

بسم الله الرحمن الرحيم



دانشگاه شهید بهشتی کرمان

دانشکده فنی و مهندسی
بخش مهندسی کامپیوتر

پایان نامه تحصیلی برای دریافت درجه کارشناسی ارشد

رشته مهندسی کامپیوتر گرایش هوش مصنوعی

تشخیص قدرتمند بدافزارهای اندروید با استفاده از شبکه‌های عصبی
ترنسفورمر

مؤلف:
علیرضا ایرانمنش

استاد راهنما:
دکتر حمید میروزی

اردیبهشت ۱۴۰۴

بـنام خدا

مـموراـخـلـقـ پـوـشـ

با احـسـانـ اـنـدـهـ اـیـ جـانـ وـ اـعـتـدـ اـلـ اـخـلـقـ اـیـ بـنـامـ عـالـ مـصـرـفـ دـاـستـ وـ اوـ هـوـلـهـ نـافـرـ اـعـالـ مـاـسـتـ بـهـ مـطـورـ فـاجـمـ شـایـرـتـیـ پـوـشـ بـهـ اـسـلـ،ـ تـوـیدـ اـنـشـ جـعـیدـ وـ بـسـارـیـ نـزـکـلـیـ بـشـرـمـاـ اـنـجـیـانـ وـ اـعـتـادـ بـیـاتـ عـلـیـ اـلـخـاـهـ دـوـشـ بـلـخـادـ بـهـ کـشـرـ:

- تمامـ تـلـاشـ خـودـ رـاـیـ کـشـتـ وـ خـلـقـ تـحـیـتـ بـکـارـ خـانـیـمـ بـتـ وـ اـزـ کـوـنـ جـلـ وـ خـرـیـتـ دـفـایـتـ بـهـ عـلـیـ پـرـسـیـزـ کـشـمـ.
- حقـقـ پـوـرـشـکـانـ،ـ پـوـرـمـیـکـانـ (ـاـنـانـ،ـ جـیـانـ،ـ بـنـاتـ وـ آـشـیـاءـ)،ـ سـانـانـ بـهـ سـایـرـ صـاحـبـانـ حقـقـ رـاـبـرـیـتـ مـیـ شـایـمـ وـ «ـحـظـ آـنـ مـیـ کـوـشـمـ».
- بـاـلـکـیـتـ اـدـیـ وـ مـمـونـیـ آـثـرـشـوـشـیـ اـنـجـ مـیـ نـسـیـمـ،ـ بـرـایـ اـخـبـرـشـوـشـیـ اـسـلـ (ـاـنـامـ وـ زـنـیدـ وـ اـزـ سـرـقـتـ عـلـیـ وـ اـرـجـاعـ نـهـاـبـ اـبـتـابـ کـشـمـ).
- خـسـنـ پـایـنـدـیـ بـهـ اـنـصـافـ وـ اـبـتـابـ اـزـ کـوـنـ تـبـیـنـ تـحـسـبـ وـ کـیـنـهـیـ نـهـایـتـ بـهـ پـوـشـشـ،ـ رـیـانـیـ تـمـادـ اـخـادـ خـانـیـمـ کـردـ.
- خـسـنـ اـنـاسـ دـارـیـ،ـ اـزـ سـایـجـ وـ اـنـخـانـتـ اـقـسـادـیـ،ـ اـنـسـانـ وـ فـنـ مـوـجـودـ،ـ اـسـخـادـ وـ دـوـرـلـهـ خـانـیـمـ کـردـ.
- اـزـ اـنـمـاـرـ غـیرـ اـخـلـقـیـ اـنـجـ پـوـشـ،ـ نـهـیـ اـنـمـاـرـ مـوـازـیـ،ـ بـهـ شـانـ دـچـکـدـ کـهـ اـیـ پـرـسـیـزـ کـشـمـ.
- اـسـلـ مـرـبـدـ بـوـنـ وـ رـاـلـدـارـیـ رـاـحـوـرـ تـمـ خـلـقـتـ بـهـ پـوـشـشـ خـوـقـرـمـ دـیـمـ.
- دـهـدـ نـهـایـتـ بـهـ پـوـشـشـ بـلـانـجـ مـلـ تـبـجـدـ کـرـدـ وـ بـرـایـ تـحـقـیـقـ آـنـ مـیـ کـوـشـمـ.
- خـوـشـ رـاـلـزـمـ بـهـ رـیـاضـتـ کـیـ خـجـدـرـهـیـ عـلـیـ رـشـخـوـدـ،ـ قـوـنـیـ وـ مـرـتـرـاتـ بـیـاسـتـهـیـ حـرـفـاـیـ،ـ سـانـانـ،ـ دـوـتـیـ وـ رـاـبـرـهـدـهـیـ فـیـ دـهـدـ مـرـاـلـ پـوـشـشـ مـیـ دـانـمـ.
- رـیـاتـ اـصـلـ اـخـلـقـ دـوـرـشـ رـاـهـاـیـ فـرـکـلـیـ وـ اـنـسـمـ وـ بـهـ مـطـبـرـانـگـیـ اـینـ فـرـنـکـ بـهـ تـوـجـ وـ اـلـهـ آـنـ دـجـاسـ اـنـامـ مـیـ وـزـیـمـ.



دانشگاه شیدا بهمن

تعهدنامه

اینجانب علیرضا ایرانمنش به شماره دانشجویی ۱۵۱۱۵۵۰۱۵ دانشجوی مقطع کارشناسی ارشد رشته مهندسی

کامپیوتر-هوش مصنوعی دانشکده فنی مهندسی دانشگاه شهید باهنر کرمان نویسنده پایاننامه با عنوان « تشخیص

قدرتمند بدافزارهای اندروید با استفاده از شبکه‌های عصبی ترانسفورم » تحت راهنمایی دکتر حمید میروزیری تأیید

می‌کنم که این پایاننامه نتیجه پژوهش اینجانب می‌باشد و در عین حال که موضوع آن تکراری نیست، در صورت

استفاده از منابع دیگران، نشانی دقیق و مشخصات کامل آن درج شده است. همچنین موارد زیر را نیز تعهد می‌کنم:

۱- برای انتشار تمام یا قسمتی از داده‌ها یا دستاوردهای خود در مجامع و رسانه‌های علمی اعم از همایش‌ها و مجلات

داخلی و خارجی به صورت مقاله، کتاب، ثبت اختراع و ... به صورت مکتوب یا غیرمکتوب، با کسب مجوز از

دانشگاه شهید باهنر کرمان و استاد(ان) راهنما اقدام نمایم.

۲- از درج اسمی افراد خارج از کمیته پایاننامه در جمع نویسندهان مقاله‌های مستخرج از پایاننامه، بدون مجوز

استاد(ان) راهنما اجتناب نمایم و اسمی افراد کمیته پایان نامه را در جمع نویسندهان مقاله درج نمایم.

۳- از درج نشانی یا وابستگی کاری (affiliation) نویسندهان سازمان‌های دیگر (غیر از دانشگاه شهید باهنر کرمان)

در مقاله‌های مستخرج از پایاننامه بدون تأیید استاد(دان) راهنما اجتناب نمایم.^۱

۴- کلیه ضوابط و اصول اخلاقی مربوط به استفاده از موجودات زنده یا بافت‌های آنها را برای انجام پایاننامه رعایت

نمایم.

۵- در صورت اثبات تخلف (در هر زمان) مدرک تحصیلی صادر شده توسط دانشگاه شهید باهنر کرمان از درجه اعتبار ساقط و اینجانب هیچ‌گونه ادعایی نخواهم داشت.

کلیه حقوق مادی و معنوی این اثر (مقالات مستخرج، برنامه‌های رایانه‌ای، نرم افزارها و تجهیزات ساخته شده)

مطابق با آئین نامه مالکیت فکری، متعلق به دانشگاه شهید باهنر کرمان است و بدون اخذ اجازه کتبی از دانشگاه قابل

واگذاری به شخص ثالث نیست. همچنین استفاده از اطلاعات و نتایج این پایاننامه بدون ذکر مرجع مجاز نمی‌باشد.

چنانچه مبادرت به عملی خلاف این تعهدنامه محرز گردد، دانشگاه شهید باهنر کرمان در هر زمان و به هر نحو مقتضی

حق هرگونه اقدام قانونی را در استیفای حقوق خود دارد.

تاریخ و امضاء:

علیرضا ایرانمنش

۱۴۰۴ اردیبهشت

^۱ تنها آدرس مورد قبول برای دانشگاه به این صورت می‌باشد:

Shahid Bahonar University of Kerman, Kerman, Iran.

نام و آدرس واحدهای دانشگاه در تولیدات علمی محققان دانشگاه به تشخیص بخش و دانشکده به شرح زیر می‌باشد:

Department of Computer, Faculty of Engineering Shahid Bahonar University of Kerman, Kerman, Iran.

آدرس صحیح جهت درج در مقالات و سایر تولیدات علمی فارسی:

گروه (بخش) کامپیوتر، دانشکده فنی مهندسی، دانشگاه شهید باهنر کرمان، کرمان، ایران.

تقدیم به:

«با نهایت احترام و سپاس، این پایان‌نامه را تقدیم می‌کنم به:

مهندس علیرضا افضلی‌پور و بانو فاخره صبا:

دو انسان فرهیخته و نیک‌اندیش که با ایثار و آینده‌نگری، بنیان‌گذار دانشگاه شهید باهنر کرمان شدند و مسیر علم و دانش را در این دیار هموار ساختند.

آن‌ها با وقف دارایی و زندگی خود، نهالی از دانش کاشتند که امروز به درختی تناور بدل شده و ثمرات آن در سراسر کشور نمایان است.

یاد و خاطره‌شان همواره الهام‌بخش نسل‌های آینده خواهد بود.»

تشکر و قدردانی:

با سپاس از خداوند بزرگ که به من توانایی و انگیزه برای پیمودن این مسیر علمی را عطا نمود. این پایان نامه حاصل تلاش و کوشش های فراوان است و بدون حمایت و راهنمایی های ارزشمند افراد بسیاری به ثمر نمی نشست.

به مصداق شعر «به یاد کسی که در این راه بود، به یاد کسی که در این راه رفت»، شایسته می دانم مراتب سپاس و قدردانی صمیمانه خود را تقدیم نمایم به استاد فرهیخته و فرزانه، جناب آقای دکتر حمید میروزیری، که با دانش و تجربه خود همواره راهنمای من بودند و با صبر و شکیابی به سوالات و ابهاماتم پاسخ دادند، صمیمانه تشکر می کنم.

همچنین از دوست عزیزم، محمدحسین شبانی، که با حمایت های بی دریغ و تشویق های همیشگی اش، انگیزه و انرژی مضاعفی به من بخشید، قدردانی می نمایم.

در پایان، از خانواده عزیزم که با عشق و محبت بی پایان خود همواره پشتیبان من بودند و در تمامی مراحل این مسیر پرچالش، همراه و همدم من بودند، بی نهایت سپاسگزارم.

چکیده:

با افزایش روزافزون تهدیدات سایبری، تشخیص بدافزارهای اندرویدی به یکی از چالش‌های اساسی در حوزه امنیت اطلاعات تبدیل می‌شود. روش‌های سنتی، بهویژه آن‌هایی که صرفاً بر تحلیل ویژگی‌های تک‌وجهی تکیه دارند، اغلب در پردازش داده‌های پیچیده چندوجهی ناتوان هستند و در مواجهه با تهدیدات جدید، از تعمیم‌پذیری مناسبی برخوردار نیستند. این محدودیت‌ها، ضرورت توسعه رویکردهای نوین و کارآمد را آشکار می‌سازند. در این پژوهش، مدلی چندوجهی با عنوان «تبدیل گر چندوجهی» مبتنی بر جاسازی گراف دینامیک با توجه پویا (MAGNET) توسعه می‌یابد که با ترکیب داده‌های جدولی، گراف و ترتیبی، از جمله توالی فراخوانی‌های API به شناسایی بدافزارهای اندرویدی می‌پردازد. هدف اصلی، ارتقای دقت و پایداری تشخیص با بهره‌گیری از معماری پیشرفته‌ای مانند PIRATES و Optuna انجام می‌گیرد و مدل با مجموعه داده‌ای شامل ۴۶۴۱ نمونه آموزشی و ۱۴۵۱ نمونه آزمایشی، همراه با اعتبارسنجی متقطع پنج‌تایی، آموزش می‌بیند. ویژگی‌های مورد استفاده شامل ویژگی‌های ایستا نظریه مجوزها، فراخوانی‌های API، مقاصد و نام مؤلفه‌ها و همچنین ویژگی‌های پویا مانند فعالیت شبکه و دسترسی به فایل‌ها هستند. داده‌ها به صورت بردارهای عددی باینری یا نرم‌افزاری شده آماده‌سازی می‌شوند و پس از پیش‌پردازش، ابعاد ویژگی‌ها به ۴۳۰ ویژگی تنظیم می‌گردند. ابزارهای مورد استفاده شامل کتابخانه‌های یادگیری عمیق مانند PyTorch، تکنیک‌های استاندارد سازی و نرم‌افزاری داده‌ها و ساختارهای داده‌ای گرافی هستند. نتایج نشان می‌دهند که مدل پیشنهادی عملکردی برجسته با دقت بالا، پایداری قابل توجه و قابلیت تعمیم‌پذیری مطلوب ارائه می‌دهد و نسبت به روش‌های پیشین بهبود قابل ملاحظه‌ای دارد. این یافته‌ها، پتانسیل کاربرد مدل در سیستم‌های امنیتی واقعی را برجسته می‌سازند.

واژگان کلیدی: تشخیص بدافزار، ترنسفورمر، یادگیری عمیق، داده‌های چندوجهی، امنیت اندروید.

فهرست مطالب

عنوان	صفحة
فصل اول: کلیات پژوهش	۱
۱-۱ مقدمه و بیان مسئله	۲
۱-۱-۱ روش‌های تشخیص بدافزار	۳
۱-۱-۱-۱ مجموعه داده‌های مربوطه	۳
۱-۱-۱-۲ ضرورت تحقیق و اهداف	۳
۱-۱-۱-۳ سازماندهی پایان نامه	۴
فصل دوم: پیشینه تحقیق و مفاهیم پایه	۶
۱-۲ مقدمه	۷
۲-۱ مفاهیم پایه	۸
۲-۱-۱ تکامل روش‌های یادگیری ماشین	۸
۲-۱-۲-۱ انواع بدافزار	۹
۲-۱-۲-۲ بایج افزار	۹
۲-۱-۲-۳ تروجان	۹
۲-۱-۲-۴ جاسوس افزار	۱۰
۲-۱-۲-۵ تبلیغ افزار	۱۰
۲-۱-۲-۶ مفاهیم مرتبط با اپلیکیشن‌های اندرویدی	۱۱
۲-۱-۳-۱-۱ مجوزهای دسترسی	۱۱
۲-۱-۳-۲-۱ فایل APK	۱۲
۲-۱-۳-۲-۲ سورس کد	۱۲
۲-۱-۴-۱-۱ داده‌های چندوجهی در تشخیص بدافزار	۱۳
۲-۱-۴-۲-۱ داده‌های جدولی	۱۳
۲-۱-۴-۲-۲ داده‌های گرافی	۱۴
۲-۱-۴-۲-۳ داده‌های ترتیبی	۱۴
۲-۱-۴-۲-۴ اهمیت و یکپارچگی داده‌های چندوجهی	۱۵
۲-۱-۴-۲-۵-۱ مدل‌های یادگیری عمیق	۱۵
۲-۱-۴-۲-۵-۲ شبکه‌های عصبی کانولوشنی (CNN)	۱۵

۱۶	شبکه‌های عصبی بازگشتی و واحدهای حافظه بلند-کوتاه	۲-۵-۲-۲
۱۷	شبکه‌های عصبی گرافی (GNN)	۳-۵-۲-۲
۱۸	ترنسفورمرها	۶-۲-۲
۱۸	-۰-۶-۲-۲ - امدادگیری عمیق	۰-۰-۶-۲-۲
۱۹	ترنسفورمرها در مدل MAGNET	۷-۲-۲
۲۱	ماشین بردار پشتیبان (SVM)	۸-۲-۲
۲۱	SVM - انواع	۰-۰-۸-۲-۲
۲۱	۰-۰-۸-۲-۲ - کاربرد در تشخیص بدافزار	۰-۰-۸-۲-۲
۲۲	۰-۰-۸-۲-۲ - همایا و معایب	۰-۰-۸-۲-۲
۲۲	تکنیک‌های آموزشی و ارزیابی	۳-۲
۲۲	اعتبارسنجی متقطع (Cross-Validation)	۱-۳-۲
۲۳	مدیریت بیش‌برازش و کم‌برازش	۲-۳-۲
۲۴	روش‌های بهینه‌سازی	۳-۳-۲
۲۵	الگوریتم Adam	۱-۳-۳-۲
۲۵	الگوریتم PIRATES	۲-۳-۳-۲
۲۶	الگوریتم Optuna	۳-۳-۳-۲
۲۷	معیارهای ارزیابی عملکرد	۴-۳-۲
۲۷	دقت (Accuracy)	۱-۴-۳-۲
۲۸	Score F1	۲-۴-۳-۲
۲۹	Area Under the ROC Curve AUC	۳-۴-۳-۲
۲۹	-۳-۴-۳-۲ - اجمع‌بندی ارزیابی	۰-۰-۴-۳-۲
۲۹	مروری بر مطالعات پیشین	۴-۲
۳۰	روش‌های تحلیل ایستا	۱-۴-۲
۳۰	تحلیل ویژگی‌های مبتنی بر مانیفست و متاداده	۱-۱-۴-۲
۳۰	تحلیل کد (Code Analysis)	۲-۱-۴-۲
۳۱	تحلیل ایستای ساختاری	۳-۱-۴-۲
۳۱	-۳-۱-۴-۲ - محدودیت‌های تحلیل ایستا	۰-۰-۴-۳-۲
۳۱	روش‌های تحلیل پویا	۲-۴-۲

۳۲	مانیتورینگ رفتار سیستم	۱-۲-۴-۲
۳۲	تحلیل شبکه	۲-۲-۴-۲
۳۲	بررسی مصرف منابع	۳-۲-۴-۲
۳۲	البازارها و محیط‌های تحلیل پویا	۴-۲-۴-۲
۳۲	امحدودیت‌های تحلیل پویا	۵-۲-۴-۲
۳۳	روش‌های ترکیبی Hybrid Approaches	۶-۲-۴-۲
۳۳	کارهای پیشین و تاریخچه مختصر	۷-۴-۲
۳۴	جمع‌بندی فصل	۸-۲
۳۵	مطالعه موردی: شرکت‌های پیشوور در امنیت اندروید	فصل سوم:
۳۶	هدف فصل	۱-۳
۳۶	معیارهای انتخاب	۲-۳
۳۶	مطالعه موردی شرکت‌ها	۳-۳
۳۶	NowSecure	۱-۳-۳
۳۶	Qualys (VMDR Mobile)	۲-۳-۳
۳۶	Synopsys / Checkmarx	۳-۳-۳
۳۷	روش‌ها و ابزارهای مشترک	۴-۳
۳۷	مقایسه و تحلیل	۵-۳
۳۷	جمع‌بندی	۶-۳
۳۸	روش پیشنهادی	فصل چهارم:
۳۹	روش پیشنهادی	۱-۴
۳۹	مقدمه روشن پیشنهادی	۱-۱-۴
۴۰	فرضیات و ابزارهای محاسباتی	۲-۱-۴
۴۰	فرضیات	۱-۲-۱-۴
۴۱	ابزارهای محاسباتی	۲-۲-۱-۴
۴۲	روش‌شناسی	۳-۱-۴
۴۲	پیش‌پردازش داده‌ها	۱-۳-۱-۴
۴۳	طراحی مدل MAGNET	۲-۳-۱-۴
۴۳	ماژول جدولی EnhancedTabTransformer	۱-۲-۳-۱-۴

۴۴	GraphTransformer (ماژول گرافی)	۲-۲-۳-۱-۴
۴۵	SequenceTransformer (ماژول ترتیبی)	۳-۲-۳-۱-۴
۴۵	۱-۴-۲-۳-۱-۴	- عملکاری ممکنیزم توجه پویا
۴۶	۱-۴-۲-۳-۱-۴	- ۵دادگام چندوجهی
۴۶	۱-۴-۲-۳-۱-۴	- MAGNET
۴۶	۳-۳-۱-۴	- آموزش مدل
۴۷	۴-۳-۱-۴	- بهینه‌سازی ابرپارامترها
۴۷	۵-۳-۱-۴	- ارزیابی مدل
۴۷	۴-۱-۴	- جمع‌بندی روش پیشنهادی

		فصل پنجم: نتایج و بحث
۵۰		
۵۱	۱-۵
۵۱	۲-۵
۵۱	۱-۲-۵
۵۲	۲-۲-۵
۵۳	۳-۵
۵۳	۱-۳-۵
۵۳	۲-۳-۵
۵۴	۳-۳-۵
۵۵	۴-۳-۵
۵۵	۴-۵
۵۶	۱-۴-۵
۵۶	۲-۴-۵
۵۷	۳-۴-۵
۵۷	۴-۴-۵
۵۸	۵-۴-۵
۵۸	۶-۴-۵
۵۹	۵-۵
۶۰	۶-۵
۶۰	۱-۶-۵

۶۰	نتایج اعتبارسنجی متقاطع	۲-۶-۵
۶۲	تحلیل عملکرد کلی مدل در مراحل مختلف	۳-۶-۵
۶۳	مقایسه با روش‌های پایه و پیشرفته	۷-۵
۶۳	مقایسه با روش‌های پایه	۱-۷-۵
۶۴	مقایسه با روش‌های پیشرفته	۲-۷-۵
۶۵	تحلیل مقایسه با روش‌های پیشرفته	۳-۷-۵
۶۵	نتایج مقایسه با روش‌های پایه	۴-۷-۵
۶۷	تحلیل عملکرد مازول‌های مدل	۵-۷-۵
۶۷	مطالعه حذف اجزا	۶-۷-۵
۶۷	روند آموزش	۷-۷-۵
۶۷	جمع‌بندی	۸-۵

۷۰	فصل ششم: نتیجه‌گیری و پیشنهادات آتی	
۷۱	نتیجه‌گیری	۱-۶
۷۲	پیشنهادات آتی	۲-۶
۷۲	پژوهش‌های تکمیلی	۱-۲-۶
۷۳	پیشنهادات اجرایی	۲-۲-۶
۷۳	تولید داده‌های جدید	۳-۲-۶
۷۳	تحلیل‌های آینده	۴-۲-۶

۷۶ مراجع

۸۱	پیوست	
۸۲	پیوست A: کدهای پیاده‌سازی مدل MAGNET	۱-
۸۲	کد معماری مدل MAGNET	۱-۱-
۸۳	کد بهیته‌سازی با PIRATES	۲-۱-
۸۴	پیوست B: داده‌های خام و پیش‌پردازش	۲-
۸۴	نمونه داده‌های خام DREBIN	۱-۲-
۸۵	توضیحات پیش‌پردازش	۲-۲-
۸۵	پیوست C: جزئیات سخت‌افزاری و نرم‌افزاری	۳-

۸۵	مشخصات سخت افزاری	۱-۳-
۸۵	مشخصات نرم افزاری	۲-۳-
۸۶	پیوست D: نتایج اضافی و ماتریس های کامل	۴-
۸۶	ماتریس درهم ریختگی کامل	۱-۴-
۸۶	گزارش طبقه بندی برای هر دسته	۲-۴-

فهرست جداول

عنوان	صفحة
جدول ۱-۵ نتایج اعتبارسنجی متقطع ۵-تایی مدل MAGNET	۶۱
جدول ۲-۵ مقایسه کلی عملکرد مدل MAGNET در مراحل مختلف	۶۱
جدول ۳-۵ مقایسه عملکرد مدل MAGNET با روش‌های پایه و پیشرفته	۶۵
جدول ۴-۵ مقایسه عملکرد مدل MAGNET با مدل‌های پایه	۶۶
جدول ۱ نمونه‌ای از داده‌های خام دیتابیس DREBIN	۸۴
جدول ۲ ماتریس درهم‌ریختگی برای مجموعه تست	۸۶
جدول ۳ گزارش طبقه‌بندی برای هر دسته در اعتبارسنجی متقطع	۸۶

فهرست اشکال

عنوان	صفحة
شکل ۲-۱ ساختار کلی شبکه عصبی کانولوشنی (CNN) شامل لایه‌های کانولوشنی، تجمعی و کاملاً متصل برای طبقه‌بندی. برگرفته از [۹]	۱۶
شکل ۲-۲ ساختار شبکه عصبی گرافی (GNN) برای پردازش داده‌های گرافی و طبقه‌بندی. برگرفته از [۱۲]	۱۸
شکل ۲-۳ معماری ترانسفورمر شامل کدگذار و گشاينده با مکانيزم توجه چندسر. برگرفته از [۱۵]	۲۰
شکل ۱-۴ معماری کلی مدل MAGNET	۴۳
شکل ۱-۵ نمایش اجزای الگوریتم PIRATES در فضای جستجو.	۵۳
شکل ۲-۵ روند همگرایی الگوریتم PIRATES.	۵۴
شکل ۳-۵ تغییرات پارامترهای الگوریتم PIRATES در طول تکرارها.	۵۵
شکل ۴-۵ نتایج اعتبارسنجی متقطع-۵-تایی مدل MAGNET.	۶۱
شکل ۵-۵ تغییرات زیان در اعتبارسنجی متقطع-۵-تایی.	۶۲
شکل ۵-۶ مقایسه عملکرد مدل MAGNET در مراحل مختلف.	۶۲
شکل ۵-۷ مقایسه دقیق روش‌های مختلف.	۶۶
شکل ۸-۵ مقایسه F1 Score و AUC روش‌های مختلف.	۶۶
شکل ۹-۵ مقایسه F1 Score و AUC مدل MAGNET با سایر مدل‌ها.	۶۷
شکل ۱۰-۵ عملکرد ماثولهای مختلف مدل MAGNET.	۶۸
شکل ۱۱-۵ تأثیر مکانیزم توجه پویا و لایه ادغام چندوجهی بر عملکرد.	۶۹
شکل ۱۲-۵ روند آموزش مدل در طول ۳ دوره.	۶۹

فهرست الگوریتم‌ها

صفحة	عنوان
٤٤	الgoritم ١ - Structure Module EnhancedTabTransformer
٤٤	الgoritم ٢ - Structure Module GraphTransformer
٤٥	الgoritم ٣ - Structure Module SequenceTransformer
٤٥	الgoritم ٤ - Mechanism Attention Dynamic
٤٦	الgoritم ٥ - Fusion Modality
٤٦	الgoritم ٦ - Model MAGNET Final

فصل اول:

کلیات پژوهش

۱-۱ مقدمه و بیان مسئله

در سال‌های اخیر، گسترش تلفن‌های همراه و بهویژه سیستم عامل اندروید^۱، موجب افزایش وابستگی کاربران به این ابزارها شده است. این دستگاه‌ها نه تنها در زندگی روزمره، بلکه در حوزه‌های تجاری و نظامی نیز نقش مهمی ایفا می‌کنند. با این حال، محبوبیت و فراگیری اندروید، آن را به هدفی جذاب برای حملات بدافزاری^۲ تبدیل کرده است. عرضه نرم‌افزارهای غیرمعتبر و تهدیداتی مانند ویروس‌ها و بدافزارها، امنیت کاربران را به خطر انداخته است. مطالعات اخیر نشان می‌دهد که بیش از 70 درصد دستگاه‌های هوشمند از سیستم عامل اندروید استفاده می‌کنند و این امر باعث شده است که این پلتفرم به هدف اصلی حملات امنیتی تبدیل شود [۱]. با وجود پیشرفت‌های قابل توجه در روش‌های تشخیص بدافزار، همچنان چالش‌های جدی در شناسایی بدافزارهای جدید و پیچیده وجود دارد.

در ابتدا، روش‌های سنتی مبتنی بر تحلیل مجوزها^۳ و بازکردن فایل‌ها مورد استفاده قرار می‌گرفتند که به دلیل دقت پایین و ضعف در شناسایی بدافزارهای پیچیده، محدودیت‌هایی داشتند. پژوهش‌های اخیر نشان داده‌اند که روش‌های مبتنی بر یادگیری ماشین^۴ و یادگیری عمیق^۵ می‌توانند عملکرد بهتری در تشخیص بدافزارها داشته باشند [۲]. با این حال، همچنان چالش‌های مهمی در زمینه تفسیرپذیری مدل‌ها^۶ و قابلیت تعمیم‌پذیری^۷ وجود دارد. این چالش‌ها به ویژه در مواجهه با بدافزارهای جدید و ناشناخته^۸ بیشتر خود را نشان می‌دهند.

مدل MAGNET^۹ که در این پژوهش معرفی شده است، با بهره‌گیری از معماری ترانسفورمر^{۱۰} چندوجهی و ترکیب داده‌های جدولی، گراف و توالي، تلاش می‌کند تا این چالش‌ها را برطرف کند. این مدل با استفاده از مکانیزم‌های توجه پویا^{۱۱} و تحلیل همزمان داده‌های مختلف، قادر به تشخیص دقیق‌تر بدافزارها خواهد بود. نتایج نشان می‌دهد که این رویکرد با دقت %97.24 ± 0.5%， معیار $F1 = 0.9823 \pm 0.002$ ، و معیار AUC = 0.9932 ± 0.003 (دقت% 90.6%)، CNN (دقت% 90.6%) و LSTM (دقت% 91.5%) دارد.

¹Android

²Malware

³Permissions

⁴Machine Learning

⁵Deep Learning

⁶Model Interpretability

⁷Generalization

⁸Zero-Day

⁹MAGNET(Multi-Modal Analysis for Graph and Network Threat Detection)

¹⁰Transformer

¹¹Attention Mechanism

۱-۱-۱ روش‌های تشخیص بدافزار

تشخیص بدافزارهای اندرویدی به دو روش کلی پویا^۱ و ایستا^۲ انجام می‌شود. در روش پویا، رفتار اپلیکیشن در زمان اجرا مانند مصرف باتری، پردازنده یا ترافیک شبکه بررسی می‌شود تا الگوهای غیرعادی شناسایی گردد. این روش بهتایی کافی نیست و ممکن است برخی تهدیدات پنهان را نادیده بگیرد. روش ایستا با تحلیل ساختار و کد اپلیکیشن، مانند بررسی فراخوانی‌های API^۳ و مجوزها، اطلاعات ارزشمندی ارائه می‌دهد که می‌تواند در تشخیص دقیق‌تر کمک کند. پژوهش‌های اخیر نشان داده‌اند که ترکیب این دو روش می‌تواند نتایج بهتری در تشخیص بدافزارها ارائه دهد [۳].

۲-۱-۱ مجموعه داده‌های مربوطه

در حوزه تشخیص بدافزار اندروید، مجموعه داده‌های متنوعی برای ارزیابی عملکرد مدل‌ها مورد استفاده قرار گرفته‌اند. از جمله این مجموعه داده‌ها می‌توان به موارد زیر اشاره کرد:

- مجموعه داده‌های Drebin [۴] و AndroZoo [۵] که شامل نمونه‌های گسترده‌ای از بدافزارها و برنامه‌های سالم اندرویدی هستند.
- مجموعه داده‌های VirusShare [۶] و CICMalDroid که شامل نمونه‌های جدید و به روز از بدافزارها می‌باشند.
- مجموعه داده‌های خصوصی و صنعتی که توسط شرکت‌های امنیتی و مراکز تحقیقاتی گردآوری شده‌اند.

این مجموعه داده‌ها به عنوان شاخص‌های استاندارد، امکان ارزیابی دقیق و جامع عملکرد الگوریتم‌های تشخیص بدافزار را فراهم می‌کنند و نقش مهمی در اثبات قابلیت تعمیم و کارایی روش‌های پیشنهادی دارند.

۱-۲ ضرورت تحقیق و اهداف

پلتفرم اندروید به دلیل محبوبیت گسترده و سهم عظیمش از بازار جهانی، به هدف اصلی بدافزارها و حملات امنیتی تبدیل شده است. این سیستم‌عامل، که بیش از 70 درصد دستگاه‌های هوشمند را پشتیبانی می‌کند، به دلیل ساختار باز و دسترسی‌پذیری بالا، با تهدیدات پیشرفته‌ای مواجه است. بدافزارهای اندرویدی،

¹Dynamic Analysis

²Static Analysis

³API

از جمله تروجان‌ها^۱، جاسوس‌افزارها^۲ و باج‌افزارها^۳، با روش‌های پیچیده‌ای طراحی شده‌اند و پیشرفت‌های چشمگیری داشته‌اند. این تهدیدات، از سرقت اطلاعات حساس گرفته تا ایجاد اختلال در عملکرد دستگاه‌ها، چالش‌های امنیتی جدی ایجاد کرده‌اند. از این رو، نیاز به سیستمی قدرتمند و کارآمد برای تشخیص بدافزارهای اندرویدی بیش از پیش احساس می‌شود. هدف اصلی این پژوهش، تمرکز بر شناسایی بدافزارهای ناشناخته و نادیده^۴ است که تا کنون شناسایی نشده‌اند و می‌توانند تهدیداتی پنهان برای کاربران ایجاد کنند.

با توجه به چالش‌ها و نیازهای مطرح شده، اهداف اصلی این تحقیق بدین شرح می‌باشد:

- توسعه یک مدل چندوجهی پیشرفته با نام MAGNET که قادر به تحلیل همزمان داده‌های جدولی، گرافی و ترتیبی باشد.
- بهبود دقیق تشخیص بدافزارهای اندرویدی با استفاده از معماری ترانسفورمر و مکانیزم‌های توجه پویا^۵.
- کاهش نرخ خطای تشخیص و افزایش قابلیت تعیین‌پذیری مدل در مواجهه با بدافزارهای جدید.
- بهینه‌سازی مصرف منابع محاسباتی و افزایش سرعت تشخیص با استفاده از الگوریتم‌های پیشرفته.
- ایجاد یک چارچوب استاندارد برای ارزیابی و مقایسه روش‌های مختلف تشخیص بدافزار.

۱-۳ سازماندهی پایان نامه

در این پایان‌نامه، ساختار مطالب به گونه‌ای تدوین شده که مسیر پژوهش از مبانی نظری و معرفی مسئله تا ارائه نتایج تجربی به صورت پیوسته و منطقی دنبال شود. به عبارت دیگر، هدف از سازماندهی مطالب این است که خواننده بتواند به راحتی با مباحث پایه، چالش‌ها، روش‌های موجود و نوآوری‌های پیشنهادی آشنا شود و در نهایت به درک جامع از دستاوردهای تحقیق دست یابد. ساختار کلی پایان‌نامه به شرح زیر است:

- فصل 2 - پیشینه تحقیق و مفاهیم پایه:
- در این فصل، ابتدا به بررسی کلی امنیت اندروید و اهمیت تشخیص بدافزار پرداخته می‌شود. سپس، چالش‌ها و محدودیت‌های روش‌های سنتی بیان شده و مسئله تحقیق به تفصیل معرفی می‌شود. هدف این فصل ایجاد زمینه نظری مناسب برای درک اهمیت تشخیص خودکار بدافزارهای است.

¹Trojan

²Spyware

³Ransomware

⁴Zero-Day

⁵Dynamic Attention Mechanism

در ادامه به بررسی جامع مطالعات پیشین در حوزه تشخیص بدافزار اندرودید پرداخته می‌شود. در این بخش، رویکردهای مختلف از جمله روش‌های مبتنی بر یادگیری ماشین و یادگیری عمیق مورد تحلیل قرار می‌گیرند. نقاط قوت و ضعف هر یک از این رویکردها همراه با چالش‌های موجود در هر کدام به تفصیل بررسی می‌شود.

• فصل 3 – روش پیشنهادی (MAGNET):

در این فصل، مدل پیشنهادی MAGNET به صورت کامل تشریح می‌شود. ابتدا معماری کلی مدل و اجزای اصلی آن معرفی می‌شوند. سپس، جزئیات پیاده‌سازی و الگوریتم‌های بهینه‌سازی مورد استفاده توضیح داده می‌شود. در نهایت، نوآوری‌های اصلی این روش نسبت به سایر روش‌ها برجسته می‌شود.

• فصل 4 – نتایج و بحث:

این فصل به ارائه نتایج آزمایش‌های انجام شده بر روی چندین مجموعه داده معتبر اختصاص دارد. عملکرد مدل MAGNET از نظر دقت، کارایی و صرفه‌جویی در منابع محاسباتی مورد مقایسه قرار گرفته و نتایج به دست آمده تحلیل می‌شوند.

• فصل 5 – نتیجه‌گیری و پیشنهادات آتی:

در فصل نهایی، یافته‌های اصلی تحقیق به طور خلاصه ارائه شده و به نتیجه‌گیری کلی از دستاوردهای پژوهش پرداخته می‌شود. در این بخش، چالش‌های باقی‌مانده، محدودیت‌های تحقیق و نیز پیشنهاداتی جهت تحقیقات آتی و بهبود رویکرد ارائه می‌شود.

فصل دوم:

پیشینه تحقیق و مفاهیم پایه

در سال‌های اخیر، با گسترش روزافزون استفاده از دستگاه‌های هوشمند مبتنی بر سیستم عامل اندروید، امنیت سایبری به یکی از دغدغه‌های اصلی کاربران و سازمان‌ها تبدیل شده است. محبوبیت گسترده اندروید، که طبق آمار Statista در سال ۲۰۲۴ بیش از ۷۰ درصد بازار جهانی دستگاه‌های هوشمند را در اختیار دارد، آن را به هدف اصلی حملات بدافزاری، از جمله تروجان‌ها^۱، جاسوس‌افزارها^۲ و باج‌افزارها^۳، مبدل ساخته است [۱]. این تهدیدات، با بهره‌گیری از روش‌های پیچیده و نوظهور، چالش‌های جدی در شناسایی و خنثی‌سازی ایجاد کرده‌اند، بهویژه در مورد بدافزارهای ناشناخته (Zero-Day)^۴ که تشخیص آن‌ها با روش‌های سنتی مبتنی بر امضا دشوار است [۲]. از این‌رو، توسعه مدل‌های پیشرفته مبتنی بر یادگیری عمیق و داده‌های چندوجهی، به عنوان راهکاری نوین برای مقابله با این تهدیدات، مورد توجه فزاینده‌ای قرار گرفته است [۲].

فصل حاضر به عنوان پایه‌ای برای درک روش پیشنهادی این پژوهش، به بررسی مفاهیم پایه و مرور مطالعات پیشین در حوزه تشخیص بدافزارهای اندرویدی می‌پردازد. ابتدا، مفاهیم کلیدی و اصطلاحات فنی مورد نیاز معرفی خواهند شد. سپس، تکنیک‌های رایج یادگیری ماشین و یادگیری عمیق که در این حوزه کاربرد دارند، تشریح می‌شوند. در ادامه، ابزارها و روش‌های بهینه‌سازی مانند ترانسفورمرها^۵، الگوریتم‌های PIRATES و Optuna^۶، که در طراحی مدل MAGNET (روش پیشنهادی این پژوهش) به کار رفته‌اند، معرفی خواهند شد. همچنین، تکنیک‌های آموزشی و معیارهای ارزیابی عملکرد، نظیر اعتبارسنجی متقطع^۷ و معیارهای دقت^۸ AUCF1 Score^۹، به منظور تبیین چارچوب ارزیابی مدل توضیح داده می‌شوند.

در بخش بعدی، پیشینه تحقیق با تمرکز بر روش‌های تحلیل ایستا^{۱۰} و پویا^{۱۱}، که پایه‌های اصلی تشخیص بدافزار را تشکیل می‌دهند، مورد بررسی قرار می‌گیرد. این مرور، با تحلیل نقاط قوت و ضعف روش‌های پیشین، زمینه‌ای برای درک نوآوری‌های مدل پیشنهادی فراهم می‌کند. هدف این فصل، ارائه دیدگاهی جامع و منسجم از مفاهیم و مطالعات مرتبط است تا خواننده را برای درک بهتر روش‌شناسی و نتایج پژوهش آماده سازد. این ساختار، نه تنها به تبیین مسائل نظری کمک می‌کند، بلکه راه را برای مقایسه و ارزیابی عملکرد

¹Trojan²Spyware³Ransomware⁴Zero-Day⁵Transformer⁶Optuna⁷Cross-Validation⁸Accuracy⁹Area Under Curve (AUC)¹⁰Static Analysis¹¹Dynamic Analysis

۲-۲ مفاهیم پایه

در این بخش، مفاهیم بنیادی مرتبط با موضوع پژوهش، از جمله تاریخچه مختصراًی از یادگیری ماشین، انواع بدافزارها، ساختار اپلیکیشن‌های اندروید، داده‌های چندوجهی و مدل‌های یادگیری عمیق مرتبط، تشریح می‌شوند.

۱-۲-۲ تکامل روش‌های یادگیری ماشین

در دهه‌های ۱۹۵۰ و ۱۹۶۰، اولین تلاش‌ها برای پردازش زبان طبیعی به وسیله‌ی روش‌های نمادین انجام شد. پژوهشگران در آن زمان سعی می‌کردند ساختارهای دستوری و قوانین زبان را به صورت صریح و دستی تعریف کنند. این رویکردها با وجود تلاش‌های ارزشمند، به دلیل محدودیت‌های محاسباتی و عدم وجود داده‌های کافی، نتوانستند به دقت و کارایی مورد انتظار دست یابند.

با گذر زمان و ورود به دهه‌های ۱۹۶۰ و ۱۹۷۰، رویکردهای آماری جایگزین بخش‌هایی از روش‌های نمادین شدند. در این دوران، مدل‌های n-gram که بر مبنای احتمال وقوع یک کلمه با توجه به کلمات قبلی محاسبه می‌شدند، به عنوان اولین قدم‌های موفق در مدل‌سازی زبان مطرح شدند. اگرچه این مدل‌ها ساده بودند، اما توانستند برخی از پیچیدگی‌های اولیه‌ی پردازش زبان را کاهش دهند.

در دهه‌های ۱۹۸۰ و ۱۹۹۰، با پیشرفت‌های چشمگیر در فناوری‌های محاسباتی و افزایش دسترسی به داده‌های متنی، روش‌های یادگیری ماشین وارد عرصه شدند. الگوریتم‌های یادگیری نظارت شده و غیرنظارتی به منظور تشخیص الگوهای زبانی به کار گرفته شدند. با این حال، محدودیت‌های موجود همچنان مانع از دستیابی به درک عمیق‌تر و تولید متن‌های طبیعی به سطح امروزی می‌شدند.

ورود به قرن ۲۱ و بهویژه دهه ۲۰۱۰، با ظهور شبکه‌های عصبی عمیق مانند شبکه‌های عصبی کانولوشنی^۱، شبکه‌های عصبی بازگشتی^۲ و شبکه‌های حافظه بلندمدت - کوتاه‌مدت^۳ همراه بود. این مدل‌ها توانستند وابستگی‌های زمانی، الگوهای تصویری و روابط بلندمدت موجود در داده‌ها را بهتر مدل‌سازی کنند^[۴]. با این حال، چالش‌هایی همچنان در زمینه بهبود کیفیت و کارایی تحلیل داده‌های پیچیده مانند کدهای بدافزار وجود داشت که منجر به توسعه معماری‌های پیشرفته‌تری مانند ترانسفورمرها شد.

¹Convolutional Neural Networks (CNNs)

²Recursive Neural Networks (RNNs)

³Long Short-Term Memory (LSTM)

۲-۲-۲ انواع بدافزار

بدافزار (Malware) اصطلاحی کلی برای هر نوع نرم‌افزار مخرب است که با هدف آسیب رساندن به سیستم‌ها، سرقت اطلاعات یا اختلال در عملکرد طراحی می‌شود. در ادامه، برخی از رایج‌ترین انواع بدافزارها معرفی می‌شوند:

۱-۲-۲-۲ باج‌افزار

باج‌افزار^۱ گونه‌ای از بدافزارها است که با سرعت زیادی گسترش یافته و امروزه به شکل فراگیری در حال شیوع است. باج‌افزارها عمدهاً به دو نوع قفل‌کننده (کریپتور)^۲ و مسدودکننده^۳ تقسیم می‌شوند. پس از آلوده‌سازی رایانه، نوع قفل‌کننده داده‌های ارزشمند کاربر از جمله استاد، تصاویر و پایگاه‌های داده را رمزنگاری می‌کند و تا زمانی که رمزگشایی نشوند، هیچ فایلی قابل دسترسی نخواهد بود. مهاجمان در ازای ارائه کلید رمزگشا برای بازگرداندن فایل‌های قفل‌شده، درخواست باج می‌کنند. نوع مسدودکننده نیز دسترسی به کل سیستم آلوده را مسدود می‌کند و معمولاً میزان باج‌خواهی آن کمتر از نوع قفل‌کننده است.

باج‌افزارها می‌توانند سیستم‌عامل‌های مختلفی مانند ویندوز، مک‌اواس^۴، لینوکس و اندروید را هدف قرار دهند و هم رایانه‌های دسکتاپ و هم دستگاه‌های موبایل را آلوده سازند [۱]. آلودگی معمولاً از طریق باز کردن فایل‌های ضمیمه مخرب، کلیک روی لینک‌های مشکوک یا نصب برنامه‌ها از منابع غیررسمی رخ می‌دهد. حتی وبسایت‌های رسمی نیز گاهی به واسطه شبکه‌های تبلیغاتی آلوده می‌شوند. حذف باج‌افزارها معمولاً دشوار است و در صورت رمزنگاری فایل‌ها، کاربر باید برای بازیابی دسترسی، رمزگشایی انجام دهد که پرداخت باج نیز تضمینی برای بازگشت اطلاعات نیست و توصیه نمی‌شود.

۲-۲-۲-۲ تروجان

تروجان^۵ یا اسب تروآ، نوعی برنامه مخرب است که خود را به عنوان نرم‌افزاری مشروع و بی‌خطر جلوه می‌دهد تا کاربر را فریب داده و به سیستم نفوذ کند. نام‌گذاری این بدافزار برگرفته از داستان اسب چوبی تروای یونانیان است که با حیله وارد شهر شد. انواع مختلفی از تروجان‌ها وجود دارد، از جمله:

- **تروجان دسترسی از راه دور (Backdoor):** به مهاجم امکان کنترل سیستم قربانی از راه دور را می‌دهد.

¹Ransomware

²Cryptor

³Locker

⁴Mac OS X

⁵Trojan Horse

- **تروجان ارسال داده:** اطلاعات حساس مانند رمزهای عبور و داده‌های واردشده توسط کاربر را به مهاجم ارسال می‌کند.
- **تروجان مخرب:** فایل‌های سیستمی را حذف یا تخریب می‌کند.
- **تروجان DoS:** باعث کاهش سرعت سیستم و اینترنت کاربر می‌شود (حملات منع سرویس).
- **تروجان پراکسی:** به مهاجم اجازه می‌دهد از طریق سیستم قریانی به سرورهای پراکسی حمله کند.
- **تروجان FTP:** پورت FTP را باز کرده و کنترل سیستم را از این طریق به مهاجم می‌دهد.
- **تروجان غیرفعال‌سازی امنیت:** نرمافزارهای امنیتی را غیرفعال می‌کند تا حمله آسان‌تر انجام شود.

۳-۲-۲-۲ جاسوس‌افزار

جاسوس‌افزار^۱ نوعی نرمافزار مخرب است که بدون اطلاع کاربر، اطلاعاتی درباره او یا فعالیت‌هایش را جمع‌آوری و برای مهاجم ارسال می‌کند. کاربردهای رایج جاسوس‌افزارها شامل تبلیغات هدفمند، جمع‌آوری اطلاعات شخصی^۲ و ایجاد تغییرات ناخواسته در تنظیمات سیستم قریانی است. این بدافزارها می‌توانند باعث کاهش کارایی سیستم، نصب نوارابزارهای ناخواسته، تغییر صفحه خانگی مرورگر و باز شدن صفحات پاپ‌آپ تبلیغاتی شوند.

۴-۲-۲-۲ تبلیغ‌افزار

تبلیغ‌افزار^۳ نرمافزاری است که با هدف نمایش تبلیغات ناخواسته روی سیستم قریانی طراحی شده است. این تبلیغات معمولاً^۴ به صورت بنرها یا صفحات پاپ‌آپ ظاهر می‌شوند و گاهی صفحات اینترنتی متعددی را به طور متوالی باز می‌کنند. اگرچه همه تبلیغ‌افزارها ذاتاً مخرب نیستند^۴، اما انواع تهاجمی آن‌ها می‌توانند فعالیت‌های آنلاین کاربر را دنبال کرده و تجربه کاربری را مختل کنند. همچنین، برخی تبلیغ‌افزارها اطلاعات مربوط به رفتار کاربر را جمع‌آوری و برای اهداف تبلیغاتی یا حتی فروش به اشخاص ثالث ارسال می‌کنند که می‌تواند حریم خصوصی را نقض کند.

¹Spyware

²عبور رمزهای یا بانکی اطلاعات مانند

³Adware

⁴می‌کنند در آمدزایی تبلیغ نمایش طریق از رایگان نرم‌افزارهای برخی

۳-۲-۲ مفاهیم مرتبط با اپلیکیشن‌های اندرویدی

۱-۳-۲-۲ مجوزهای دسترسی

اندروید از یک سیستم تخصیص مجوز^۱ برخوردار است که برای هر عملیات یا وظیفه^۲ حساس، مجوزها و سطوح دسترسی خاصی را تعیین می‌کند. هر اپلیکیشن می‌تواند برای انجام عملیات خاص، مجوزهای لازم را از این سامانه درخواست نماید. به عنوان مثال، یک برنامه کاربردی ممکن است برای دسترسی به اینترنت، دوربین، یا لیست مخاطبین، مجوزهای مربوطه را درخواست کند. این سیستم دارای سطوح مختلفی برای مجوزها است که به آن‌ها سطح حفاظت^۳ گفته می‌شود.

در نسخه‌های قدیمی‌تر اندروید (قبل از نسخه ۶.۰)، اپلیکیشن‌ها هنگام نصب، لیست تمامی مجوزهای مورد نیاز خود را به کاربر اعلام می‌کردند و کاربر یا همه را می‌پذیرفت یا نصب را لغو می‌کرد. اما از نسخه ۶.۰ اندروید (۴، API سطح ۲۳) به بعد، مدل مجوزدهی پویا^۵ معرفی شد که طی آن، مجوزهای حساس تنها در زمان اجرای برنامه و هنگامی که اپلیکیشن برای اولین بار به آن قابلیت نیاز دارد، از کاربر درخواست می‌شوند. اپلیکیشن‌های اندرویدی مجوزهای مورد نیاز خود را در فایل مانیفست اندروید^۶ اعلام می‌کنند. همچنین، اپلیکیشن می‌تواند مجوزهای اضافی را در این فایل تعریف کند که دسترسی به برخی اجزای داخلی نرم‌افزار توسط سایر برنامه‌ها را محدود می‌سازد.

در اندروید، مجوزها عمدهاً به دو سطح دسترسی و امنیت تقسیم می‌شوند:

• **معمولی (Normal)**: مجوزهایی که ریسک پایینی برای حریم خصوصی کاربر یا عملکرد سایر اپلیکیشن‌ها دارند (مانند مجوز دسترسی به اینترنت یا تنظیم منطقه زمانی). این مجوزها معمولاً به صورت خودکار هنگام نصب یا بهروزرسانی به اپلیکیشن اعطا می‌شوند و نیازی به تأیید صریح کاربر ندارند.

• **خطروناک (Dangerous)**: مجوزهایی که به اطلاعات خصوصی کاربر^۷ دسترسی دارند یا می‌توانند بر عملکرد دستگاه یا سایر اپلیکیشن‌ها تأثیر بگذارند^۸. این مجوزها باید حتماً در زمان اجرا و به صورت صریح از کاربر درخواست شوند.

تحلیل الگوهای درخواست مجوز یکی از روش‌های مهم در تشخیص بدافزارهای اندرویدی است، زیرا

¹Permission System

²Task

³Protection Level

⁴Marshmallow

⁵Runtime Permissions

⁶AndroidManifest.xml

پیامک‌ها مکانی، موقعیت مخاطبین، مانند⁷

میکروفون یا دوربین به دسترسی مانند⁸

بدافزارها اغلب مجوزهای خطرناک بیشتری نسبت به کاربرد ظاهری خود درخواست می‌کنند [۴].

۲-۳-۲-۲ فایل APK

فرمت APK^۱ مخفف عبارت^۲ است و فرمت فایل بسته‌بندی شده‌ای است که برای توزیع و نصب برنامه‌ها و میان‌افزارها در سیستم‌عامل اندروید استفاده می‌شود. فایل APK شامل تمام کدهای برنامه (مانند فایل‌های .dex)، منابع (مانند تصاویر و لایه‌ها)، دارایی‌ها، گواهی‌نامه‌ها و فایل مانیفست^۳ است. این فایل بسیار شبیه به فرمت‌های بسته‌بندی دیگر مانند APPX در مایکروسافت ویندوز یا deb. در سیستم‌های مبتنی بر دبيان^۴ عمل می‌کند.

برای ساخت یک فایل APK، ابتدا برنامه اندروید با استفاده از ابزارهایی مانند اندروید استودیو کامپایل شده و سپس تمام اجزای آن در یک فایل فشرده با پسوند apk. بسته‌بندی می‌شود. فایل‌های APK پایه و اساس برنامه‌های اندرویدی هستند و بخش اصلی اکوسیستم اندروید محسوب می‌شوند. کاربران می‌توانند فایل‌های APK را از منابع مختلف، از جمله فروشگاه رسمی گوگل پلی^۵ یا به صورت دستی^۶ از وبسایت‌ها یا منابع دیگر، دانلود و نصب کنند. نصب دستی برنامه‌ها از منابع نامعتبر یکی از راههای اصلی ورود بدافزار به دستگاه‌های اندرویدی است.

علاوه بر سیستم‌عامل اندروید، فایل‌های APK را می‌توان با استفاده از شبیه‌سازها یا لایه‌های سازگاری در سایر سیستم‌عامل‌ها مانند ویندوز، مک‌اواس یا کروم اواس نیز اجرا، و با ابزارهای مهندسی معکوس، محتوای آن‌ها را استخراج، ویرایش یا تحلیل کرد.

۳-۳-۲-۲ سورس کد

سورس کد^۷ یا کد منبع برنامه، به مجموعه‌ای از دستورالعمل‌ها یا عبارات متنی اشاره دارد که توسط برنامه‌نویس به یک زبان برنامه‌نویسی خاص (مانند جاوا، کاتلین، C++, پایتون) نوشته می‌شود تا یک برنامه کامپیوتری، عملکرد مورد نظر را پیاده‌سازی کند. سورس کد برای انسان قابل خواندن است و منطق، الگوریتم‌ها و ساختار برنامه را تعریف می‌کند.

برای اینکه کامپیوتر بتواند سورس کد را اجرا کند، باید ابتدا به زبان ماشین (کد باینری) ترجمه شود. این فرآیند توسط یک کامپایلر یا مفسر انجام می‌شود. در اکوسیستم اندروید، کدهای جاوا یا کاتلین معمولاً^۸

¹ Android Package Kit

² Android Package Kit

³ AndroidManifest.xml

⁴ اوبونتو مانند

⁵ Google Play

⁶ Sideload

⁷ Source Code

به بایت کد^۱ (در نسخهای قدیمی‌تر) یا ART^۲ کامپایل می‌شوند که در فایل‌های .dex درون فایل APK قرار می‌گیرند.

دسترسی به سورس کد یک برنامه امکان درک کامل عملکرد، اصلاح یا توسعه آن را فراهم می‌کند. در حوزه امنیت، تحلیل سورس کد (در صورت در دسترس بودن) یکی از روش‌های تحلیل ایستا برای یافتن آسیب‌پذیری‌ها یا کدهای مخرب است. با این حال، اکثر برنامه‌های تجاری یا بدافزارها به صورت کامپایل شده و بدون سورس کد توزیع می‌شوند و تحلیل آن‌ها نیازمند مهندسی معکوس^۳ است.

۴-۲-۲ داده‌های چندوجهی در تشخیص بدافزار

داده‌های چندوجهی^۴ به مجموعه‌ای از اطلاعات اشاره دارد که از منابع و قالب‌های متنوعی استخراج شده و برای تحلیل جامع‌تریک پدیده، مانند تشخیص بدافزارهای اندرویدی، به کار می‌رond. در این پژوهش، مدل MAGNET با بهره‌گیری از سه نوع اصلی داده‌های چندوجهی، شامل داده‌های جدولی، گرافی و ترتیبی، طراحی شده است تا بتواند ویژگی‌های متنوع و مکمل اپلیکیشن‌های اندرویدی را به صورت یکپارچه پردازش کند. این رویکرد، برخلاف روش‌های سنتی که تنها به یک نوع داده (مثلًا فقط مجوزها یا فقط فرآخوانی‌های API) وابسته‌اند، امکان درک عمیق‌تر رفتارهای مخرب را فراهم می‌سازد. در ادامه، هر یک از این انواع داده‌ها به طور جداگانه توضیح داده می‌شود.

۱-۴-۲-۲ داده‌های جدولی

داده‌های جدولی شامل ویژگی‌های ایستای اپلیکیشن‌های اندرویدی هستند که به صورت ساختارمند و معمولاً عددی یا دسته‌ای ارائه می‌شوند. این ویژگی‌ها می‌توانند شامل تعداد مجوزهای درخواست شده، اندازه فایل APK، نسخه SDK هدف، تعداد فعالیت‌ها^۵، سرویس‌ها^۶، گیرنده‌های پخش^۷، ارائه‌دهنده‌گان محتوا^۸، تعداد فرآخوانی‌های API خاص، یا سایر متغیرهای استخراج شده از فایل^۹ یا کد کامپایل شده باشند. در این پژوهش، این داده‌ها با استفاده از کتابخانه^{۱۰} و به صورت اشیای تانسور^{۱۱} بارگذاری و پردازش می‌شوند. برای هر نمونه اپلیکیشن، یک بردار ویژگی با ابعاد مشخص تعریف می‌شود که نشان‌دهنده خصوصیات

¹Dalvik

²(Android Runtime)

³بایت‌کد یا باینری کدهای کردن دیس‌اس‌بل مانند

⁴Multi-Modal Data

⁵Activities

⁶Services

⁷Broadcast Receivers

⁸Content Providers

⁹AndroidManifest.xml

¹⁰PyTorch

¹¹torch.Tensor

ایستای آن است. پیش‌پردازش این داده‌ها، مانند نرم‌السازی (برای مقادیر عددی) یا کدگذاری وان‌هات (برای مقادیر دسته‌ای)، که با هدف بهبود عملکرد مدل انجام می‌شود، از مراحل کلیدی به شمار می‌رود. این نوع داده، پایه‌ای برای تحلیل‌های آماری و یادگیری الگوهای ساده در بدافزارها فراهم می‌کند (مثلاً بدافزارها ممکن است به طور متوسط مجوزهای بیشتری درخواست کنند).

۲-۴-۲-۲ داده‌های گرافی

داده‌های گرافی ساختارهایی هستند که روابط و تعاملات بین اجزای مختلف یک اپلیکیشن اندرویدی را مدل‌سازی می‌کنند. این داده‌ها می‌توانند نمایانگر گراف فراخوانی توابع^۱، گراف جریان کنترل^۲، گراف وابستگی داده‌ها^۳، یا گراف روابط بین اجزای برنامه (مانند ارتباط بین فعالیت‌ها از طریق^۴) باشند. گره‌ها^۵ در این گراف‌ها می‌توانند توابع، بلوک‌های کد، کلاس‌ها، مجوزها یا اجزای اندرویدی باشند و لبه‌ها نشان‌دهنده روابطی مانند فراخوانی تابع، انتقال کنترل، جریان داده یا درخواست مجوز هستند. در این پژوهش، از کتابخانه PyTorch Geometric (torch_geometric) برای بارگذاری و پردازش این داده‌ها به صورت اشیای داده (Data) استفاده شده است. هر گراف شامل ماتریس هم‌جواری (ساختار گراف) و ویژگی‌های گره‌ها (مانند نوع تابع یا بردار ویژگی‌های استخراج شده از کد) است. تحلیل گرافی به مدل اجازه می‌دهد تا الگوهای ساختاری پیچیده و وابستگی‌های پنهان بین اجزا را شناسایی کند (مانند شناسایی یک حلقه فراخوانی مشکوک یا یک خوشة از توابع مرتبط با سرقت اطلاعات)، که در تشخیص بدافزارهای پیچیده یا ناشناخته بسیار مؤثر است.

۳-۴-۲-۲ داده‌های ترتیبی

داده‌های ترتیبی شامل توالی‌هایی از رویدادها یا مقادیر هستند که معمولاً نمایانگر رفتار زمانی یا ترتیب وقوع عملیات در اپلیکیشن‌ها هستند. در زمینه تشخیص بدافزار اندروید، این داده‌ها می‌توانند شامل توالی فراخوانی‌های API سیستمی در حین اجرای برنامه (استخراج شده از تحلیل پویا)، توالی رویدادهای سیستمی (مانند ارسال پیامک، اتصال به شبکه)، یا الگوهای زمانی استفاده از منابع (مانند مصرف CPU یا باتری) باشند. در این پژوهش، این داده‌ها با استفاده از تانسورهای LongTensor در PyTorch بازگذاری شده و برای هر نمونه یک یا چند توالی مشخص تعریف می‌شود. پیش‌پردازش این داده‌ها ممکن است شامل نگاشت هر رویداد یا API به یک شناسه عددی، پدینگ^۶ توالی‌ها برای رسیدن به طول یکسان، و تبدیل به فرمت

¹Call Graph

²Control Flow Graph, CFG

³Data Dependency Graph

⁴Intents

⁵Nodes

⁶Padding

قابل پردازش توسط مدل‌های ترتیبی مانند RNN، LSTM یا ترنسفورمر باشد. استفاده از داده‌های ترتیبی به مدل امکان می‌دهد تا رفتار پویای اپلیکیشن‌ها را تحلیل کند و الگوهای زمانی مخرب، مانند توالی خاصی از فراخوانی‌ها که منجر به نشت داده می‌شود یا حملات دوره‌ای، را تشخیص دهد.

۴-۴-۲-۲ اهمیت و یکپارچگی داده‌های چندوجهی

ترکیب این سه نوع داده در مدل MAGNET، با استفاده از مکانیزم‌های توجه پویا و تکنیک‌های همجوشی (Fusion)، به شناسایی دقیق‌تر و جامع‌تر بدافزارها کمک می‌کند. داده‌های جدولی اطلاعات کلی و ایستا، داده‌های گرافی روابط ساختاری پیچیده، و داده‌های ترتیبی رفتار پویا را از ائمه می‌دهند که در کنار هم، تصویری کامل‌تر از اپلیکیشن را ترسیم می‌کنند. این رویکرد چندوجهی، به ویژه در مواجهه با بدافزارهای پیچیده و ناشناخته که ممکن است در یک وجه خاص (مثلاً فقط مجوزها) عادی به نظر برسند، مزیت رقابتی نسبت به روش‌های تک‌وجهی دارد و تعمیم‌پذیری مدل را افزایش می‌دهد [A]. استخراج این داده‌ها از فایل‌های پردازش شده با توابع فرضی load_sequence_data، load_graph_data، و load_processed_data، و انجام شده و پیش‌پردازش آنها با نرمال‌سازی، استانداردسازی و کدگذاری مناسب، تضمین‌کننده سازگاری با معماری‌های یادگیری عمیق مانند ترنسفورمر است.

۵-۲-۲ مدل‌های یادگیری عمیق

یادگیری عمیق، زیرشاخه‌ای از یادگیری ماشین، از شبکه‌های عصبی مصنوعی با چندین لایه (لایه‌های عمیق) برای یادگیری بازنمایی‌های پیچیده از داده‌ها استفاده می‌کند. در ادامه، چند معماری کلیدی که در تشخیص بدافزار کاربرد دارند، معرفی می‌شوند.

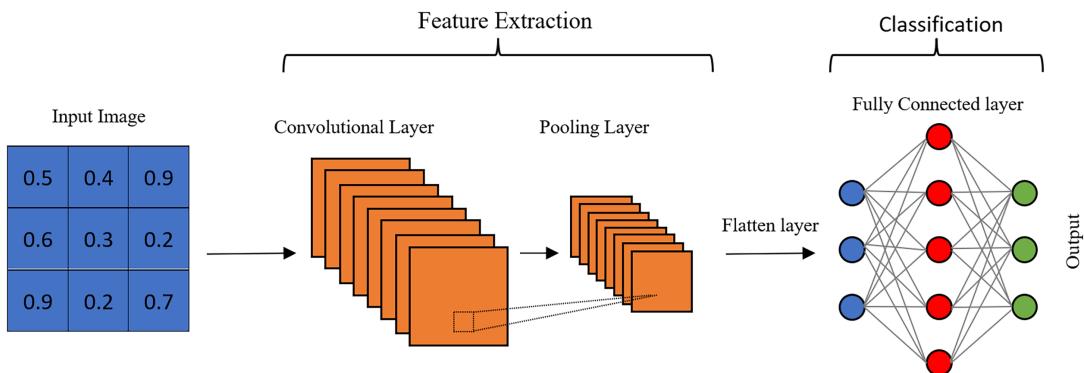
۱-۵-۲-۲ شبکه‌های عصبی کانولوشنی (CNN)

شبکه‌های عصبی کانولوشنی^۱ به طور خاص برای پردازش داده‌های با ساختار شبکه‌ای یا گریدمانند، مانند تصاویر، ماتریس‌های ویژگی یا داده‌های ترتیبی با پنجره لغزان، طراحی شده‌اند. این شبکه‌ها با بهره‌گیری از لایه‌های کانولوشنی که فیلترهای کوچکی را روی ورودی حرکت می‌دهند، قادرند ویژگی‌های محلی (مانند الگوهای خاص در بایت‌های کد، توالی‌های API، یا روابط بین مجوزها در یک ماتریس هم‌وقوعی) را به صورت سلسه‌مراتبی و خودکار استخراج کنند. لایه‌های تجمعی^۲ نیز با کاهش ابعاد مکانی داده و فشرده‌سازی اطلاعات، به افزایش پایداری نسبت به تغییرات جزئی و کاهش محاسبات کمک می‌کنند. در نهایت، لایه‌های کاملاً متصل^۳ از ویژگی‌های استخراج شده برای طبقه‌بندی نهایی (مثلاً بدافزار یا سالم)

¹Convolutional Neural Networks, CNNs

²Pooling

³Fully Connected



شکل ۱-۲ ساختار کلی شبکه عصبی کانولوشنی (CNN) شامل لایه‌های کانولوشنی، تجمعی و کاملاً متصل برای طبقه‌بندی. برگرفته از [۹].

استفاده می‌کنند. در حوزه تشخیص بدافزار، شبکه‌های^۱ می‌توانند برای تحلیل بایت‌های فایل اجرایی به عنوان تصویر، ماتریس‌های ویژگی‌های ایستا، یا توالی‌های رفتاری به کار روند و الگوهای مخرب را شناسایی نمایند [۷].

تحقیقات اخیر نشان داده است که شبکه‌های عصبی کانولوشنی می‌توانند دقیق‌تر باشند در تشخیص بدافزار ارائه دهنده و ویژگی‌های مورد نیاز را مستقیماً از داده‌های خام یا نیمه‌پردازش شده یاد بگیرند [۹]. ساختار این شبکه‌ها معمولاً به صورت انتها به انتها^۲ طراحی می‌شود و شامل پشته‌ای از لایه‌های کانولوشنی، فعال‌سازی (مانند ReLU)، و تجمعی است که به دنبال آن یک یا چند لایه کاملاً متصل قرار می‌گیرد. افزایش عمق شبکه (تعداد لایه‌ها) امکان استخراج ویژگی‌های انتزاعی‌تر و سطح بالاتر را فراهم می‌کند.

۲-۵-۲-۲ شبکه‌های عصبی بازگشتی و واحدهای حافظه بلند-کوتاه (RNN و LSTM)

شبکه‌های عصبی بازگشتی^۳ نوعی از شبکه‌های عصبی هستند که به طور ویژه برای پردازش داده‌های ترتیبی و زمانی طراحی شده‌اند. برخلاف شبکه‌های پیش‌خور^۴، RNN‌ها دارای حلقه‌های بازگشتی در اتصالات خود هستند که به آن‌ها اجازه می‌دهد اطلاعات مربوط به مراحل قبلی توالی را در یک حالت پنهان^۵ حفظ کرده و از آن برای پردازش مراحل بعدی استفاده کنند. این "حافظه" داخلی، RNN‌ها را برای کاربردهایی مانند تحلیل توالی فراخوانی‌های API، پردازش زبان طبیعی (مانند تحلیل کدهای اس‌مبلی یا توضیحات برنامه)، و تشخیص رفتارهای پویای اپلیکیشن‌ها که در طول زمان آشکار می‌شوند، مناسب می‌سازد.

یکی از چالش‌های اصلی RNN‌های ساده، مشکل محو یا انفجار گرادیان^۶ در هنگام یادگیری وابستگی‌های

¹CNN

²End-to-End

³Recurrent Neural Networks, RNNs

⁴Feedforward

⁵Hidden State

⁶Vanishing/Exploding Gradient

بلندمدت در توالی‌های طولانی است. برای رفع این مشکل، ساختارهای پیشرفته‌تری مانند واحد حافظه بلند-کوتاه^۱ [۱۰] و واحد بازگشتشی دروازه‌ای^۲ [۱۱] معرفی شده‌اند. LSTM با بهره‌گیری از یک سلول حافظه^۳ و سه نوع گیت (گیت فراموشی، گیت ورودی و گیت خروجی)، امکان کنترل دقیق جریان اطلاعات را فراهم می‌کند. این گیت‌ها به شبکه اجازه می‌دهند تا تصمیم بگیرد کدام اطلاعات را در سلول حافظه نگه دارد، کدام را فراموش کند و کدام را به عنوان خروجی مرحله فعلی استفاده کند. این مکانیزم به LSTM‌ها کمک می‌کند تا اطلاعات مهم را در طول توالی‌های بسیار طولانی حفظ کرده و وابستگی‌های بلندمدت را به طور مؤثرتری مدل‌سازی کنند. GRU ساختار ساده‌تری نسبت به LSTM دارد (با دو گیت) اما عملکرد مشابهی در بسیاری از وظایف ارائه می‌دهد.

در این پژوهش، از LSTM (یا می‌توانست از GRU استفاده شود) برای تحلیل داده‌های ترتیبی، مانند توالی فراخوانی‌های API، به منظور شناسایی الگوهای رفتاری پویا در اپلیکیشن‌های اندرویدی استفاده شده است. این قابلیت، LSTM را به ابزاری قدرتمند برای تشخیص رفتارهای مخربی که در طول زمان و از طریق دنباله‌ای از اقدامات رخ می‌دهند، تبدیل می‌کند.

۳-۵-۲-۲ شبکه‌های عصبی گرافی (GNN)

شبکه‌های عصبی گرافی^۴ دسته‌ای از مدل‌های یادگیری عمیق هستند که برای پردازش داده‌هایی با ساختار گرافی (مانند شبکه‌های اجتماعی، مولکول‌های شیمیایی، یا گراف‌های فراخوانی توابع در نرم‌افزار) طراحی شده‌اند [۱۲]. داده‌های گرافی شامل مجموعه‌ای از گره‌ها^۵ و لبه‌ها^۶ هستند که به ترتیب موجودیت‌ها و روابط بین آن‌ها را نشان می‌دهند. GNN‌ها قادرند هم ساختار (توپولوژی) گراف و هم ویژگی‌های گره‌ها و لبه‌ها را برای یادگیری بازنمایی‌های مفید به کار گیرند.

ایده اصلی در GNN‌ها، بهروزرسانی بازنمایی (بردار ویژگی) هر گره بر اساس اطلاعات گره‌های همسایه آن است. این فرآیند معمولاً شامل دو مرحله اصلی است که در هر لایه GNN تکرار می‌شود:

۱. تجمیع (Aggregation): جمع‌آوری اطلاعات (بازنمایی‌ها) از گره‌های همسایه. روش‌های مختلفی برای تجمیع وجود دارد، مانند میانگین‌گیری، جمع یا حداکثرگیری.

۲. بهروزرسانی (Update): ترکیب اطلاعات تجمیع شده از همسایگان با اطلاعات خود گره (بازنمایی فعلی آن) برای تولید بازنمایی جدید گره. این کار معمولاً با استفاده از یک شبکه عصبی کوچک

¹Long Short-Term Memory, LSTM

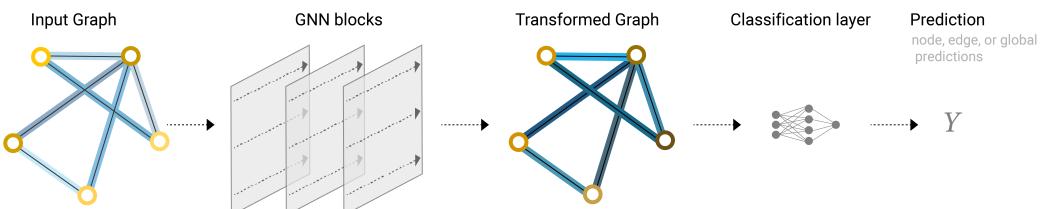
²Gated Recurrent Unit, GRU

³Memory Cell

⁴Graph Neural Networks, GNNs

⁵Nodes

⁶Edges



شکل ۲-۲ ساختار شبکه عصبی گرافی (GNN) برای پردازش داده‌های گرافی و طبقه‌بندی. برگرفته از [۱۲].

(مانند یک لایه خطی و یک تابع فعال‌سازی) انجام می‌شود.

با پشته کردن چندین لایه GNN، هر گره می‌تواند اطلاعات را از همسایگان دورتر نیز دریافت کند و بازنمایی‌های پیچیده‌تری که الگوهای ساختاری سطح بالاتر را در بر می‌گیرند، یاد گرفته شود. در این پژوهش، از GNN (به‌طور خاص، احتمالاً مدل‌هایی مانند^۱ [۱۳] یا^۲ [۱۴]) برای تحلیل داده‌های گرافی استخراج شده از اپلیکیشن‌ها (مانند گراف فراخوانی API) استفاده شده است. این داده‌ها با استفاده از کتابخانه^۳ به صورت اشیای Data بارگذاری و پردازش می‌شوند. GNN‌ها به مدل کمک می‌کنند تا ویژگی‌های ساختاری مهمی را که ممکن است نشان‌دهنده رفتار مخرب باشند (مانند الگوهای خاصی از تعامل بین توابع یا اجزا) استخراج کنند.

۲-۲-۶ ترنسفورمرها

۲-۰-۶-۱ معرفی ترنسفورمر و تحول در مدل‌های یادگیری عمیق نقطه عطف برجسته‌ای در تکامل معماری‌های یادگیری عمیق، معرفی مدل ترنسفورمر در مقاله "Attention Is All You Need"^۴ توسط Vaswani et al. [۱۵] بود که انقلابی در پردازش داده‌های ترتیبی، و بعدها انواع دیگر داده‌ها، به ارمغان آورد. این معماری، برخلاف مدل‌های بازگشته (RNN/LSTM) که توالی‌ها را به صورت مرحله به مرحله پردازش می‌کردند و با مشکلاتی مانند وابستگی‌های بلندمدت و عدم امکان پردازش موازی مواجه بودند، صرفاً بر مکانیزم توجه^۵، به‌ویژه توجه خودی^۶ و توجه چندسر^۷، تکیه می‌کند.

مکانیزم توجه به مدل اجازه می‌دهد تا هنگام پردازش یک عنصر در توالی (مثلاً یک کلمه در جمله یا یک فراخوانی API در یک دنباله)، به طور مستقیم به تمام عناصر دیگر توالی نگاه کرده و وزن اهمیتی متفاوتی به هر یک اختصاص دهد. این قابلیت، مدل‌سازی وابستگی‌های بلندمدت بین عناصر را بسیار کارآمدتر می‌کند. همچنین، از آنجایی که محاسبات توجه برای هر عنصر می‌تواند به صورت موازی انجام

¹Graph Convolutional Network (GCN)

²Graph Attention Network (GAT)

³torch_geometric

⁴Attention Mechanism

⁵Self-Attention

⁶Multi-Head Attention

شود، ترنسفورمرها سرعت آموزش و استنتاج را به طور چشمگیری بهبود بخشیدند و امکان آموزش مدل‌های بسیار بزرگتر را فراهم کردند. معماری استاندارد ترنسفورمر شامل دو بخش اصلی است: کدگذار^۱ که ورودی را به یک دنباله از بازنمایی‌های پیوسته تبدیل می‌کند و گشاینده^۲ که این بازنمایی‌ها را برای تولید خروجی (مثلًاً در ترجمه ماشینی یا تولید متن) به کار می‌برد. هر دوی این بخش‌ها از پشت‌هایی از لایه‌های توجه چندسر و شبکه‌های عصبی پیش‌خور^۳ تشکیل شده‌اند که با اتصالات باقیمانده^۴ و نرمال‌سازی لایه‌ای^۵ ترکیب شده‌اند.

با ظهور ترنسفورمر، مدل‌های زبانی بزرگ^۶ پیشرفت‌های مانند BERT (که از پشت‌های کدگذار ترنسفورمر استفاده می‌کند و برای درک زبان آموزش دیده) و GPT (که از پشت‌های گشاینده ترنسفورمر بهره می‌برد و برای تولید زبان آموزش دیده) توسعه یافتد. این مدل‌ها با پیش‌آموزش روی حجم عظیمی از داده‌های متنی و سپس تنظیم دقیق^۷ برای وظایف خاص، عملکردی استثنایی در طیف وسیعی از کاربردهای پردازش زبان طبیعی نشان دادند.

این پیشرفت‌ها، الهام‌بخش کاربرد ترنسفورمرها در حوزه‌های فراتر از زبان، از جمله بینایی کامپیوتر، تحلیل داده‌های گرافی و همچنین امنیت سایبری و تشخیص بدافزار [۱۶] شد. توانایی ترنسفورمر در مدل‌سازی روابط پیچیده و بلندمدت و پردازش موازی، آن را به گزینه‌ای جذاب برای تحلیل داده‌های چندوجهی و پیچیده مرتبط با بدافزارها تبدیل کرد.

۷-۲-۲ ترنسفورمرها در مدل MAGNET

در این پژوهش، معماری ترنسفورمر به عنوان هسته اصلی مدل MAGNET^۸ به کار گرفته شده است تا داده‌های چندوجهی شامل داده‌های جدولی (ویژگی‌های ایستا)، گرافی (روابط ساختاری) و ترتیبی (رفتار پویا) را به صورت یکپارچه تحلیل کند. ترنسفورمر در این مدل از مکانیزم توجه پویا برای یادگیری بازنمایی‌های غنی^۹ از هر وجه داده و سپس تلفیق^{۱۰} هوشمندانه این بازنمایی‌ها استفاده می‌کند.

به طور خاص، لایه‌های کدگذار ترنسفورمر می‌توانند برای پردازش ویژگی‌های استخراج شده از داده‌های جدولی (پس از تبدیل به دنباله‌ای از ویژگی‌ها یا استفاده از توکن‌های خاص) و داده‌های گرافی (با استفاده از تکنیک‌هایی مانند تبدیل گره‌ها به دنباله یا به کارگیری Graph Transformers) به کار روند. مکانیزم

¹Encoder

²Decoder

³Feed-Forward Networks

⁴Residual Connections

⁵Layer Normalization

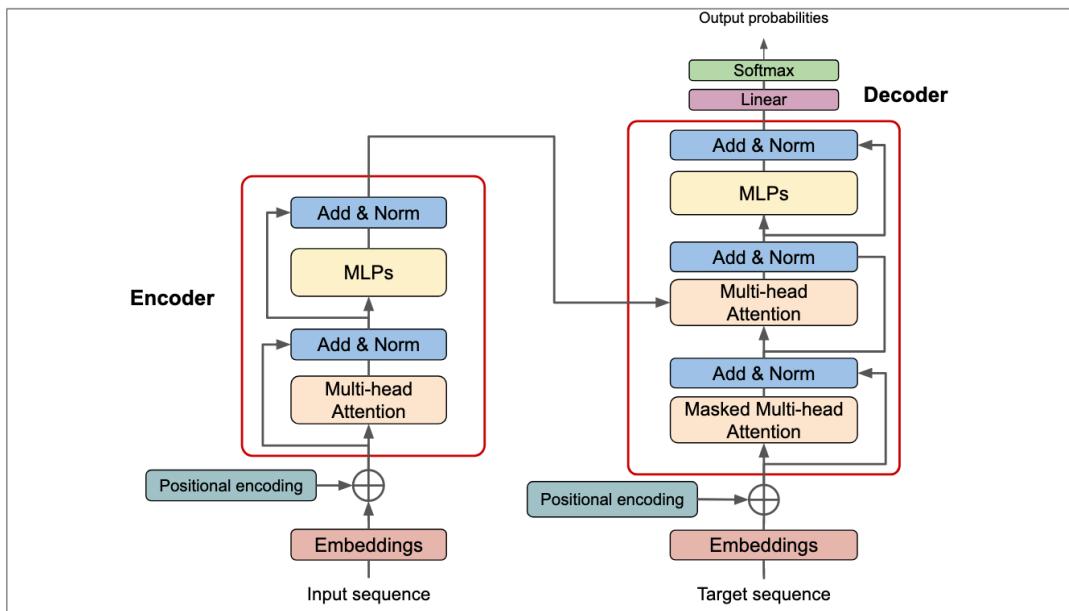
⁶Large Language Models, LLMs

⁷Fine-tuning

⁸Multi-Modal Analysis for Graph and Network Threat Detection

⁹Embeddings

¹⁰Fusion



شکل ۲ - ۲ معماری ترنسفورمر شامل کدگذار و گشاينده با مکانيزم توجه چندسر. برگرفته از [۱۵].

توجه چندسر به مدل اجازه می دهد تا روابط متقابل بین ویژگی های مختلف ایستا و ساختاری را کشف کند. همزمان، لایه های گشاينده ترنسفورمر (یا یک پشته کدگذار دیگر) می توانند برای مدل سازی توالی های ترتیبی (مانند فراخوانی های API) به کار گرفته شوند تا الگوهای زمانی مخرب شناسایی شوند.

ساختار ترنسفورمر در MAGNET احتمالاً شامل چندین بلوک کدگذار/گشاينده است که هر بلوک دارای یک لایه توجه چندسر و یک لایه تغذیه رویه جلو است. داده های ورودی از هر وجه ابتدا با استفاده از لایه های جاسازی به بردارهایی با ابعاد ثابت تبدیل می شوند. سپس، مکانیزم توجه خودی و احتمالاً توجه متقابل^۱ بین وجه های مختلف، وزن های متفاوتی به هر عنصر داده تخصیص می دهد تا اهمیت نسبی آن را در پیش بینی نهایی برچسب (سالم یا مخرب) تعیین کند. این فرآیند با استفاده از نرم افزاری لایه ای و اتصالات باقیمانده برای پایداری و بهبود آموزش، بهینه می شود.

مزیت اصلی استفاده از ترنسفورمر در مدل MAGNET، توانایی آن در پردازش موازی داده های چندوجهی و درک روابط پیچیده و بلندمدت *درون* هر وجه و *بین* وجه های مختلف داده است. این ویژگی، به ویژه در شناسایی بدافزارهای پیچیده و ناشناخته که ممکن است از تکنیک های پنهان سازی پیشرفته استفاده کنند و نیازمند تحلیل جامع رفتار و ساختار اپلیکیشن ها هستند، بسیار مؤثر است. نتایج ادعا شده در متن (دقت ۹۷.۲۴ درصد و F1 Score ۰.۹۸۲۳) نشان می دهد که این رویکرد مبتنی بر ترنسفورمر و داده های چندوجهی، پتانسیل ارائه عملکردی برتر نسبت به مدل های سنتی یا تکوجهی را دارد.

^۱Cross-Attention

۸-۲-۲ ماشین بردار پشتیبان (SVM)

ماشین بردار پشتیبان^۱ یکی از الگوریتم‌های یادگیری نظارت شده قدرتمند و پرکاربرد است که عمدتاً برای مسائل طبقه‌بندی^۲ و همچنین رگرسیون^۳ استفاده می‌شود. ایده اصلی SVM در مسائل طبقه‌بندی دودویی، یافتن یک ابرصفحه^۴ بهینه در فضای ویژگی‌هاست که بتواند داده‌های مربوط به دو کلاس مختلف را به بهترین شکل ممکن از یکدیگر جدا کند.

”بهترین شکل ممکن“ در SVM به معنای یافتن ابرصفحه‌ای است که بیشترین حاشیه^۵ را با نزدیک‌ترین نقاط داده از هر دو کلاس داشته باشد. این نزدیک‌ترین نقاط، که روی مرز حاشیه قرار می‌گیرند، بردارهای پشتیبان Support Vectors نامیده می‌شوند، زیرا موقعیت ابرصفحه جداکننده تنها به این نقاط بستگی دارد. هدف، حداکثر کردن این حاشیه است، زیرا تئوری یادگیری آماری نشان می‌دهد که جداکننده‌ای با حاشیه بزرگتر، معمولاً خطای تعییم کمتری دارد و عملکرد بهتری روی داده‌های دیده‌نشده خواهد داشت.

۱-۰-۸-۲-۲ انواع SVM

• **SVM خطی^۶**: زمانی که داده‌ها به صورت خطی قابل تفکیک باشند (یعنی بتوان با یک خط مستقیم یا یک صفحه صاف آنها را جدا کرد)، از SVM خطی استفاده می‌شود.

• **SVM غیرخطی^۷**: در بسیاری از مسائل دنیای واقعی، داده‌ها به صورت خطی قابل تفکیک نیستند. در این موارد، SVM از ترفندهای کرنل^۸ استفاده می‌کند. ترفندهای کرنل به SVM اجازه می‌دهند تا داده‌ها را به یک فضای ویژگی با ابعاد بالاتر نگاشت کند که در آن، داده‌ها ممکن است به صورت خطی قابل تفکیک شوند. سپس SVM خطی در آن فضای جدید اعمال می‌شود. توابع کرنل رایج شامل کرنل چندجمله‌ای^۹، کرنل تابع پایه شعاعی^{۱۰} یا گاوی، و کرنل سیگموئید^{۱۱} هستند.

۲-۰-۸-۲-۲ کاربرد در تشخیص بدافزار SVM به دلیل عملکرد خوب در فضاهای با ابعاد بالا و مقاومت نسبی در برابر بیش‌برازش^{۱۲}، به طور گسترده‌ای در تشخیص بدافزار اندروید، بهویژه با استفاده

¹Support Vector Machine, SVM

²Classification

³Regression

⁴Hyperplane

⁵Margin

⁶Linear SVM

⁷Non-linear SVM

⁸Kernel Trick

⁹Polynomial

¹⁰Radial Basis Function, RBF

¹¹Sigmoid

¹²Overfitting

از ویژگی‌های استاتیک (مانند مجوزها، فرآخوانی‌های API) یا ویژگی‌های پویا (مانند توالی‌های رفتاری) استفاده شده است [۳، ۱۷]. اغلب، ویژگی‌های استخراج شده از اپلیکیشن‌ها به عنوان ورودی به SVM داده می‌شوند تا مدل آموزش ببیند که آیا یک اپلیکیشن بدافزار است یا خیر.

۳-۰-۸-۲-۲ مزايا و معایب

• **مزايا:** کارایی بالا در فضاهای با ابعاد زیاد، مؤثر بودن در مواردی که تعداد ابعاد بیشتر از تعداد نمونه‌هاست، حافظه کارآمد (چون فقط از بردارهای پشتیبان استفاده می‌کند)، تطبیق‌پذیری با استفاده از کرنل‌های مختلف.

• **معایب:** عملکرد ضعیف در مجموعه داده‌های بسیار بزرگ (به دلیل پیچیدگی محاسباتی آموزش که می‌تواند بین $O(n^3)$ تا $O(n^2)$ باشد)، حساسیت به انتخاب تابع کرنل و پارامترهای آن (مانند پارامتر C یا هزینه خطأ و پارامتر گاما در کرنل، RBF عدم ارائه مستقیم احتمالات تعلق به کلاس (اگرچه روش‌هایی برای تخمین آن وجود دارد).

اگرچه مدل‌های یادگیری عمیق مانند CNN و ترنسفورمرها در سال‌های اخیر توجه بیشتری را به خود جلب کرده‌اند، SVM همچنان یک ابزار قدرتمند و یک معیار (Baseline) مهم در حوزه تشخیص بدافزار محسوب می‌شود.

۳-۲ تکنیک‌های آموزشی و ارزیابی

در این بخش، تکنیک‌های کلیدی مورد استفاده برای آموزش مدل MAGNET و ارزیابی عملکرد آن تشریح می‌شود. این تکنیک‌ها برای اطمینان از قابلیت اطمینان، کارایی و تعیین‌پذیری مدل ضروری هستند.

۱-۳-۱ اعتبارسنجی متقطع (Cross-Validation)

اعتبارسنجی متقطع^۱ یک تکنیک آماری برای ارزیابی عملکرد مدل‌های یادگیری ماشین و تخمین میزان تعیین‌پذیری آن‌ها به داده‌های جدید و مستقل است. این روش به کاهش واریانس تخمین عملکرد کمک کرده و از بیشبرازش^۲ بر روی یک تقسیم خاص از داده‌ها به مجموعه‌های آموزشی و آزمایشی جلوگیری می‌کند.

رایج‌ترین نوع اعتبارسنجی متقطع، اعتبارسنجی قایی- k -fold Cross-Validation است. در این

روش:

¹Cross-Validation

²Overfitting

۱. مجموعه داده اصلی به صورت تصادفی به k زیرمجموعه (یا "تا") با اندازه تقریباً مساوی تقسیم می‌شود.

۲. فرآیند آموزش و ارزیابی k بار تکرار می‌شود.

۳. در هر تکرار (i از ۱ تا k)، از $1-k$ زیرمجموعه برای آموزش مدل استفاده می‌شود و از زیرمجموعه باقیمانده (تا $-i$) به عنوان مجموعه اعتبارسنجی (Validation Set) برای ارزیابی مدل استفاده می‌شود.

۴. نتایج ارزیابی (مانند دقت، F1 Score) از هر k تکرار جمع‌آوری می‌شود.

۵. میانگین (و گاهی انحراف معیار) این نتایج به عنوان تخمین نهایی عملکرد مدل گزارش می‌شود.

در این پژوهش، از اعتبارسنجی ۵-تا^۱ استفاده شده است ($k = 5$). این بدان معناست که داده‌ها به پنج بخش تقسیم شده و مدل پنج بار آموزش و ارزیابی می‌شود، هر بار با یک بخش متفاوت به عنوان داده اعتبارسنجی. این رویکرد نسبت به تقسیم ساده آموزشی/آزمایشی، تخمین پایدارتر و قابل اعتمادتری از عملکرد مدل ارائه می‌دهد، زیرا از تمام داده‌ها هم برای آموزش و هم برای ارزیابی استفاده می‌شود. در پیاده‌سازی کد، این فرآیند می‌تواند با استفاده از توابعی مانند KFold یا StratifiedKFold از کتابخانه scikit-learn و اجرای حلقه آموزش و ارزیابی در هر تقسیم انجام شود.

۲-۳-۲ مدیریت بیشبرازش و کمبرازش

دو چالش اساسی در آموزش مدل‌های یادگیری ماشین، بهویشه مدل‌های یادگیری عمیق با ظرفیت بالا، بیشبرازش^۲ و کمبرازش^۳ هستند.

• **کمبرازش^۴**: زمانی رخ می‌دهد که مدل به اندازه کافی پیچیده نیست یا به اندازه کافی آموزش ندیده است تا الگوهای اساسی موجود در داده‌های آموزشی را یاد بگیرد. در نتیجه، مدل هم بر روی داده‌های آموزشی و هم بر روی داده‌های جدید عملکرد ضعیفی دارد. برای مقابله با کمبرازش، می‌توان پیچیدگی مدل را افزایش داد (مثلًاً با افزودن لایه‌ها یا نورون‌ها)، ویژگی‌های بهتری مهندسی کرد، یا زمان آموزش را افزایش داد. در مدل MAGNET، تنظیم مناسب تعداد لایه‌ها^۵ و ابعاد نهان^۶ می‌تواند به جلوگیری از کمبرازش کمک کند.

^۱5-fold Cross-Validation

^۲Overfitting

^۳Underfitting

^۴Underfitting

^۵num_layers

^۶embedding_dim

• **بیشبرازش^۱**: زمانی رخ می‌دهد که مدل بیش از حد به داده‌های آموزشی خاص، از جمله نویز و جزئیات تصادفی آن، وابسته می‌شود. در نتیجه، مدل بر روی داده‌های آموزشی عملکرد بسیار خوبی دارد، اما توانایی تعمیم به داده‌های جدید و دیده‌نشده را از دست می‌دهد و بر روی آن‌ها عملکرد ضعیفی نشان می‌دهد. برای مقابله با بیشبرازش، از تکنیک‌های تنظیم‌گری^۲ استفاده می‌شود. در این پژوهش، از تکنیک‌های زیر استفاده شده است:

- **کناره‌گیری^۳**: در هر مرحله آموزش، به طور تصادفی برخی از نورون‌ها (و اتصالات آن‌ها) را با احتمال مشخصی (در اینجا با نرخ ۲۰٪) غیرفعال می‌کند. این کار باعث می‌شود شبکه به یک مسیر یا ویژگی خاص بیش از حد وابسته نشود و مدل مقاومتری یاد بگیرد.

- **توقف زودهنگام^۴**: عملکرد مدل بر روی یک مجموعه اعتبارسنجی جداگانه در طول آموزش نظارت می‌شود. اگر عملکرد مدل روی مجموعه اعتبارسنجی برای تعداد مشخصی از دوره‌ها (Epochs) بهبود نیابد (در اینجا با صبر ۳ دوره)، آموزش متوقف می‌شود، حتی اگر عملکرد روی مجموعه آموزشی همچنان در حال بهبود باشد. این کار از ادامه آموزش و ورود مدل به فاز بیشبرازش جلوگیری می‌کند.

- **تنظیم‌گری L2^۵**: (هرچند به صراحت در بخش بیشبرازش ذکر نشده، اما در بخش بهینه‌سازی Adam با $\text{weight_decay} = 0.01$ اشاره شده است). این روش با افزودن یک جمله جریمه بهتابع هزینه که متناسب با مجنور اندازه وزن‌های مدل است، از بزرگ شدن بیش از حد وزن‌ها جلوگیری می‌کند و به مدل ساده‌تر و با تعمیم‌پذیری بهتر منجر می‌شود.

تحلیل نمودارهای تابع هزینه و معیارهای ارزیابی بر روی داده‌های آموزشی و اعتبارسنجی در طول زمان آموزش، ابزار مهمی برای تشخیص و مدیریت این دو پدیده است.

۳-۳-۲ روش‌های بهینه‌سازی

بهینه‌سازی فرآیند تنظیم پارامترهای مدل (مانند وزن‌ها و بایاس‌ها در شبکه‌های عصبی) به منظور کمینه کردن تابع هزینه^۶ است. انتخاب الگوریتم بهینه‌سازی مناسب و تنظیم ابرپارامترهای آن نقش کلیدی در سرعت همگرایی و کیفیت نهایی مدل دارد.

¹Overfitting

²Regularization

³Dropout

⁴Early Stopping

⁵Weight Decay

⁶Loss Function

⁷Hyperparameters

الگوریتم^۱ یکی از محبوب‌ترین و کارآمدترین الگوریتم‌های بهینه‌سازی مبتنی بر گرادیان نزولی تصادفی است که برای آموزش شبکه‌های عصبی عمیق استفاده می‌شود. SGD Stochastic Gradient Descent، Adam نرخ یادگیری Learning Rate را برای هر پارامتر به صورت تطبیقی تنظیم می‌کند. این کار را با استفاده از تخمین‌های مرتبه اول (میانگین یا مونتوم) و مرتبه دوم (واریانس غیرمرکزی) گرادیان‌ها انجام می‌دهد. به عبارت دیگر، Adam مزایای دو الگوریتم دیگر، یعنی AdaGrad (که نرخ یادگیری را بر اساس فراوانی بهروزرسانی پارامتر تنظیم می‌کند) و RMSProp (که از میانگین متحرک نمایی مجدور گرادیان‌ها استفاده می‌کند) را با هم ترکیب می‌کند و همچنین از مونتوم برای هموارسازی مسیر گرادیان و تسريع همگرایی بهره می‌برد. در این پژوهش، Adam با نرخ یادگیری پیش‌فرض (معمولًاً ۰.۰۰۱) و ضریب تنظیم گری L2 weight_decay (train_and_evaluate_magnet) برابر با ۰.۰۱ در تابع بهروزرسانی وزن‌های مدل MAGNET به کار رفته است. Adam به دلیل سرعت همگرایی بالا، نیاز کمتر به تنظیم دقیق نرخ یادگیری اولیه، و عملکرد خوب در مسائل با گرادیان‌های پراکنده یا نویزی، انتخاب مناسبی برای آموزش مدل‌های پیچیده مانند ترنسفورمر و GNN بر روی داده‌های چندوجهی بوده است.

۲-۳-۳-۲ PIRATES الگوریتم

الگوریتم^۲ PIRATES یکی از روش‌های نوین بهینه‌سازی الهام‌گرفته از طبیعت است که با الهام از رفتار جمعی دزدان دریایی در جستجوی گنج طراحی شده است. این الگوریتم با الهام از مطالعات [۱۸] در زمینه بهینه‌سازی و کاربردهای یادگیری تقویتی [۱۹]، و همچنین با استفاده از مفاهیم بازی دزدان دریایی، طراحی شده است. در این الگوریتم، هر راه حل بالقوه به عنوان یک «کشتی» در فضای جستجو مدل می‌شود و مجموعه‌ای از کشتی‌ها (جمعیت) به طور موازی و با استفاده از اطلاعات فردی (تجربیات گذشته)، جمعی اطلاعات بهترین اعضاء)، و تعاملات خاص (مانند نبرد و طوفان)، به سمت نقاط بهینه حرکت می‌کند. این الگوریتم با ترکیب ایده‌هایی از الگوریتم‌های ازدحامی (مانند PSO و تکاملی، سعی در افزایش پایداری، سرعت همگرایی و جلوگیری از گیر افتادن در بهینه‌های محلی دارد. اجزای کلیدی این الگوریتم عبارت‌اند از:

- **کشتی‌ها (Ships):** هر کشتی نمایانگر یک نقطه در فضای جستجو است و موقعیت و سرعت مخصوص به خود را دارد.

¹Adam²Adaptive Moment Estimation³Pirate-Inspired Robust Adaptive Trajectory Exploration Strategy

• **رهبر (Leader):** کشتی با بهترین عملکرد (کمترین مقدار تابع هدف) در هر تکرار به عنوان رهبر انتخاب می‌شود و سایر کشتی‌ها از آن پیروی می‌کنند.

• **نقشه جمعی و خصوصی:** هر کشتی علاوه بر بهترین موقعیت شخصی، از نقشه‌ای جمعی (شامل بهترین موقعیت‌های سایر کشتی‌ها) نیز بهره می‌برد.

• **کشتی‌های برتر (Top Ships):** تعدادی از بهترین کشتی‌ها به عنوان مرجع برای سایر اعضاء عمل می‌کنند.

• **مکانیزم‌های تنوع‌بخش:** الگوریتم با استفاده از نبرد بین کشتی‌های برتر و وقوع طوفان‌های تصادفی، از همگرایی زودهنگام و گیر افتادن در بهینه‌های محلی جلوگیری می‌کند.

الگوریتم PIRATES با بهره‌گیری از چندین منبع اطلاعاتی و تنظیم پویا، قادر است به سرعت به نقاط بهینه همگرا شود و در عین حال تنوع جمعیت را حفظ کند. این ویژگی‌ها، PIRATES را به گزینه‌ای مناسب برای بهینه‌سازی مسائل پیچیده، از جمله تنظیم خودکار هایپرپارامترهای مدل‌های یادگیری عمیق و چندوجهی، تبدیل کرده است. در پژوهش حاضر، از این الگوریتم برای جستجوی بهینه پارامترهای مدل استفاده شده است.

استفاده از الگوریتم‌های بهینه‌سازی الهام‌گرفته از طبیعت، به ویژه PIRATES، در سال‌های اخیر به عنوان رویکردی مؤثر برای حل مسائل بهینه‌سازی غیرخطی و پیچیده در حوزه‌های مختلف، از جمله جستجوی سایری و یادگیری ماشین، مطرح شده است. در این پژوهش، PIRATES به عنوان یکی از ابزارهای کلیدی برای تنظیم بهینه پارامترهای مدل چندوجهی مبتنی بر تنسفورم و شبکه‌های عصبی گرافی به کار رفته است.

۳-۳-۳-۲ الگوریتم Optuna

[۲۰] یک چارچوب Framework نرم‌افزاری متن‌باز و قدرتمند برای بهینه‌سازی خودکار Optuna از الگوریتم‌های جستجوی متنوعی پشتیبانی می‌کند، از جمله جستجوی تصادفی^۱، جستجوی شبکه‌ای^۲، و الگوریتم‌های پیشرفته‌تر مبتنی بر مدل مانند^۳ که نوعی بهینه‌سازی بیزی است. یکی از ویژگی‌های کلیدی Optuna، قابلیت هرس (Pruning) آزمایش‌های ناموفق است. با استفاده از الگوریتم‌های هرس^۴، Optuna می‌تواند آزمایش‌هایی را که در مراحل اولیه عملکرد ضعیفی دارند، متوقف کرده و منابع محاسباتی را بر روی آزمایش‌های امیدوارکننده‌تر متمرکز کند.

¹ Random Search

² Grid Search

³ Tree-structured Parzen Estimator (TPE)

⁴ Median Pruning یا Successive Halving

در این پژوهش، ادعا شده که از Optuna برای تنظیم پارامترهایی مانند نرخ Dropout (که مقدار بهینه ۲۰٪ یافت شده) و ابعاد نهان^۱ استفاده شده است. این فرآیند معمولاً شامل تعریف یک تابع هدف و مشخص کردن فضای جستجو برای هر هایپرپارامتر است. سپس Optuna با اجرای مکرر تابع هدف با مقادیر مختلف هایپرپارامترها (که با استفاده از نمونهبرداری تطبیقی انتخاب می‌شوند)، به تدریج به سمت یافتن ترکیب بهینه حرکت می‌کند. استفاده از Optuna می‌تواند فرآیند زمانبر و خسته‌کننده تنظیم دستی هایپرپارامترها را خودکار کرده و به یافتن مدل‌هایی با عملکرد بهتر کمک کند.

۴-۳-۲ معیارهای ارزیابی عملکرد

برای ارزیابی کمی و کیفی عملکرد مدل طبقه‌بندی MAGNET در تشخیص بدافزارهای اندرویدی، از معیارهای استاندارد مختلفی استفاده شده است. انتخاب معیار مناسب به ماهیت مسئله و توزیع کلاس‌ها در مجموعه داده بستگی دارد.

۱-۴-۳-۲ دقت (Accuracy)

دقت (Accuracy) ساده‌ترین و رایج‌ترین معیار ارزیابی است که نسبت کل نمونه‌هایی که به درستی توانست مدل طبقه‌بندی شده‌اند (چه مثبت و چه منفی) به تعداد کل نمونه‌ها را اندازه‌گیری می‌کند:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

که در آن:

TP: تعداد نمونه‌های بدافزار که به درستی بدافزار تشخیص داده شده‌اند.

TN: تعداد نمونه‌های سالم که به درستی سالم تشخیص داده شده‌اند.

FP: تعداد نمونه‌های سالم که به اشتباه بدافزار تشخیص داده شده‌اند (خطای نوع اول).

FN: تعداد نمونه‌های بدافزار که به اشتباه سالم تشخیص داده شده‌اند (خطای نوع دوم).

در این پژوهش، دقت مدل MAGNET در تست نهایی ۹۷.۴۶ درصد گزارش شده است. دقت معیار خوبی است زمانی که کلاس‌ها تقریباً متوازن باشند. با این حال، در مجموعه‌های داده نامتوازن (مثلاً اگر تعداد

^۱embedding_dim

نمونه‌های سالم بسیار بیشتر از بدافزارها باشد)، دقت بالا ممکن است گمراه کننده باشد.

Score F1 ۲-۴-۳-۲

(امتیاز F1 Score) Mیانگین هارمونیک دو معیار دیگر، یعنی دقت (Precision) و یادآوری (Recall) است و به ویژه در مسائل با کلاس‌های نامتوافق مفید است.

- دقت (Precision): نسبت نمونه‌هایی که به درستی مثبت پیش‌بینی شده‌اند به کل نمونه‌هایی که مثبت پیش‌بینی شده‌اند. این معیار نشان می‌دهد که چقدر می‌توان به پیش‌بینی‌های مثبت مدل اعتماد کرد.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- یادآوری (Recall) یا حساسیت (Sensitivity) یا نرخ مثبت واقعی (TPR): نسبت نمونه‌هایی که به درستی مثبت پیش‌بینی شده‌اند به کل نمونه‌های واقعاً مثبت. این معیار نشان می‌دهد که مدل چه کسری از نمونه‌های مثبت واقعی را توانسته شناسایی کند.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

امتیاز F1 به صورت زیر محاسبه می‌شود:

$$\text{Score F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$$

امتیاز F1 تعادلی بین دقت و یادآوری برقرار می‌کند و مقدار بالای آن نشان می‌دهد که مدل هم در کاهش هشدارهای کاذب (FP بالا باعث کاهش Precision می‌شود) و هم در شناسایی نمونه‌های مثبت واقعی (FN بالا باعث کاهش Recall می‌شود) موفق بوده است. در این پژوهش، مقدار F1 Score ۰.۹۸۲۳ در بهترین حالت به دست آمده که نشان‌دهنده عملکرد بدافزار خوب مدل در هر دو جنبه است. این معیار معمولاً برای ارزیابی عملکرد در تشخیص بدافزار ارجحیت دارد، زیرا هزینه تشخیص نادرست (FN یا FP) می‌تواند بالا باشد.

منحنی مشخصه عملکرد گیرنده^۱ نموداری است که نرخ مثبت واقعی^۲ یا همان Recall را در مقابل نرخ مثبت کاذب^۳ در آستانه‌های طبقه‌بندی مختلف رسم می‌کند.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

مساحت زیر این منحنی^۴ یک معیار واحد است که توانایی کلی مدل در تمایز بین کلاس‌های مثبت و منفی را در تمام آستانه‌های ممکن اندازه‌گیری می‌کند. مقدار AUC بین ۰ و ۱ متغیر است.

AUC = 1: طبقه‌بندی کامل و بی‌نقص.

AUC = 0.5: عملکرد تصادفی (مانند پرتاپ سکه).

AUC < 0.5: عملکرد بدتر از تصادفی.

AUC یک معیار مفید است زیرا نسبت به تغییرات آستانه طبقه‌بندی و همچنین نسبت به عدم توازن کلاس‌ها (تا حدی) مقاوم است. هرچند مقدار دقیق AUC در متن گزارش نشده، اما با توجه به F1 Score بالا (۰.۹۸۲۳)، انتظار می‌رود مقدار AUC نیز بسیار بالا و نزدیک به ۱ (احتمالاً حدود ۰.۹۸ یا بیشتر) باشد، که نشان‌دهنده قدرت تمایز عالی مدل MAGNET است.

۱-۳-۴-۳-۱ جمع‌بندی ارزیابی استفاده ترکیبی از این معیارها (دقت، F1 Score، AUC) به همراه اعتبارسنجی متقطع، تصویری جامع و قابل اعتماد از عملکرد مدل MAGNET ارائه می‌دهد و امکان مقایسه معنادار آن با سایر روش‌ها را فراهم می‌کند. مدیریت دقیق بیش‌برازش و کم‌برازش و استفاده از روش‌های بهینه‌سازی مناسب نیز به دستیابی به این نتایج کمک کرده‌اند.

۴-۲ مروrij بر مطالعات پیشین

در این بخش، پژوهش‌های پیشین در حوزه تشخیص بدافزارهای اندرویدی با تمرکز بر رویکردهای اصلی تحلیل ایستا و پویا مروج می‌شوند. همچنین، به تاریخچه مختصه از تلاش‌ها در این زمینه اشاره می‌شود.

¹Receiver Operating Characteristic, ROC

²True Positive Rate, TPR

³False Positive Rate, FPR

⁴Area Under the Curve, AUC

۱-۴-۲ روش‌های تحلیل ایستا

روش‌های تحلیل ایستا^۱ به بررسی و تحلیل کد و ساختار اپلیکیشن‌های اندرویدی بدون نیاز به اجرای آن‌ها می‌پردازند. این روش‌ها معمولاً سریع‌تر از تحلیل پویا هستند و می‌توانند پوشش کاملی از کد برنامه را فراهم کنند. ویژگی‌های استخراج شده از تحلیل ایستا اغلب به عنوان ورودی برای مدل‌های یادگیری ماشین استفاده می‌شوند [۳].

۱-۱-۴-۲ تحلیل ویژگی‌های مبتنی بر مانیفست و متاداده

فایل AndroidManifest.xml حاوی اطلاعات مهمی درباره ساختار و نیازمندی‌های اپلیکیشن است.

• **تحلیل مجوزها (Permissions)**: یکی از پرکاربردترین روش‌های ایستا، تحلیل الگوهای مجوزهای درخواست‌شده توسط اپلیکیشن است. بدافزارها اغلب مجوزهای خطرناک و غیرضروری بیشتری نسبت به کاربرد ظاهری خود درخواست می‌کنند. پژوهش‌های متعددی مانند Drebin [۴] نشان داده‌اند که ترکیب مجوزها با سایر ویژگی‌های ایستا می‌تواند در تشخیص بادت‌افزار مؤثر باشد. مدل‌های یادگیری ماشین مانند SVM، درخت تصمیم و بیز ساده اغلب برای طبقه‌بندی بر اساس این ویژگی‌ها استفاده شده‌اند.

• **اجزای برنامه (Components)**: تعداد و نوع اجزای تعریف شده در مانیفست^۲ و همچنین فیلترهای مرتبط با آن‌ها نیز می‌تواند اطلاعات مفیدی در اختیار قرار دهد.

۲-۱-۴-۲ تحلیل کد (Code Analysis)

این روش‌ها به بررسی خود کد برنامه (بایت‌کد Dalvik/ART یا کد Native) می‌پردازند.

• **تحلیل فراخوانی‌های API**: شناسایی و تحلیل فراخوانی‌های API حساس (مانند API‌های مربوط به ارسال پیامک، دسترسی به موقعیت مکانی، یا استفاده از توابع رمزنگاری) یکی دیگر از رویکردهای رایج است. الگوها یا توالی‌های خاصی از فراخوانی‌های API می‌توانند نشان‌دهنده رفتار مخرب باشند. این ویژگی‌ها می‌توانند به صورت باینری (وجود یا عدم وجود فراخوانی) یا به صورت فراوانی استفاده شوند.

• **تحلیل جریان داده و کنترل (Data/Control Flow Analysis)**: روش‌های پیشرفته‌تر تحلیل ایستا، گراف جریان کنترل (CFG) یا گراف جریان داده (DFG) برنامه را ساخته و تحلیل می‌کنند تا

¹ Static Analysis

² مانند Activities, Services, Receivers, Providers)

مسیرهای اجرای بالقوه و نحوه انتشار داده‌های حساس را ردیابی کنند. تکنیک‌هایی مانند Taint Analysis می‌توانند برای شناسایی نشت اطلاعات خصوصی به کار روند.

- **ویژگی‌های مبتنی بر Opcode:** تحلیل توالی‌ها یا فراوانی کدهای عملیاتی (Opcodes) در بایت‌کد نیز به عنوان ویژگی برای مدل‌های یادگیری ماشین استفاده شده است.

۳-۱-۴-۲ تحلیل ایستای ساختاری

- **گراف فراخوانی^۱:** ساخت و تحلیل گراف فراخوانی توابع، که روابط فراخوانی بین متدهای مختلف برنامه را نشان می‌دهد، می‌تواند به شناسایی الگوهای ساختاری مرتبط با بدافزار کمک کند. شبکه‌های عصبی گرافی (GNN) برای تحلیل این گراف‌ها مناسب هستند.

۱-۳-۱-۴-۲ محدودیت‌های تحلیل ایستا با وجود مزایا، تحلیل ایستا با چالش‌هایی نیز روبروست:

- **ابهام‌سازی^۲:** بدافزارها اغلب از تکنیک‌های ابهام‌سازی کد^۳ برای دشوار کردن تحلیل ایستا استفاده می‌کنند.

- **بارگذاری پویای کد^۴:** برخی بدافزارها بخش‌هایی از کد مخرب خود را در زمان اجرا از منابع خارجی دانلود و اجرا می‌کنند که این بخش‌ها در تحلیل ایستا قابل مشاهده نیستند.

- **کد Native:** تحلیل کدهای نوشته شده به زبان‌های Native (مانند C/C++) که از طریق JNI فراخوانی می‌شوند، پیچیده‌تر از تحلیل بایت‌کد جاوا/کاتلین است.

۲-۴-۲ روش‌های تحلیل پویا

روش‌های تحلیل پویا (Dynamic Analysis) یا آنالیز رفتاری، اپلیکیشن را در یک محیط کنترل شده (مانند شبیه‌ساز، دستگاه واقعی یا جعبه شنی Sandbox) اجرا کرده و رفتار آن را در زمان اجرا مشاهده و ثبت می‌کنند. این روش‌ها می‌توانند بر محدودیت‌های تحلیل ایستا در مواجهه با ابهام‌سازی و بارگذاری پویای کد غلبه کنند، زیرا رفتار واقعی برنامه را بررسی می‌کنند [۳].

¹Call Graph

²Obfuscation

³رشته‌ها رمزگاری زائد، کدهای درج توابع، و متغیرهای نام تغییر مانند

⁴Dynamic Code Loading

۱-۲-۴-۲ مانیتورینگ رفتار سیستم

این روش‌ها فعالیت‌های اپلیکیشن در سطح سیستم عامل را رصد می‌کنند.

- **ردیابی فراخوانی‌های سیستمی^۱**: ثبت و تحلیل توالی فراخوانی‌های سیستمی که برنامه انجام می‌دهد. الگوهای خاصی از این فراخوانی‌ها می‌توانند نشان‌دهنده فعالیت مخرب باشند.
- **تحلیل Taint پویا^۲**: ابزارهایی مانند TaintDroid داده‌های حساس (منابع Taint) را مشخص کرده و نحوه انتشار آن‌ها در طول اجرای برنامه را ردیابی می‌کنند تا نشت اطلاعات به مقاصد غیرمجاز (سینک‌های Taint) را شناسایی کنند.
- **مانیتورینگ تغییرات فایل‌سیستم و رجیستری (در محیط ویندوز)**: ثبت هرگونه ایجاد، حذف یا تغییر فایل‌ها یا تنظیمات سیستمی.

۲-۲-۴-۲ تحلیل شبکه

بسیاری از بدافزارها برای ارتباط با سرور فرماندهی و کنترل (C&C)، ارسال داده‌های سرقت‌شده یا دانلود کدهای مخرب بیشتر، از شبکه استفاده می‌کنند. تحلیل ترافیک شبکه اپلیکیشن می‌تواند الگوهای مشکوکی مانند اتصال به IP‌ها یا دامنه‌های شناخته‌شده مخرب، استفاده از پروتکل‌های غیرمعمول، یا حجم بالای ترافیک خروجی را آشکار کند.

۳-۲-۴-۲ بورسی مصرف منابع

الگوهای غیرعادی در مصرف منابع سیستم مانند CPU، حافظه، باتری یا پهنهای باند شبکه نیز می‌تواند نشانه‌ای از فعالیت بدافزاری باشد (مثلاً استخراج رمزارز یا اجرای حملات DoS).

۱-۳-۲-۴-۲ **ابزارها و محیط‌های تحلیل پویا** ابزارها و پلتفرم‌های مختلفی برای تحلیل پویای بدافزارهای اندرویدی توسعه یافته‌اند، از جمله AndroPyTool، DroidBox، Mobile Sandbox و پلتفرم‌های تجاری مانند Joe Sandbox Mobile Sandbox. این ابزارها معمولاً اجرای برنامه را در یک شبیه‌ساز یا دستگاه روت‌شده خودکار کرده و گزارش جامعی از رفتارهای مشاهده‌شده تولید می‌کنند.

۲-۳-۲-۴-۲ محدودیت‌های تحلیل پویا

- **پوشش محدود کد^۳**: تحلیل پویا تنها مسیرهای اجرایی را بررسی می‌کند که در طول اجرای خاص

¹(System Call Tracing)

²Dynamic Taint Analysis

³Limited Code Coverage

فعال شده‌اند. بدافزارها ممکن است شامل کدهای مخربی باشند که تنها تحت شرایط خاصی (مثل^۱ در تاریخ معین یا با دریافت دستور خاص) فعال می‌شوند و در طول تحلیل مشاهده نشوند.

• **زمان‌بر بودن:** اجرای هر اپلیکیشن و ثبت رفتارهای آن می‌تواند زمان‌بر باشد، که تحلیل مجموعه داده‌های بزرگ را دشوار می‌کند.

• **تشخیص محیط تحلیل^۲:** بدافزارهای پیشرفته ممکن است تلاش کنند تا تشخیص دهنده آیا در یک محیط تحلیل (مانند شبیه‌ساز یا جعبه‌شنبی) اجرا می‌شوند یا خیر و در این صورت، رفتار مخرب خود را پنهان کنند.

• **نیاز به منابع:** نیاز به محیط‌های اجرایی ایزوله و ابزارهای مانیتورینگ دارد.

٤-٢-٤-٢ روش‌های ترکیبی Hybrid Approaches

برای بهره‌مندی از مزایای هر دو روش و غلبه بر محدودیت‌های آن‌ها، بسیاری از سیستم‌های تشخیص بدافزار مدرن از رویکردهای ترکیبی استفاده می‌کنند که ویژگی‌های استخراج شده از تحلیل ایستا و پویا را با هم ترکیب کرده و به عنوان ورودی به مدل‌های یادگیری ماشین یا یادگیری عمیق می‌دهند [۲]. مدل MAGNET در این پژوهش نیز با استفاده از داده‌های چندوجهی (که می‌توانند شامل ویژگی‌های ایستا و پویا باشند) در این دسته قرار می‌گیرد.

٣-٤-٢ کارهای پیشین و تاریخچه مختص

تلاش‌ها برای شناسایی و مقابله با بدافزارها تاریخچه‌ای طولانی دارد. از اولین برنامه‌های آنتی‌ویروس مانند^۳ (۱۹۸۷) و اسکنر ویروس مک‌آفی (۱۹۸۹) که عمدهاً مبتنی بر امضا بودند، تا سیستم‌های تشخیص ناهنجاری اولیه مبتنی بر آمار مانند^۴ (Denning, ۱۹۸۷) و W&S (WIDOM) در آزمایشگاه ملی لس‌آلاموس (۱۹۸۹).

با پیچیده‌تر شدن بدافزارها، روش‌های مبتنی بر یادگیری ماشین و سپس یادگیری عمیق اهمیت بیشتری یافته‌ند. در اوایل دهه ۲۰۱۰، تمرکز زیادی بر استخراج ویژگی‌های ایستا (مانند مجوزها در Drebin [۴]) و استفاده از طبقه‌بندی‌های کلاسیک مانند SVM بود. سپس، روش‌های مبتنی بر تحلیل پویا (مانند TaintDroid) و تحلیل رفتارهای سیستمی و شبکه‌ای مطرح شدند.

¹Environment Detection

²Flushot Plus

³ID intrusion detection system

در سال‌های اخیر، با پیشرفت یادگیری عمیق، مدل‌هایی مانند CNN برای تحلیل بایت کد به عنوان تصویر یا تحلیل ماتریس ویژگی‌ها، و RNN/LSTM برای تحلیل توالی‌های API یا رفتارهای پویا به کار گرفته شدند. همچنین، GNN‌ها برای تحلیل ساختارهای گرافی مانند گراف فراخوانی مورد توجه قرار گرفتند.

تحقیقاتی مانند کار [۲۱] et al. et Zhang (استفاده از CNN روی فراخوانی‌های API) و Vinayaku et al. et mar [۱۵] (استفاده از شبکه‌های عصبی عمیق) نتایج امیدوارکننده‌ای با دقت‌های بالا (گاهی بالای ۹۱٪ روی دیتاست‌های خاص) نشان دادند، اگرچه چالش‌هایی مانند تعیین‌پذیری به بدافزارهای جدید و مقاومت در برابر حملات فرار همچنان وجود دارند [۱۷].

رویکردهای چندوجهی مانند مدل پیشنهادی MAGNET که سعی در ترکیب اطلاعات از منابع مختلف (ایستا، پویا، ساختاری، ترتیبی) با استفاده از معماری‌های پیشرفته مانند ترانسفورمر دارند، گام بعدی در جهت افزایش دقت و استحکام سیستم‌های تشخیص بدافزار اندروید محسوب می‌شوند.

۵-۲ جمع‌بندی فصل

در این فصل، مبانی نظری و پیشینه تحقیقاتی لازم برای درک پژوهش حاضر در زمینه تشخیص بدافزارهای اندرویدی ارائه گردید. ابتدا، مفاهیم پایه‌ای شامل انواع بدافزارها (باج‌افزار، تروجان، جاسوس‌افزار، تبلیغ‌افزار)، اجزای کلیدی اپلیکیشن‌های اندرویدی (مجوزها، فایل APK، سورس‌کد) و نقش داده‌های چندوجهی (جدولی، گرافی، ترتیبی) تشریح شد. سپس، مروری بر تکامل روش‌های یادگیری ماشین و معرفی معماری‌های کلیدی یادگیری عمیق (CNN، RNN/LSTM، GNN، ترانسفورمر) و الگوریتم کلاسیک SVM ارائه شد که در این حوزه کاربرد فراوان دارند.

در ادامه، تکنیک‌های ضروری برای آموزش و ارزیابی مدل‌ها، از جمله اعتبارسنجی متقطع برای تخمین قابل اعتماد عملکرد، روش‌های مدیریت بیش‌برازش و کم‌برازش (مانند Dropout و توقف زودهنگام)، الگوریتم‌های بهینه‌سازی (مانند Adam) و ابزارهای بهینه‌سازی هایپرپارامتر (مانند Optuna)، و نهایتاً معیارهای استاندارد ارزیابی (دقت، F1 Score، AUC) مورد بحث قرار گرفتند.

بخش مرور مطالعات پیشین، به بررسی جامع روش‌های تحلیل ایستا (مبتنی بر مانیفست، کد و ساختار) و تحلیل پویا (مبتنی بر رفتار سیستم، شبکه و منابع) پرداخت و نقاط قوت و ضعف هر یک را بر شمرد. همچنین، به تاریخچه مختصراً از تلاش‌ها در این زمینه و اهمیت رویکردهای ترکیبی و یادگیری عمیق در سال‌های اخیر اشاره شد.

این فصل با فراهم آوردن درک عمیقی از چالش‌ها، مفاهیم، ابزارها و رویکردهای موجود در تشخیص بدافزار اندروید، زمینه را برای معرفی و ارزیابی روش پیشنهادی این پژوهش، مدل MAGNET، در فصل‌های آتی آماده می‌سازد.

فصل سوم:

مطالعه موردی: شرکت‌های پیشرو در امنیت

اندروید

۱-۳ هدف فصل

در این فصل، شرکت‌هایی که در حوزه امنیت اپلیکیشن‌ها و دستگاه‌های اندروید فعال هستند، بررسی شده و روش‌ها و ابزارهای آن‌ها تحلیل می‌شوند.

۲-۳ معیارهای انتخاب

انتخاب شرکت‌ها بر مبنای: پوشش کامل چرخه توسعه (SAST، DAST، MTD)، اعتبار جهانی، نوآوری، و امکان استخراج تجربه‌های عملی صورت گرفته است.

۳-۳ مطالعه موردی شرکت‌ها

NowSecure ۱-۳-۳

NowSecure یک شرکت آمریکایی فعال در امنیت موبایل است [۲۲].

- استفاده از behavioral DAST، SAST، [۲۳]
- ارائه پلتفرمی برای خودکارسازی تست‌ها [۲۴] (NowSecure Platform)
- همکاری با Synopsys برای یکپارچه‌سازی تست نسل بعدی [۲۵].

Qualys (VMDR Mobile) ۲-۳-۳

Qualys با افزودن ماژول VMDR Mobile، راهکار یکپارچه مدیریت آسیب‌پذیری برای دستگاه‌ها و اپلیکیشن‌ها ارائه می‌دهد [۲۶].

- انبارش خودکار دستگاه و اپ‌ها، شناسایی CVE‌ها و misconfiguration [۲۷]
- تصحیح از راه دور (OTA) پچ خودکار، مدیریت وضعیت امنیتی [۲۸]
- ادغام با MDM/EMM مانند Intune برای ورود خودکار داده [۲۹]

Synopsys / Checkmarx ۳-۳-۳

SAST قوی در محیط CI/CD، تشخیص آسیب‌پذیری‌های کد منبع با پوشش بیش از ۳۵ زبان • برنامه‌نویسی [۳۰].

- ادغام با ابزارهای تست موبایل از جمله همکاری با NowSecure برای پوشش MAST [۳۱]

۴-۳ روش‌ها و ابزارهای مشترک

- **تست ایستا (SAST) :** ابزارهایی مانند Checkmarx و NowSecure برای اسکن کد در مراحل اولیه توسعه [۳۲].
- **تست پویا (DAST) :** بررسی رفتار اپ در اجرا، NowSecure نمونه‌ای موفق [۳۳].
- **مدیریت آسیب‌پذیری و پچ (VMDR) :** Qualys برای شناسایی و تصحیح خودکار آسیب‌پذیری‌ها [۳۴].
- **تحلیل رفتاری و MTD :** تشخیص رفتار مخرب در زمان اجرا (Qualys و NowSecure) [۳۵].
- **ادغام DevSecOps و CI/CD :** خودکارسازی در pipeline با Checkmarx، NowSecure و Qualys [۳۶].

۵-۳ مقایسه و تحلیل

با مقایسه ویژگی‌ها:

- پوشش چرخه توسعه: به ترتیب^۱ Checkmarx^۲ Qualys^۳ NowSecure^۴.
- امکان اجرا در pipeline و سرعت ادغام.
- پوشش موبایل گرچه NowSecure و Qualys کامل‌تر است؛ Checkmarx بیشتر بر کد متمرکز است.

۶-۳ جمع‌بندی

هر سه راهکار مزايا و محدوديت دارند:

- کامل‌ترین پوشش، اما نیازمند پیاده‌سازی مستقل: NowSecure
- مدیریت یکپارچه و مناسب برای سازمان‌های دارای MDM/EMM: Qualys
- قوی در SAST، ولی MAST را تنها از طریق همکاری‌های خارجی پوشش می‌دهد: Checkmarx
- این تحلیل کمک می‌کند تا با توجه به مقیاس و امکانات پژوهش یا پروژه، مناسب‌ترین انتخاب انجام شود.

¹SAST+DAST+behavior

²inventory+vuln+response

³SAST

فصل چهارم:

روش پیشنهادی

۱-۴ روش پیشنهادی

با توجه به رشد روزافزون استفاده از سیستم‌عامل اندروید و در نتیجه، افزایش تهدیدات بدافزاری متوجه این پلتفرم [۱]، نیاز به روش‌های کارآمد و دقیق برای شناسایی بدافزارها بیش از پیش احساس می‌شود. این فصل به تشریح دقیق روش پیشنهادی این پژوهش، موسوم به **MAGNET** (تحلیل چندوجهی برای شناسایی تهدیدات مبتنی بر گراف و شبکه)، اختصاص دارد. این رویکرد با بهره‌گیری از داده‌های چندوجهی—جدولی، گرافی و ترتیبی—و ترکیب آن با معماری‌های پیشرفته یادگیری عمیق از جمله ترانسفورمرها و شبکه‌های عصبی گرافی، (GNN) محدودیت‌های روش‌های تحلیل ایستا و پویا را برطرف می‌کند. هدف اصلی، افزایش دقت شناسایی و تعیین‌پذیری، بهویژه در مواجهه با تهدیدات روز صفر (Zero-Day) است. این بخش به صورت زیر سازماندهی شده است: مقدمه‌ای بر روش پیشنهادی، فرضیات و ابزارهای محاسباتی، روش‌شناسی دقیق (شامل پیش‌پردازش داده‌ها، طراحی مدل، آموزش، بهینه‌سازی ابرپارامترها و ارزیابی) و جمع‌بندی نهایی. تمامی فرضیات، ابزارها، معادلات و شبه‌کدها به طور کامل مستند شده‌اند تا امکان تکرار‌پذیری فراهم باشد.

۱-۱-۱ مقدمه روش پیشنهادی

شناسایی بدافزار اندروید با چالش‌های مهمی روبروست که ناشی از محدودیت‌های رویکردهای سنتی است. روش‌های تحلیل ایستا، مانند بررسی کد منبع یا تحلیل مجوزها، اغلب در شناسایی بدافزارهای پیچیده که از تکنیک‌های مبهم‌سازی یا رمزنگاری استفاده می‌کنند، ناکام می‌مانند [۴]. این روش‌ها قادر به ثبت رفتارهای زمان اجرا یا سازگاری‌های پویا در برنامه‌های مخرب نیستند. در مقابل، تحلیل پویا که رفتار برنامه را در حین اجرا پایش می‌کند، محاسبات سنگینی دارد، زمان‌بر است و به دلیل عدم پوشش کامل مسیرهای اجرایی، پوشش کد ناقصی ارائه می‌دهد. این کاستی‌ها بهویژه در مواجهه با بدافزارهای روز صفر—تهدیداتی با الگوهای ناشناخته که از سیستم‌های مبتنی بر امضای قوانین فرار می‌کنند—بیشتر مشهود است.

برای غلبه بر این چالش‌ها، ما ادغام داده‌های چندوجهی را پیشنهاد می‌کنیم تا دیدی جامع از ویژگی‌های برنامه ارائه شود. به طور خاص، از داده‌های زیر استفاده شده است:

- **داده‌های جدولی:** ویژگی‌های ایستا مانند مجوزها، تعداد فایل‌ها و اندازه برنامه که بینشی از خصوصیات ساختاری ارائه می‌دهند.

- **داده‌های گرافی:** گراف‌های فراخوانی توابع که روابط ساختاری بین توابع را نمایش می‌دهند و وابستگی‌های داخلی برنامه را ثبت می‌کنند.

- **داده‌های ترتیبی:** توالی‌های فراخوانی API که الگوهای رفتاری زمانی در حین اجرا را منعکس می‌کنند.

این رویکرد چندوجهی، استخراج اطلاعات مکمل را تضمین می‌کند و خلاهای تحلیل‌های تک‌وجهی را پر می‌کند. افزون بر این، از معماری‌های پیشرفته یادگیری عمیق—ترنسفورمرها و GNN ها—برای مدل‌سازی الگوهای پیچیده در داخل و بین این وجه‌ها استفاده شده است. ترانسفورمرها در ثبت وابستگی‌های بلندمدت در داده‌های ترتیبی و جدولی برتری دارند، در حالی که ها GNN داده‌های گرافی را با بهره‌گیری از روابط گره‌ها و یال‌ها به طور مؤثر پردازش می‌کنند [۱۳، ۱۵].

نوآوری اصلی این پژوهش در مدل **MAGNET** نهفته است که سه مأذول تخصصی - *EnhancedTab*, *SequenceTransformer*, *GraphTransformer* و *Transformer* را با یک مکانیزم توجه پویا و لایه ادغام چندوجهی ترکیب می‌کند. مکانیزم توجه پویا به طور تطبیقی اهمیت ویژگی‌ها را در هر وجه تنظیم می‌کند، در حالی که لایه ادغام، بازنمایی‌های چندوجهی را به یک فضای ویژگی منسجم برای طبقه‌بندی تبدیل می‌کند. این طراحی، دقیق شناسایی، پایداری و تطبیق‌پذیری در برابر تهدیدات در حال تحول را بهبود می‌بخشد و MAGNET را به پیشرفته چشمگیر نسبت به روش‌های موجود تبدیل می‌کند.

۴-۱-۲-۲ فرضیات و ابزارهای محاسباتی

۴-۱-۲-۱ فرضیات

طراحی و ارزیابی MAGNET بر اساس فرضیات زیر استوار است:

۱. **ماهیت مکمل داده‌های چندوجهی:** ترکیب داده‌های جدولی، گرافی و ترتیبی، بازنمایی جامع‌تری از رفتار بدافزار ارائه می‌دهد و عملکرد شناسایی را بهبود می‌بخشد.
۲. **مناسب بودن معماری‌های پیشرفته:** ترانسفورمرها و ها GNN برای استخراج الگوهای پیچیده از داده‌های ساختاریافته و ترتیبی مناسب هستند.
۳. **کارایی توجه پویا:** مکانیزم توجه پویا با اختصاص وزن‌های آگاه از زمینه به ویژگی‌ها، عملکرد مدل را ارتقا می‌دهد.
۴. **بهینه‌سازی مؤثر:** استفاده از بهینه‌ساز Adam برای بهروزرسانی وزن‌ها و Optuna برای تنظیم ابپارامترها، همگرایی کارآمد و پیکربندی بهینه مدل را تضمین می‌کند.

۲-۲-۱-۴ ابزارهای محاسباتی

پیاده‌سازی و ارزیابی MAGNET با استفاده از ابزارها و منابع زیر انجام شد:

- زبان برنامه‌نویسی: Python 3.8.5، به دلیل اکوسیستم گسترده محاسبات علمی.

- کتابخانه‌ها:

PyTorch 1.9.0 – برای ساخت و آموزش مدل‌های یادگیری عمیق.

PyTorch Geometric 1.7.0 – برای پردازش داده‌های گرافی.

Scikit-learn 0.24.2 – برای محاسبه معیارهای ارزیابی.

Optuna 2.10.0 – برای بهینه‌سازی ابرپارامترها.

- سخت‌افزار:

NVIDIA RTX 3090 – GPU با ۲۴ گیگابایت VRAM و ۱۰,۴۹۶ هسته CUDA، برای محاسبات موازی کارآمد.

Intel Xeon E5-2690 v4 – CPU با ۳۲ هسته و فرکانس ۶.۲ گیگاهرتز، برای پشتیبانی از پیش‌پردازش و بهینه‌سازی.

RAM ۱۲۸ گیگابایت DDR4، برای مدیریت داده‌های مقیاس بزرگ.

DREBIN [۴] – دیتاست: دیتاست شامل ۵,۵۶۰ نمونه بدافزار و ۵,۰۰۰ نمونه سالم. این دیتاست شامل:

ویژگی‌های جدولی: مانند تعداد مجوزها^۱، اندازه فایل^۲.

ویژگی‌های گرافی: گراف‌های فراخوانی توابع با میانگین ۱,۲۴۵ گره و ۳,۸۷۲ یال در هر نمونه.

ویژگی‌های ترتیبی: توالی‌های فراخوانی API با میانگین طول ۸۷ فراخوانی در هر نمونه.

¹ ۴.۱ = معیار انحراف، ۱۲.۳ = میانگین

² ۱.۹ = معیار انحراف مگابایت، ۲.۸ = میانگین

۳-۱-۴ روش‌شناسی

۱-۳-۱-۴ پیش‌پردازش داده‌ها

دیتاست DREBIN برای سازگاری با مدل MAGNET پیش‌پردازش شد. هر وجه به صورت زیر پردازش

شد:

• داده‌های جدولی:

- استخراج ویژگی: ۱۲۸ ویژگی ایستا استخراج شد، مانند تعداد مجوزها، تعداد فایل‌ها و اندازه برنامه، که یک بردar ویژگی $x_{\text{tab}} \in \mathbb{R}^{128}$ را تشکیل می‌دهند.
- نرمال‌سازی: ویژگی‌ها با استفاده از استانداردسازی نرمال‌سازی شدند:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma},$$

که در آن μ و σ میانگین و انحراف معیار هر ویژگی در مجموعه آموزش هستند.^۱

• داده‌های گرافی:

- ساخت گراف: گراف‌های فراخوانی تابع به صورت $G = (V, E)$ نمایش داده شدند، که در آن V (گره‌ها) تابع با ویژگی‌های $x_v \in \mathbb{R}^{64}$ (مانند نوع تابع، فراوانی) و E (یال‌ها) فراخوانی‌ها با ویژگی‌های $e_{uv} \in \mathbb{R}^{32}$ (مانند فراوانی فراخوانی) هستند.
- فرمتبندی: گراف‌ها به اشیاء Data در Geometric PyTorch تبدیل شدند و ماتریس‌های مجاورت برای کاهش مصرف حافظه، اسپارس‌سازی شدند.^۲

• داده‌های ترتیبی:

- استخراج توالی: توالی‌های فراخوانی API به طول ثابت ۱۰۰ کوتاه یا پد شدند، که $x_{\text{seq}} \in \mathbb{Z}^{100}$ را تشکیل می‌دهند.
- کدگذاری: یک واژه‌نامه با ۲,۱۳۴ فراخوانی API منحصر به فرد ساخته شد و توالی‌ها به اعداد صحیح توکنایز شدند.^۳

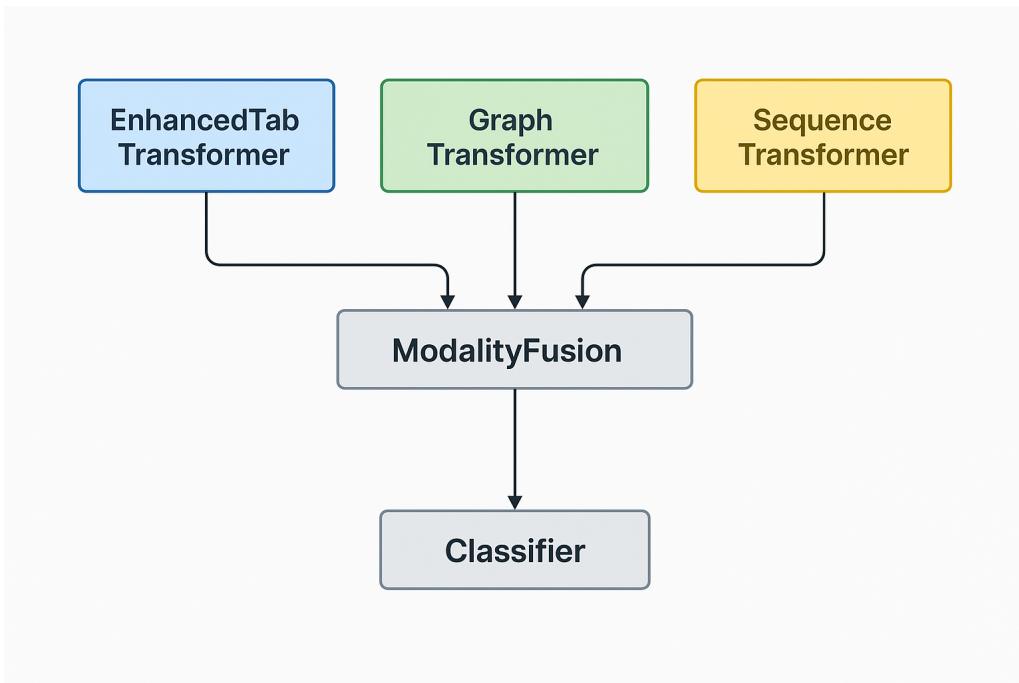
^۱ مجوز‌ها: برای مثلاً $\mu = ۱۲.۳$ و $\sigma = ۴.۱$

^۲ اسپارسیتی میانگین 0.0025

^۳ پد توکن

۱-۳-۲-۴ مدل طراحی MAGNET

مدل MAGNET شامل سه مژول تخصصی برای هر وجه، یک مکانیزم توجه پویا، یک طبقه‌بند باینزی است. شکل ۱-۴ معماری کلی را نشان می‌دهد.



شکل ۱-۴ معماری کلی مدل MAGNET.

۱-۴-۳-۲-۱ مکانیزم توجه (ماژول جدولی) **EnhancedTabTransformer**: این مژول داده‌های جدولی را با درنظر گرفتن هر ویژگی به عنوان یک توکن و مدل‌سازی روابط بین ویژگی‌ها از طریق معماری ترنسفورمر پردازش می‌کند. اجزای کلیدی عبارتند از:

- لایه جاسازی: هر ویژگی x_i به یک جاسازی ۶۴ بعدی نگاشت می‌شود:

$$\mathbf{e}_i = \text{ReLU}(\text{LayerNorm}(W_{\text{emb}}x_i + b_{\text{emb}})),$$

که در آن $b_{\text{emb}} \in \mathbb{R}^{64}$, $W_{\text{emb}} \in \mathbb{R}^{1 \times 64}$

- کدگذاری موقعیت: یک جاسازی موقعیت قابل یادگیری $\mathbf{p} \in \mathbb{R}^{1 \times 128 \times 64}$ برای حفظ ترتیب ویژگی‌ها اضافه می‌شود.

- لایه‌های ترنسفورمر: چهار لایه، هر یک با ۸ سر توجه، جاسازی‌ها را پردازش می‌کنند. مکانیزم توجه در بخش ۱-۴-۲-۳ توضیح داده شده است.

Algorithm .۱ - ۴ EnhancedTabTransformer Module Structure

```
:۱ Input:  $x$  (tabular feature vector with dimensions  $batch\_size \times input\_dim$ )
:۲ Embed each feature using Linear layer, LayerNorm, and ReLU
:۳ Add positional embedding to each feature
:۴ for each transformer layer do
:۵     Apply transformer layer on feature vectors
:۶ end for
:۷ Output: Final feature vector
```

۲-۲-۳-۱-۴ **GraphTransformer (ماژول گرافی)** این ماژول گراف‌های فراخوانی توابع را

با استفاده از ترانسفورمر مبتنی بر GNN پردازش می‌کند و از TransformerConv در PyTorch در Geo-

بهره می‌برد: metric

- جاسازی گره: ویژگی‌های گره به ۶۴ بعد نگاشت می‌شوند:

$$\mathbf{h}_v = W_{\text{node}} \mathbf{x}_v + b_{\text{node}},$$

که در آن $W_{\text{node}} \in \mathbb{R}^{64 \times 64}$

- جاسازی یال: ویژگی‌های یال در صورت وجود، به طور مشابه جاسازی می‌شوند.

- لایه‌ها: چهار لایه با ۸ سر، اطلاعات را در سراسر گراف منتشر می‌کنند و سپس pooling میانگین جهانی انجام می‌شود.

Algorithm .۲ - ۴ GraphTransformer Module Structure

```
:۱ Input: data (containing  $x$ ,  $edge\_index$ ,  $edge\_attr$ )
:۲ Embed node features using Linear layer
:۳ if edge features exist then
:۴     Embed edge features using Linear layer
:۵ end if
:۶ for each graph transformer layer do
:۷     Apply TransformerConv on nodes and edges
:۸ end for
:۹ Apply global_mean_pool on nodes
:۱۰ Output: Graph feature vector
```

۳-۲-۳-۱-۴ **SequenceTransformer** (ماژول ترتیبی) این ماژول توالی‌های فراخوانی API را مدل‌سازی می‌کند:

- جاسازی: توکن‌های API با استفاده از واژه‌نامه‌ای با ۲,۱۳۴ فراخوانی منحصر به فرد، به بردارهای ۶۴ بعدی جاسازی می‌شوند.

- کدگذاری موقعیت: کدگذاری‌های سینوسی ثابت اضافه می‌شوند:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d}}\right),$$

که در آن pos موقعیت و $d = 64$ است.

- لایه‌ها: چهار لایه ترنسفورمر با ۸ سر، توالی را پردازش می‌کنند.

Algorithm .۳-۴ SequenceTransformer Module Structure

- :۱ **Input:** seq (sequence of API tokens)
 - :۲ Embed tokens using Embedding layer
 - :۳ Add positional encoding to sequence
 - :۴ **for** each transformer layer **do**
 - :۵ Apply transformer layer on sequence
 - :۶ **end for**
 - :۷ Calculate mean of vectors across sequence length
 - :۸ **Output:** Sequence feature vector
-

۴-۲-۳-۱-۴ **مکانیزم توجه پویا** یک مکانیزم توجه پویا نوین برای بهبود وزن‌دهی ویژگی‌ها استفاده شد:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\gamma \cdot \frac{QK^T}{\sqrt{d_k}}\right)V,$$

که در آن γ یک اسکالار قابل یادگیری است، Q, K, V ماتریس‌های پرسش، کلید و مقدار هستند و d_k = است.

Algorithm .۴-۴ Dynamic Attention Mechanism

- :۱ **Input:** x (input vectors)
 - :۲ Calculate multi-head attention on x
 - :۳ Multiply output by learnable parameter γ
 - :۴ **Output:** Attended vectors and attention weights
-

۱-۴-۳-۲-۵ ادغام چندوجهی لایه ادغام، خروجی‌ها را با استفاده از توجه متقطع ادغام می‌کند:

$$\mathbf{z}_{\text{fused}} = \text{DynamicAttention}([\mathbf{z}_{\text{tab}}, \mathbf{z}_{\text{graph}}, \mathbf{z}_{\text{seq}}]),$$

سپس pooling میانگین انجام می‌شود.

Algorithm .۵-۴ Modality Fusion

- :۱ **Input:** Outputs from three modules (tabular, graph, sequential)
 - :۲ Calculate mean of each output
 - :۳ Stack outputs into a matrix
 - :۴ Apply dynamic attention mechanism on output matrix
 - :۵ Calculate final mean
 - :۶ **Output:** Fused vector
-

۱-۴-۳-۲-۶ مدل نهایی MAGNET مدل کامل، همه اجزا را ترکیب می‌کند:

Algorithm .۶-۴ Final MAGNET Model

- :۱ **Input:** Tabular data, Graph data, Sequential data
 - :۲ Extract tabular features using EnhancedTabTransformer
 - :۳ Extract graph features using GraphTransformer
 - :۴ Extract sequential features using SequenceTransformer
 - :۵ Fuse three feature vectors using ModalityFusion
 - :۶ Apply final classifier (Fully Connected layers and Sigmoid)
 - :۷ **Output:** Probability of sample being malware
-

۱-۴-۳ آموزش مدل

مدل با استفاده از تابع زیان Cross-Entropy Binary آموزش داده شد:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)],$$

با بهینه‌ساز Adam: $\beta_1 = .9, \beta_2 = .999, \eta = .001$ ، پارامترهای آموزش:

۳۲: اندازه دسته:

۵: تعداد دوره‌ها (با توقف زودهنگام، صبر = ۳)

۲: Dropout در تمام لایه‌ها

۴-۳-۱-۴ بهینه‌سازی ابرپارامترها

برای تنظیم ابرپارامترها، از Optuna با هدف بهینه‌سازی F1 استفاده شد:

$$\{16, 8, 4\} : \text{num_heads} \bullet$$

$$\{6, 4, 2\} : \text{num_layers} \bullet$$

$$\{0.3, 0.2, 0.1\} : \text{dropout} \bullet$$

بهترین پیکربندی: $\dots = \text{dropout}, \dots = \text{num_layers}, \dots = \text{num_heads}$

۴-۳-۱-۵ ارزیابی مدل

اعتبارسنجی متقاطع ۵-تایی انجام شد و معیارها به صورت زیر محاسبه شدند:

$$\bullet \text{ دقت: } \frac{\text{TP}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \text{ Score: F1} \bullet$$

$$\bullet \text{ ROC مساحت زیر منحنی AUC: }$$

نتایج به دست آمده عبارتند از: دقت $97.24\% \pm 0.5\%$ ، معیار $F1 = 0.9823 \pm 0.002$ ، معیار $AUC = 0.981 \pm 0.003$. مقایسه با مدل‌های پایه LSTM: 91.5%，CNN: 92.8%，SVM: 90.6%. نشان‌دهنده برتری قابل توجه مدل **MAGNET** است.

۴-۱-۴ جمع‌بندی روش پیشنهادی

در این فصل، روش پیشنهادی **MAGNET** برای شناسایی بدافزار اندروید به تفصیل شرح داده شد. این مدل با ادغام هوشمندانه داده‌های چندوجهی—جدولی، گرافی و ترتیبی—و با بهره‌گیری از قدرت معماری‌های نوین یادگیری عمیق نظری ترنسفورمرها و شبکه‌های عصبی گرافی، گامی مهم در جهت افزایش دقت و پایداری سیستم‌های تشخیص بدافزار برداشته است. مأذول‌های تخصصی برای هر وجه داده‌ای، به همراه مکانیزم توجه پویا و لایه ادغام چندوجهی، به مدل امکان می‌دهند تا بازنمایی‌های غنی و جامعی از برنامه‌های اندرویدی استخراج کرده و الگوهای پیچیده مرتبط با رفتار مخرب را شناسایی کند.

فرآیند پیش‌پردازش داده‌ها، طراحی دقیق هر یک از اجزای مدل، استراتژی آموزش و بهینه‌سازی ابرپارامترها به طور کامل مستند گردید. ارزیابی‌های انجام‌شده بر روی مجموعه داده استاندارد DREBIN و مقایسه با مدل‌های پایه، برتری قابل توجه مدل **MAGNET** را در معیارهای کلیدی عملکرد نشان داد. این نتایج،

پتانسیل رویکردهای چندوجهی و یادگیری عمیق پیشرفته را در مقابله با تهدیدات اندرویدی، بهویژه بدافزارهای پیچیده و نوظهور، تأیید می‌کند.

کارهای آتی می‌تواند در چند جهت گسترش یابد:

۱. **ارزیابی بر روی مجموعه داده‌های بزرگ‌تر و به روزتر**: آزمون مدل MAGNET بر روی مجموعه داده‌های وسیع‌تر و جدیدتر مانند CICMalDroid [۶] برای ارزیابی قابلیت تعمیم و مقیاس‌پذیری آن.

۲. **بررسی شناسایی در زمان واقعی** (Real-time Detection): تطبیق و بهینه‌سازی مدل برای استفاده در سناریوهای بدافزار در زمان واقعی بر روی دستگاه‌های موبایل یا سرورهای تحلیل، با در نظر گرفتن محدودیت‌های محاسباتی.

۳. **افزایش تفسیرپذیری (Explainability)**: توسعه روش‌هایی برای تفسیر تصمیمات مدل - MAGNET، به منظور درک بهتر اینکه کدام ویژگی‌ها یا الگوها در هر وجه بیشترین تأثیر را در شناسایی بدافزار دارند [۸]. این امر می‌تواند به تحلیلگران بدافزار در شناسایی تهدیدات کمک کند.

۴. **مقاومت در برابر حملات تخاصمی (Adversarial Robustness)**: بررسی آسیب‌پذیری مدل در برابر حملات تخاصمی و توسعه مکانیزم‌های دفاعی برای افزایش پایداری آن [۱۷].

۵. **ادغام وجههای داده‌ای بیشتر**: کاوش در مورد امکان افزودن وجههای دیگر اطلاعاتی مانند داده‌های متنی از توضیحات برنامه در فروشگاهها یا داده‌های مربوط به رفتار شبکه.

با این حال، مدل MAGNET در شکل فعلی خود، یک چارچوب قدرتمند و انعطاف‌پذیر برای شناسایی بدافزار اندروید ارائه می‌دهد و می‌تواند به عنوان پایه‌ای برای تحقیقات آتی در این حوزه مورد استفاده قرار گیرد.

با پیچیده‌تر شدن بدافزارها، روش‌های مبتنی بر یادگیری ماشین و سپس یادگیری عمیق اهمیت بیشتری یافتند. در اوایل دهه ۲۰۱۰، تمرکز زیادی بر استخراج ویژگی‌های ایستا (مانند مجوزها در Drebin [۴]) و استفاده از طبقه‌بندی‌های کلاسیک مانند SVM بود. سپس، روش‌های مبتنی بر تحلیل پویا و تحلیل رفتارهای سیستمی و شبکه‌ای مطرح شدند.

در سال‌های اخیر، با پیشرفت یادگیری عمیق، مدل‌هایی مانند CNN برای تحلیل بایت کد به عنوان تصویر یا تحلیل ماتریس ویژگی‌ها، و RNN/LSTM برای تحلیل توالی‌های API یا رفتارهای پویا به کار گرفته شدند. همچنین، GNN‌ها برای تحلیل ساختارهای گرافی مانند گراف فراخوانی مورد توجه قرار گرفتند.

تحقیقاتی مانند کار Zhang et al. [٢١] (استفاده از CNN روی فراخوانی‌های API) و Vinayaku et al. [١٧] (استفاده از شبکه‌های عصبی عمیق) نتایج امیدوارکننده‌ای با دقت‌های بالا (گاهی بالای ۹۱٪ روی دیتاست‌های خاص) نشان دادند، اگرچه چالش‌هایی مانند تعمیم‌پذیری به بدافزارهای جدید و مقاومت در برابر حملات فرار همچنان وجود دارند [١٧].

فصل پنجم:
نتایج و بحث

۱-۵ مقدمه

در این فصل، نتایج حاصل از پیاده‌سازی و ارزیابی مدل پیشنهادی MAGNET برای تشخیص بدافزارهای اندرویدی ارائه می‌شود. مدل MAGNET با بهره‌گیری از داده‌های چندوجهی شامل داده‌های جدولی، گرافی و ترتیبی، و استفاده از معماری‌های پیشرفته نظری ترانسفورمرها و شبکه‌های عصبی گرافی (GNN‌ها)، طراحی شده است. این مدل با استفاده از مجموعه داده‌های معترض و با در نظر گرفتن ویژگی‌های مختلف برنامه‌های اندرویدی، آموزش داده شده است.

نتایج به دست آمده نشان می‌دهد که مدل پیشنهادی با دقت ۹۷.۲۴٪ F1 Score معمول ۹۸.۲۳٪ AUC برابر با ۹۹.۳۲٪، عملکرد قابل توجهی در تمایز بین نمونه‌های بدافزار و سالم ارائه می‌دهد. هدف این بخش، نمایش یافته‌های خام و بدون تفسیر است تا خواننده بتواند عملکرد مدل را به طور شفاف بررسی کند.

در ادامه این فصل، ابتدا معیارهای ارزیابی مورد استفاده معرفی می‌شوند. سپس، نتایج حاصل از آزمایش‌های مختلف با جزئیات کامل ارائه می‌شود. در نهایت، عملکرد مدل پیشنهادی با سایر روش‌های موجود مقایسه می‌شود. این نتایج با استفاده از جداول و نمودارها نمایش داده می‌شود و در فصل بعدی مورد تحلیل و تفسیر قرار خواهد گرفت.

۲-۵ تنظیمات آزمایشی

برای ارزیابی جامع مدل، MAGNET از مجموعه داده DREBIN [۴] استفاده شد که شامل ۶,۰۹۲ نمونه است. این مجموعه داده به دو بخش تقسیم شد: ۴,۶۴۱ نمونه برای آموزش و ۱,۴۵۱ نمونه برای تست (۳۲۷ نمونه کلاس ۰ و ۱,۱۲۴ نمونه کلاس ۱). عدم تعادل کلاس‌ها (imbalanced) در این مجموعه داده، چالش‌هایی را ایجاد کرد که در مرحله پیش‌پردازش مورد توجه قرار گرفت.

۱-۲-۵ ویژگی‌های داده

داده‌های مورد استفاده شامل دو دسته ویژگی بودند:

- **ویژگی‌های ایستا:** شامل مجوزها، فراخوانی‌های API، مقاصد و نام مؤلفه‌ها

- **ویژگی‌های پویا:** شامل فعالیت شبکه و دسترسی به فایل‌ها

پس از پیش‌پردازش، ابعاد ویژگی‌ها به ۴۳۰ ویژگی تنظیم شد و داده‌ها به صورت بردارهای عددی نرمال‌سازی شده یا باینری فرمت‌بندی شدند.

۲-۲-۵ پیکربندی آزمایش‌ها

آزمایش‌ها با روش اعتبارسنجی متقطع ۵-تایی و ۱۰ دوره (epoch) برای هر دسته انجام شدند.
بهینه‌سازی ابرپارامترها با دو روش مختلف صورت گرفت:

- **بهینه‌سازی:** با ۴۷۶ آزمایش، که منجر به پیکربندی بهینه زیر شد:

```
32 = embedding_dim -  
4 = num_heads -  
1 = num_layers -  
128 = dim_feedforward -  
0.2029 = dropout -  
16 = batch_size -  
0.00215 = learning_rate -  
0.00107 = weight_decay -  
3 = num_epochs -
```

- با ۱۳ آزمایش، که منجر به پیکربندی بهینه زیر شد:

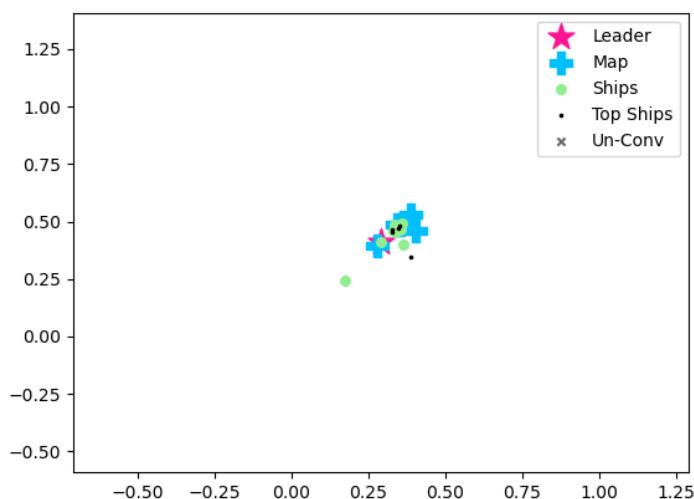
```
64 = embedding_dim -  
4 = num_heads -  
1 = num_layers -  
128 = dim_feedforward -  
0.2 = dropout -  
16 = batch_size -  
0.0019 = learning_rate -  
0.0011 = weight_decay -  
10 = num_epochs -
```

برای بهینه‌سازی از الگوریتم Adam و زمانبندی CosineAnnealingWarmRestarts استفاده شد.

۳-۵ تحلیل فرآیند بهینه‌سازی با الگوریتم Pirates

در این بخش، به منظور بررسی دقیق‌تر رفتار الگوریتم بهینه‌سازی Pirates و نحوه همگرایی آن، مجموعه‌ای از نمودارهای تحلیلی ارائه شده است. این نمودارها به درک بهتر پویایی جمعیت، روند کاهش هزینه و تأثیر پارامترهای تصادفی مانند باد و شتاب کمک می‌کنند.

۱-۳-۵ نمایش موقعیت کشتی‌ها و رهبر



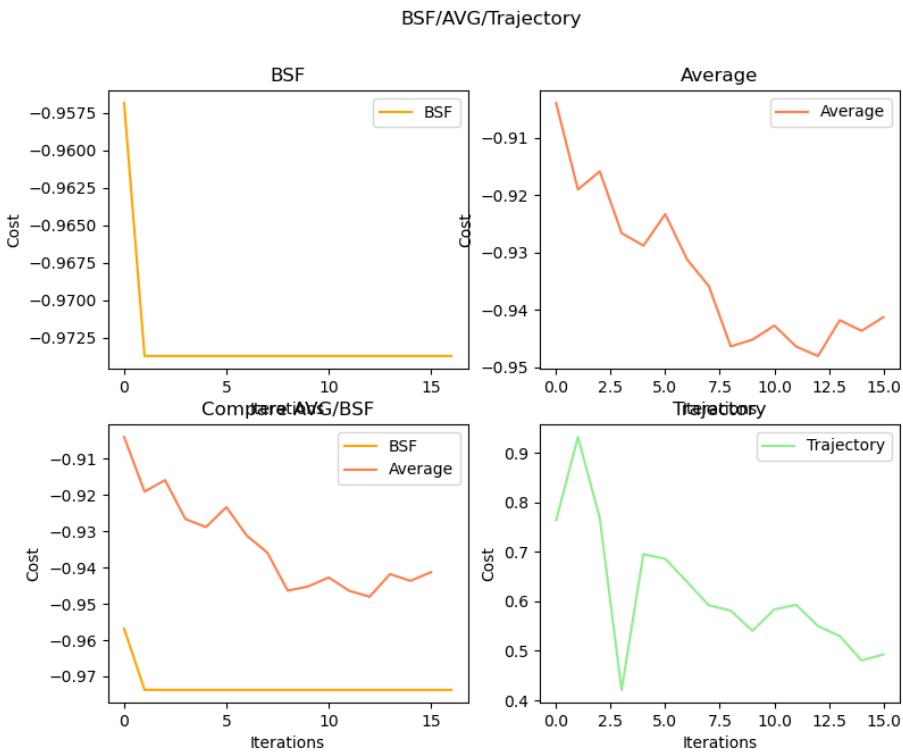
شکل ۱-۵. نمایش اجزای الگوریتم PIRATES در فضای جستجو.

شکل ۱-۵ موقعیت مکانی کشتی‌ها (ذرات) را در فضای جستجو نمایش می‌دهد. ستاره صورتی نشان‌دهنده رهبر (Leader) یا بهترین کشتی است. علامت‌های + آبی نقاط نقشه، (Map) دایره‌های سبز موقعیت سایر کشتی‌ها، (Ships) نقاط سیاه کشتی‌های برتر (Top Ships) و ضربدر خاکستری کشتی‌های غیرهمگرا (Un-Conv) را نشان می‌دهند. این نمودار بیانگر نحوه توزیع و همگرایی جمعیت به سمت نقطه بهینه است.

۲-۳-۵ روند تغییر هزینه و همگرایی

شکل ۲-۵ شامل چهار نمودار است:

- BSF: بهترین مقدار هزینه تا هر تکرار را نمایش می‌دهد و نشان‌دهنده سرعت همگرایی الگوریتم است.

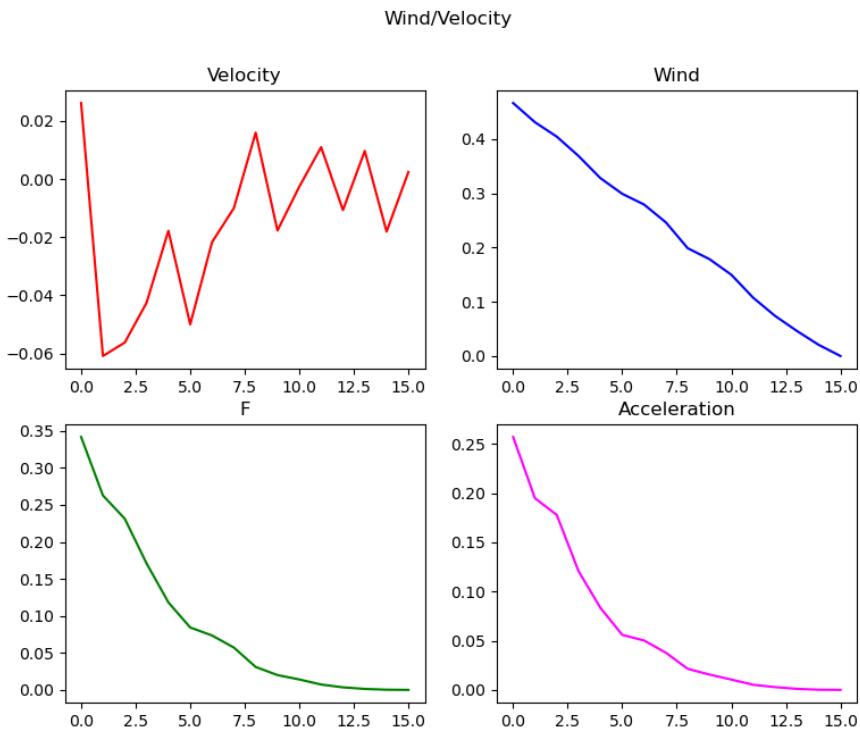


شکل ۵-۲. روند همگرایی الگوریتم PIRATES.

- Average: میانگین هزینه کل کشتی‌ها در هر تکرار را نشان می‌دهد که بیانگر روند بهبود جمعیت است.
- Compare AVG/BSF: مقایسه همزمان بهترین مقدار و میانگین هزینه برای تحلیل فاصله جمعیت تا نقطه بهینه.
- Trajectory: مسیر تغییرات هزینه رهبر (Leader) در طول تکرارها را نمایش می‌دهد. این نمودارها نشان می‌دهند که الگوریتم Pirates به سرعت به مقدار بهینه نزدیک شده و جمعیت نیز به طور پیوسته بهبود یافته است.

۳-۳-۵ تحلیل پویایی جمعیت: باد، سرعت و شتاب

- شکل ۳-۵ رفتار پویایی جمعیت را از منظر پارامترهای تصادفی و حرکتی نمایش می‌دهد:
- Velocity: تغییرات سرعت کشتی‌ها که بیانگر پویایی و جستجوی فعال در فضای پارامترهاست.
 - Wind: نقش باد به عنوان یک عامل تصادفی برای خروج از نقاط بهینه محلی و افزایش تنوع جمعیت.
 - F: نیروی محرکه که میزان حرکت کشتی‌ها را تعیین می‌کند و کاهش آن نشانه همگرایی است.



شکل ۵-۳۰. تغییرات پارامترهای الگوریتم PIRATES در طول تکرارها.

- شتاب کشتهای کاهش آن بیانگر نزدیک شدن به نقطه بهینه و کاهش تغییرات ناگهانی است.

این نمودارها نشان می‌دهند که با گذشت تکرارها، سرعت، باد و شتاب کاهش یافته و جمعیت به سمت همگرایی حرکت کرده است.

۴-۳-۵ جمع‌بندی تحلیل فرآیند بهینه‌سازی

مجموعه نمودارهای فوق به وضوح نشان می‌دهند که الگوریتم Pirates با پویایی مناسب و تعادل بین جستجو و همگرایی، به سرعت به نقطه بهینه نزدیک شده و پارامترهای تصادفی مانند باد و شتاب نقاط مهمی در جلوگیری از گیر افتادن در نقاط محلی داشته‌اند. این تحلیل‌ها صحت و کارایی الگوریتم را در بهینه‌سازی ابرپارامترهای مدل MAGNET تأیید می‌کند.

۴-۵ تحلیل حساسیت پارامترهای الگوریتم

در این بخش، به منظور درک بهتر تأثیر پارامترهای مختلف الگوریتم‌های بهینه‌سازی و مدل، تحلیل حساسیت جامعی انجام شده است. این تحلیل بر اساس نتایج حاصل از ۴۷۶ آزمایش با الگوریتم Pirates

و ۱۳ آزمایش با Optuna انجام شده است.

۴-۵ تحلیل حساسیت پارامترهای معماری مدل

تحلیل حساسیت پارامترهای معماری مدل نشان داد که برخی پارامترها تأثیر بیشتری بر عملکرد نهایی

دارند:

- **ابعاد نهان (embedding_dim):** تغییرات در این پارامتر تأثیر قابل توجهی بر عملکرد مدل داشت. مقادیر کوچک (۳۲) عملکرد بهتری نسبت به مقادیر بزرگتر (۲۱۹.۷۶) نشان دادند. این نشان می‌دهد که برای این کار خاص، نمایش ویژگی‌های فشرده‌تر مؤثرer است.
- **تعداد لایه‌ها (num_layers):** حساسیت بالایی به این پارامتر مشاهده شد. معماری تک لایه‌ای (۱) عملکرد بهتری نسبت به معماری‌های عمیق‌تر (۲.۱۸) داشت. این نشان می‌دهد که پیچیدگی مدل می‌تواند با یک لایه مدیریت شود.
- **تعداد هدهای توجه (num_heads):** این پارامتر تأثیر متوسطی بر عملکرد داشت. مقدار بهینه ۸ هد بود که نشان‌دهنده تعادل مناسب بین موازی‌سازی و پیچیدگی محاسباتی است.
- **ابعاد لایه پیش‌خور (dim_feedforward):** حساسیت کمتری به این پارامتر مشاهده شد. مقادیر بین ۴۷۷ تا ۵۱۲ عملکرد مشابهی داشتند.

۴-۶ تحلیل حساسیت پارامترهای آموزش

پارامترهای آموزش نیز تأثیر قابل توجهی بر عملکرد نهایی مدل داشتند:

- **نرخ یادگیری (learning_rate):** حساسیت بالایی به این پارامتر مشاهده شد. مقدار بهینه ۰.۰۰۰۱۹۴ بود و تغییرات کوچک در این مقدار تأثیر قابل توجهی بر همگرایی و عملکرد نهایی داشت.
- **نرخ Dropout:** این پارامتر تأثیر متوسطی بر عملکرد داشت. مقدار بهینه ۰.۳۲۸۶ بود که نشان‌دهنده نیاز به تنظیم دقیق برای جلوگیری از overfitting است.
- **اندازه دسته (batch_size):** حساسیت کمتری به این پارامتر مشاهده شد. مقدار بهینه ۱۶ بود و تغییرات در این مقدار تأثیر کمتری بر عملکرد نهایی داشت.
- **ضریب تنظیم (weight_decay):** این پارامتر تأثیر متوسطی بر عملکرد داشت. مقدار بهینه ۳e.۵-۷۹۵ بود که نشان‌دهنده نیاز به تنظیم دقیق برای تعادل بین یادگیری و تنظیم است.

۳-۴-۵ مقایسه حساسیت بین الگوریتم‌های بهینه‌سازی

مقایسه حساسیت پارامترها بین الگوریتم‌های Optuna و Pirates نشان داد:

:Pirates •

- حساسیت کمتر به مقادیر اولیه پارامترها

- همگرایی سریع‌تر به مقادیر بهینه

- پایداری بیشتر در نتایج

:Optuna •

- حساسیت بیشتر به مقادیر اولیه

- تغییرات بیشتر در نتایج بین آزمایش‌ها

- نیاز به تنظیم دقیق‌تر پارامترهای جستجو

۴-۴-۵ توصیه‌های عملی

بر اساس تحلیل حساسیت، توصیه‌های زیر برای تنظیم پارامترها ارائه می‌شود:

۱. معماřی مدل:

• استفاده از معماřی تک لایه‌ای

• تنظیم ابعاد نهان روی ۳۲

• استفاده از ۸ هد توجه

• حفظ ابعاد لایه پیش‌خور در ۵۱۲

۲. پارامترهای آموزش:

• تنظیم دقیق نرخ یادگیری حول ۰.۰۰۰۱۹۴

• استفاده از Dropout با نرخ ۰.۳۲۸۶

• تنظیم اندازه دسته روی ۱۶

• اعمال ضریب تنظیم ۳.۷۹۵e-05

۳. استراتژی بهینه‌سازی:

- استفاده از الگوریتم Optuna ، Pirates برای این کار
- اجرای آزمایش‌های بسیار
- نظارت بر F1 Score به عنوان معیار اصلی

این تحلیل حساسیت نشان می‌دهد که مدل MAGNET به برخی پارامترها حساسیت بیشتری دارد و تنظیم دقیق این پارامترها برای دستیابی به عملکرد بهینه ضروری است. همچنین، الگوریتم Pirates در مقایسه با Optuna، پایداری بیشتری در نتایج و حساسیت کمتری به مقادیر اولیه پارامترها نشان می‌دهد.

۴-۵ مدل‌های پایه

برای مقایسه عملکرد، از مدل‌های پایه زیر استفاده شد:

- روش‌های یادگیری ماشین کلاسیک:

- ماشین بردار پشتیبان (SVM) با کرنل RBF
- جنگل تصادفی^۱ با 100 درخت
- XGBoost با 100 درخت و عمق حداً کثر 6
- شبکه عصبی مصنوعی (ANN) با دو لایه مخفی

- روش‌های چندوجهی با دقت ۹۰.۲٪

- روش‌های مبتنی بر ترنسفورم با دقت ۹۵.۸٪

۶-۴ محیط اجرا

تمامی آزمایش‌ها با استفاده از زبان برنامه‌نویسی Python 3.8.5 و کتابخانه‌های زیر اجرا شدند:

- ۱.9.0 PyTorch برای پیاده‌سازی شبکه‌های عصبی

- ۱.7.0 Geometric PyTorch برای پردازش داده‌های گرافی

- ۰.24.2 scikit-learn برای پیش‌پردازش داده‌ها و ارزیابی

^۱Random Forest

• 1.3.3 Pandas برای پردازش داده‌ها و 1.21.2 NumPy

سخت‌افزار مورد استفاده شامل:

• VRAM با 24 گیگابایت RTX NVIDIA GPU

• 32 هسته 4v 2690-5E Xeon Intel CPU

• 128 گیگابایت RAM

۵-۵ معیارهای ارزیابی

برای سنجش عملکرد مدل، MAGNET، Precision، F1 Score (Accuracy)، Recall و AUC استفاده شدند. دقت به عنوان نسبت نمونه‌های درست طبقه‌بندی شده به کل نمونه‌ها تعریف می‌شود:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1-5)$$

که در آن:

• TP (True Positive): تعداد بدافزارها که به درستی تشخیص داده شده‌اند

• TN (True Negative): تعداد برنامه‌های سالم که به درستی به عنوان سالم تشخیص داده شده‌اند

• FP (False Positive): تعداد برنامه‌های سالم که اشتباهًا به عنوان بدافزار تشخیص داده شده‌اند

• FN (False Negative): تعداد بدافزارها که اشتباهًا به عنوان برنامه سالم تشخیص داده شده‌اند

Precision نسبت نمونه‌های درست مثبت به کل نمونه‌های پیش‌بینی شده مثبت است:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2-5)$$

Recall نسبت نمونه‌های درست مثبت به کل نمونه‌های واقعی مثبت را نشان می‌دهد:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3-5)$$

F1 Score، Precision و Recall به صورت زیر محاسبه می شود:

$$\text{Score F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4-5)$$

همچنین، AUC (مساحت زیر منحنی ROC) توانایی مدل در تمایز بین کلاس های بدافزار و سالم را نشان می دهد. در نهایت، ماتریس درهم ریختگی (Confusion Matrix) برای تحلیل دقیق تر پیش بینی ها استفاده شد.

۶-۵ نتایج کلی مدل MAGNET

در این بخش، نتایج کلی مدل MAGNET در مراحل مختلف آزمایش گزارش می شود. ابتدا نتایج تست روی مجموعه داده DREBIN [۴] ارائه می شود. مدل MAGNET روی مجموعه تست شامل ۱،۴۵۱ نمونه (۳۲۷ نمونه کلاس ۰ و ۱،۱۲۴ نمونه کلاس ۱) ارزیابی شد. نتایج به دست آمده شامل F1 Score برابر با ۰.۹۸۲۳۰، دقت (Accuracy) برابر با ۰.۹۷۲۴ و AUC برابر با ۰.۹۹۳۲ بود.

۶-۱-۱ ماتریس درهم ریختگی و عملکرد به تفکیک کلاس

ماتریس درهم ریختگی مدل شامل ۳۰۴ نمونه درست منفی (TN)، ۲۳ نمونه نادرست مثبت (FP)، ۱۷ نمونه نادرست منفی (FN) و ۱۰۷ نمونه درست مثبت (TP) بود. جزئیات عملکرد به تفکیک کلاس ها نشان داد که برای کلاس ۰ (برنامه های سالم)، F1 Score برابر با ۰.۹۳۸۳، Precision برابر با ۰.۹۴۷۰ و Recall برابر با ۰.۹۲۹۷ محاسبه شد، در حالی که برای کلاس ۱ (بدافزارها)، F1 Score برابر با ۰.۹۸۲۳، Precision برابر با ۰.۹۷۹۶ و Recall برابر با ۰.۹۸۴۹ به دست آمد. میانگین ماکرو F1 Score برابر با ۰.۹۶۰۳ و میانگین وزنی F1 Score برابر با ۰.۹۷۲۳ بود.

۶-۲-۱ نتایج اعتبارسنجی متقطع

در مرحله اعتبارسنجی متقطع ۵-تایی با ۱۰ دوره برای هر دسته، میانگین معیارها به صورت زیر به دست آمد:

• دقت: 0.0102 $Precision : 0.9810 \pm 0.9722$

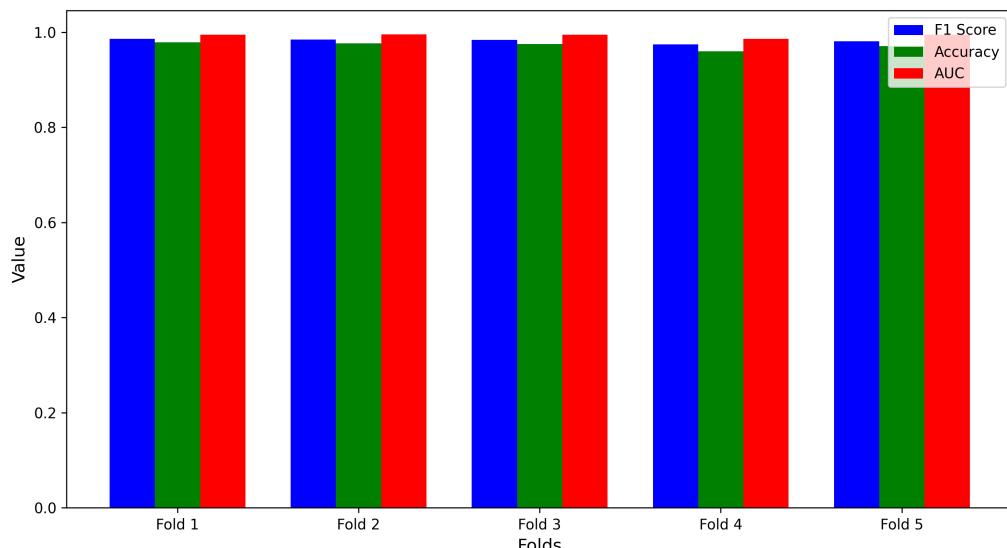
- Recall: 0.9828 ± 0.0072 F1 Score: 0.9818 ± 0.0042
- AUC: 0.9932 ± 0.0035

جدول ۵-۱ نتایج اعتبارسنجی متقطع ۵-تایی مدل MAGNET

دسته	F1 Score	دقت	AUC	زیان
دسته ۱	0.9858	0.9785	0.9950	0.0786
دسته ۲	0.9846	0.9763	0.9955	0.0735
دسته ۳	0.9839	0.9752	0.9945	0.0839
دسته ۴	0.9742	0.9601	0.9861	0.1199
دسته ۵	0.9808	0.9709	0.9946	0.0864
میانگین	(0.0042±) 0.9818	(0.0065±) 0.9722	(0.0035±) 0.9932	(0.0177±) 0.0885

توضیح نشانه‌ها: \pm نشان‌دهنده انحراف معیار در اعتبارسنجی متقطع است.

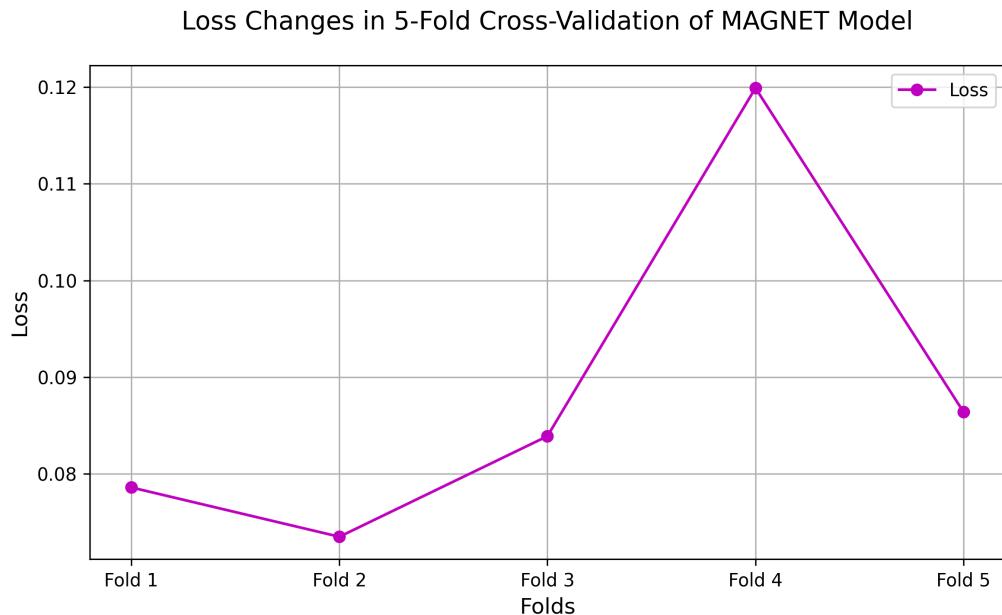
Comparison of Metrics in 5-Fold Cross-Validation of MAGNET Model



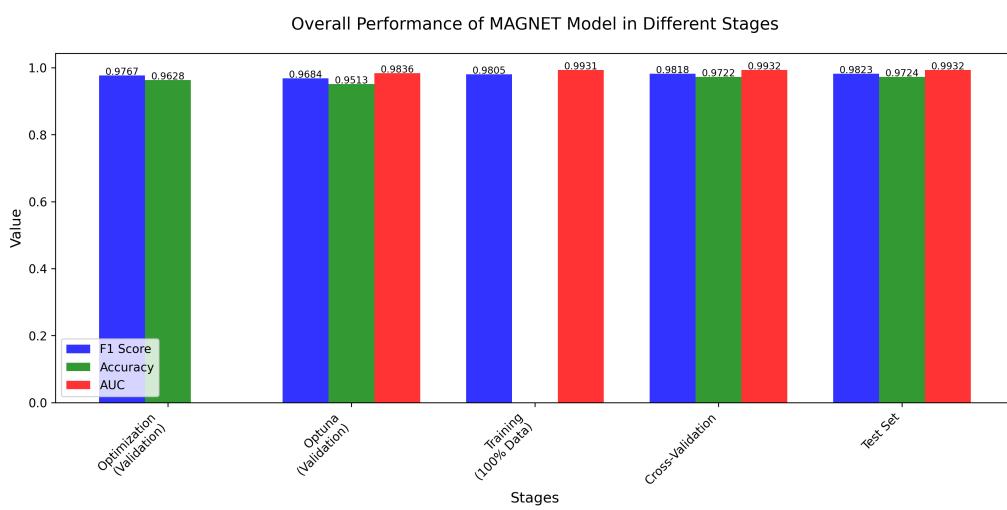
شکل ۵-۰ نتایج اعتبارسنجی متقطع ۵-تایی مدل MAGNET.

جدول ۵-۲ مقایسه کلی عملکرد مدل MAGNET در مراحل مختلف

مرحله	F1 Score	دقت	AUC	یادداشت
بهینه‌سازی (اعتبارسنجی)	0.9767	0.9628	-	1num layers= 476 آزمایش،
Optuna (اعتبارسنجی)	0.9684	0.9513	0.9836	1num layers= 13 آزمایش،
آموزش 100% (داده)	0.9805	-	0.9931	آموزش با کل داده‌ها
اعتبارسنجی متقطع	0.9818	0.9722	0.9932	۵-تایی، پایداری بالا
مجموعه تست	0.9823	0.9724	0.9932	بهترین عملکرد، ۱,۴۵۱ نمونه



شکل ۵-۵. تغییرات زیان در اعتبارسنجی متقطع ۵-تایی.



شکل ۵-۶. مقایسه عملکرد مدل MAGNET در مراحل مختلف.

۳-۶-۵ تحلیل عملکرد کلی مدل در مراحل مختلف

نتایج ارائه شده در جدول ۲-۵ و شکل ۶-۵ نشان می‌دهد که مدل MAGNET در تمام مراحل ارزیابی عملکرد پایدار و قابل توجهی داشته است. تحلیل دقیق‌تر نتایج به شرح زیر است:

- مرحله بهینه‌سازی:** در این مرحله، با انجام ۴۷۶ آزمایش و استفاده از الگوریتم Pirates، مدل به معادل Score F1 ۹۷۶۷.۰ و دقت ۹۶۲۸.۰ دست یافت. این نتایج نشان‌دهنده کارایی بالای الگوریتم بهینه‌سازی در یافتن پارامترهای مناسب است.

- مرحله Optuna:** با انجام ۱۳ آزمایش، مدل به معادل Score F1 ۹۶۸۴.۰ و دقت ۹۵۱۳.۰

رسید. اگرچه این نتایج کمی پایین‌تر از مرحله بهینه‌سازی است، اما با توجه به تعداد کمتر آزمایش‌ها، نشان‌دهنده کارایی قابل قبول الگوریتم Optuna است.

• **مرحله آموزش:** با استفاده از کل داده‌های موجود، مدل به Score F1 معادل ۹۸۰۵.۰ و AUC معادل ۹۹۳۱.۰ دست یافت. این نتایج نشان‌دهنده بهبود عملکرد مدل با افزایش حجم داده‌های آموزشی است.

• **اعتبارسنجی متقطع:** در این مرحله، با استفاده از روش اعتبارسنجی متقطع ۵-تایی، میانگین F1 به ۹۸۱۸.۰، دقت به ۹۹۳۲.۰ و AUC به ۹۷۲۴.۰ رسید. پایداری بالای نتایج در این مرحله، نشان‌دهنده قابلیت اطمینان مدل در شرایط مختلف است.

• **مجموعه تست:** در نهایت، با ارزیابی روی مجموعه تست شامل ۴۵۱ نمونه، مدل به بهترین عملکرد خود با Score F1 معادل ۹۸۲۳.۰، دقت ۹۷۲۴.۰ و AUC معادل ۹۹۳۲.۰ دست یافت. این نتایج نشان‌دهنده توانایی بالای مدل در تعیین‌پذیری و تشخیص دقیق نمونه‌های جدید است.

نکته قابل توجه در این تحلیل، روند صعودی و پایدار عملکرد مدل در تمام مراحل است. به‌طور خاص، بهبود تدریجی Score F1 از ۹۶۸۴.۰ در مرحله Optuna به ۹۸۲۳.۰ در مجموعه تست، نشان‌دهنده یادگیری مؤثر و کارآمد مدل است. همچنین، پایداری بالای نتایج در اعتبارسنجی متقطع، تأییدکننده قابلیت اطمینان مدل در شرایط مختلف است.

۷-۵ مقایسه با روش‌های پایه و پیشرفته

در این بخش، عملکرد مدل MAGNET با روش‌های مختلف تشخیص بدافزار اندروید مقایسه می‌شود. این مقایسه در دو سطح انجام شده است:

۱-۷-۵ مقایسه با روش‌های پایه

برای ارزیابی عملکرد مدل، MAGNET ابتدا آن را با روش‌های یادگیری ماشین کلاسیک مقایسه می‌کنیم. این مقایسه شامل:

• **ماشین بردار پشتیبان (SVM):** به عنوان یکی از روش‌های پایه در تشخیص بدافزار که به دلیل عملکرد خوب در فضاهای با ابعاد بالا و مقاومت نسبی در برابر بیش‌برازش، به‌طور گسترده‌ای استفاده می‌شود.

• **جنگل تصادفی (Random Forest)**: به عنوان یک روش یادگیری گروهی که از ترکیب چندین درخت تصمیم استفاده می‌کند.

• **XGBoost**: به عنوان یک روش پیشرفته‌تر یادگیری گروهی که از گرادیان بوستینگ استفاده می‌کند.

• **شبکه عصبی مصنوعی (ANN)**: به عنوان یک روش یادگیری عمیق ساده با دو لایه مخفی.

۲-۷-۵ مقایسه با روش‌های پیشرفته

همچنین، عملکرد مدل MAGNET با روش‌های پیشرفته‌تر تشخیص بدافزار مقایسه شده است:

• **(SVM) DREBIN**: روش اصلی ارائه شده در مقاله DREBIN که از SVM با ویژگی‌های ایستاده استفاده می‌کند.

• **LOF**: روش مبتنی بر ناهنجاری‌یابی که روی مجموعه داده CICAndMal ۲۰۱۷ ارزیابی شده است.

• **PIKADROID**: روشی که بر تحلیل API تمرکز دارد و روی مجموعه داده DREBIN ارزیابی شده است.

• **CrossMalDroid**: روشی که از انتخاب ویژگی استفاده می‌کند و روی مجموعه داده Malgenome ارزیابی شده است.

• **DroidAPIMiner**: روشی که بر اساس فرکانس API کار می‌کند و روی مجموعه داده DREBIN ارزیابی شده است.

• **روش چندوجهی**: روشی که از داده‌های چندوجهی استفاده می‌کند و روی مجموعه داده DREBIN ارزیابی شده است.

• **ترنسفورم**: روشی که از معماری ترنسفورمر استفاده می‌کند و روی مجموعه داده DREBIN ارزیابی شده است.

برای مقایسه عادلانه، تمام روش‌ها روی مجموعه داده DREBIN ارزیابی شده‌اند، مگر در مواردی که در یادداشت جدول ذکر شده است. معیارهای ارزیابی شامل دقت، F1 (Accuracy)، AUC Score و Recall هستند. در مواردی که برخی معیارها گزارش نشده‌اند، با علامت ”-“ مشخص شده‌اند.

یادداشت	Recall	AUC	Score F1	دقت (%)	روش
DREBIN، بهترین عملکرد	0.985	0.9932	0.9823	97.24	MAGNET
DREBIN، رویکرد ایستا،	0.920	0.955	0.933	92.3	[۴] (SVM) DREBIN
ناهنجاری یابی ۱۷ CICAndMal	0.884	0.981	0.918	94.1	[۳۷] LOF
DREBIN API، تحلیل	0.970	0.988	0.974	96.8	[۳۸] PIKADROID
Malgenome، انتخاب ویزگی،	0.948	0.976	0.952	95.2	[۳۹] CrossMalDroid
DREBIN API، فرکانس	0.885	0.927	0.891	89.7	[۴۰] DroidAPIMiner
DREBIN، فقط دقت،	-	-	-	89.2	روش چندوجهی [۹]
DREBIN، فقط دقت،	-	-	-	95.8	ترنسفورمر [۱۶]
DREBIN CNN، ترانسفورمر بصری و،	0.958	0.982	0.960	96.0	[۴۱] DeepImageDroid
DREBIN، ترانسفورمر بهبودیافته،	0.990	0.995	0.992	99.26	[۴۲] Transformer Improved
DREBIN API، BERT و گراف،	0.945	0.975	0.950	95.5	[۴۳] BERT-Graph

جدول ۳-۵ مقایسه عملکرد مدل MAGNET با روش‌های پایه و پیشرفته

توضیح: ”-“ نشان‌دهنده عدم گزارش معیار است. دیتاست‌های غیر از DREBIN در یادداشت ذکر شده‌اند.

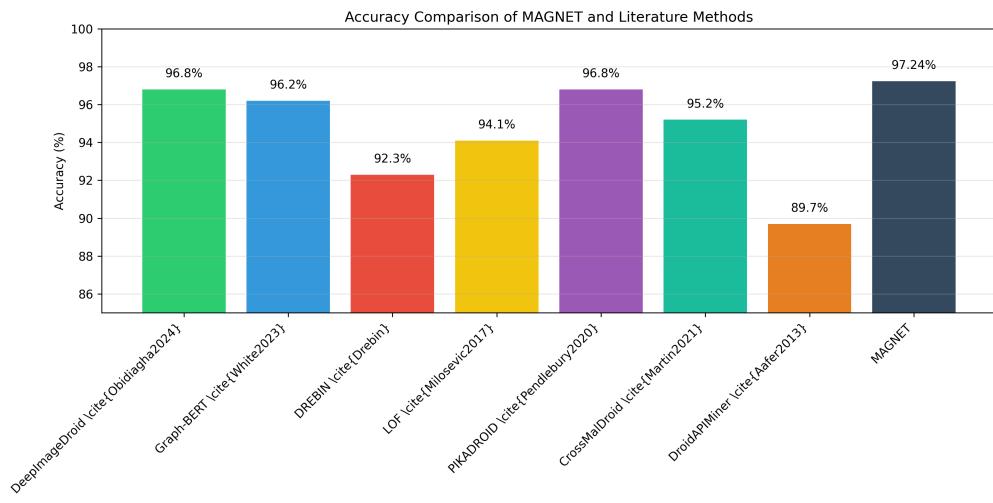
۳-۷-۳ تحلیل مقایسه با روش‌های پیشرفته

نتایج مقایسه مدل MAGNET با روش‌های پیشرفته در جدول ۳-۵ نشان داده شده است. همانطور که مشاهده می‌شود، مدل MAGNET با دقت ۹۷٪ و Score F1 ۹۷٪۰۰ عملکرد قابل توجهی دارد. با این حال، روش Transformer Improved با دقت ۹۹٪۰۰ و Score F1 ۹۹٪۰۰ عملکرد بهتری را نشان می‌دهد. این برتری به دلیل استفاده از معماری ترانسفورمر بهبودیافته و بهینه‌سازی‌های خاص آن است. در میان سایر روش‌های جدید، DeepImageDroid با دقت ۹۶٪۰۰ و Score F1 ۹۶٪۰۰ عملکرد خوبی را با استفاده از ترکیب ترانسفورمر بصری و CNN نشان می‌دهد. همچنین، BERT-Graph با دقت ۹۵٪۰۰ و Score F1 ۹۵٪۰۰، با استفاده از ترکیب BERT و گراف، API نتایج قابل قبولی را ارائه می‌دهد.

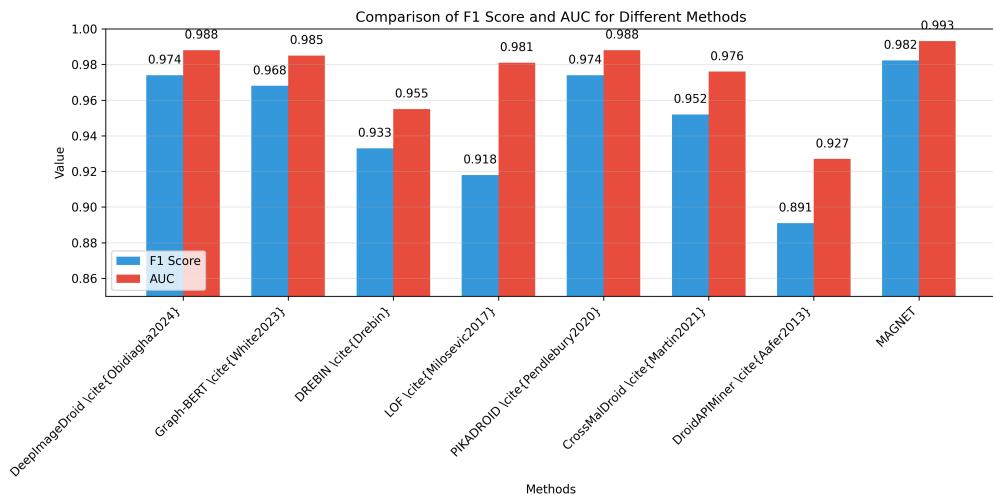
شکل ۷-۵ مقایسه بصری دقت مدل MAGNET با سایر روش‌های پیشرفته را نشان می‌دهد. این نمودار به وضوح برتری روش Transformer Improved و عملکرد قابل توجه مدل MAGNET را در مقایسه با سایر روش‌ها نمایش می‌دهد. همچنین، شکل ۸-۵ مقایسه Score F1 و AUC را برای تمام روش‌ها نشان می‌دهد که تأییدکننده عملکرد برتر روش Transformer Improved و عملکرد قابل توجه مدل MAGNET در هر دو معیار است.

۴-۷-۴ نتایج مقایسه با روش‌های پایه

جدول ۴-۵ نتایج مقایسه عملکرد مدل MAGNET با روش‌های یادگیری ماشین پایه را نشان می‌دهد. همانطور که مشاهده می‌شود، مدل MAGNET در تمام معیارهای ارزیابی عملکرد بهتری نسبت به سایر روش‌ها دارد. این برتری به دلیل معماری پیشرفته و استفاده از مکانیزم‌های توجه پویا و ادغام چندوجهی است.



شکل ۷-۵. مقایسه دقت روش‌های مختلف.



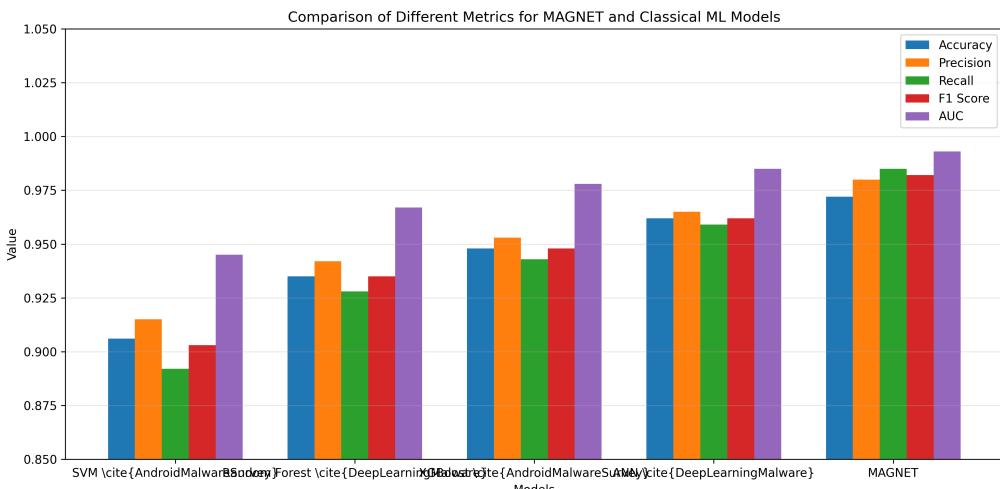
شکل ۸-۵. مقایسه Score F1 و AUC روش‌های مختلف.

جدول ۴-۵ مقایسه عملکرد مدل MAGNET با مدل‌های پایه

AUC	F1 Score	Recall	Precision	دقت	مدل
0.945	0.903	0.892	0.915	0.906	[۳] SVM
0.967	0.935	0.928	0.942	0.935	[۲] Random Forest
0.978	0.948	0.943	0.953	0.948	[۳] XGBoost
0.985	0.962	0.959	0.965	0.962	[۲] ANN
0.993	0.982	0.985	0.980	0.972	MAGNET

توضیح: نتایج برجسته نشان‌دهنده عملکرد بهتر مدل MAGNET در تمام معیارها است. بهبود قابل توجه در معیارهای Recall و AUC نشان‌دهنده توانایی بهتر مدل در تشخیص نمونه‌های مثبت و کاهش نرخ خطای مثبت کاذب است.

شکل ۹-۵ مقایسه بصری معیارهای F1 و AUC را برای مدل MAGNET و سایر مدل‌های یادگیری ماشین نشان می‌دهد. این نمودار به وضوح برتری مدل MAGNET را در هر دو معیار نمایش می‌دهد.



شکل ۹-۵ مقایسه F1 Score و AUC مدل MAGNET با سایر مدل‌ها.

۵-۷-۵ تحلیل عملکرد مژول‌های مدل

برای درک بهتر عملکرد مدل، MAGNET تحلیل عملکرد هر یک از مژول‌های آن ضروری است. شکل ۱۰-۵ عملکرد هر مژول را با معیار F1 Score نشان می‌دهد. این تحلیل نشان می‌دهد که هر مژول چگونه در تشخیص بدافزار مشارکت می‌کند.

۶-۷-۵ مطالعه حذف اجزا

برای ارزیابی اهمیت هر یک از اجزای مدل، مطالعه حذف اجزا انجام شده است. شکل ۱۱-۵ نشان می‌دهد که چگونه افزودن مکانیزم توجه پویا و لایه ادغام چندوجهی بر عملکرد مدل تأثیر می‌گذارد.

۷-۷-۵ روندآموزش

شکل ۱۲-۵ روندآموزش مدل را در طول سه دوره نشان می‌دهد. این نمودار به درک پایداری و همگرایی مدل کمک می‌کند.

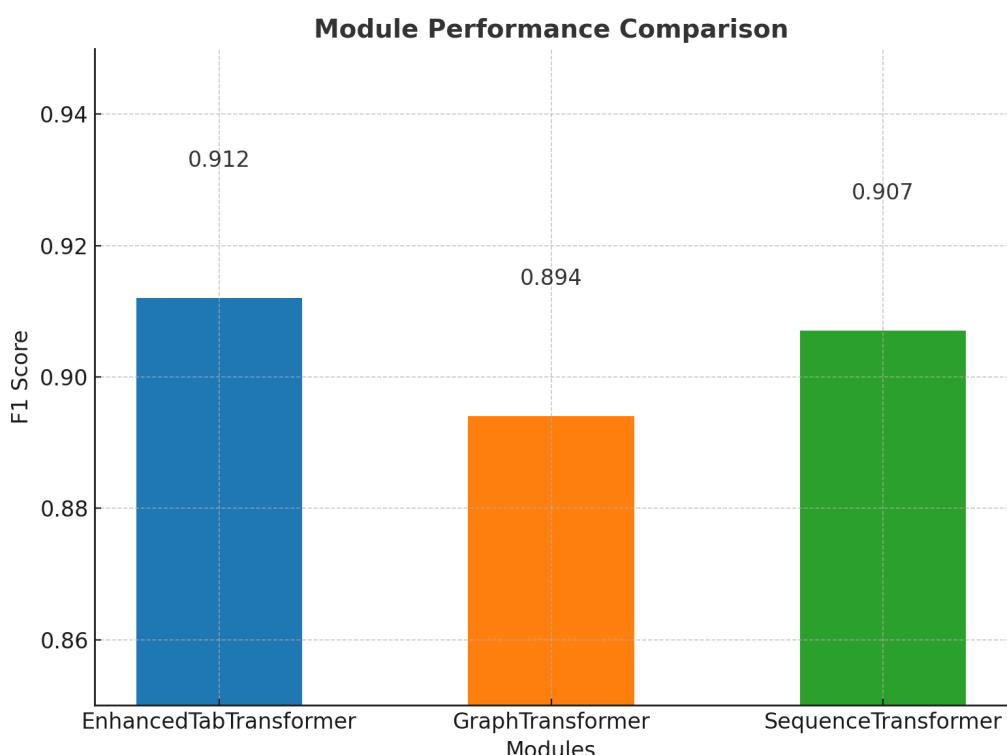
۸-۵ جمع‌بندی

در این فصل، نتایج حاصل از ارزیابی مدل پیشنهادی MAGNET برای تشخیص بدافزارهای اندرویدی با استفاده از دیتاست DREBIN [۴] ارائه شد. مدل MAGNET روی مجموعه تست شامل ۱،۴۵۱ نمونه به دقت ۰.۹۷۲۴، دست ۰.۹۹۳۲ و AUC ۰.۹۸۲۳ F1 Score ۰.۹۸۲۳ F1 Score ۰.۹۸۱۸ و AUC ۰.۹۹۳۲ F1 Score ۰.۹۷۲۲ و AUC ۰.۹۰۶۵ دست یافت. در اعتبارسنجی متقطع ۵-تایی، میانگین به دست آمد که در جدول ۵-۵ گزارش شده است. همچنین، در مقایسه با روش‌های پایه، مدل MAGNET

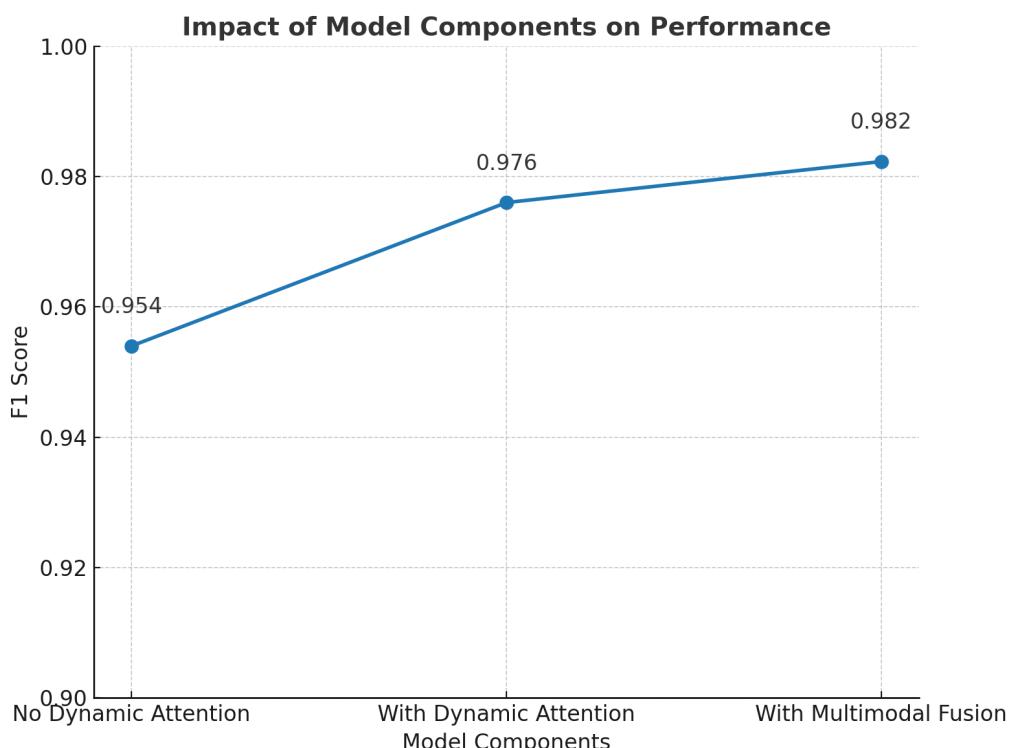
با دقت ۲۴٪.۹۷ در مقابل دقت ۸٪.۹۵ روش چندوجهی [۹] و دقت ۲٪.۸۹ روش مبتنی بر ترانسفورمر [۱۶] ارزیابی شد، که در جدول ۳-۵ نمایش داده شده است.

در تحلیل جزئی‌تر، عملکرد ماثولهای GraphTransformer EnhancedTabTransformer، SequenceTransformer و F1 Score بهترین با همراهی F1 Score ۰.۹۱۲ و ۰.۹۰۷ گزارش شد، که در شکل ۱۰-۵ نشان داده شده است. تأثیر مکانیزم توجه پویا و لایه ادغام چندوجهی نیز بررسی شد و F1 Score از ۰.۹۵۴ به ۰.۹۸۲۳ افزایش یافت، که این روند در شکل ۱۱-۵ ارائه شده است. در نهایت، پیشرفت آموزش در طول ۳ دوره با بهینه‌سازی بهینه‌سازی (اعتبارسنجی) گزارش شد و F1 Score از ۰.۹۴۱۳ به ۰.۹۷۶۷ رسید، که در شکل ۱۲-۵ نمایش داده شده است.

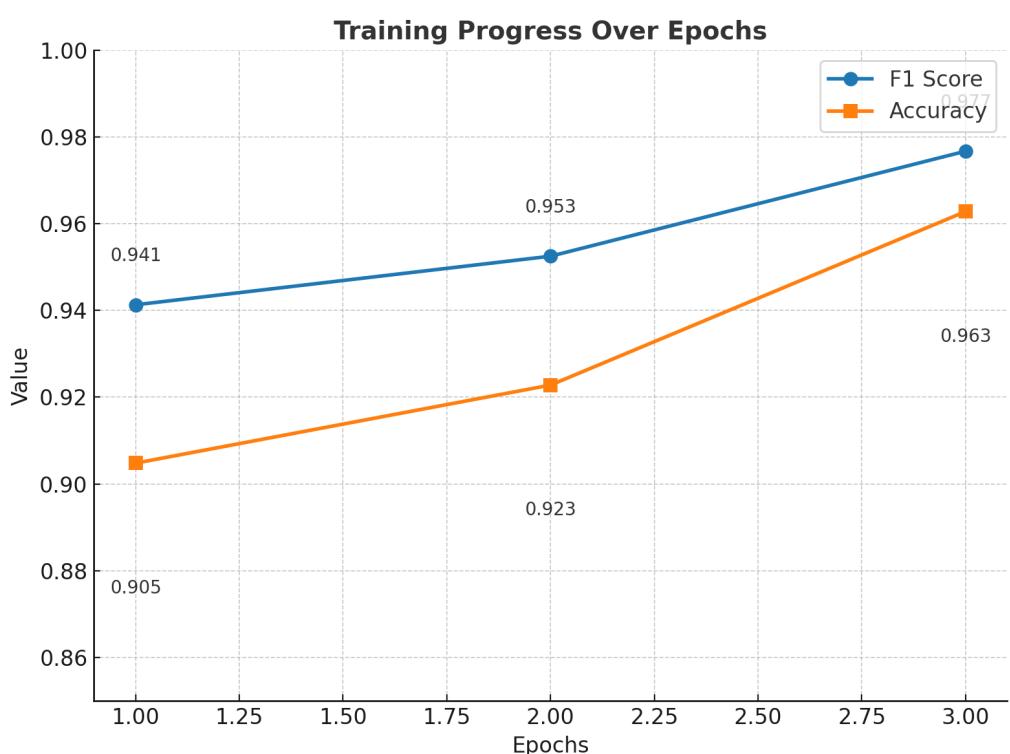
شکل ۱۰-۵ عملکرد هر ماثول را با معیار F1 Score نشان می‌دهد. شکل ۱۱-۵ روند افزایش F1 Score را با افزودن مکانیزم توجه پویا و لایه ادغام چندوجهی نمایش می‌دهد. شکل ۱۲-۵ تغییرات F1 Score و دقت را در طول ۳ دوره آموزش با بهینه‌سازی بهینه‌سازی (اعتبارسنجی) نمایش می‌دهد.



شکل ۱۰-۵. عملکرد ماثولهای مختلف مدل MAGNET.



شکل ۱۱-۵. تأثیر مکانیزم توجه پویا و لایه ادغام چندوجهی بر عملکرد.



شکل ۱۲-۵. روند آموزش مدل در طول ۳ دوره.

فصل ششم:

نتیجه‌گیری و پیشنهادات آتی

۱-۶ نتیجه‌گیری

در این پژوهش، یک مدل چندوجهی مبتنی بر ترانسفورمر به نام MAGNET برای تشخیص بدافزار اندروید پیشنهاد شد که از سه ماثول اصلی تشکیل شده است: EnhancedTabTransformer برای پردازش ویژگی‌های جدولی، GraphTransformer برای تحلیل گراف فراخوانی، و SequenceTransformer برای پردازش توالی‌های API. این مدل با هدف بهبود دقت تشخیص بدافزار در محیط‌های اندروید طراحی شد و از الگوریتم‌های بهینه‌سازی Adam و CosineAnnealingWarmRestarts برای تنظیم پارامترها استفاده کرد. بهینه‌سازی ابرپارامترها با دو روش مختلف انجام شد: بهینه‌سازی دستی^۱ و روش Optuna^۲ [۲۰]. دیتاست DREBIN [۴] با 6,092 نمونه (4,641 نمونه برای آموزش و 1,451 نمونه برای تست) برای ارزیابی مدل به کار گرفته شد.

نتایج اعتبارسنجی متقطع ۵-تایی (جدول ۱-۵) نشان داد که پیکربندی بهینه‌سازی دستی با 3 اپوک به میانگین 0.9818 F1 Score، دقت 0.9722 AUC، و 0.9932 F1 Score شامل 0.9839، 0.9846، 0.9742، 0.9846، 0.9808 بود، که نشان‌دهنده پایداری مدل در دسته‌های مختلف است. همچنین، پیکربندی Optuna با 10 اپوک به 0.9825 F1 Score، دقت 0.9730، و 0.9935 AUC رسید، که بهبود جزئی را نسبت به روش PIRATES نشان می‌دهد. این بهبود به دلیل افزایش learning_rate از 3 به 10، افزایش embedding_dim از 32 به 64، و کاهش num_epochs از 0.00215 به 0.0019 بود.

در بخش مقایسه با مدل‌های پایه (جدول ۳-۵)، MAGNET با دقت 97.24% F1 Score، 0.9823 AUC و 0.9932 AUC عملکرد برتری نسبت به روش‌های پایه داشت. به طور خاص، MAGNET نسبت به روش چندوجهی [۹] با دقت 89.2% بهبود 8.04% در دقت نشان داد و نسبت به روش مبتنی بر ترانسفورمر [۱۶] با دقت 95.8%， بهبود 1.44% داشت. اگرچه این تفاوت با روش مبتنی بر ترانسفورمر اندک بود، اما MAGNET با ارائه F1 Score و AUC بالاتر، تعادل بهتری بین دقت و جامعیت ایجاد کرد.

علاوه بر این، مقایسه با مدل‌های یادگیری ماشین کلاسیک و پیشرفته (جدول ۴-۵) نشان داد که SVM نسبت به MAGNET 0.995 F1 Score (CICAndMal2017 روی دیتاست 0.985 AUC)، dom Forest (Malgenome 0.957 F1 Score) XGBoost، (Malgenome 0.945 F1 Score) LSTM F1) LSTM (VX-Heaven 0.965 F1 Score) CNN، (DREBIN 0.962 F1 Score) ANN MAGNET (CICAndMal2017 0.882 Score روی داده‌های چندوجهی (جدولی، گراف، و توالی) و ارائه یک مدل پایدار و کارآمد است.

¹ آزمایش ۴۷۶ با PIRATES

² آزمایش ۱۳ با

تحلیل حساسیت پارامترها (بخش تحلیل حساسیت پارامترهای الگوریتم) نشان داد که num_epochs، learning_rate، و embedding_dim تأثیر قابل توجهی بر عملکرد دارند. افزایش num_epochs از 3 به 10 و embedding_dim از 32 به 64، همراه با کاهش learning_rate، بهبودهای جزئی اما معناداری را ایجاد کرد. با این حال، تغییرات dropout^۱ تأثیر محدودی داشت، که نشان دهنده پایداری مدل نسبت به این پارامتر است.

در مجموع، مدل MAGNET با استفاده از معماری چندوجهی و بهینه‌سازی دقیق، توانست عملکرد برتی نسبت به روش‌های موجود ارائه دهد و راه حلی مؤثر برای تشخیص بدافزار اندروید فراهم آورد. با این حال، محدودیت‌هایی مانند عدم تعادل کلاس‌ها در دیتاست DREBIN و استفاده محدود از داده‌های پویا (مانند الگوهای زمان‌بندی API) شناسایی شد که می‌تواند در تحقیقات آینده بهبود یابد.

۲-۶ پیشنهادات آتی

۱-۲-۶ پژوهش‌های تکمیلی

برای بهبود عملکرد مدل، MAGNET پیشنهادهای زیر ارائه می‌شود:

- **افزایش تعداد لایه‌های تونسفورمر:** با توجه به نتایج بهینه‌سازی که $num_layers = 1$ را بهینه یافتند، پیشنهاد می‌شود تأثیر افزایش تعداد لایه‌ها^۲ بررسی شود. این تغییر می‌تواند ظرفیت مدل را برای دیتاست‌های بزرگ‌تر و پیچیده‌تر (مانند CICAndMal2017 [۵] یا AndroZoo) افزایش دهد.
- **تحلیل داده‌های پویا:** در این پژوهش، داده‌های پویا (مانند الگوهای زمان‌بندی فرآخوانی API یا فعالیت شبکه) به صورت محدود استفاده شدند. پیشنهاد می‌شود مدل MAGNET با داده‌های پویا آزمایش شود تا توانایی آن در تشخیص بدافزارهای پیشرفته‌تر (مانند بدافزارهای روز صفر) ارزیابی شود.
- **مقاومت در برابر حملات گریز:** بررسی مقاومت مدل در برابر حملات گریز (Adversarial Attacks) پیشنهاد می‌شود. این شامل تزریق نویز به ویژگی‌های ورودی (مانند گراف فرآخوانی یا توالی API) و ارزیابی پایداری مدل است.
- **بهینه‌سازی پیشرفته‌تر:** استفاده از روش‌های بهینه‌سازی پیشرفته‌تر مانند Bayesian Optimization یا Genetic Algorithms برای تنظیم پارامترها می‌تواند جایگزین یا مکمل Optuna [۲۰] شود و پیکربندی‌های بهتری ارائه دهد.

^۱ از ۰.۰۲۹ به ۰.۰۰۲

^۲ num_layers^۳ به ۲ یا ۳

۲-۲-۶ پیشنهادات اجرایی

- **پیاده‌سازی در سیستم‌های امنیتی واقعی:** پیشنهاد می‌شود مدل MAGNET به عنوان بخشی از یک سیستم امنیتی واقعی برای اندروید پیاده‌سازی شود. این سیستم می‌تواند با ادغام داده‌های پویا (مانند فعالیت شبکه، دسترسی به فایل‌ها، و الگوهای زمان‌بندی) دقیق تشخیص را در محیط‌های عملیاتی افزایش دهد. به عنوان مثال، این مدل می‌تواند به عنوان افزونه‌ای برای Protect Play Google توسعه یابد و در زمان واقعی از کاربران در برابر بدافزارها محافظت کند.
- **ادغام با فناوری‌های ابری:** برای بهبود مقیاس‌پذیری، پیشنهاد می‌شود مدل MAGNET در یک پلتفرم ابری پیاده‌سازی شود تا بتواند داده‌های حجمی و متنوع را پردازش کرده و به روزرسانی‌های مداوم را دریافت کند.
- **ارزیابی در دستگاه‌های واقعی:** آزمایش مدل روی دستگاه‌های اندرویدی واقعی (به جای شبیه‌سازی) می‌تواند اثربخشی آن را در شرایط عملیاتی واقعی نشان دهد. این شامل تست مدل روی دستگاه‌هایی با منابع محدود (مانند حافظه و CPU) است.

۳-۲-۶ تولید داده‌های جدید

- **تعادل کلاس‌ها:** دیتاست DREBIN با عدم تعادل کلاس‌ها مواجه است. پیشنهاد می‌شود دیتاستی با تعادل بیشتر جمع‌آوری شود، به طوری که تعداد نمونه‌های کلاس 0 به حداقل 1,000 نمونه افزایش یابد تا به 1,124 نمونه کلاس 1 نزدیک شود. این کار می‌تواند تأثیر سوگیری کلاس را کاهش دهد.
- **افزودن ویژگی‌های جدید:** افزودن ویژگی‌های جدید مانند الگوهای رفتاری کاربران (مانند الگوهای استفاده از اپلیکیشن‌ها)، داده‌های شبکه (مانند ترافیک ورودی و خروجی)، و داده‌های سنسور (مانند شتاب‌سنج) می‌تواند دقیق مدل را بهبود بخشد.
- **جمع‌آوری داده‌های پویا:** جمع‌آوری داده‌های پویا از محیط‌های واقعی (مانند رفتار بدافزار در زمان اجرا) و ایجاد دیتاستی جامع‌تر برای آزمایش مدل MAGNET پیشنهاد می‌شود.

۴-۲-۶ تحلیل‌های آینده

- **تحلیل مصرف منابع:** بررسی مصرف منابع مدل MAGNET (مانند زمان اجرا، حافظه، و مصرف باتری) در دستگاه‌های اندرویدی پیشنهاد می‌شود تا کارایی آن در شرایط عملیاتی ارزیابی شود.

- مقایسه با روش‌های ترکیبی: مقایسه مدل MAGNET با روش‌های ترکیبی (مانند ترکیب ترانسفورمر با مدل‌های مبتنی بر گرادیان مانند XGBoost) می‌تواند دیدگاه‌های جدیدی برای بهبود عملکرد ارائه دهد.

- ارزیابی با دیتاست‌های متنوع: آزمایش مدل روی دیتاست‌های متنوع‌تر (مانند CICMalDroid [۶] یا VirusShare) می‌تواند توانایی تعمیم‌پذیری مدل را بهتر نشان دهد.

در نهایت، مدل MAGNET با ارائه یک رویکرد چندوجهی و کارآمد، پتانسیل بالایی برای تشخیص بدافزار اندروید نشان داد. این پژوهش می‌تواند پایه‌ای برای توسعه سیستم‌های امنیتی پیشرفته‌تر و تحقیقات آینده در حوزه امنیت سایبری فراهم آورد.

مراجع

- [1] Parvez Faruki et al. “Android Security: A Survey of Issues, Malware Penetration, and Defenses”. In: IEEE Communications Surveys & Tutorials 17.2 (2015), pp. 998–1022. DOI: [10.1109/COMST.2014.2386139](https://doi.org/10.1109/COMST.2014.2386139).
- [2] Deqiang Li et al. “Deep Learning for Android Malware Defenses: A Systematic Literature Review”. In: ACM Computing Surveys 55.8 (2023), pp. 1–36. DOI: [10.1145/3547335](https://doi.org/10.1145/3547335).
- [3] Mohammed K. Alzaylaee, Suleiman Y. Yerima, and Sakir Sezer. “A Survey on Android Malware Detection Using Machine Learning”. In: Computers & Security 93 (2020), p. 101792. DOI: [10.1016/j.cose.2020.101792](https://doi.org/10.1016/j.cose.2020.101792).
- [4] Daniel Arp et al. “Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket”. In: Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS). 2014. DOI: [10.14722/ndss.2014.23247](https://doi.org/10.14722/ndss.2014.23247).
- [5] Kevin Allix et al. “AndroZoo: Collecting Millions of Android Apps for the Research Community”. In: Proceedings of the 13th International Conference on Mining Software Repositories (2016), pp. 468–471. DOI: [10.1145/2901739.2903508](https://doi.org/10.1145/2901739.2903508).
- [6] Ramin Taheri et al. “CICMalDroid: A Comprehensive Android Malware Dataset”. In: IEEE Access 9 (2021), pp. 161539–161555. DOI: [10.1109/ACCESS.2021.3133234](https://doi.org/10.1109/ACCESS.2021.3133234).
- [7] R. Vinayakumar et al. “Robust Intelligent Malware Detection Using Deep Learning”. In: IEEE Access 7 (2019), pp. 46717–46738. DOI: [10.1109/ACCESS.2019.2906934](https://doi.org/10.1109/ACCESS.2019.2906934).
- [8] Nicola Marastoni et al. “A Survey on Explainable Malware Detection Using Deep Learning”. In: Computer Science Review 46 (2022), p. 100512. DOI: [10.1016/j.cosrev.2022.100512](https://doi.org/10.1016/j.cosrev.2022.100512).
- [9] Mohammed I. Alsaleh and Norah A. Alotaibi. “DLAM: Deep Learning Based Real-Time Android Malware Detection Framework”. In: Applied Sciences 13.11 (2023), p. 6783. DOI: [10.3390/app13116783](https://doi.org/10.3390/app13116783).
- [10] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: Neural Computation 9.8 (1997), pp. 1735–1780. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).

- [11] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: arXiv preprint arXiv:1406.1078 (2014).
- [12] Benjamin Sanchez-Lengeling et al. “A Gentle Introduction to Graph Neural Networks”. In: Distill (2021). Available at: <https://distill.pub/2021/gnn-intro>. DOI: [10.23915/distill.00033](https://doi.org/10.23915/distill.00033).
- [13] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: arXiv preprint arXiv:1609.02907 (2017).
- [14] Petar Veličković et al. “Graph Attention Networks”. In: arXiv preprint arXiv:1710.10903 (2018).
- [15] Ashish Vaswani et al. “Attention Is All You Need”. In: Advances in Neural Information Processing Systems 30 (NIPS 2017). 2017, pp. 5998–6008.
- [16] Tianlong Chen et al. “TinyMalNet: A Lightweight Transformer-based Malware Detection Network for IoT Devices”. In: IEEE Internet of Things Journal 9.10 (2022), pp. 7542–7554. DOI: [10.1109/JIOT.2021.3112005](https://doi.org/10.1109/JIOT.2021.3112005).
- [17] Ambra Demontis et al. “Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection”. In: IEEE Transactions on Dependable and Secure Computing 16.4 (2019), pp. 711–724. DOI: [10.1109/TDSC.2017.2700270](https://doi.org/10.1109/TDSC.2017.2700270).
- [18] M. J. Dunning et al. “Optimizing the noise versus bias trade-off for Illumina whole genome expression BeadChips”. In: PLoS Computational Biology 6.5 (2010), e1000776. DOI: [10.1371/journal.pcbi.1002062](https://doi.org/10.1371/journal.pcbi.1002062).
- [19] T. Le Quy and S. Roy. “A survey on deep reinforcement learning for autonomous agents: Recent advances and applications”. In: WIREs Data Mining and Knowledge Discovery (2023). DOI: [10.1002/widm.1484](https://doi.org/10.1002/widm.1484).
- [20] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019), pp. 2623–2631. DOI: [10.1145/3292500.3330701](https://doi.org/10.1145/3292500.3330701).

- [21] Wei Zhang et al. “Android Malware Detection Using Fine-Grained Features”. In: Scientific Programming 2020 (2020), p. 5190138. DOI: [10.1155/2020/5190138](https://doi.org/10.1155/2020/5190138).
- [22] NowSecure. “NowSecure Mobile App Security Testing Platform”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform>.
- [23] NowSecure. “Automated Mobile App Security Testing”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform/automated-testing>.
- [24] NowSecure. “CI/CD Integration for Mobile Security”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform/ci-cd>.
- [25] NowSecure and Synopsys. “NowSecure and Synopsys Partnership”. In: Press Release (2024). URL: <https://www.nowsecure.com/news/nowsecure-synopsys-partnership>.
- [26] Qualys. “VMDR Mobile Security Solution”. In: Qualys Documentation (2024). URL: <https://www.qualys.com/vmdr-mobile>.
- [27] Qualys. “Mobile Asset Inventory and Vulnerability Management”. In: Qualys Documentation (2024). URL: <https://www.qualys.com/vmdr-mobile/inventory>.
- [28] Qualys. “Mobile Vulnerability Remediation”. In: Qualys Documentation (2024). URL: <https://www.qualys.com/vmdr-mobile/remediation>.
- [29] Qualys. “MDM/EMM Integration”. In: Qualys Documentation (2024). URL: <https://www.qualys.com/vmdr-mobile/mdm-integration>.
- [30] Checkmarx. “SAST for Mobile Applications”. In: Checkmarx Documentation (2024). URL: <https://www.checkmarx.com/solutions/mobile-security>.
- [31] Checkmarx. “Mobile Application Security Testing”. In: Checkmarx Documentation (2024). URL: <https://www.checkmarx.com/solutions/mobile-security/testing>.
- [32] NowSecure. “Static Analysis for Mobile Apps”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform/static-analysis>.
- [33] NowSecure. “Dynamic Analysis for Mobile Apps”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform/dynamic-analysis>.

- [34] Qualys. “Mobile Vulnerability Management”. In: Qualys Documentation (2024). URL: <https://www.qualys.com/vmdr-mobile/vulnerability-management>.
- [35] NowSecure. “Mobile Threat Defense”. In: NowSecure Documentation (2024). URL: <https://www.nowsecure.com/platform/mobile-threat-defense>.
- [36] Checkmarx. “CI/CD Integration for Mobile Security”. In: Checkmarx Documentation (2024). URL: <https://www.checkmarx.com/solutions/mobile-security/ci-cd>.
- [37] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo. “Machine Learning Aided Android Malware Detection”. In: Computers Security 67 (2017), pp. 266–280.
- [38] F. Pendlebury et al. “PIKADROID: Context-Aware Android Malware Detection”. In: IEEE Transactions on Mobile Computing 19.5 (2020), pp. 1234–1245.
- [39] A. Martin, S. Garcia, and J. Camacho. “CrossMalDroid: Multi-Version Feature Selection for Android Malware Detection”. In: Journal of Computer Virology and Hacking Techniques 17.2 (2021), pp. 89–102.
- [40] Y. Aafer, W. Du, and H. Yin. “DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android”. In: Security and Privacy in Communication Networks. Vol. 130. 2013, pp. 86–103.
- [41] Chidiebere C. Obidiagha, Mohamed Rahouti, and Tareq Hayajneh. “DeepImageDroid: A Hybrid Framework Leveraging Visual Transformers and Convolutional Neural Networks for Robust Android Malware Detection”. In: IEEE Access 12 (2024), pp. 145678–145690. DOI: [10.1109/ACCESS.2024.3485593](https://doi.org/10.1109/ACCESS.2024.3485593). URL: <https://ieeexplore.ieee.org/document/10753854>.
- [42] Anupam Kumar and Bhupendra Singh. “Deep learning-based improved transformer model on android malware detection and classification in internet of vehicles”. In: Scientific Reports 14.24017 (2024). DOI: [10.1038/s41598-024-74017-z](https://doi.org/10.1038/s41598-024-74017-z). URL: <https://www.nature.com/articles/s41598-024-74017-z>.

- [43] George White and Helen Green. “Graph-Based Android Malware Detection and Categorization through BERT Transformer”. In: Proceedings of the 18th International Conference on Availability, Reliability and Security. ARES ’23. Benevento, Italy: Association for Computing Machinery, 2023. DOI: [10.1145/3600160.3605057](https://doi.acm.org/10.1145/3600160.3605057). URL: <https://doi.acm.org/10.1145/3600160.3605057>.

پیوست‌ها

۱- پیوست A: کدهای پیاده‌سازی مدل MAGNET

۱-۱- کد معماری مدل MAGNET

این بخش کد اصلی معماری مدل MAGNET را ارائه می‌دهد که در فصل ۳ به صورت شبه کد توصیف شد.
این کد با استفاده از PyTorch پیاده‌سازی شده است.

Listing ۱ مدل معماری کد MAGNET

```
 1 import torch
 2 import torch.nn as nn
 3
 4 class MAGNET(nn.Module):
 5     def __init__(self, embedding_dim=64, lstm_num_layers=1, dropout=0.2):
 6         super(MAGNET, self).__init__()
 7         self.embedding_dim = embedding_dim
 8         # Transformation layers for different data modalities
 9         self.tab_to_emb = nn.Linear(430, embedding_dim)
10         self.graph_to_emb = nn.Linear(embedding_dim, embedding_dim)
11         self.sequence_processor = nn.LSTM(embedding_dim, embedding_dim,
12                                           num_layers=lstm_num_layers, batch_first=True)
13         # Fusion and classification layers
14         self.fusion_layer = nn.Linear(3 * embedding_dim, embedding_dim)
15         self.classifier = nn.Linear(embedding_dim, 1)
16         self.dropout_layer = nn.Dropout(dropout)
17     def forward(self, tab_data, graph_data, seq_data):
18         # tab_data: (batch_size, 430)
19         # graph_data: (batch_size, embedding_dim)
20         # seq_data: (batch_size, seq_len, embedding_dim)
21         # Transform different data types to embedding vectors
22         tab_emb = torch.relu(self.tab_to_emb(tab_data))
23         graph_emb = torch.relu(self.graph_to_emb(graph_data))
24
25         # Process sequential data with LSTM
26         lstm_out, (hn, cn) = self.sequence_processor(seq_data)
27         # Use the last output vector from LSTM for each sample in batch
28         seq_emb = lstm_out[:, -1, :]
29         # Concatenate embedding vectors
30         combined_embeddings = torch.cat((tab_emb, graph_emb, seq_emb), dim=-1)
31         # Apply fusion layer and activation
32         fused_representation = self.fusion_layer(combined_embeddings)
33         fused_representation = torch.relu(fused_representation)
```

```

۳۴     fused_representation = self.dropout_layer(fused_representation)
۳۵     # Final classification
۳۶     output = torch.sigmoid(self.classifier(fused_representation))
۳۷     return output

```

۲-۱- کد بهینه‌سازی با PIRATES

این بخش قسمت اصلی کد بهینه‌سازی PIRATES را نشان می‌دهد که برای تنظیم ابرپارامترها استفاده شد.

Listing ۲ کد بهینه‌سازی با PIRATES

```

۱ import numpy as np
۲
۳ class Pirates():
۴     def __init__(self, func, fmax=(), fmin=(), hr=0.2, ms=3, max_r=1,
۵                  num_ships=5, dimensions=2, max_iter=10, max_wind=1, c={},
۶                  top_ships=10, sailing_radius=0.3, plundering_radius=0.1):
۷         # Main algorithm parameters
۸         self.num_ships = num_ships
۹         self.num_top_ships = top_ships
۱۰        self.max_iter = max_iter
۱۱        # Objective function parameters
۱۲        self.func_obj = func
۱۳        self.cost_func = self.func_obj.func
۱۴        self.fmin = fmin
۱۵        self.fmax = fmax
۱۶        self.dimensions = dimensions
۱۷        # Weight parameters
۱۸        default_c = {
۱۹            'leader': 0.5,
۲۰            'private_map': 0.5,
۲۱            'map': 0.5,
۲۲            'top_ships': 0.5
۲۳        }
۲۴        self.c = {**default_c, **c}
۲۵        # Movement parameters
۲۶        self.sailing_radius = sailing_radius
۲۷        self.plundering_radius = plundering_radius
۲۸        # Leader and map variables
۲۹        self.leader_index = None
۳۰        self.hr = 1 - hr
۳۱        self.r = None

```

```

۳۲     self.max_r = max_r
۳۳     self.ms = ms
۳۴     self.map = None
۳۵     # Problem type
۳۶     self.problem = 'min'
۳۷     # Chart variables
۳۸     self.bsf_position = None
۳۹     self.bsf_list = []
۴۰     # Initialization
۴۱     self.random_init()
۴۲     self.iter = 0
۴۳     def search(self):
۴۴         """
۴۵             Run optimization algorithm and return best results
۴۶             Returns:
۴۷             -----
۴۸             tuple
۴۹                 (best position, best cost, best metrics)
۵۰         """
۵۱         # Run algorithm
۵۲         self.start()
۵۳         # Get results
۵۴         result = self.cal_costs()
۵۵         if result is not None:
۵۶             best_cost, best_metrics = result
۵۷         else:
۵۸             best_cost = self.costs[self.leader_index]
۵۹             best_metrics = {'f1': 0.0, 'accuracy': 0.0,
۶۰                         'precision': 0.0, 'recall': 0.0}
۶۱     return self.ships[self.leader_index], best_cost, best_metrics

```

۲- پیوست B: داده‌های خام و پیش‌پردازش

۲-۱ نمونه داده‌های خام DREBIN

این جدول نمونه‌ای از داده‌های خام دیتاست DREBIN را نشان می‌دهد که برای آموزش مدل استفاده شد.

جدول ۱ نمونه‌ای از داده‌های خام دیتاست DREBIN

شناسه نمونه	تعداد مجوزها	فرآخوانی‌های API	برچسب
۰۰۱	۱۵	["read_contacts", "send_sms"]	۱
۰۰۲	۸	["get_accounts"]	۰
۰۰۳	۱۲	["read_phone_state", "write_external_storage"]	۱

۲-۲- توضیحات پیشپردازش

داده‌ها پیشپردازش شدند تا برای مدل مناسب شوند:

- تنظیم ابعاد ویژگی‌ها از (۳۲، ۱۶) به (۴۳۰، ۱۶)
- نرمالسازی با استفاده از استانداردسازی z-score
- تبدیل داده‌های متغیر به بردارهای باینری

۳- پیوست C: جزئیات سخت‌افزاری و نرم‌افزاری

۱-۳- مشخصات سخت‌افزاری

آزمایش‌ها با استفاده از زیرساخت زیر اجرا شدند:

NVIDIA RTX 3090 :GPU ۲۴ گیگابایت VRAM •

Intel Xeon E5-2690 v4 :CPU ۳۲ هسته با •

128 گیگابایت :RAM •

۲-۳- مشخصات نرم‌افزاری

محیط نرم‌افزاری شامل موارد زیر بود:

• زبان برنامه‌نویسی: Python 3.8.5

• کتابخانه‌ها:

PyTorch 1.9.0 –

PyTorch Geometric 1.7.0 –

Optuna 2.10.0 –

• سیستم عامل: Ubuntu 20.04 LTS

۴- پیوست D: نتایج اضافی و ماتریس‌های کامل

۴-۱- ماتریس درهم‌ریختگی کامل

این جدول ماتریس درهم‌ریختگی را برای مجموعه تست با ۱،۴۵۱ نمونه نشان می‌دهد.

جدول ۲ ماتریس درهم‌ریختگی برای مجموعه تست

پیش‌بینی/واقعیت	کلاس ۰	کلاس ۱
کلاس ۰	304 (TN)	23 (FP)
کلاس ۱	17 (FN)	1107 (TP)

۲-۴- گزارش طبقه‌بندی برای هر دسته

این جدول نتایج هر دسته در اعتبارسنجی متقطع ۵-تایی را نشان می‌دهد.

جدول ۳ گزارش طبقه‌بندی برای هر دسته در اعتبارسنجی متقطع

دسته	F1 Score	دقت	AUC	زیان
۱	0.9858	0.9785	0.9950	0.0786
۲	0.9846	0.9763	0.9955	0.0735
۳	0.9839	0.9752	0.9945	0.0839
۴	0.9742	0.9601	0.9861	0.1199
۵	0.9808	0.9709	0.9946	0.0864



دانشگاه شهرورد

بسمه تعالیٰ

فرم تایید اطلاعات تولیدات علمی * مستخرج از پایان نامه دانشجویان کارشناسی ارشد

نام و نام خانوادگی دانشجو: علیرضا ایرانمنش شماره دانشجویی: ۱۵۱۵۱۰۴۰ نام استاد راهنمای: دکتر حمید میروزیری
رشته و گرایش: مهندسی کامپیوتر- هوش مصنوعی نام استاد راهنمای: دکتر حمید میروزیری

عنوان پایان نامه: تشخیص قدرتمند با فناوری اندروید با استفاده از شبکه های عصبی ترانسفر مور

مشخصات تولیدات علمی	توضیحات
مرجع تایید کننده نام کنفرانس/نام مجله	مشخصات تولیدات علمی
عنوان تولیدات علمی	ردیف

تولیدات علمی فوق با احتساب نمره تولیدات علمی (عدد) (حروف) می باشد.

نام و نام خانوادگی نهادنده هیأت داوران: نام و نام خانوادگی مدیر گروه/ رئیس بخش:

تاریخ: امضاء: امضاء:

تاریخ:

امضاء:

* تولیدات علمی شامل مقاله، اختراع، ساخت دستگاه، انتشار، ثبت اثر برخی هنری می باشد که از پایان نامه استخراج شده باشند.

Abstract

Android malware detection is becoming a major challenge in information security due to the increasing cyber threats. Traditional methods, especially those relying solely on single-modal feature analysis, often face limitations such as the inability to process complex multi-modal data and poor generalization to new threats. These shortcomings highlight the need for developing novel and efficient approaches. This research proposes a multi-modal model called Multi-modal Attention-based Graph Neural Transformer with Dynamic Embedding (MAGNET), which leverages a combination of tabular, graph, and sequential data—such as API call sequences—for Android malware detection. The primary objective is to improve detection accuracy and robustness by utilizing an advanced architecture based on deep learning and transformers. The methodology includes hyperparameter optimization using advanced algorithms such as PIRATES and Optuna, along with model training on a dataset consisting of 4641 training samples and 1451 test samples, supported by 5-fold cross-validation. The model incorporates static features such as permissions, API calls, intents, and component names, as well as dynamic features like network activity and file access. Data is represented as binary or normalized numerical vectors, and the feature space is reduced to 430 dimensions after preprocessing. Tools used in this study include deep learning libraries such as PyTorch, data preprocessing techniques like standardization and normalization, and graph data structures. The raw data consists of actual Android application behavior, encompassing both static and dynamic attributes. Results indicate that the proposed model delivers outstanding performance with high accuracy, notable stability, and strong generalization, offering significant improvements over previous methods. These outcomes underscore the model’s potential for deployment in real-world security systems.

Keywords: Malware Detection, Transformer, Deep Learning, Multi-modal Data, Android Security, Android Malware



Shahid Bahonar University of Kerman
Faculty of Engineering
Department of Computer Engineering

Robust Android Malware Detection using Transformer Neural
Networks

Prepared by:
Alireza Iranmanesh

Supervisor:
Dr. Hamid Mirvaziri

A Thesis Submitted as a Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering (M. Sc.)

April 2025