Scientific Journal of Electronical & Cyber Defence
Vol. 11, No. 2, Summer 2023, Serial No. 42

Mobile botnets detection using deep learning techniques

M. Ghanavati Nasab, M. Ghazvini*, F. Ghasemian

* Associate Professor, Shahid Bahonar University of Kerman, Kerman, Iran

(Received: 12/06/2022, Accepted: 21/01/2023)

ABSTRACT

Smartphones are now well integrated with advanced capabilities and technologies such as the Internet. Today, due to the facilities and capabilities and the widespread use of smart mobile devices, mobile security has become a vital issue worldwide. Smartphones are not properly protected compared to computers and computer networks, and users do not consider security updates. Recently, mobile devices and networks have been targeted by one of the most dangerous cyber threats known as botnets. Mobile Bantent An enhanced example of Boutons has the ability to perform malicious operations such as denial of service attacks, data theft, eavesdropping, and more. Bunters use three communication protocols: HTTP, SMS and Bluetooth to communicate with each other; So when users are not connected to the Internet, botnets are able to communicate with each other. In this study, to identify mobile batonet from 14 Android baton families, including 1932 samples of Android mobile devices applications and 4304 samples of safe and secure Android mobile devices applications have been used. Application permissions were extracted for reverse engineering to automatically classify and detect types of botnets, then based on these permissions, each application was converted to an equivalent image using the proposed method. Labeled images were then used to train convolutional neural networks. The results of evaluation and comparison of this method with classical methods including backup vector machine and decision tree showed that the proposed method is able to achieve higher efficiency in detecting different types of botnets and separating it from healthy programs.

Keywords: Botnet, mobile security, security, mobile botnet, botnet detection, convolutional network.

* Corresponding Author Email: Mghazvini@gmail.com

نشریه علمی "پدافند الکترونیکی و سایبری"
سال یازدهم، شماره ۲، تابستان ۱۴۰۲، ص۳۱-۴۳

علمی – پژوهشی

تشخیص بات‌نت‌ها با استفاده از فنون یادگیری عمیق

مریم قنواتی نسب[۱]، مهدیه قزوینی[۲*]، فهیمه قاسمیان[۳]

۱- دانشجوی کارشناسی ارشد، ۲- دانشیار، ۳- استادیار، دانشگاه شهید باهنر کرمان، کرمان، ایران

(دریافت: ۱۴۰۱/۰۳/۲۲، پذیرش: ۱۴۰۱/۱۱/۰۱)

چکیده

امروزه به دلیل اتصال تلفن‌های همراه هوشمند به اینترنت و وجود قابلیت‌ها و امکانات مختلف در این تلفن‌ها، حفظ امنیت این دستگاه‌ها به یک چالش مهم تبدیل شده است. چرا که معمولاً در این دستگاه‌ها انواع داده‌های خصوصی که مرتبط با حریم شخصی افراد است ثبت و ذخیره می‌شود. در سال‌های اخیر این دستگاه‌ها مورد هدف یکی از خطرناک‌ترین حملات سایبری قرار گرفته‌اند که بات‌نت نام دارد. بات‌نت‌ها توانایی انجام عملیات مخربی چون ربودن و استراق سمع و حملات انکار سرویس را دارند. ازاین‌رو شناسایی به‌موقع بات‌نت‌ها تأثیر زیادی در حفظ امنیت تلفن‌های همراه دارد. در این مقاله روشی جدید برای شناسایی بات‌نت‌ها از برنامه‌های سالم اندروید و همچنین تشخیص نوع بات‌نت از میان ۱۴ نوع مختلف از خانواده بات‌نت‌ها ارائه شده است. در این روش ابتدا با استفاده از مهندسی معکوس، لیست مجوزهای برنامه استخراج شده، سپس بر اساس این لیست مجوزها تصویر معادل برنامه ایجاد می‌شود. به این ترتیب مجموعه‌ای از تصاویر به دست می‌آید که با استفاده از شبکه عصبی کانولوشنال ارائه شده، این تصاویر طبقه‌بندی و نوع برنامه کاربردی مشخص می‌شود. نتایج حاصل از مقایسه و ارزیابی این روش با روش‌های سنتی یادگیری ماشین چون ماشین بردار پشتیبان و درخت تصمیم نشان داد که روش ارائه شده کارایی بالاتری در تشخیص انواع بات‌نت‌ها و جداسازی آن از برنامه‌های سالم دارد.

کلیدواژه‌ها: بات‌نت، امنیت تلفن همراه، امنیت، بات‌نت تلفن همراه، تشخیص بات‌نت، شبکه کانولوشن

۱- مقدمه

امروزه تلفن‌های همراه هوشمند به خوبی با اینترنت ادغام شده‌اند و با استفاده‌ی روزافزون تلفن‌های هوشمند در مقیاس جهانی، امنیت تلفن‌های همراه به موضوعی حیاتی تبدیل شده است. حملات و تهدیدات این دستگاه‌ها به اشکال مختلف مانند ویروس، باتنت‌ها و غیره انجام می‌شود. بااین‌وجود، بات‌نت‌ها تهدیدی جدی برای تلفن‌های همراه هوشمند و شبکه‌های ارتباطی‌اند [۱-۳]. یک بات‌نت شامل برنامه‌های مخرب کوچکی است که بدون جلب‌توجه کاربران و به‌صورت خودکار، رایانه‌ها یا دستگاه‌های تلفن همراه را آلوده می‌کند و برای اهداف مختلف استفاده می‌شود. باتنت‌ها با استفاده از کانال کنترل و فرمان با یکدیگر ارتباط برقرار می‌کنند. تفاوت اصلی بین بات‌نت و سایر تهدیدها در این است که آن‌ها به‌صورت پویا توسط یک مهاجم به نام مدیر بات کنترل می‌شوند. صرفنظر از اندازه که ارتباط مستقیمی با پیچیدگی و هدف آن‌ها دارد، باتنت‌ها بیش‌تر برای انجام فعالیت‌های مخرب در فضای مجازی ایجاد می‌شوند. به‌دلیل الگوی ارتباطی خوب مدیریت‌شده در بین عناصر بات‌نت و همچنین حملات هماهنگ، می‌توان آن‌ها را به‌عنوان نمونه سازمان‌یافته‌ی جرائم سایبری طبقه‌بندی کرد [۴, ۵]. بات‌نت‌ها بر روی زیرساخت‌های شبکه کار می‌کنند، رایج‌ترین اهداف آن‌ها رایانه‌های با نظارت کم، رایانه‌هایی با ارتباط پهنای باند بالا، سرویس‌دهنده‌های دانشگاه و رایانه‌های خانگی است. استفاده گسترده از تلفن‌های همراه هوشمند متصل به اینترنت، مدیران بات را به زیرساخت‌های موبایل نیز ترغیب کرده است. گزارش‌های بسیاری نشان می‌دهد نسل جدید بات‌نت‌ها و جرائم سایبری سازمان‌یافته مبتنی بر کانال کنترل و فرمان، شبکه‌های تلفن همراه را هدف قرار داده‌اند [۶-۸]. تاکنون رویکردهای مختلفی برای شناسایی بات‌نت‌ها ارائه شده است و یادگیری عمیق یکی از موفق‌ترین روش‌ها در زمینه شناسایی بات‌نت‌های جدید است [۶, ۷]. در پژوهش جاری، بات‌نت‌های اندروید برای مقابله با یکی از حیاتی‌ترین چالش‌های امنیتی در تلفن‌های همراه اندرویدی، شناسایی و دسته‌بندی شده‌اند. بدین صورت که با استفاده از مجوزهای برنامه‌های اندروید، روشی برای ایجاد تصاویر برنامه پیشنهاد شده است. مدل شبکه عصبی می‌تواند برنامه‌های بات‌نت را از برنامه‌های امن تنها با استفاده از مجوزهای آن‌ها متمایز و طبقه‌بندی کند. یکی از مزایای مدل شبکه عصبی پیشنهادی، دسته بندی بات‌نت‌ها علاوه بر شناسایی آن‌ها است که در کارهای مشابه به آن پرداخته نشده و یا کمتر مورد توجه قرار گرفته است (شکل (۱)).

* رایانامه نویسنده مسئول: Mghazvini@gmail.com

شکل (۱). تشخیص و طبقه بندی برنامه های اندروید در روش پیشنهادی

۲- کارهای مرتبط

روش‌های گوناگونی برای شناسایی باتنت‌ها ارائه شده است که بسیاری از آن‌ها مبتنی بر روش‌های هوش مصنوعی و یادگیری ماشین هستند و به میزان قابل توجهی بهتر از روش‌های قدیمی عمل می‌کنند[۶، ۷]. به‌عنوان مثال مودی و همکاران[۹] یک الگوریتم یادگیری خود-تطبیقی هوشمند را با استفاده از ماشین بردار پشتیبان و بهینه‌سازی ازدحام ذرات برای شناسایی باتنت‌های اندروید پیشنهاد کردند. در روش پیشنهادی آن‌ها، به طور همزمان از پنج استراتژی مختلف برای تنظیم پارامترهای ماشین بردار پشتیبان بر اساس الگوریتم PSO استفاده می‌شود و با انتخاب بهترین استراتژی‌ها، پاسخ‌های بهینه برای پارامترهای ماشین بردار پشتیبان و ویژگی‌های مناسب برای شناسایی باتنت‌های اندروید به دست می‌آید.

منگ و همکاران [۱۰] الگوریتم MBotCS را برای شناسایی باتنت‌های موبایل با استفاده از روش‌های یادگیری ماشین و پنج طبقه بندی کننده مختلف NB، J48، SVM، MLP و kNN استفاده کرده و یک طبقه‌بندی کننده بلادرنگ با استفاده از ترافیک ثبت شده از سوی برنامه ارائه دادند. این طبقه‌بندی کننده برنامه ورودی را به دو دسته سالم و آلوده طبقه‌بندی می‌کند. طبق آزمایش‌ها، فرایند تشخیص، زمانی کارآمدتر بود که طبقه‌بندی‌کننده‌های یادگیری ماشین با ترافیک یک باتنت مشخص آموزش داده شدند. علاوه بر این، فرایند تشخیص در ترافیک جریان شبکه کارآمدتر از ترافیک بسته بود و الگوریتم‌های یادگیری ماشین مؤثرتر از الگوریتم‌های اتمی بودند. رویکرد پیشنهادی عبدالله و همکاران [۱۱] از مجوزهای درخواست شده در برنامه اندروید به‌عنوان ویژگی و از الگوریتم بهره اطلاعاتی برای انتخاب مهم‌ترین ویژگی‌ها استفاده می‌کند. سپس از الگوریتم‌های طبقه‌بندی نایو بیز، جنگل تصادفی و J48 برای طبقه‌بندی برنامه‌های اندروید به‌عنوان باتنت یا برنامه‌های خوش‌خیم استفاده می‌شود. نتایج تجربی نشان می دهد که جنگل تصادفی بالاترین دقت تشخیص برابر با ۹۴٫۶٪ با کمترین نرخ مثبت کاذب ۰٫۰۹۹ را نتیجه می‌دهد.

در[۱۲] سیستمی به نام ABIS (سیستم شناسایی باتنت اندروید) برای شناسایی باتنت‌های اندروید، توسعه داده شده که با تحلیل فایل‌های APK و با استفاده از ابزار وکا، ویژگی‌های مهم و بهترین الگوریتم با بالاترین دقت در بین هفت الگوریتم یادگیری ماشین را پیدا می‌کند. البته انواع دیگر بدافزارها را نمی‌توان در این کار شناسایی کرد زیرا کل مجموعه داده آموزشی فقط شامل باتنت‌های اندروید است. علاوه بر این، هدف باتنت‌های تلفن همراه عمدتاً سرقت اطلاعات حساس کاربر است، برخلاف سایر بدافزارها، مانند ویروس، کرم، تروجان و باج‌افزار که هدفشان از بین بردن و غیرفعال کردن ماشین قربانی است. چارچوب پیشنهادی انور و همکاران [۱۳] ویژگی‌های استاتیک شامل هش MD5، مجوزها، گیرنده‌های چندپخشی و سرویس‌های پس‌زمینه را از برنامه‌های اندروید استخراج می‌کند. سپس شش نوع مختلف از الگوریتم‌های یادگیری ماشین را به مجموعه داده‌ها اعمال می‌کند تا مدل‌های طبقه‌بندی را استخراج نموده و برنامه‌ها را به‌عنوان باتنت یا خوش‌خیم طبقه‌بندی نماید. بهترین نتیجه از الگوریتم طبقه‌بندی کننده نایو بیز برای مجوزها به دست می‌آید، در حالی که J-48 برای گیرنده‌های پخش از هر دو نوع خوش خیم و باتنت نتایج بهتری را به دست می‌دهد. نویسندگان [۱۴] با استفاده از ۱۴۵ ویژگی شامل ۱۳۸ مجوز درخواستی و هفت سطح حفاظتی برای مجوزهای اندروید با کمک چهار طبقه‌بند (جنگل تصادفی، شبکه‌های عصبی پرسپترون چندلایه، درخت‌های تصمیم‌گیری، و بیز) روشی برای تشخیص بات نت‌های اندروید ارائه نمودند. نتایج تجربی نشان داد که طبقه‌بند جنگل تصادفی بهترین نتایج را برای هر دو گروه از ویژگی‌ها دارد. در [۱۵] روشی مبتنی بر مجوزهای اندروید و شبکه‌های عصبی کانولوشنال(CNN) برای شناسایی برنامه‌های باتنت اندروید پیشنهاد شده که در آن برای نمایش هر برنامه، یک تصویر بر اساس همزمانی مجوزهای داده‌شده به آن برنامه ساخته می‌شود. CNN پیشنهادی نیز یک طبقه‌بند باینری است که با استفاده از تصاویر به‌دست آمده آموزش داده شده است. با ارزیابی این روش روی ۵۴۵۰ برنامه اندرویدی شامل نمونه‌های باتنت و خوش خیم، نتایج به دست آمده دقت ۹۷٫۲ درصد و RF برابر با ۹۶ درصد را نشان می دهد. در پژوهش [۱۶] یک سیستم تشخیص باتنت اندروید مبتنی بر یادگیری عمیق با استفاده از شبکه‌های عصبی کانولوشن پیشنهاد شده است که برخلاف مطالعات قبلی که فقط از مجوزهای برنامه استفاده می‌کنند، این سیستم بر اساس ۳۴۲ ویژگی است که مجوزها، فراخوان‌های API، فرمان‌ها، فایل‌های اضافی و اهداف را نشان می‌دهند. علاوه بر این، بردارهای ویژگی بدون تبدیل به تصویر، مستقیماً برای آموزش مدل‌های CNN یک بعدی استفاده می‌شوند که این سبب کاهش بار محاسباتی است.

آگنیل و همکاران(Agniel, Arnold et al. 2022) روشی برای

تشخیص بات نت‌ها در بستر اینترنت اشیا ارائه داده‌اند. بدین صورت که ابتدا اطلاعات ترافیکی را استخراج نموده و به تصویر تبدیل می‌کنند. هر تصویر از ارتفاعی که نشان‌دهنده هر بستهٔ مورد بررسی و عرضی که نشان‌دهنده ویژگی‌های هر بسته است، تشکیل می‌شود. پس از جمع‌آوری داده‌های کافی در یک بازه زمانی، ویژگی‌ها به پیکسل‌های RGB تبدیل می‌شوند. همچنین بار شبکه بررسی می‌شود که اگر شبکه بیش از حد بارگذاری شده باشد، منتظر می‌ماند تا بسته‌های بیشتری تصویر را پر کنند. سپس تصویر، فشرده‌سازی شده و به شبکه کانولوشن دوبعدی وارد می‌شود. نتایج شبیه‌سازی حاکی از حساسیت ۹۸,۹۸٪ است. در [17] سیستمی برای شناسایی بات‌نت‌های اندروید از طریق متن کاوی خودکار فایل‌های مانیفست به‌دست‌آمده از برنامه‌ها پیشنهاد شده است. روش پیشنهادی از تکنیک‌های پردازش زبان طبیعی برای استخراج ویژگی‌ها از فایل‌های مانیفست استفاده می‌کند و یک مدل طبقه‌بندی مبتنی بر یادگیری عمیق برای شناسایی برنامه‌های بات‌نت استفاده می‌شود. مدل طبقه‌بندی با استفاده از CNN و طبقه‌بندی‌کننده یادگیری ماشین سنتی مانند ماشین بردار پشتیبان، جنگل تصادفی یا KNN پیاده‌سازی می‌شود. برای ارزیابی از ۳۸۵۸ برنامه اندرویدی متشکل از ۱۹۲۹ بات‌نت و ۱۹۲۹ نمونه خوش‌خیم استفاده شده که بهترین عملکرد کلی با مدل ترکیبی CNN-SVM با دقت متوسط ۹۶,۹٪ به دست آمد و بنابراین از طبقه‌بندی‌کننده‌های یادگیری ماشین منفرد بهتر عمل کرد. نویسندگان [۱۸] تشخیص حمله بات‌نت اندروید را با استفاده از الگوریتم‌های یادگیری عمیق، شبکه عصبی کانولوشنال (CNN) و شبکه عصبی مصنوعی (ANN) مقایسه نمودند. این آزمایش بر روی ۱۹۲۹ برنامه بات‌نت و ۴۸۷۳ برنامه کاربردی خوش خیم با استفاده از دسته‌های مختلف از مجوزها انجام شد. این تحقیق چندین معیار عملکرد مانند دقت، صحت، پوشش، امتیاز F1 را برای شناسایی بهترین طبقه‌بندی‌کننده عملکرد استفاده کرد. در پایان مطالعه، طبقه‌بندی‌کننده ANN به‌عنوان بهترین طبقه‌بندی‌کننده برای تشخیص حمله بات‌نت اندروید با بالاترین دقت تشخیص ۹۶,۳۵٪ شناخته شد در حالی که دقت تشخیص به‌دست‌آمده توسط CNN برابر با ۹۵,۴۴٪ است. علاوه بر این، معیارهای عملکرد به دست آمده از تشخیص حمله بات‌نت اندروید با استفاده از CNN و ANN بهتر از معیارهای به‌دست‌آمده از مطالعات قبلی بود که از الگوریتم‌های کلاسیک یادگیری ماشین برای تشخیص حمله بات‌نت اندروید استفاده می‌کردند.

گسترش روزافزون تهدید بات‌نت و توسعه بسترهای جدید استقرار بات‌نت مانند اینترنت اشیا، لزوم مقابله را نشان می‌دهد. پژوهش‌هایی که در حوزه تشخیص بات‌نت مبتنی بر روش‌های یادگیری ماشین و یادگیری عمیق انجام شده است؛ نشان می‌دهد این روش‌ها کارایی لازم را جهت تشخیص بات‌نت دارند. این در حالی است که عدم وجود یک مجموعه دادگان استاندارد در این حوزه، یکی از چالش‌ها در سامانه‌های تشخیص بات‌نت است که موجب افزایش نرخ خطا و کاهش نرخ تشخیص در محیط واقعی می‌شود. در [۱۹]، ترافیک عادی و بات‌نت با ارائه روشی مبتنی بر بردار فاصله مینکوفسکی تحلیل شده است. نتایج نشان می‌دهد که جریان ترافیک عادی، مرحله انتخاب و استخراج ویژگی را با تغییر در اهمیت ویژگی‌ها مؤثر می‌کند. این روش به ویژگی‌ها بر اساس نزدیک نمودن بردارهای رفتاری بات-بات و دور نمودن بردارهای رفتاری بات‌عادی امتیاز می‌دهد. نتایج این آزمایش‌ها بر روی ده مجموعه دادگان عادی و سه مجموعه دادگان بات، نشان داد امتیاز یک ویژگی در محیط‌هایی با ترافیک عادی متفاوت بیش از ۵۰٪ افزایش یا کاهش دارد.

معرفی و مطالعه روش‌های بیشتر در زمینه تشخیص بات‌نتها در[۷, ۸, ۲۰, ۲۱] آمده است.

۳- روش تحقیق

ازآنجایی‌که دستگاه‌های تلفن همراه به‌ویژه گوشی‌های هوشمند تمایل دارند برای دوره‌های طولانی برخط باشند، زیرساخت مناسبی برای عملکرد بات‌نت‌ها در زمانی که به خطر افتاده‌اند، فراهم می‌کنند. بات‌نت‌های موبایل با استفاده از پیامک یا دستورات مبتنی بر وب و کانال‌های کنترل و فرمان، هدایت و کنترل می‌شوند و برای اهداف مختلفی مانند حملات انکار سرویس توزیع‌شده، حملات فیشینگ، توزیع هرزنامه، سرقت اطلاعات محرمانه و غیره استفاده می‌شوند. حملات انکار سرویس، حجم و فرکانس بالایی دارند و بنابراین به‌راحتی توسط سیستم‌های تشخیص نفوذ شبکه شناسایی می‌شوند. برخی حملات بات‌نت‌ها با فرکانس پایین و حجم کم ترافیک شبکه مشخص نمی‌شوند و شناسایی آن‌ها چالش‌انگیز است؛ بنابراین رویکردهای تکمیلی برای تشخیص مبتنی بر شبکه برای تقویت دفاع در برابر آلودگی و حملات بات‌نت موبایل موردنیاز است. ازآنجایی‌که بدافزار موبایل همچنان در حال افزایش و پیچیده‌تر شدن است، تلاش‌های تحقیقاتی برای شناسایی و کاهش بدافزار اندروید در سال‌های اخیر بیشتر شده است. تاکنون روش‌هایی برای تشخیص بات‌نت ارائه‌شده است. چنین سیستم‌هایی برای آموزش ماشین به ویژگی‌های استخراج‌شده به‌صورت ایستا یا پویا متکی هستند. در بسیاری از موارد، این ویژگی‌ها، دست‌ساز هستند. با تکامل سیستم‌عامل اندروید، بسیاری از این ویژگی‌های دست‌ساز ممکن است منسوخ شوند و کل فرایند استخراج ویژگی نیاز به مهندسی مجدد داشته باشد. استفاده از روش‌های پردازش

تصویر برای استخراج ویژگی‌ها از بازنمایی مبتنی بر تصویر برنامه دارای مزیت مشخصی است که نیاز به تکیه بر ویژگی‌های دست‌ساز برای ساخت مدل‌های یادگیری ماشینی را حذف می‌کند. علاوه بر این، با رویکرد مبتنی بر تصویر، برای انطباق با تکامل سیستم‌عامل نیاز به تغییر کمی یا بدون تغییر خواهد بود و این منجر به کارایی طولانی‌مدت در مقایسه با سیستم‌های مبتنی بر ویژگی‌های دست‌ساز می‌شود.

روش پیشنهادی بر اساس متدولوژی CRISP-ML ارائه شده است. این متدولوژی شامل مراحل زیر است:

- فهم داده
- آماده‌سازی و جمع‌آوری داده
- مدل‌سازی
- ارزیابی

۳-۱- فهم داده

مجوز یک مفهوم انتزاعی برای اتصال عملیات به منابع است. از مجوزها برای بررسی الزامات سیستم استفاده می‌شود. توسعه‌دهنده باید مجوزهای استفاده در برنامه‌های خود را اعلام کند. مجوزهای اعلام شده برای آشکارکردن خطرات احتمالی برنامه‌های در حال نصب مفید و مؤثر هستند. به طور پیش‌فرض، یک برنامه اندروید با صفر مجوز شروع می‌شود. هنگامی که برنامه نیاز به استفاده از هر یک از ویژگی‌های محافظت شده دستگاه (ارسال درخواست شبکه، دسترسی به دوربین، ارسال پیامک و غیره) داشته باشد، باید مجوز مناسب را از کاربر برای انجام این کار دریافت کند. از بررسی برخی از برنامه‌های معروف دانلود شده از فروشگاه Google Play مشخص شد که ۱۵۲ مجوز استاندارد وجود دارد که توسط برنامه‌های مختلف استفاده می‌شود. این مجوزها به چهار دسته تقسیم می‌شوند: عادی، خطرناک، امضا و امضا-یا-سیستم. مجوزهای عادی کم‌خطر هستند که به برنامه‌های اندرویدی اجازه می‌دهند تا اقداماتی را انجام دهند که برای کاربران نهایی مضر نیستند، مانند VIBRATE، WAKE_LOCK و SET_WALLPAPER. مجوزهای خطرناک مجوزهایی هستند که توانایی انجام فعالیت‌های مضر را دارند. مانند CALL_PHONE، READ_SMS، SEND_SMS، CAMERA. این برنامه‌ها می‌توانند بدون اطلاع مالک، با دستگاه‌های دیگر تماس بگیرند، بخوانند، ارسال کنند، دریافت کنند و پیامک بنویسند. مجوزهای امضا مجوزهایی هستند که پس از امضای یک گواهی به برنامه‌های اندرویدی اختصاص می‌یابد. مجوزهایی که فقط می‌توانند به برنامه‌های موجود در تصویر سیستم اندروید اعطا شوند یا با همان گواهی امضا شده‌اند، مجوزهای Signature-OR-System نامیده می‌شوند. INSTALL_PACK- AGE یکی از نمونه‌های این مجوزها است. مجوز INTERNET بیشترین مجوز مورد نیاز را هم در برنامه‌های بات‌نت و هم در برنامه‌های کاربردی خوش‌خیم است. READ_PHONE_STATE دومین مجوز خطرناک است که توسط بدافزارها درخواست شده است. اکثر بات‌نت‌ها برای گروه پیامک درخواست می‌کنند که حاوی SEND-_SMS، RECEIVE_SMS، WRITE_SMS، و READ_SMS است. همه این مجوزهای خطرناک می‌توانند با ترکیب سایر ویژگی‌ها به کاربر آسیب برسانند.

این که یک برنامه از چه مجوزهایی استفاده می‌کند می‌تواند به‌عنوان یک ویژگی متمایزکننده برای تشخیص این که آیا برنامه یک برنامه بات‌نات یا سالم است مورد استفاده قرار گیرد. بنابراین بایستی مجموعه‌ای از برنامه‌های سالم و آلوده به بات‌نت را به‌عنوان مجموعه داده آموزشی جمع‌آوری کرده و لیست مجوزهای مورد استفاده در هر برنامه را استخراج کرد.

۳-۲- آماده‌سازی و جمع‌آوری داده

مجموعه داده برنامه‌های کاربردی تلفن همراه اندروید آلوده به بدافزار بات از ISCX بارگیری شدند. مجموعه داده‌ها، ترکیبی از نمونه‌های بات‌نت از پروژه بدافزار ژنوم و وبلاگ امنیتی بدافزار VirusTotal است. به‌طورکلی، مجموعه داده برنامه‌های کاربردی آلوده به بدافزار بات، شامل ۱۴ خانواده از بات‌نت‌ها و ۱۹۳۲ نمونه است. برای دسترسی به مجوزهای برنامه‌های کاربردی اندروید، از مهندسی معکوس فایل‌های APK استفاده شد.

۳-۲-۱- مهندسی معکوس

مهندسی معکوس یک فایل APK، برای دسترسی به محتوا، تصاویر و پایگاه داده، فایل‌ها و لایه‌های طراحی، کدهای XML، کدهای جاوا یا کاتلین، طراحی ساختار برنامه موردنظر، رسیدن به کدهای منبع برنامه‌های کاربردی و اعمال تغییرات در آن استفاده می‌شود. ابزارهای متعددی برای مهندسی معکوس برنامه‌های کاربردی اندروید وجود دارند. مجموعه داده‌ی برنامه‌های کاربردی اندروید آلوده به بدافزار بات، با استفاده از ابزار APKtool مهندسی معکوس شدند. ۲۰۴ مجوز متمایز و منحصربه‌فرد به دست آمد؛ که بیش‌ترین فراوانی به مجوز اینترنت تعلق داشت. سپس این مجوزها برای ایجاد تصاویر برنامه‌های کاربردی مورد استفاده قرار گرفتند. مجموعه دادهٔ برنامه‌های کاربردی بی‌خطر و بدون هیچ‌گونه آلودگی به بدافزار، ویروس و غیره از کگل[1] بارگیری شد. این مجموعه داده شامل ۴۳۰۴ برنامه کاربردی بی-

[1] Kaggle

خطر است. مجوزهای برنامه کاربردی بی‌خطر نیز استخراج شد که در نتیجه ۱۹۸۱ مجوز منحصربه‌فرد و متمایز به دست آمد. مجوزهای برنامه‌های کاربردی اندروید در مراحل بعدی برای ایجاد تصاویر برنامه‌های کاربردی مورداستفاده قرار گرفتند. الگوریتم (۱) نحوهٔ ایجاد ماتریس مجوزها را نشان می‌دهد.

Algorithm1 Create a merge list of permissions

Input: Botnet permissions BP
Normal permissions NP,

Output: Merged permissions MP

Procedure:
1: i × k ← 0 × 0
2: BI ← Botnet permissions. length;
3: NI ← Normal permissions. length;
4: MP: newlist();
5: While i < BI:
5.1: MP. append(BI[i])
5.2: i ← i + 1
6: End While
7: While k < NI:
7.1: MP. append(NI[k])
7.2: k ← k + 1
8: End While;
9: Calculate the frequency of list elements MP;
10: Select 61 permissions with the highest frequency

الگوریتم (۱). الگوریتم ساخت لیست مجوزها

۳-۲-۲- ایجاد تصاویر

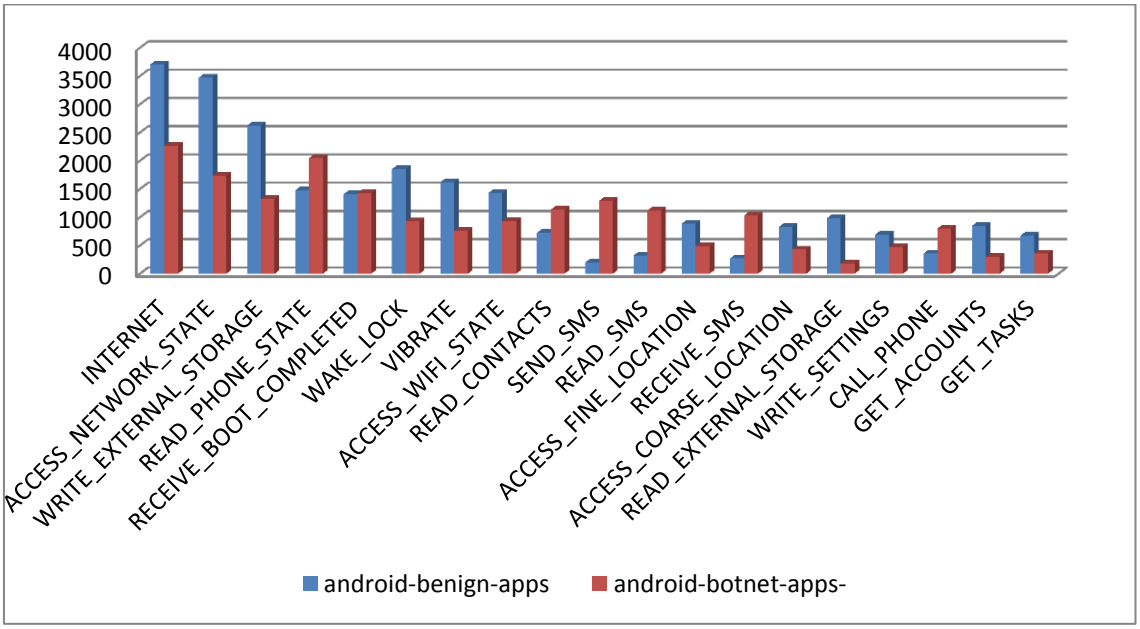
در مراحل قبلی مجوزهای برنامه‌های کاربردی اندروید استخراج و فراوانی مجوزها محاسبه شد. سپس ۶۱ مجوز با بیشترین فراوانی برای ایجاد ماتریس تصاویر انتخاب شد که این ۶۱ مجوز، به‌عنوان برچسب سطر و ستون‌های ماتریس قرار داده شدند (شکل (۲)). به صورتی که سطر و ستون‌های هم‌نام متناظر هستند. برای ایجاد ماتریس برنامه‌های کاربردی، به‌ازای هر برنامه کاربردی یک ماتریس صفر با ابعاد (۶۱×۶۱) در نظر گرفته شد. اگر هر برنامه کاربردی، مجوزهای موجود در لیست مجوزها (۶۱ مجوز موردنظر) را دارا بود با توجه به الگوریتم (۱)، درایه‌های موردنظر ماتریس برنامهٔ کاربردی مقدار یک قرار داده شد. بعد از تشکیل ماتریس برنامه‌های کاربردی، نوبت به ایجاد تصاویر برنامه‌های کاربردی بر اساس ماتریس به‌دست‌آمده در مرحله قبل رسید. الگوریتم (۲) روش پیشنهادی جهت تولید تصاویر متنی بر مجوزها را نشان می‌دهد. همچنین چند نمونه از تصاویر تولیدی برای برنامه‌های بی‌خطر و برنامه‌های بات‌نت در شکل‌های (۳) و (۴) دیده می‌شوند.

Algoritm.2 Create Images of Botnet applications according to MP lists

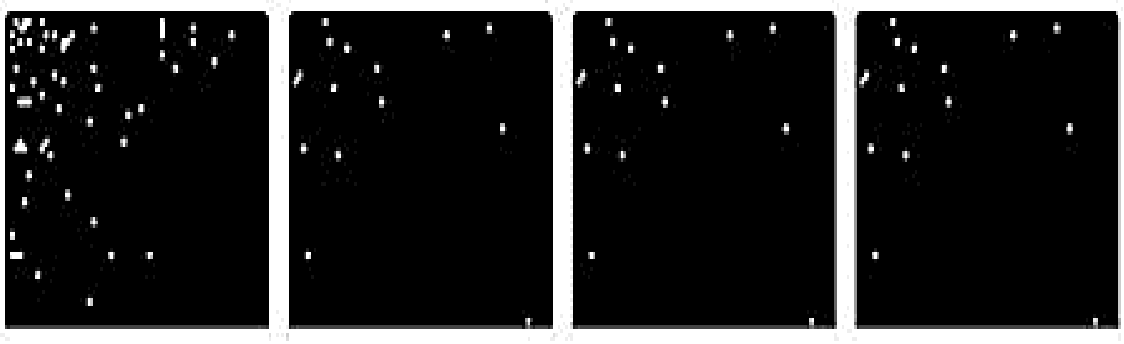
Input: Merged permissions MP,
Normal permissions NP,
Botnet permissions BP,
Output: Images of applications

```
1.  I × J ← 0 × 0 ;
2.  List1 ← new list () ;
3.  L ← Botnet permissions. lenght ;
4.  if l == 0 :
      4.1. data ← np. zeros (61,61)
      4.2. Save image (data);
5.  If <> 0 ;
      5.1. data ← np. zeros (61,61)
      5.2. For j in range (0, l) :
      5.3.     If BP in MP :
      5.4.         row = vec [j]
      5.5.         col = vec [j + 1 ]
      5.6.         data [row , col ] = 1
      5.7.         Save image (data) ;
```

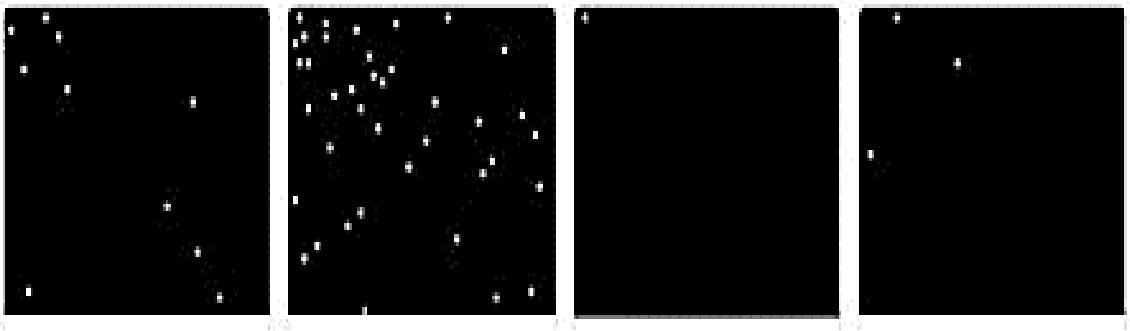
الگوریتم (۲) . الگوریتم ساخت تصاویر برنامه‌های کاربردی



شکل (۲). فراوانی مجوزهای برنامه‌های بی‌خطر و باتنت



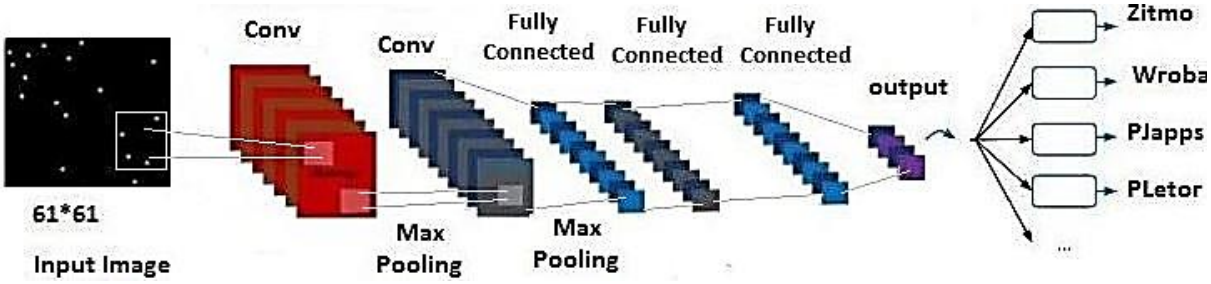
شکل (۳). نمونه‌هایی از تصاویر برنامه‌های کاربردی آلوده به بات‌نت



شکل (۴). نمونه‌هایی از تصاویر برنامه‌های کاربردی بی‌خطر

۴- مدل‌سازی

برای تشخیص و دسته‌بندی بات‌نت‌ها از شبکه عصبی عمیق (یادگیری با نظارت) استفاده شد. در یادگیری با نظارت، مدل با استفاده از مثال‌ها آموزش می‌بیند و برای آموزش نیاز به داده‌های برچسب‌گذاری شده دارد. بنابراین مجموعه داده‌ها برای شبکهٔ عصبی عمیق برچسب‌گذاری شدند. سپس مجموعه داده‌ها به سه قسمت داده آموزشی، اعتبار¬سنجی و تست با نسبت‌های ۶۰ درصد، ۲۰ درصد، ۲۰ درصد تقسیم‌بندی شدند. در نهایت مدل با استفاده از شبکه‌های عصبی کانولوشن که روشی بسیار کارآمد در زمینهٔ پردازش تصویر است مشخص شد. در شکل (۵) معماری این مدل نشان‌داده‌شده است. مدل شبکه عصبی به‌صورت ترتیبی و متشکل از لایه‌های کانولوشن، پولینگ و لایه‌های کاملاً متصل است. لایه کانولوشن، وظیفه ضرب داخلی بین ورودی‌ها و پارامترهای هر نورون و به‌طورکلی عملیات کانولوشن در هر لایه را انجام می‌دهد. سپس خروجی شبکه محاسبه می‌شود و در جهت تنظیم پارامترهای شبکه برای محاسبه میزان خطای شبکه مورداستفاده قرار می‌گیرد. بدین صورت که نتایج با استفاده از تابع خطا با پاسخ‌های صحیح مورد مقایسه قرار می‌گیرد و خطا محاسبه می‌شود. با استفاده از شبکه کانولوشن فیلترهای هر لایه وابستگی مکانی هر تصویر را به دست می‌آورند. در اولین لایه کانولوشن، ویژگی‌های سطح پایین به دست می‌آید. از لایه‌های پولینگ به‌منظور کاهش ابعاد در جهت کاهش محاسبات پردازش داده‌ها استفاده می‌کنیم. در مدل پیشنهادی از پولینگ حداکثری به‌منظور برگرداندن حداکثر مقدار تصویر توسط کرنل استفاده شد. در لایه‌های بعدی ویژگی‌های سطح بالا برای دسته‌بندی تصاویر به دست می‌آید. بعد از آخرین لایه پولینگ، لایه‌های تماماً متصل قرار گرفته‌اند که ویژگی‌های دوبعدی را به بردار ویژگی‌های یک‌بعدی برای استخراج ویژگی‌ها و نمایش ویژگی‌ها تبدیل می‌کنند. این لایه، نتیجه آموزش شبکه را در قالب یک بردار با اندازه مشخص ارائه می‌دهد، از این بردار برای دسته‌بندی مجموعه داده برنامه‌های کاربردی (تصاویر) استفاده‌شده است. مجموعه داده‌ها به‌صورت دسته‌ای در هر دوره برای آموزش و تست در شبکه عصبی اجرا شدند. تصویری از مدل شبکه پیشنهادی در شکل (۶) آمده است. پس از پایان فاز آموزش، مدل درک کاملی از تصاویر، تناظر با برنامه‌های کاربردی به دست می‌آورد و می‌توان از آن برای تشخیص و دسته‌بندی بات‌نت‌ها استفاده کرد. درنهایت نیز نتایج یادگیری شبکه عصبی توسط معیارهایی ازجمله کاپا، دقت، صحت، فراخوانی مورد ارزیابی قرار گرفت.



شکل (۵). معماری مدل پیشنهادی

Layer (type)	Output Shape	Param #
Input_11 (Input Layer)	(None, 61, 61, 1)	0
Conv2d_30 (Conv2D)	(None, 61, 61, 12)	120
Max_pooling_30 (Maxpooling2D)	(None, 61, 61, 12)	0
Conv2d_32 (Conv2D)	(None, 61, 61, 48)	5232
Max_pooling_32 (Maxpooling2D)	(None, 31, 31, 48)	0
Flatten_10 (Flatten)	(None , 46128)	0
dense_30 (Dense)	(None , 48)	2214192
dense_31 (Dense)	(None , 24)	1176
dense_32 (Dense)	(None , 14)	350
Total params : 2, 221, 070		
Trainable params : 2, 221, 070		
None-trainable param: 0		

شکل (۶). تصویری از مدل شبکه عصبی

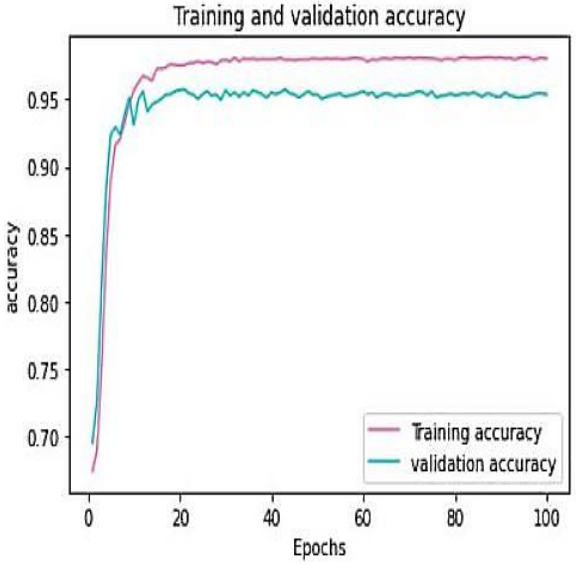
۵- ارزیابی

برای ارزیابی کارایی روش پیشنهادی، شبکه کانولوشن در ۱۰۰ دوره، در محیط آزمایشگاهی تحت آموزش، ارزیابی و تست قرار گرفت. آموزش شبکه عصبی را با پارامترهای مختلف: نرخ یادگیری، تعداد لایه کانولوشن، تعداد لایه‌های کاملاً متصل، تعداد نورون، نوع بهینه‌ساز و غیره تکرار کردیم. تا الگوریتمی با بهترین دقت و نرخ خطای کمتر انتخاب گردد. شکل (۷) نمودار تغییرات دقت و شکل (۸) تغییرات loss را در ۱۰۰ دوره نمایش می‌دهد.

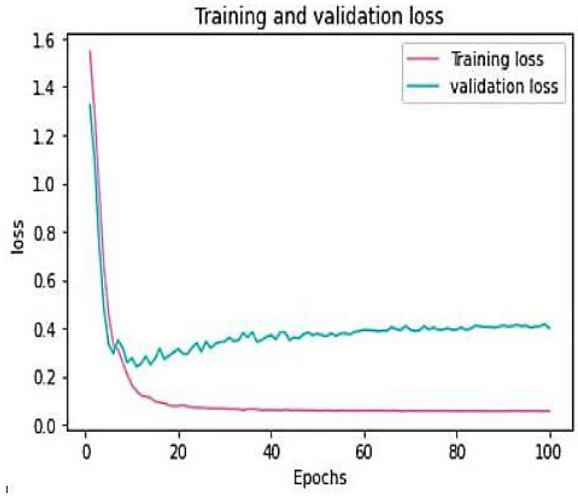
همان‌طور که مشاهده می‌شود، شبکه عصبی در شکل (۷) با یک نرخ صعودی متعادل در حال رشد است و از جایی به بعد دیگر، تغییرات قابل‌توجهی در وزن‌ها مشاهده نمی‌شود.

۵-۱- معیارهای ارزیابی

برای سنجش بهتر نتایج، از معیارهای صحت[1]، دقت[2]، امتیاز F1 (میانگین هارمونیک دو معیار صحت و دقت)، معیار کاپا، ماتریس درهم‌ریختگی، منحنی مشخصه عملکرد[3]، فراخوانی یا حساسیت[4] استفاده‌شده است. نتایج را بعد از تحلیل می‌توان به دو گروه داده مثبت و منفی تقسیم کرد. روش‌های آزمون و ارزیابی، نتایج یک آزمایش را به این دو دسته تقسیم می‌کند. سپس کارایی الگوریتم با استفاده از معیارها و شاخص‌های موردنظر، قابل‌اندازه‌گیری و توصیف می‌شوند.



شکل (۷). نمودار accuracy در ۱۰۰ دوره



شکل (۸). نمودار loss در ۱۰۰ دوره

[1] Precision
[2] Accuracy
[3] ROC
[4] Recall

TP: تعداد مثبت صحیح است. مشاهداتی که بخشی از کلاس مثبت هستند و درست پیش‌بینی‌شده‌اند.

FP: تعداد مثبت کاذب، مشاهدات پیش‌بینی‌شده، بخشی از کلاس مثبت هستند ولی در واقعیت بخشی از کلاس منفی تشخیص داده شده‌اند. در اینجا خطای نوع اول رخداده است.

TN: تعداد منفی صحیح، مشاهداتی که بخشی از کلاس منفی هستند و درست پیش‌بینی شده‌اند.

FN: تعداد منفی کاذب، مشاهدات پیش‌بینی‌شده که بخشی از کلاس منفی باشند ولی اشتباه به کلاس مثبت تعلق یافته‌اند. در اینجا خطای نوع دوم اتفاق افتاده است.

- **دقت:** دقت به این معناست که مدل تا چه اندازه خروجی را درست پیش‌بینی می‌کند. با نگاه‌کردن به‌دقت، بلافاصله می‌توان دریافت که آیا مدل درست آموزش‌دیده است یا خیر و کار آیی آن به‌طورکلی چگونه است؛ اما این معیار اطلاعات جزئی در مورد کار آیی مدل ارائه نمی‌دهد.

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

- **صحت:** وقتی مدل نتیجه را مثبت پیش‌بینی می‌کند، این نتیجه تا چه اندازه درست است، زمانی که ارزش مثبت‌های کاذب، بالا باشد، معیار صحت، معیار مناسبی خواهد بود. اگر مدلی صحت پایینی داشته باشد. نتیجه این امر این است که این مدل، بسیاری از نمونه‌ها را به‌اشتباه تشخیص داده است.

$$Precision = \frac{TP}{(TP + FP)} \quad (۲)$$

- **فراخوانی:** زمانی که ارزش منفی‌های کاذب بالا باشد، معیار حساسیت، معیار مناسبی خواهد بود. فرض کنیم مدلی برای تشخیص داشته باشیم. اگر این مدل حساسیت پایینی داشته باشد به این معنی است که این مدل نمونه‌های زیادی را به‌اشتباه صحیح تشخیص می‌دهد.

$$Recall = \frac{TP}{(TP + FN)} \quad (۳)$$

- **امتیاز F1:** یک معیار مناسب برای ارزیابی دقت یک مدل است. این معیار حساسیت و صحت را باهم در نظر می‌گیرد. معیار F1 در بهترین حالت، یک و در بدترین حالت صفر است.

$$\mathrm{F1} = \frac{2 \times Precission \times Recall}{Precission + Recall} \quad (۴)$$

- **ماتریس درهم‌ریختگی:** انتخاب یک معیار برای کارایی، به مسئله‌ای که سعی در حل آن داریم، وابسته است. اگر فرض کنیم ۱۰۰ نمونه داده در دسترس باشد. این داده‌ها تک‌تک به مدل داده شده و به‌ازای هرکدام، یک کلاس به‌عنوان خروجی دریافت شود. کلاس پیش‌بینی‌شده توسط مدل و کلاس واقعی داده‌ها را می‌توان در یک جدول نمایش داد. به این جدول، ماتریس اختلاط گفته می‌شود.
- **کاپا:** یکی دیگر از این معیارها به نام امتیاز کاپا که به معیار Cohen's Kappaنیز معروف است. این معیار یک معیار مناسب، برای ارزیابی کیفیت الگوریتم‌های دسته بندی چند کلاسه است. معیار کاپا، الگوریتم طبقه‌بندی ما را با یک الگوریتم دسته‌بندی تصادفی مقایسه می‌کند و به ما می‌گویند که الگوریتم طبقه‌بندی، چقدر از یک الگوریتم تصادفی، بهتر عمل کرده است؛ بنابراین معیار کاپا می‌تواند عملکرد الگوریتم‌های چند کلاسه را ارزیابی کند. این معیار قادر است که تعداد نمونه‌ها را در یک مجموعه‌ی داده‌ی نامتوازن نیز در نظر بگیرد و کیفیت الگوریتم را با توجه به تعداد نمونه‌ها بیان کند. پس برای مسائلی با طبقه‌های مختلف و همچنین مسائلی که داده‌های نامتوازن دارند، می‌توان از این معیار استفاده کرد. جدول (۱) عملکردهای مختلف بر اساس معیار کاپا را ارائه می‌دهد. که بر اساس معیار کاپا به مقدار ۹۱ درصد حاصل شد.

جدول (۱). عملکرد معیار کاپا

عملکرد خیلی بد	کمتر از ۰
عملکرد ضعیف	بین ۰ تا ۲۰
عملکرد متوسط روبه‌پایین	بین ۲۱ تا ۴۰
عملکرد متوسط	بین ۴۱ تا ۶۰
عملکرد خوب	بین ۶۱ تا ۸۰
عملکرد عالی	بین ۸۱ تا ۱۰۰

۶- نتایج مدل بر اساس معیارهای ارزیابی

چون مسئله مورد نظر از نظر تعداد نمونه‌ها در هر کلاس نامتوازن است در کنار معیار دقت از معیارهای .Macro Avg (میانگین کلان) و .Weighted Avg (میانگین وزنی) نیز استفاده نمودیم. معیارها به‌صورت زیر تعریف می‌شوند:

$$Macro\ Avg\ Precision = \frac{\sum_{i=1}^{N} Precision_i}{N} \quad (۵)$$

$$Macro\ Avg\ Recall = \frac{\sum_{i=1}^{N} Recall_i}{N} \quad (۶)$$

$$Macro\ Avg\ F1-Score\ = \frac{\sum_{i=1}^{N} F1-Score_i}{N} \quad (۷)$$

Macro Avg بر طبق معیار صحت ۹۵ درصد، بر اساس معیار فراخوانی، ۸۸ درصد و بر اساس امتیاز F1، ۹۱ درصد به دست آمد. همچنین Weighted Avg بر اساس ۳ معیار صحت، فراخوانی و امتیاز F1، ۹۶ درصد حاصل شد (شکل (۹)). ماتریس درهم‌ریختگی در شکل (۱۰) آمده است.

جدول (۲). نتایج بر اساس TPR، FNR، FPR،TNR

	TPR	FNR	FPR	TNR
۱	۰/۹۸۸۳	۰/۰۱۱۶	۰/۰۵۹۷	۰/۹۴۰۲
۲	۰/۰۰۰۰	۰/۰۰۰۰	۰/۰۰۰۰	۱/۰۰۰۰
۳	۰/۹۰۴۱	۰/۰۹۵۸	۰/۰۱۲۷	۰/۹۸۷۲
۴	۰/۸۳۰۱	۰/۱۶۹۸	۰/۰۰۰۸	۰/۹۹۹۱
۵	۰/۸۵۰۰	۰/۱۵۰۰	۰/۰۰۰۰	۱/۰۰۰۰
۶	۰/۹۷۵۰	۰/۰۲۵۰	۰/۰۰۰۸	۰/۹۹۹۱
۷	۱/۰۰۰۰	۰/۰۰۰۰	۰/۰۰۱۶	۰/۹۹۸۳
۸	۰/۶۹۳۸	۰/۳۰۶۱	۰/۰۰۸۳	۰/۹۹۱۶
۹	۰/۹۴۱۱	۰/۰۵۵۸	۰/۰۰۰۰	۱/۰۰۰۰
۱۰	۰/۶۰۰۰	۰/۴۰۰۰	۰/۰۰۰۰	۱/۰۰۰۰
۱۱	۰/۶۶۶۶	۰/۳۳۳۳	۰/۰۰۰۸	۰/۹۹۹۱
۱۲	۱/۰۰۰۰	۰/۰۰۰۰	۰/۰۰۰۰	۱/۰۰۰۰
۱۳	۰/۹۵۰۰	۰/۰۵۰۰	۰/۰۰۰۰	۱/۰۰۰۰
۱۴	۰/۹۳۷۵	۰/۰۶۲۵	۰/۰۰۰۰	۱/۰۰۰۰

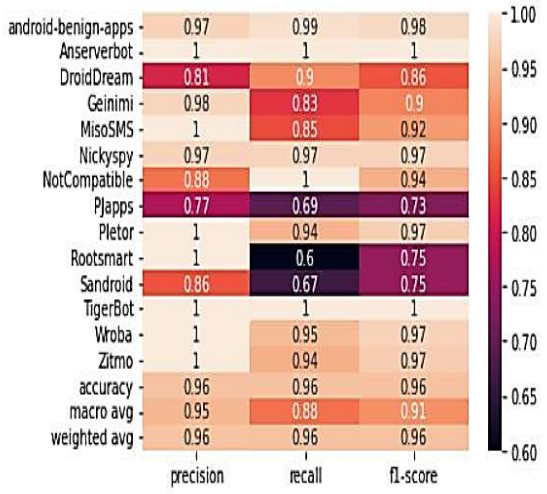
TPR، نرخ تولید داده‌های درست (برای دستهٔ مثبت) وFPR ، نرخ تولید خطای دسته‌بندی (برای دستهٔ مثبت)، TNR نرخ تولید داده‌های درست (برای دستهٔ منفی) وFNR، نرخ تولید خطای دسته‌بندی (برای دستهٔ منفی)، با توجه به روابط زیر محاسبه گردیده و در جدول (۲) نشان داده شده است.

$$TPR\ = \frac{TP}{TP+FN} \quad (۸)$$

$$FPR\ = \frac{FP}{FP+TN} \quad (۹)$$

$$TNR\ = \frac{TN}{TN+FP} \quad (۱۰)$$

$$FNR\ = \frac{FN}{TP+FN} \quad (۱۱)$$



شکل (۹). نمایش نتایج با استفاده از کتابخانه Seaborn

```
[[851   0   3   1   0   1   2   2   0   0   1   0   0   0]
 [  0  49   0   0   0   0   0   0   0   0   0   0   0   0]
 [  3   0  66   0   0   0   0   4   0   0   0   0   0   0]
 [  7   0   0  44   0   0   0   2   0   0   0   0   0   0]
 [  2   0   0   0  17   0   0   1   0   0   0   0   0   0]
 [  1   0   0   0   0  39   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  15   0   0   0   0   0   0   0]
 [  4   0  11   0   0   0   0  34   0   0   0   0   0   0]
 [  1   0   0   0   0   0   0   0  16   0   0   0   0   0]
 [  1   0   1   0   0   0   0   0   0   3   0   0   0   0]
 [  2   0   0   0   0   0   0   1   0   0   6   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  19   0   0]
 [  1   0   0   0   0   0   0   0   0   0   0   0  19   0]
 [  1   0   0   0   0   0   0   0   0   0   0   0   0  15]]
```

شکل (۱۰). ماتریس درهم‌ریختگی

۶-۱- الگوریتم‌های مقایسه شونده

برای ارزیابی، مدل را با الگوریتم‌های کلاسیک ازجمله ماشین بردار پشتیبان و درخت تصمیم مورد مقایسه قراردادیم.

۶-۱-۱- الگوریتم درخت تصمیم

درخت تصمیم از یک ساختار درخت‌مانند استفاده می‌کند که داده‌های برچسب‌گذاری شده را مدل می‌کند. ساختار آن از برگ‌ها و شاخه‌ها تشکیل‌شده است که درواقع به ترتیب نشان‌دهنده طبقه‌بندی‌ها و ترکیب ویژگی‌هایی است که منجر به آن طبقه‌بندی می‌شود. در طول طبقه‌بندی، یک ورودی بدون برچسب با آزمایش مقادیر ویژگی آن در برابر گره‌های درخت تصمیم طبقه‌بندی می‌شود. دو پیاده‌سازی الگوریتمی محبوب برای درخت تصمیم شامل Decision Trees ID3 و C4.5 است که بر اساس این که از چه معیاری برای انتخاب ویژگی بهتر در گره‌های میانی استفاده می‌کنند با یکدیگر متفاوت هستند.

جدول (۳). نتایج حاصل ماشین بردار پشتیبان

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
۰/۹۷۷	۰/۰۷۸	۰/۹۶۶	۰/۹۷۷	۰/۹۷۱	۰/۹۰۶	۰/۹۵۱	۰/۹۶۰	۰
۱/۰۰۰	۰/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱
۰/۷۹۵	۰/۰۰۵	۰/۹۰۶	۰/۷۹۵	۰/۸۴۷	۰/۸۴۰	۰/۹۴۱	۰/۷۴۷	۲
۰/۷۳۶	۰/۰۰۲	۰/۹۵۱	۰/۷۳۶	۰/۸۳۰	۰/۸۳۱	۰/۹۳۲	۰/۷۳۰	۳
۰/۰۰۹	۰/۰۰۰	۱/۰۰۰	۰/۹۰۰	۰/۹۴۷	۰/۹۴۸	۰/۹۹۱	۰/۹۱۲	۴
۰/۹۲۵	۰/۰۰۲	۰/۹۴۹	۰/۹۲۵	۰/۹۳۷	۰/۹۳۵	۰/۹۷۱	۰/۸۸۶	۵
۰/۹۳۳	۰/۰۰۷	۰/۶۰۹	۰/۹۳۳	۰/۷۳۷	۰/۷۵۰	۰/۹۴۲	۰/۵۶۹	۶
۰/۸۹۸	۰/۰۱۱	۰/۷۷۲	۰/۸۹۸	۰/۸۳۰	۰/۸۲۵	۰/۹۶۰	۰/۷۰۳	۷
۱/۰۰۰	۰/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۹۹۷	۸
۰/۸۰۰	۰/۰۰۰	۱/۰۰۰	۰/۸۰۰	۰/۸۸۹	۰/۸۹۴	۰/۹۷۹	۰/۸۰۴	۹
۰/۵۵۶	۰/۰۰۰	۱/۰۰۰	۰/۵۵۶	۰/۷۱۴	۰/۷۴۴	۰/۹۱۱	۰/۵۹۵	۱۰
۱/۰۰۰	۰/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱۱
۱/۰۰۰	۰/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱۲
۱/۰۰۰	۰/۰۰۲	۰/۸۴۲	۱/۰۰۰	۰/۹۱۴	۰/۹۱۷	۰/۹۹۹	۰/۸۴۲	۱۳
۰/۹۴۸	۰/۰۵۵	۰/۹۵۱	۰/۹۴۸	۰/۹۴۷	۰/۹۰۲	۰/۹۵۶	۰/۹۱۸	Avg

جدول (۴). نتایج حاصل از روش درخت تصمیم C4.5

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
۰/۹۷۸	۰/۱۳۵	۰/۹۴۲	۰/۹۷۸	۰/۹۶۰	۰/۸۶۵	۰/۹۴۴	۰/۹۵۴	۰
۰/۹۸۰	۰/۰۰۰	۱/۰۰۰	۰/۹۸۰	۰/۹۹۰	۰/۹۸۹	۰/۹۳۸	۰/۹۸۰	۱
۰/۷۵۳	۰/۰۰۲	۰/۹۶۵	۰/۷۵۳	۰/۸۴۶	۰/۸۴۵	۰/۹۷۰	۰/۸۷۱	۲
۰/۷۵۵	۰/۰۰۳	۰/۹۳۰	۰/۷۷۵	۰/۸۳۳	۰/۸۳۲	۰/۸۹۲	۰/۷۳۹	۳
۰/۹۰۰	۰/۰۰۰	۱/۰۰۰	۰/۹۰۰	۰/۹۴۷	۰/۹۴۸	۰/۹۶۴	۰/۹۰۲	۴
۰/۷۷۵	۰/۰۰۱	۰/۹۶۹	۰/۷۷۵	۰/۸۶۱	۰/۸۶۳	۰/۹۲۰	۰/۷۹۸	۵
۰/۹۳۳	۰/۰۱۰	۰/۵۳۸	۰/۹۳۳	۰/۶۸۳	۰/۷۰۵	۰/۹۶۲	۰/۵۰۳	۶
۰/۸۵۷	۰/۰۰۸	۰/۸۰۸	۰/۸۵۷	۰/۸۳۲	۰/۸۲۵	۰/۹۶۴	۰/۸۴۲	۷
۰/۹۴۱	۰/۰۰۱	۰/۹۴۱	۰/۹۴۱	۰/۹۴۱	۰/۹۴۰	۰/۹۵۲	۰/۸۸۷	۸
۰/۸۰۰	۰/۰۰۰	۱/۰۰۰	۰/۸۰۰	۰/۸۸۹	۰/۸۹۴	۰/۹۰۰	۰/۸۰۱	۹
۰/۴۴۴	۰/۰۰۲	۰/۶۶۷	۰/۴۴۴	۰/۵۳۳	۰/۵۴۲	۰/۸۲۷	۰/۳۰۳	۱۰
۰/۸۴۲	۰/۰۰۰	۱/۰۰۰	۰/۸۴۲	۰/۹۱۴	۰/۹۱۷	۱/۰۰۰	۰/۹۶۷	۱۱
۰/۹۰۰	۰/۰۰۰	۱/۰۰۰	۰/۹۰۰	۰/۹۴۷	۰/۹۴۸	۰/۹۶۹	۰/۹۰۳	۱۲
۰/۹۳۸	۰/۰۰۰	۱/۰۰۰	۰/۹۳۸	۰/۹۶۸	۰/۹۶۸	۰/۹۸۰	۰/۹۳۹	۱۳
۰/۹۳۳	۰/۰۹۴	۰/۹۳۷	۰/۹۳۳	۰/۹۳۳	۰/۸۶۷	۰/۹۴۶	۰/۹۱۹	Avg

جدول (۵). مقایسه مقادیر صحت، فراخوانی و امتیاز F1 در سه الگوریتم

Class	Precision SVM	Precision C4.5	Precision Model	Recall SVM	Recall C4.5	Recall Model	F1 SVM	F1 C4.5	F1 Model
Class 1	۰/۹۶۶	۰/۹۴۲	۰/۹۷۰	۰/۹۷۷	۰/۹۷۸	۰/۹۹۰	۰/۹۷۱	۰/۹۶۰	۰/۹۸۰
Class 2	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۹۸۰	۱/۰۰۰	۱/۰۰۰	۰/۹۹۰	۱/۰۰۰
Class 3	۰/۹۰۶	۰/۹۶۵	۰/۸۱۰	۰/۷۹۵	۰/۷۵۳	۰/۹۰۰	۰/۸۴۷	۰/۸۴۶	۰/۸۶۰
Class 4	۰/۹۵۱	۰/۹۳۰	۰/۹۸۰	۰/۷۳۶	۰/۷۵۵	۰/۸۳۰	۰/۸۳۰	۰/۸۳۳	۰/۹۰۰
Class 5	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۹۰۰	۰/۹۰۰	۰/۸۵۰	۰/۹۴۷	۰/۹۴۷	۰/۹۲۰
Class 6	۰/۹۴۹	۰/۹۶۹	۰/۹۷۰	۰/۹۲۵	۰/۷۷۵	۰/۹۷۰	۰/۹۳۷	۰/۸۶۱	۰/۹۷۰
Class 7	۰/۶۰۹	۰/۵۳۸	۰/۸۸۰	۰/۹۳۳	۰/۹۳۳	۱/۰۰۰	۰/۷۳۷	۰/۶۸۳	۰/۹۴۰
Class 8	۰/۷۷۲	۰/۸۰۸	۰/۷۷۰	۰/۸۹۸	۰/۸۵۷	۰/۶۹۰	۰/۸۳۰	۰/۸۳۲	۰/۷۳۰
Class 9	۱/۰۰۰	۰/۹۴۱	۱/۰۰۰	۱/۰۰۰	۰/۹۴۱	۰/۹۴۰	۱/۰۰۰	۰/۹۴۱	۰/۹۷۰

Class	Precision SVM	Precision C4.5	Precision Model	Recall SVM	Recall C4.5	Recall Model	F1 SVM	F1 C4.5	F1 Model
Class 10	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۸۰۰	۰/۸۰۰	۰/۶۰۰	۰/۸۸۹	۰/۸۸۹	۰/۷۵۰
Class 11	۱/۰۰۰	۰/۶۶۷	۰/۸۶۰	۰/۵۵۶	۰/۴۴۴	۰/۶۷۰	۰/۷۱۴	۰/۵۳۳	۰/۷۵۰
Class 12	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۸۴۲	۱/۰۰۰	۱/۰۰۰	۰/۹۱۴	۱/۰۰۰
Class 13	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۹۰۰	۰/۹۵۰	۱/۰۰۰	۰/۹۴۷	۰/۹۷۰
Class 14	۰/۸۴۲	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰	۰/۹۳۸	۰/۹۴۰	۰/۹۱۴	۰/۹۶۸	۰/۹۷۰
Average	۰/۹۵۱	۰/۹۳۷	۹۶۰	۰/۹۴۸	۰/۹۳۳	۰/۹۶۰	۰/۹۴۷	۰/۹۳۳	۹۶۰

جدول (۶). مقایسه مقادیرFP، TP،ROC در سه الگوریتم

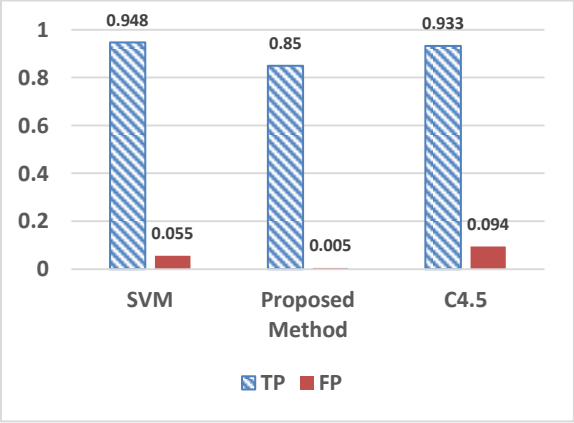
Class	TP SVM	TP C4.5	TP Model	FP SVM	FP C4.5	FP Model	ROC SVM	ROC C4.5	ROC Model
Class 1	۰/۹۷۷	۰/۹۷۸	۰/۹۸۸	۰/۰۷۸	۰/۱۳۵	۰/۰۵۹	۰/۹۵۱	۰/۹۴۴	۰/۹۸۰
Class 2	۱/۰۰۰	۰/۹۸۰	۱/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۱/۰۰۰	۰/۹۳۸	۱/۰۰۰
Class 3	۰/۷۹۵	۰/۷۵۳	۰/۹۰۴	۰/۰۰۵	۰/۰۰۲	۰/۰۱۲	۰/۹۴۱	۰/۹۷۰	۰/۹۹۰
Class 4	۰/۷۳۶	۰/۷۵۵	۰/۸۳۰	۰/۰۰۲	۰/۰۰۳	۰/۰۰۰	۰/۹۳۲	۰/۸۹۲	۰/۹۷۰
Class 5	۰/۹۰۰	۰/۹۰۰	۰/۸۵۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۹۹۱	۰/۹۶۴	۰/۹۰۰
Class 6	۰/۹۲۵	۰/۷۵۵	۰/۹۷۵	۰/۰۰۲	۰/۰۰۱	۰/۰۰۰	۰/۹۷۱	۰/۹۲۰	۰/۹۹۰
Class 7	۰/۹۳۳	۰/۹۳۳	۱/۰۰۰	۰/۰۰۷	۰/۰۱۰	۰/۰۰۱	۰/۹۴۲	۰/۹۶۲	۱/۰۰۰
Class 8	۰/۸۹۸	۰/۸۵۷	۰/۶۹۳	۰/۰۱۱	۰/۰۰۸	۰/۰۰۸	۰/۹۶۰	۰/۹۶۴	۰/۹۵۰
Class 9	۱/۰۰۰	۰/۹۴۱	۰/۹۴۱	۰/۰۰۰	۰/۰۰۱	۰/۰۰۰	۱/۰۰۰	۰/۹۵۲	۱/۰۰۰
Class 10	۰/۸۰۰	۰/۸۰۰	۰/۶۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۹۷۹	۰/۹۰۰	۰/۸۶۰
Class 11	۰/۵۵۶	۰/۴۴۴	۰/۶۶۶	۰/۰۰۰	۰/۰۰۲	۰/۰۰۰	۰/۹۱۱	۰/۸۲۷	۰/۹۷۰
Class 12	۱/۰۰۰	۰/۸۴۲	۱/۰۰۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۱/۰۰۰	۱/۰۰۰	۱/۰۰۰
Class 13	۱/۰۰۰	۰/۹۰۰	۰/۹۵۰	۰/۰۰۰	۰/۰۰۰	۰/۰۰۰	۱/۰۰۰	۰/۹۶۹	۰/۹۹۰
Class 14	۱/۰۰۰	۰/۹۳۸	۰/۹۳۷	۰/۰۰۲	۰/۰۰۰	۰/۰۰۰	۰/۹۹۹	۰/۹۸۰	۱/۰۰۰
Average	۰/۹۴۸	۰/۹۳۳	۰/۸۵۰	۰/۰۵۵	۰/۰۹۴	۰/۰۰۵	۰/۹۵۶	۰/۹۴۶	۰/۹۷۰

۲-۱-۶- ماشین بردار پشتیبان

SVM داده‌های ورودی را با یافتن یک ابر صفحه در فضای با ابعاد بالاتر از ابعاد مجموعه ویژگی‌های موجود در مجموعه داده طبقه‌بندی می‌کند. این تکنیک داده‌های ورودی را که به‌صورت غیرخطی به کلاس‌های جداگانه تقسیم می‌شوند، با اعمال انواع مختلف توابع هسته مانند توابع پایه خطی، چندجمله‌ای، گاوسی و شعاعی به یک فضا با ابعاد بالاتر نگاشت می‌کند با این امید که بتوان در این فضا به‌صورت خطی داده‌ها را از هم جدا کرد. SVM در مقایسه با مدل‌های دیگر، قابلیت تعمیم بهتری دارد و ازاین‌رو می‌تواند در شرایطی که تعداد نمونه‌های آموزشی کمتر است و داده‌ها دارای تعداد زیادی ویژگی هستند، استفاده شود.

۳-۱-۶- مقایسه نتایج با روش‌های کلاسیک

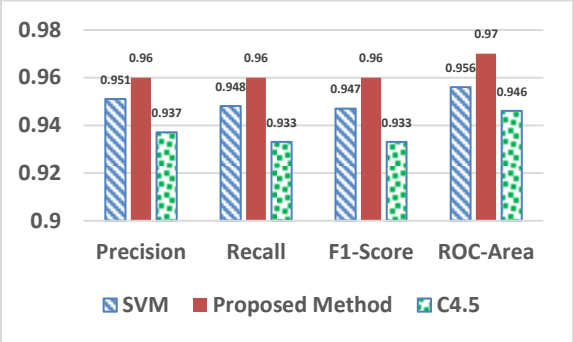
در نمودار و جداول زیر به مقایسه مدل پیشنهادی با روش‌های کلاسیک درخت تصمیم C4.5 و ماشین بردار پشتیبان SVM پرداخته شده است. در شکل (۱۳)، نمودار به مقایسه مقادیر TP و FP در الگوریتم پیشنهادی، درخت تصمیم C4.5 و SVM می‌پردازد. نتایج به‌دست‌آمده از اجرای مدل الگوریتم پیشنهادی و الگوریتم ماشین بردار پشتیبان و درخت تصمیم C4.5 به‌منظور تشخیص اندروید بات‌نت‌ها با توجه به معیارهای صحت، دقت و امتیاز F1 در جدول (۶) آورده شده است. همچنین نتایج به‌دست‌آمده از سه الگوریتم با توجه به FP،TP و منحنی ROC در جدول (۷) آورده شده است.



شکل (۱۳). نمودار مقایسه مقادیر TP و FP

۷- نتایج و بحث

در این پژوهش به توصیف چگونگی ایجاد تصاویر برنامه‌های کاربردی بر اساس مجوزها برای تشخیص و دسته‌بندی باتنت‌ها با استفاده از شبکه عصبی عمیق و یادگیری با نظارت پرداخته شد. روش پیشنهادی با استفاده از زبان برنامه‌نویسی پایتون اجرا شد. سپس به بررسی نتایج بر اساس معیارهای صحت، دقت، فراخوانی و امتیاز F1 برای سنجش و مقایسه با روش‌های کلاسیک ماشین بردار پشتیبان و درخت تصمیم c4.5و کارهای مرتبط در این زمینه انجام گرفت. نتایج روش‌های کلاسیک در جداول (۳) و (۴) آورده شده است. مشاهده شد روش پیشنهادی بر اساس معیارهای صحت، فراخوانی و امتیاز F1 و مقادیر نمودار ROC جداول (۵) و (۶) با روش‌های کلاسیک عملکرد بهتری (شکل (۱۵)) دارد. روش پیشنهادی در مقایسه با کارهای مرتبط (جدول (۷)) بر اساس معیارهای مختلف به نتایج بسیار بهتری دست پیدا کرده است.



شکل (۱۴). مقایسه بر اساس معیارهای ارزیابی

جدول (۷). مقایسه مدل پیشنهادی با کارهای مرتبط

Ref.	Botnet /Benign	Acc	Prec.	Rec.	F1 Score	Kappa
[12]	۱۹۲۶ /۱۵۰	-	۰/۹۷۰	۰/۹۶۹	-	-
[15]	۱۸۰۰ /۳۶۵۰	۹۷/۲	۰/۹۵۵	۰/۹۶۰	۰/۹۵۷	-
[14]	۱۶۳۵ /۱۶۳۵	۹۷/۳	۰/۹۸۷	۰/۹۵۷	-	-
[16]	۱۹۲۹ /۴۸۷۳	۹۸/۹	۹۸۳	۰/۹۷۸	۰/۹۸۱	-
[11]	۱۵۰۵ /۸۵۰	-	۰/۹۳۱	۰/۹۴۶	-	-
[13]	۱۴۰۰ /۱۴۰۰	۹۵/۱	۰/۹۷۰	۰/۸۲۷	-	-
Model	۱۹۳۲ /۴۳۰۴	۹۶	۰/۹۶	۰/۰۹۶	۰/۹۶	۰/۹۱۵

۸- نتیجه‌گیری

با گسترش تلفن همراه هوشمند، بدافزارها و باتنت‌های مربوط به آن‌ها روزبه‌روز در حال گسترش و توسعه می‌باشند. این حفره امنیتی از آنجایی به وجود می‌آید که کاربران بدون توجه به مسائل امنیتی اقدام به نصب و استفاده از این برنامه‌های کاربردی می‌نمایند و به مسائل امنیتی توجهی ندارند. تاکنون رویکردهای مختلفی برای شناسایی آن‌ها ارائه‌شده است؛ و روش‌های مختلفی به بررسی و حل این مشکل پرداخته‌اند که یادگیری عمیق، موفق‌ترین روش‌ها در زمینهٔ شناسایی باتنت‌های جدید را ارائه داده است و بسیار بهتر از سایر روش‌های قدیمی‌تر عمل می‌کند. در این پایان‌نامه، به تشخیص و دسته‌بندی باتنت‌های تلفن همراه اندروید، برای مقابله با یکی از مهم‌ترین چالش‌های امنیتی در تلفن همراه اندروید پرداختیم. بدین صورت که با استفاده از مجوزهای برنامه‌های کاربردی اندروید و مهندسی معکوس برنامه-های اندروید یک روش برای ایجاد تصاویر هر برنامه کاربردی پیشنهادشده است. درمجموع ۶۲۳۶ نمونه شامل ۴۳۰۴ نمونه برنامه کاربردی بی‌خطر و ۱۹۳۲ نمونه برنامه کاربردی آلوده به باتنت جمع‌آوری شدند. سپس از شبکه عصبی کانولوشن برای دسته‌بندی باتنت‌ها بر اساس تصاویر تولیدشده از روی مجوزها استفاده شد. نتایج حاصل از ارزیابی و مقایسه روش ارائه‌شده با سایر روش‌ها نشان داد که مدل پیشنهادشده، تنها با استفاده از مجوزهای اندروید قادر به تشخیص و دسته‌بندی برنامه‌های کاربردی باتنت از بی‌خطر است. از مزیت مدل شبکه عصبی پیشنهادی، طبقه‌بندی باتنت‌ها علاوه بر شناسایی آن‌ها هست که در کارهای مشابه به آن پرداخته نشده یا کم‌تر موردتوجه قرار گرفته است. همچنین نتایج حاصل از ارزیابی نشان داد که کارایی روش پیشنهادی بر اساس معیارهای ارزیابی مختلف بالاتر از سایر روش‌ها است.

۹- مراجع

[1] M. La Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE communications surveys & tutorials*, vol. 15, no. 1, pp. 446-471, 2012.

[2] Y. Zeng, K. G. Shin, and X. Hu, "Design of SMS commanded-and-controlled and P2P-structured mobile botnets," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, 2012, pp. 137-148.

[3] A. Ezzatneshan, S. Kamel Tabbakh Farizani, M. Kheirabadi, and R. Ghaemi, "Providing a new solution to botnet detection in a Markov chain-based network," *Electronic and Cyber Defense*, vol. 9, no. 3, pp. 59-71, 2021.(in Persian)

[4] H. Akbari and S. Safavi, "Estimate botnet using vicarious servers in distributed denial of service attacks," *Journal of Electronical & Cyber Defence*,

pp. 95-109.(in Persian)

[5] A. Flo and A. Josang, "Consequences of botnets spreading to mobile devices," in *Short-Paper Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009)*, 2009, pp. 37-43: Citeseer.

[6] N. Sakthipriya, V. Govindasamy, and V. Akila, "Review of Deep Learning Approaches for IoT Botnet Detection," in *Proceedings of International Conference on Communication and Computational Technologies*, 2023, pp. 521-533: Springer.

[7] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, p. 100529, 2023.

[8] S. Hamzenejadi, M. Ghazvini, and S. Hosseini, "Mobile botnet detection: a comprehensive survey," *International Journal of Information Security*, pp. 1-39, 2022.

[9] M. Moodi, M. Ghazvini, and H. Moodi, "A hybrid intelligent approach to detect android botnet using smart self-adaptive learning-based PSO-SVM," *Knowledge-Based Systems*, vol. 222, p. 106988, 2021.

[10] X. Meng and G. Spanoudakis, "MBotCS: A mobile botnet detection system based on machine learning," in *International Conference on Risks and Security of Internet and Systems*, 2015, pp. 274-291: Springer.

[11] Z. Abdullah, M. M. Saudi, and N. B. Anuar, "ABC: android botnet classification using feature selection and classification algorithms," *Advanced Science Letters*, vol. 23, no. 5, pp. 4717-4720, 2017.

[12] C. Tansettanakorn, S. Thongprasit, S. Thamkongka, and V. Visoottiviseth, "ABIS: a prototype of android botnet identification system," in *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, 2016, pp. 1-5: IEEE.

[13] S. Anwar, J. M. Zain, Z. Inayat, R. U. Haq, A. Karim, and A. N. Jabir, "A static approach towards mobile botnet detection," in *2016 3rd International Conference on Electronic Design (ICED)*, 2016, pp. 563-567: IEEE.

[14] J. f. Alqatawna and H. Faris, "Toward a detection framework for android botnet," in *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 2017, pp. 197-202: IEEE.

[15] S. Hojjatinia, S. Hamzenejadi, and H. Mohseni, "Android botnet detection using convolutional neural networks," in *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, 2020, pp. 1-6: IEEE.

[16] S. Y. Yerima and M. K. Alzaylaee, "Mobile botnet detection: a deep learning approach using convolutional neural networks," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, 2020, pp. 1-8: IEEE.

[17] S. Y. Yerima and Y. To, "A deep learning-enhanced botnet detection system based on Android manifest text mining," in *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, 2022, pp. 1-6: IEEE.

[18] S. Balasunthar and Z. Abdullah, "Comparison of Convolutional Neural Network and Artificial Neural Network for Android Botnet Attack Detection," *Applied Information Technology And Computer Science*, vol. 3, no. 2, pp. 32-49, 2022.

[19] M. Faghihniya, R. Jalaei, and H. Shojaee Yas, "Kavosh: Offering an Analysis Method and the Impact of Normal Network Traffic on Selection and Extraction Based on the Minkowski Distance," *Electronic and Cyber Defense*, vol. 9, no. 1, pp. 137-147, 2021.(in Persian)

[20] A. M. Almuhaideb and D. Y. Alynanbaawi, "Applications of artificial intelligence to detect android botnets: A Survey," *IEEE Access*, vol. 10, pp. 71737-71748, 2022.

[21] R. Mishra and S. K. Jha, "Survey on Botnet Detection Techniques," in *Internet of Things and Its Applications*: Springer, 2022, pp. 441-449.