدل را بهبود بخشیده است. این عوامل روش ارائه‌شده را مناسب استفاده در سامانه‌های واقعی، ابزارهای قابل حمل و بازارهای اندرویدی می‌سازد.

در گام‌های آینده، افزایش تعداد داده‌ها و بهبود عملکرد مدل‌ها از اهداف اصلی است. با تمرکز بر بدافزارهای مبهم‌شده، تنوع داده‌ها افزایش خواهد یافت تا مدل در شناسایی این نوع بدافزارها نیز عملکرد بهتری داشته باشد؛ همچنین، ترکیب تحلیل ایستا و پویا در قالب یک مدل یادگیری عمیق واحد، فرایند آموزش را ساده‌تر و اطلاعات استخراج‌شده را غنی‌تر خواهد کرد؛ علاوه‌براین، توسعهٔ مدل‌های توضیح‌دهنده برای شناسایی بخش‌های مخرب فایل‌ها و ایجاد ابزارهای پیش‌بینی رفتار برنامه‌ها، از دیگر اهداف آینده است که می‌تواند امنیت کاربران را افزایش چشم‌گیری دهد.

درنهایت، تولید یک ابزار برون‌خط بر اساس مدل آموزش‌دیده، برای استفادهٔ کاربران و بازارهای برنامه‌های اندرویدی در دستور کار است. با توجه به حجم مناسب مدل نهایی، امکان استفاده از آن بر روی گوشی‌های تلفن همراه فراهم است و می‌تواند ابزاری کارآمد و قابل‌اعتماد برای شناسایی بدافزارها باشد. این رویکردها نه‌تنها کارایی مدل را بهبود می‌بخشند، بلکه به کاربران اطمینان بیشتری در استفاده از برنامه‌های اندرویدی می‌دهند؛ بااین‌حال، پژوهش حاضر نیز با برخی محدودیت‌ها مواجه بوده‌است؛ ازجمله، ارزیابی مدل تنها بر روی مجموعه‌دادهٔ CIC انجام و اگرچه شامل بدافزارهای واقعی و مبهم‌شده است، اما امکان دارد در مواجهه با انواع خاص دیگر بدافزارها، نیاز به تنظیم مجدد مدل وجود داشته باشد؛ همچنین، روش ارائه‌شده تنها مبتنی‌بر تحلیل ایستا بوده و رفتار زمان‌اجرای برنامه‌ها را به‌طور مستقیم درنظر نمی‌گیرد که این مسئله نیازمند انجام تحلیل پویا است.

در گام‌های آینده این مسیر پژوهشی، می‌توان موارد مختلفی را انجام داد؛ ازجمله:

- ارزیابی روش پیشنهادی بر روی مجموعه‌داده‌های دیگر مانند Drebin، Androzoo و MalGenome برای بررسی پایداری مدل
- ترکیب تحلیل ایستا و پویا در یک چهارچوب یک‌پارچه: برای این هدف، از ابزارهایی مانندDroidBox یا TaintDroid برای استخراج رفتارهای زمان اجرا استفاده خواهد شد و این داده‌ها می‌توانند برای آموزش مدل‌های یادگیری عمیق استفاده شوند.
- به‌کارگیری مدل‌های قابل توضیح[1] برای شناسایی نواحی مؤثر در تصمیم‌گیری مدل و ارائهٔ شفافیت بیشتر در دلایل تشخیص
- توسعهٔ یک ابزار برون‌خط و سبک برای کاربران عادی و توسعه‌دهندگان، با قابلیت اجرای مدل بر روی گوشی‌های هوشمند

در مجموع، پژوهش حاضر با بهره‌گیری از تلفیق هوشمندانهٔ تحلیل ایستا، پردازش صوتی، تصویرسازی رنگی از داده‌های طیفی و یادگیری انتقالی، روشی دقیق، مقاوم و عملیاتی برای تشخیص بدافزارهای اندرویدی ارائه داده‌است که می‌تواند گامی مؤثر در افزایش امنیت کاربران و ارتقای سطح دفاع سایبری در سکوی اندروید به‌شمار آید.

۶-مراجع 6-References

[1] J. Li et al., "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, 2018.

[2] N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and api calls," *in Proc. IEEE 25th Int. Conf. Tools with Artificial Intelligence (ICTAI)*, Washington, DC, USA, 2013, pp. 556–560.

[3] Y. Zhou et al., "Taming information-stealing smartphone applications (on Android)," in *Trust and Trustworthy Computing: 4th Int. Conf.,* TRUST 2011, Pittsburgh, PA, USA, June 22–24, 2011, vol. 6700, pp. 423–434.

[4] H.-J. Zhu et al., "DroidDet: effective and robust detection of Android malware using static analysis along with rotation forest model," *Neurocomputing,* vol. 272, pp. 638–646, 2018.

[5] V. M. Afonso et al., "Identifying Android malware using dynamically obtained features," *J. Comput. Virol. Hacking Tech.,* vol. 11, pp. 9–17, 2015.

[1] Explainable AI

[6] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, no. 3, pp. 891–919, 2018.
[7]G. Canfora, F. Mercaldo, and C. A. Visaggio, "Mobile malware detection using op-code frequency histograms," in 2015 12th Int. Joint *Conf. on e-Business and Telecommunications (ICETE)*, 2015, pp. 1–7.
[8]M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based Android malware detection using real devices," C*omputers & Security*, vol. 89, p. 101663, 2020.
[9]J. Kim et al., "MAPAS: a practical deep learning-based Android malware detection system," *Int. J. Inf. Secur.*, vol. 21, no. 4, pp. 725–738, 2022.
[10] X. Xiao et al., "Android malware detection based on system call sequences and LSTM," *Multimed. Tools Appl.*, vol. 78, pp. 3979–3999, 2019.
[11] S. I. Imtiaz et al., "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network," *Future Gener. Comput. Syst.*, vol. 115, pp. 844–856, 2021.
[12] D. T. Dehkordy and A. Rasoolzadegan, "A new machine learning-based method for Android malware detection on imbalanced dataset," *Multimed. Tools Appl.*, vol. 80, pp. 24533–24554, 2021.
[13] L. Shen et al., "Self-attention based convolutional-LSTM for Android malware detection using network traffics grayscale image," *Appl. Intell.*, 2022, pp. 1–23.
[14] K. Bakour and H. M. Ünver, "DeepVisDroid: Android malware detection by hybridizing image-based features with deep learning techniques," *Neural Comput. Appl.*, vol. 33, pp. 11499–11516, 2021.
[15] Y. Ding et al., "Android malware detection method based on bytecode image," *J. Ambient. Intell. Humaniz. Comput.*, 2020, pp. 1–10.
[16] T. Hsien-De Huang and H.-Y. Kao, "R2-d2: Color-inspired convolutional neural network (CNN)-based Android malware detections," in *Proc. IEEE Int. Conf. on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 2017–2026.
[17] D. Vasan et al., "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Networks*, vol. 171, p. 107138, 2020.
[18] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *J. Comput. Virol. Hacking Tech*., vol. 16, no. 2, pp. 157–171, 2020.
[19] P. Yadav et al., "EfficientNet convolutional neural networks-based Android malware detection," *Computers & Security*, vol. 115, p. 102622, 2022.
[20] J. Feng et al., "A two-layer deep learning method for Android malware detection using network traffic," *IEEE Access*, vol. 8, pp. 125786–125796, 2020.
[21] M. Farrokhmanesh and A. Hamzeh, "Music classification as a new approach for malware detection," *J. Comput. Virol. Hacking Tech*, vol. 15, pp. 77–96, 2019.
[22] F. Mercaldo, A. Santone, "Audio signal processing for android malware detection and family identification," *J. Comput. Virol. Hacking Tech.*, vol. 17, no. 2, pp. 139–152, 2021.
[23] R. Casolare et al., "Mobile Family Detection through Audio Signals Classification," in *Proc. SECRYPT*, 2021.
[24] A. Azab and M. Khasawneh, "Msic: malware spectrogram image classification," *IEEE Access*, vol. 8, pp. 102007–102021, 2020.
[25] P. Tarwireyi, A. Terzoli, and M.O. Adigun, "Using multi-audio feature fusion for android malware detection," *Computers & Security*, vol. 131, p. 103282, 2023.
[26] T. Irino and R.D. Patterson, "A time-domain, level-dependent auditory filter: The gammachirp," *The Journal of the Acoustical Society of America*, vol. 101, no. 1, pp. 412–419, 1997.
[27] P. Tarwireyi, A. Terzoli, and M.O. Adigun, "BarkDroid: Android malware detection using bark frequency Cepstral coefficients," *Indonesian Journal of Information Systems*, vol. 5, no. 1, pp. 48–63, 2022.
[28] X. Zhao and D. Wang, "Analyzing noise robustness of MFCC and GFCC features in speaker identification," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, Canada, 2013.
[29] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[30] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[31] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.
[32] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," in *Proc. Int. Conf. Machine Learning (ICML)*, PMLR, 2021.
[۳۳] قاسمی معصومه، حری عباس، بصیری محمداحسان، «تشخیص بدافزارهای اندرویدی با رویکرد تخلیهٔ بار در محاسبات ابری»، *فصلنامهٔ پردازش علائم و داده‌ها*. ۲۱ (۳)، صص ۱۳۷-۱۴۸، ۱۴۰۳.
[33] M. Ghasemi, A. Horri, and M. E. Basiri, "Detecting Android malware with offloading approach in cloud computing," *J. Signal & Data Processing.*, vol. 21, no. 3, pp. 137-148, 2024.
[۳۴] دی پیر محمود، «ریسک سنج: ابزاری برای سنجش دقیق میزان ریسک امنیتی برنامه‌ها در دستگاه‌های همراه»، *فصلنامهٔ پردازش علائم و داده‌ها*، ۱۴ (۳)، صص ۲۳-۳۶، ۱۳۹۶.
[34] M. Deypir, "RiskMeter: A Tool for Measuring Precise Security Risk Values of Mobile Device Applications," *J. Signal & Data Process.*, vol. 14, no. 3, pp. 23–36, 2017.

علی علیائی طرقبه مدرک کارشناسی و کارشناسی ارشد خود را در رشتهٔ مهندسی کامپیوتر گرایش نرم‌افزار به‌ترتیب در سال‌های ۱۴۰۰ و ۱۴۰۳ از دانشگاه فردوسی مشهد دریافت کرد. ایشان در حال حاضر دانشجوی دکترای مهندسی کامپیوتر گرایش نرم‌افزار در دانشگاه فردوسی مشهد است. زمینه‌های پژوهشی مورد علاقهٔ ایشان مهندسی نرم‌افزار، هوش مصنوعی و یادگیری عمیق است.
نشانی رایانامهٔ ایشان عبارت است از:
ali.olyaei@mail.um.ac.ir

عباس رسول‌زادگان کارشناسی خود را در رشتهٔ مهندسی کامپیوتر - مهندسی نرم‌افزار از دانشگاه هوانوردی تهران در سال ۱۳۸۳، کارشناسی ارشد را در رشتهٔ مهندسی کامپیوتر - مهندسی نرم‌افزار از دانشگاه صنعتی امیرکبیر در سال ۱۳۸۶ و دکترای خود را در رشتهٔ مهندسی کامپیوتر – نرم‌افزار از همان دانشگاه در سال ۱۳۹۲ دریافت کرد. ایشان در حال حاضر دانشیار گروه مهندسی کامپیوتر دانشگاه فردوسی مشهد است. زمینه‌های پژوهشی مورد علاقه ایشان عبارت‌اند از: مهندسی کیفیت نرم‌افزار و تست هوشمند نرم‌افزار.
نشانی رایانامهٔ ایشان عبارت است از:
rasoolzadegan@um.ac.ir
برای اطلاعات بیشتر دربارهٔ ایشان، به تارنمای SQLab مراجعه کنید:
https://SQLab.um.ac.ir