پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

تشخیص نفوذ به شبکه با ارایه یک الگوریتم یادگیري افزایشی مبتنی بر درخت تصمیم

نگارنده

رایان فرصت

استاد راهنما: هدیه ساجدی

پایان‌نامه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

تاریخ:مرداد ۱۴۰۲

چکیده

الگوریتم های یادگیری ماشین در سالهای اخیر در زمینه ساخت سیستم های تشخیص نفوذ (IDS) بسیار محبوب و قابل اتکا بوده اند. با این حال، بیشتر مدل‌ها به صورت استاتیک می‌باشند و با استفاده از مجموعه‌داده‌های حاوی تمام نفوذهای هدف آموزش می‌بینند و اگر نفوذهای جدید ظاهر شوند، این مدل‌های آموزش‌دیده باید با استفاده از مجموعه‌داده‌های قدیمی و جدید دوباره آموزش ببینند تا تمام نفوذها را به‌درستی دسته‌بندی کنند. در واقعیت‌های جهان واقعی، تهدیدهای جدید به طور مداوم ظاهر می‌شوند. بنابراین، الگوریتم‌های یادگیری ماشینی استفاده‌شده برای (IDS) باید توانایی یادگیری تدریجی بر روی داده های جدید را داشته باشند. برای حل این مسئله، ما برای هنگامی که این نفوذهای جدید ظاهر می‌شوند T-DFNN را پیشنهاد می‌دهیم. T-DFNN یک الگوریتم قابل‌یادگیری است که حمله های جدیدی که ظاهر می‌شوند را به‌صورت تدریجی یاد می‌گیرد. یک مدل T-DFNN از چندین مدل شبکه عصبی روبه جلو عمیق DFNN تشکیل‌شده است که به‌صورت ساختار شبیه به درخت به هم متصل شده‌اند. ما الگوریتم پیشنهادی خود را با استفاده از مجموعه‌داده CICIDS2017، یک مجموعه‌داده تخریب شبکه در دسترس عموم و پرکاربرد که ترافیک های سالم و انواع نفوذهای شبکه را پوشش می‌دهد، مورد بررسی قرار دادیم. نتایج تجربی نشان داد که الگوریتم T-DFNN قادر است به‌صورت تدریجی نفوذهای جدید را یاد بگیرد و اثر فراموشی(catastrophic forgetting) را کاهش دهد. میانگین امتیاز F1 مدل T-DFNN برای هر بار آموزش دوباره 0.85 بوده است. علاوه‌براین، مدل پیشنهادی ما مزایایی را در چندین جنبه نسبت به سایر مدل‌ها دارد. نسبت به مدل‌های DFNN و درخت هوفدینگ آموزش‌دیده با مجموعه‌داده‌ای که فقط حاوی آخرین نفوذهای هدف است، مدل T-DFNN پیشنهادی ما امتیازهای F1 بالاتری دارد. علاوه‌براین، مدل T-DFNN پیشنهادی ما زمان‌های آموزش قابل‌ملاحظه کمتری نسبت به مدل DFNN آموزش‌دیده با مجموعه‌داده‌ای که تمام نفوذهای هدف را حاوی می‌باشد، دارد. هرچند چندین عامل می‌تواند بر مدت زمان فرآیند آموزش تأثیر بگذارد، اما الگوریتم T-DFNN نتایجی امیدوارکننده در حل مسئله های تشخیص نفوذ شبکه نشان داده است[۱].

پیشگفتار

همانطور که در چکیده اشاره کردیم. مشکل فراموشی آموخته های قبلی یکی از بزرگترین مشکلات نه تنها در زمینه امنیت بلکه در تمام زمینه هایی که داده ها به صورت مستمر به مدل داده میشوند وجود دارد. این مطالعه مقاله (M. Data and M. Aritsugi, "T-DFNN: An Incremental Learning Algorithm for Intrusion Detection Systems") را بررسی میکند که سعی کرده این مشکل را با ارایه معماری جدیدی به نام یادگیری افزایشی حل کند. ابتدا مفاهیم به کار رفته در مقاله را توضیح میدهیم. داده بکار رفته در مقاله را بررسی میکنیم و ساختار مدلی که پیاده سازی کرده ایم را شرح میدهیم. الگوریتم ارایه شده در مقاله را به طور کامل بررسی میکنیم و در آخر با کمک مدل پیاده شده الگوریتم را ارزیابی میکنیم تا تاییدی بر عملکرد الگوریتم ارایه شده باشد.

فهرست مطالب

فصل ۱

مفاهیم مقدماتی

۱.۱ تشخیص نفوذ

سیستم تشخیص نفوذ یک ابزار امنیتی است که برای شناسایی و هشدار در مورد فعالیت های مشکوک یا مخربی که در یک شبکه یا یک سیستم رخ می دهد طراحی شده است. این در کنار سایر اقدامات امنیت سایبری، مانند فایروال ها و نرم افزار آنتی ویروس، برای ایجاد یک دفاع قوی در برابر تهدیدات سایبری عمل می کند. برخلاف فایروال‌هایی که به عنوان یک مانع خارجی عمل می‌کنند، یک سیستم تشخیص نفوذ رویکردی را اتخاذ می‌کند که بر شبکه داخلی متمرکز است و به طور مداوم ترافیک و الگوهای رفتاری را برای نشانه‌های دسترسی غیرمجاز یا فعالیت‌های غیرعادی تجزیه و تحلیل می‌کند[۲].

- سیستم‌های تشخیص نفوذ مبتنی بر شبکه: این سیستم‌ها در نقاط استراتژیک شبکه مستقر می‌شوند و ترافیک ورودی و خروجی را برای شناسایی الگوهای مشکوک تجزیه و تحلیل می‌کنند. آنها به طور منفعلانه شبکه را رصد می کنند، بسته ها، سرصفحه ها و بارها را برای حمله شناخته شده یا رفتار غیرعادی بررسی می کنند. این سیستم‌ها برای شناسایی تهدیدهایی که ممکن است از دفاع‌های محیطی سنتی عبور کنند، کارا هستند[۳].
- سیستم‌های تشخیص نفوذ مبتنی بر میزبان: بر خلاف سیستم‌های تشخیص نفوذ مبتنی بر شبکه، این سیستم‌ها بر روی سیستم‌های میزبان مانند سرورها یا نقاط پایانی انتقال ترافیک عمل می‌کنند. این سیستم‌ها گزارش‌ها، فایل‌ها و فعالیت‌های سیستم را برای شناسایی رفتار غیرمعمول، تلاش‌های دسترسی غیرمجاز، یا نشانه‌های آلودگی بدافزار نظارت می‌کنند. این سیستم‌ها می‌توانند بینش عمیق‌تری نسبت به فعالیت‌هایی که روی سرویس خاص رخ می‌دهند، ارائه دهند[۳].

۲.۱ تاریخچه تشخیص حملات

۳.۱ حمله های متداول

حملات به سیستم‌های تشخیص نفوذ یک چالش بزرگ در حوزه امنیت سایبری است. این حملات با هدف تضعیف اثربخشی و قابلیت اطمینان سیستم‌های تشخیص نفوذ سازماندهی می‌شوند و مانع از توانایی آن‌ها برای شناسایی و جلوگیری از فعالیت‌های مخرب در یک شبکه یا سیستم می‌شوند. متجاوزان از تکنیک‌های پیچیده مختلفی استفاده می‌کنند که چند نمونه را در زیر میبینیم.

- حملات انکار سرویس (DoS) و انکار سرویس توزیع شده (DDoS): هدف حملات DoS و DDoS این است که منابع یک سیستم را تحت الشعاع قرار دهند و آن را برای کاربران قانونی و عادی غیرقابل دسترس کنند. در یک حمله DoS، یک منبع واحد مقدار زیادی ترافیک ایجاد می کند، در حالی که حملات DDoS شامل چندین منبع است که با هم کار می کنند. IDS این حملات را با نظارت بر جهش های ناگهانی در ترافیک ورودی، الگوهای غیرمعمول و کاهش قابل توجه عملکرد شبکه شناسایی می کند[۴].

- بدافزارها و ویروس ها: بدافزارها مانند ویروس ها، کرم ها و تروجان ها تهدیدی قابل توجه برای شبکه ها و سیستم ها هستند. IDS ها از تشخیص مبتنی بر امضا برای شناسایی بدافزارهای شناخته شده با مقایسه الگوها با پایگاه داده امضاها استفاده می کنند. تجزیه و تحلیل رفتاری همچنین برای شناسایی بدافزارهایی که قبلاً دیده نشده بودند بر اساس اقدامات مشکوک یا تغییرات در رفتار سیستم استفاده می شود.

- تزریق SQL: تزریق SQL یک حمله تحت وب است که در آن مهاجم کد SQL مخرب را به قسمت ورودی تزریق می کند تا پایگاه داده را دستکاری کند. IDS ها می توانند چنین حملاتی را با تجزیه و تحلیل الگوهای پرس و جو، شناسایی توالی های ورودی غیرمعمول و اعتبارسنجی ورودی های کاربر برای جلوگیری از دسترسی غیرمجاز به پایگاه داده شناسایی کنند.

- اسکریپت بین سایتی (XSS): حملات XSS شامل تزریق اسکریپت های مخرب به صفحات وب است که توسط سایر کاربران مشاهده می شود. IDS ها از تطبیق الگو و اکتشافی برای شناسایی چنین حملاتی با تجزیه و تحلیل محتوای HTML برای کدهای مشکوک یا برچسب های اسکریپت استفاده می کنند.

- سرریز بافر: حملات سرریز بافر از آسیب‌پذیری در برنامه‌ها یا سرویس‌ها با بارگذاری بیش از حد بافرها با داده‌های بیش از حد، سوء استفاده می‌کنند و به مهاجمان اجازه می‌دهند کدهای مخرب را اجرا کنند. IDS ها می توانند از تشخیص مبتنی بر امضا، تجزیه و تحلیل رفتار و تشخیص ناهنجاری برای شناسایی تلاش های سرریز بافر استفاده کنند[۵].

- حملات Brute Force: در حملات brute force، مهاجمان سعی می‌کنند با آزمایش سیستماتیک همه ترکیب‌های ممکن از نام‌های کاربری و رمزهای عبور، به یک سیستم دسترسی غیرمجاز به دست آورند. IDS ها می توانند چنین حملاتی را با تجزیه و تحلیل الگوهای ورود، شناسایی تلاش های مکرر ناموفق برای ورود به سیستم و اجرای سیاست های قفل حساب شناسایی کنند.
- حملات Man-in-the-Middle (MitM): حملات MitM شامل رهگیری و دستکاری ارتباطات بین دو طرف است که به مهاجمان اجازه می دهد داده ها را استراق سمع یا تغییر دهند. IDS ها می توانند این حملات را با نظارت بر دستگاه های غیرمجاز که خود را بین جریان های ارتباطی قرار می دهند یا با شناسایی الگوهای ارتباطی غیرعادی شناسایی کنند[۶].

۴.۱ فراموشی فاجعه‌بار(Catastrophic Forgetting)

فراموشی فاجعه آمیز که به عنوان تداخل فاجعه آمیز یا رونویسی فاجعه آمیز نیز شناخته می شود، پدیده ای است که در شبکه های عصبی مصنوعی و مدل های یادگیری ماشین از جمله مدل های یادگیری عمیق مشاهده می شود. این اثر به تمایل یک مدل به فراموشی کامل اطلاعات آموخته شده قبلی هنگام آموزش داده های جدید و نامرتبط اشاره دارد.

شبکه‌های عصبی و مدل‌های یادگیری عمیق با تنظیم وزن‌ها و پارامترهای خود برای به حداقل رساندن خطا بین پیش‌بینی‌هایشان و برچسب حقیقی یک مجموعه داده یاد می‌گیرند. هنگامی که یک مدل بر روی یک مجموعه داده خاص آموزش داده می شود، به تدریج وزن های خود را برای ثبت الگوها و روابط موجود در آن داده ها تطبیق می دهد و این امکان را بوجود میاورد که پیش بینی های دقیقی انجام دهد.

با این حال، مشکل زمانی به وجود می‌آید که مدل بر روی یک مجموعه داده جدید آموزش داده می‌شود و یا وظیفه یادگیری مجموعه جدیدی از الگوها را بر عهده میگیرد. از آنجایی که وزن های مدل برای یادگیری اطلاعات جدید به روز می شوند، ممکن است به طور قابل توجهی از مقادیر قبلی خود منحرف شوند و باعث شود مدل آنچه را که قبلاً آموخته است "فراموش کند". درواقع، اطلاعات تازه آموخته شده، دانش قدیمی را بازنویسی می کند و منجر به از دست دادن عملکرد در کار قبلی می شود.

فراموشی فاجعه آمیز یک چالش مهم در سناریوهای یادگیری مستمر است، که در آن انتظار می رود مدل ها از دنباله ای از وظایف بیاموزند بدون اینکه دانش به دست آمده از وظایف قبلی را فراموش کنند. این مانع از توانایی مدل برای حفظ طیف گسترده ای از دانش می شود و توانایی آن را برای سازگاری در محیط های پویا محدود می کند.

یادگیری افزایشی یک حوزه اساسی از تحقیقات در حال حاضر است، زیرا هدف آن ساخت سیستم‌های هوش مصنوعی است که می‌توانند در طول زمان یاد بگیرند و با آن سازگار شوند، درست

مانند آنچه انسان‌ها هنگام کسب مهارت‌های جدید بدون فراموش کردن آنچه قبلاً آموخته‌اند، انجام می‌دهند[۷].

۵.۱ یادگیری افزایشی

در علوم کامپیوتر، یادگیری افزایشی روشی از یادگیری ماشینی است که در آن داده های ورودی به طور مداوم برای گسترش دانش مدل موجود و برای آموزش بیشتر مدل استفاده می شود. این یک تکنیک پویا از یادگیری نظارت شده و یادگیری بدون نظارت است که می تواند زمانی که داده های آموزشی به تدریج در دسترس می شوند یا اندازه آن خارج از محدودیت های حافظه سیستم است، اعمال شوند. الگوریتم هایی که می توانند یادگیری افزایشی را تسهیل کنند به عنوان الگوریتم های یادگیری ماشین افزایشی شناخته می شوند[۸][۹].

باید توجه داشته باشیم که اصطلاح یادگیری افزایشی در چندین مفهوم مانند رشد تدریجی شبکه و هرس کردن، یادگیری آنلاین یا یادگیری مجدد کلاس های اشتباه نمونه ها استفاده میشود. اصطلاح یادگیری افزایشی در این مطالعه به یک الگوریتم یادگیری ماشینی اشاره دارد که دو شرط زیر را برطرف میکند

- می تواند اطلاعات جدیدی را بیاموزد، به عنوان مثال، حمله ای با نوع تازه
- می تواند دانش قبلی را حفظ کند یا به عبارت دیگر، نباید دچار فراموشی فاجعه بار که پیشتر گفتیم بشود[۱۰].

۶.۱ شبکه های عمیق

یادگیری عمیق زیر شاخه ای از یادگیری ماشین است که شامل آموزش شبکه های عصبی مصنوعی برای یادگیری و پیش بینی از داده ها می شود.شبکه عصبی از ساختار و عملکرد مغز انسان الهام گرفته شده است، جایی که نورون های به هم پیوسته برای پردازش اطلاعات و تصمیم گیری با هم کار می کنند.

در طول فرآیند آموزش، شبکه عصبی به طور مکرر از طریق feed-forward، محاسبه lost و back-propagation پارامترهای خود را تنظیم میکند. این روند برای چندین دوره ادامه می یابد تا زمانی که مدل به حالتی همگرا شود که ضرر به حداقل برسد و بتواند پیش بینی های دقیقی انجام دهد.

اصطلاح "عمیق" در یادگیری عمیق به عمق شبکه عصبی، یعنی تعداد لایه های پنهان اشاره دارد. مدل‌های یادگیری عمیق معمولاً لایه‌های پنهان زیادی دارند که به آنها امکان می‌دهد نمایش سلسله مراتبی داده‌ها را بیاموزند. این سلسله مراتب ویژگی ها به مدل کمک می کند تا به طور

خودکار ویژگی های مربوطه را از داده های ورودی خام استخراج کند[۱۱]. یادگیری عمیق در کارهای مختلف مانند تشخیص تصویر، پردازش زبان طبیعی، تشخیص گفتار و انجام بازی ها موفقیت چشمگیری از خود نشان داده است. در دسترس بودن مجموعه داده های بزرگ، منابع محاسباتی قدرتمند و پیشرفت در الگوریتم ها به پذیرش گسترده و موفقیت یادگیری عمیق در حل مسائل پیچیده دنیای واقعی کمک کرده است[۱۲].

۷.۱ درخت هافدینگ

درخت هافدینگ، همچنین به عنوان درخت تصمیم بسیار سریع (VFDT) شناخته می‌شود، نوعی از الگوریتم یادگیری ماشین به صورت تدریجی یا آنلاین است که برای وظایف یادگیری نظارت‌شده مانند دسته‌بندی استفاده می‌شود. این الگوریتم به‌طور خاص برای کاربردهایی که داده‌ها به‌صورت پیوسته تولید مشوند و یا پردازش آن‌ها به صورت دسته‌ای غیرعملی یا ناکارآمد است، مناسب است؛ زیرا این امکان را فراهم می‌کند که پیش‌بینی‌های سریع تر انجام شود.

الگوریتم درخت هافدینگ به‌طور تدریجی درخت‌های تصمیم را ساخته و نیازی به حضور تمام داده‌ها به‌صورت یکجا ندارد. این رویکرد یادگیری تدریجی به درخت هافدینگ اجازه می‌دهد که از نمونه‌های جدید بعد از ورود آن‌ها، تطبیق‌پذیر و یاد بگیرد.

ویژگی‌های کلیدی درخت‌های هافدینگ عبارتند از:

- معیار تقسیم گره: درخت‌های هافدینگ از مرز هافدینگ برای تصمیم‌گیری در مورد زمان تقسیم گره استفاده می‌کنند. مرز هافدینگ یک محدودیت اطمینان آماری است که به ما می‌گوید چند نمونه برای برآورد توزیع احتمال ویژگی به صورت دقیق نیاز است. این ویژگی مطمئن می‌شود که درخت تصمیم تصمیمات خود را بر اساس تعداد آماری کافی از نمونه‌ها میگیرد.
- یادگیری پیوسته: الگوریتم درخت هافدینگ می‌تواند به صورت پیوسته درخت تصمیم را با نمونه‌های جدید به‌روز کند.
- کارآیی حافظه: این الگوریتم تنها نیاز به ذخیره آماره‌ها برای هر گره دارد، نه کل داده‌های آموزشی، که باعث می‌شود که کارآیی حافظه بهینه شود و برای کنترل جریان داده‌های بزرگ مناسب باشد.
- تطبیق گره برگ: هنگامی که نمونه‌ی جدید وارد می‌شود، ممکن است منجر به تقسیم جدید در درخت شود و باعث رشد درخت شود. با این حال، اگر نمونه‌ی جدید نیاز به تقسیم جدید نداشته باشد، می‌توان از آن برای تنظیم آماره‌های گره برگ مربوطه استفاده کرد.

- بازبرش: برای جلوگیری از بیش‌برازش و کنترل رشد درخت، درخت‌های هافدینگ ممکن است گاهی اوقات بازبرش انجام دهند تا شاخه‌های غیرضروری را حذف کنند.

گام‌های اساسی ساخت یک درخت هافدینگ به‌صورت زیر است:

- آماده‌سازی: یک گره ریشه برای نمایش کل مجموعه داده‌ها ایجاد کنید.
- مشاهده: نمونه‌ها را یکی‌یکی مشاهده کنید.
- تقسیم: اگر نمونه‌ی جدید وارد می‌شود، درخت بررسی می‌کند

  که آیا نیاز به تقسیم گره دارد یا خیر، بر اساس مرز هافدینگ. اگر نیاز به تقسیم باشد، بهترین ویژگی برای تقسیم انتخاب می‌شود.

- رشد: گره‌های جدید بر اساس نمونه‌های مشاهده شده و ویژگی‌های مربوطه به درخت اضافه می‌شوند.
- بازبرش: گاهی اوقات برش درخت انجام می‌شود تا از بیش‌برازش جلوگیری شود و جمعیت درخت حفظ شود.

درخت‌های هافدینگ در بسیاری از سناریوهای یادگیری آنلاین مؤثر بوده‌اند، جایی که داده‌ها به صورت جریان‌های پیوسته وارد می‌شوند و نیاز به پردازش بهینه آن‌ها است. آن‌ها به ویژه در کاربردهایی مانند شناسایی نفوذ در شبکه، سیستم‌های تطبیقی و تحلیل داده‌های به صورت زمان‌واقع مناسب هستند.

با این حال، مهم است به‌یاد داشت که درخت‌های هافدینگ ممکن است هنگامی که کل مجموعه داده به‌صورت یکجا در دسترس است، به‌خوبی عمل نکنند، به ویژه زمانی که داده‌ها ثابت هستند (یعنی بدون نوسان مفهومی). در چنین مواردی، الگوریتم‌های یادگیری مناسب‌تری (مانند درخت‌های تصمیم سنتی مانند CART) ممکن است وجود داشته باشند[۱۳].

فصل ۲

دادگان

۱.۲ مرجع

مجموعه داده CICIDS (مؤسسه کانادایی برای سیستم تشخیص نفوذ امنیت سایبری) یک مجموعه داده در دسترس عموم است که برای تحقیق و ارزیابی سیستم‌های تشخیص نفوذ (IDS) و سیستم‌های پیشگیری از نفوذ (IPS) استفاده می‌شود. این مجموعه داده توسط موسسه کانادایی امنیت سایبری در دانشگاه نیوبرانزویک ایجاد شده است.

مجموعه داده CICIDS شامل داده های ترافیک شبکه است که در یک محیط کنترل شده که در آن انواع مختلفی از حملات سایبری علیه آن انجام شده است، گرفته شده است. این داده ها طیف وسیعی از سناریوهای حمله را پوشش می دهد و شامل ترافیک خوش خیم (عادی) و ترافیک مخرب است. این مجموعه داده اطلاعات برچسب‌گذاری شده‌ای در مورد حملات ارائه می‌کند و آن را برای آموزش و آزمایش مدل‌های تشخیص نفوذ و پیشگیری مبتنی بر یادگیری ماشین ارزشمند می‌کند.

برخی از انواع رایج حملات موجود در مجموعه داده CICIDS است Distributed Denial of Service (DDoS) injection SQL Brute Force و سایر اشکال حملات مبتنی بر شبکه میباشد.

۲.۲ ویژگی ها

ما یازده معیار داریم که برای ایجاد یک مجموعه داده معیار قابل اعتماد ضروری هستند. در ادامه، این معیارها را به اختصار بیان می کنیم:

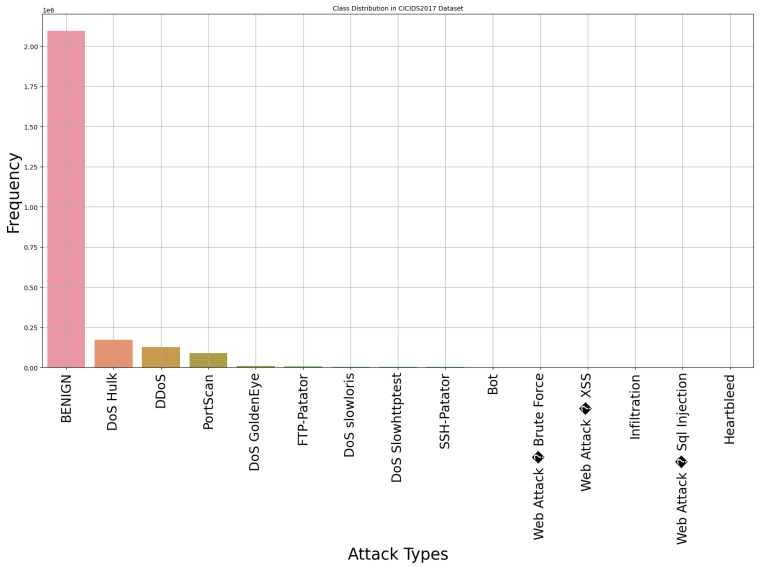
- پیکربندی شبکه: یک توپولوژی شبکه کامل شامل مودم، فایروال، سوئیچ ها، روترها و وجود انواع سیستم عامل ها مانند ویندوز، اوبونتو و Mac OS X می باشد.

- ترافیک: کنترل ترافیک در ۱۲ ماشین مختلف در شبکه طعمه و حملات واقعی از شبکه حمله کننده.
- مجموعه داده با لیبل: برچسب های سالم و مخرب را برای هر روز در دسترس داریم. همچنین جزئیات زمان حمله در داده گزارش شده است.
- تعامل: با داشتن دو شبکه مختلف هم ترافیک خارجی (ارتباطات اینترنتی)، و هم داخلی (LAN) را پوشش می دهیم.
- Capture: از آنجایی که از پورتmirror مانند سیستم tapping استفاده شده، تمام ترافیک ها در سرور ذخیره سازی ضبط و ثبت شده است.
- پروتکل‌های موجود: تمام پروتکل‌های رایج موجود مانند پروتکل‌های HTTP HTTPS FTP SSH و ایمیل را فراهم شده است.
- تنوع حمله: شامل رایج ترین حملات بر اساس گزارش McAfee در سال ۲۰۱۶، مبتنی بر وب، Brute, DoS, DDoS, Infiltration, Heart-bleed Bot, Scan که در این مجموعه داده پوشش داده شده است.
- ناهمگونی: در طول اجرای حملات، ترافیک شبکه از سوییچ اصلی و حافظه خالی و تماس های سیستمی از همه ماشین های قربانی گرفته شده است.
- مجموعه ویژگی: بیش از ۸۰ ویژگی جریان شبکه را از ترافیک شبکه تولید شده با استفاده از CICFlowMeter استخراج کرده و مجموعه داده جریان شبکه به عنوان یک فایل CSVتحویل داده شده است.
- MetaData: مجموعه داده ای را که شامل زمان، حملات، جریان ها و برچسب ها را به طور کامل توضیح داده است.

[۱۴]

۳.۲ تحلیل

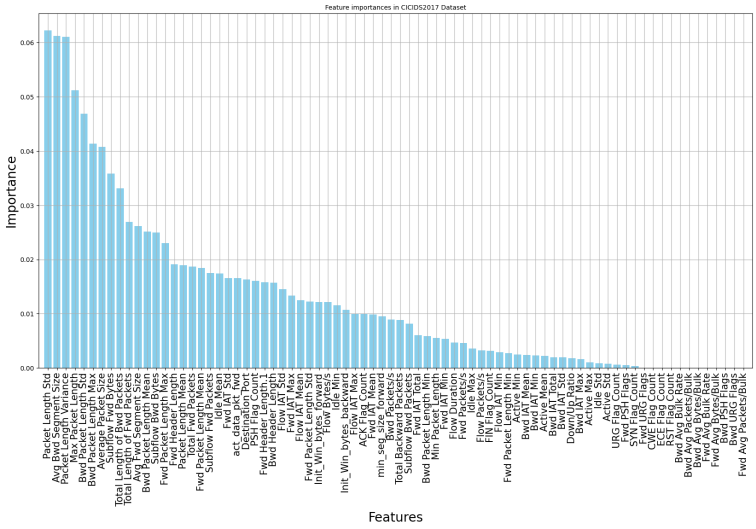
در ابتدا فراوانی حمله ها را بررسی میکنیم تا دیدی بهتر از داده ها داشته باشیم[شکل ۱.۲][۱۵].



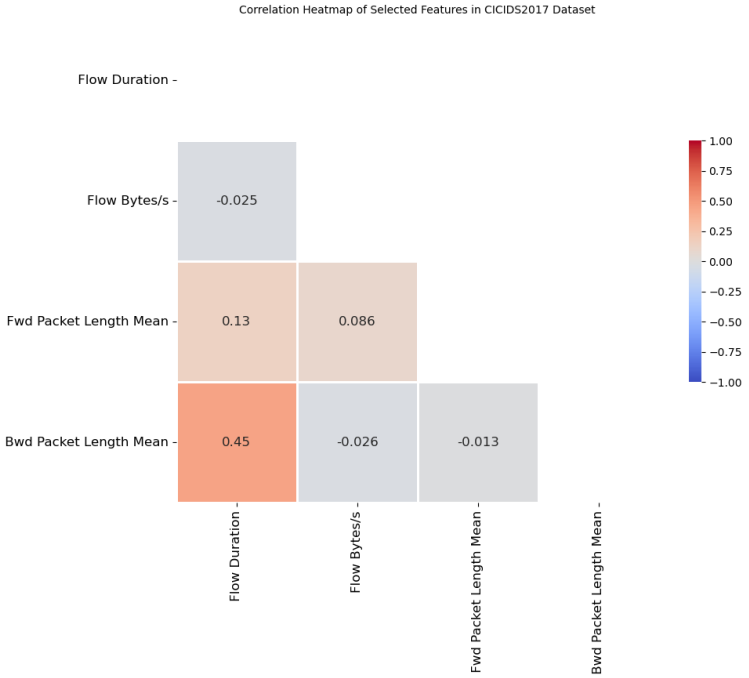
شکل ۱.۲: در این شکل فراوانی هر نوع ترافیک موجود در داده را نشان داده ایم[۱]

با استفاده از RandomForestClassifier دادگان را آموزش دادیم تا اهمیت هر ویژگی را ببینیم.[شکل ۲.۲]

ارتباط چند ویژگی تاثیرگذارتر را بررسی میکنیم[شکل ۳.۲].



شکل ۲.۲: تاثیر هر ویژگی بر روی طبقه بندی بر اساس الگوریتم درخت تصمیم گیری[۱۵]



شکل ۳.۲: همبستگی ویژگی ها با بیشترین تاثیر در طبقه بندی با همدیگر[۱۵]

۴.۲ پیش پردازش

در ابتدا تمام فایل هارا میخوانیم و همه را به یک دیتافریم تبدیل میکنیم. ویژگی هایی که در آموزش استفاده میشوند را از ستون لیبل جدا میکنیم و در دو دیتافریم قرار میدهیم. نام ستون هارا تصحیح میکنیم و لیبل هارا از رشته به عدد تغییر میدهیم.

دادگان آموزش را با استفاده از روش Min Max عادی سازی میکنیم تا مقیاس تمام ویژگی ها یکی شود.

$$MinMax(X) = \frac{X - X_{min}}{X_{max}X_{min}}$$

داده ها را با استفاده از متد train test split با نسبت ۲ به ۸ دادگان آموزش و تست تقسیم میکنیم.

برای مقادیر خالی میانگین هر ستون و برای مقادیر بینهایت دوبرابر بیشینه آن ستون را لحاظ میکنیم.

تصمیم گرفیم ۶ ویژگی زیر را از داده حذف کنیم. جریان شناسه (Flow ID)، پروتکل (Protocol)، زمان نما (Timestamp)، آدرس آی‌پی مبدا (Source IP)، آدرس آی‌پی مقصد (Destination IP) و پورت مبدا (Source Port). از دید توپولوژی شبکه، مقادیر این ویژگی‌ها با سناریوهای واقعی متفاوت خواهد بود زیرا این مجموعه داده در یک شبکه ایزوله تولید شده اند. علاوه بر این، ما ۲۸۸۶۰۲ ردیف با برچسب‌های گم شده را از مجموعه داده حذف کردیم. پس از حذف داده‌های بی‌استفاده و بدون برچسب، مجموعه داده نهایی شامل ۲۸۳۰۷۴۳ ردیف و ۷۸ ویژگی بود. در مجموعه داده CICI2017، پانزده کلاس وجود دارد، یکی از این کلاس‌ها ترافیک معمولی (benign traffic) و بقیه چهارده نوع مختلف از ترافیک نفوذ (intrusion traffic) هستند. جدول زیر توزیع ترافیک در این مجموعه داده را نشان می‌دهد.

Batch	Class label	Encoded class label	Sample		
			Training	Evaluation	Total
1	Benign	0	1,818,477	454,620	2,273,097
	FTP-patator	1	6,350	1,588	7,938
	Bot	2	1,573	393	1,966
	Web attack-XSS	3	522	130	652
2	DoS-hulk	4	184,858	46,215	231,073
	Port scan	5	127,144	31,786	158,930
	DDoS	6	102,421	25,606	128,027
	Web attack-brute force	7	1,206	301	1,507
3	DoS-golden eye	8	8,234	2,059	10,293
	SSH-patator	9	4,718	1,179	5,897
	DoS-slowloris	10	4,637	1,159	5,796
	DoS-slowhttptest	11	4,399	1,100	5,499
4	Infiltration	12	29	7	36
	Web attack-SQL injection	13	17	4	21
	Heartbleed	14	9	2	11
	Total		2,264,594	566,149	2,830,743

شکل ۴.۲: تعداد نمونه های هر کلاس در دسته بندی های ایجاد شده[۱]

فصل ۳

ساختار شبکه عصبی

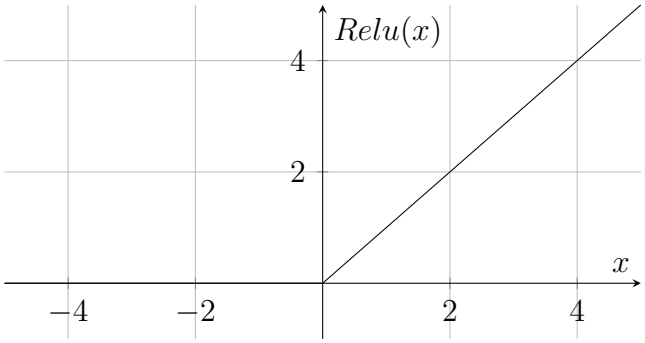
۱.۳ معماری

T-DFNN از ساختار درخت مانند پیروی میکند که متشکل از یک گره ریشه و چندین گره برگ و داخلی است هنگامی که یک مدل T-DFNN دوباره آموزش می بیند، گره ها برای حفظ اطلاعات گذشته بدون تغییر باقی میمانند و در عوض گره های جدید ایجاد می شوند. این گره های تازه ایجاد شده سپس با گره های موجود متصل می شوند تا یک درخت ایجاد کنند. هر گره جدید تنها روی داده هایی آموزش میبیند که گره والد آن ارسال کرده. در این مورد در بخش بعد بیشتر بحث خواهیم کرد.

هر گره یک شبکه عصبی با سه لایه است. لایه اول ۱۲۰ و لایه دوم ۷۸ نورون دارد. تعداد نورون لایه اخر با توجه به گره ای که شبکه در آن قرار دارد متفاوت است. لایه اول از فعال ساز relu و لایه دوم از softmax استفاده میکند.

$$Relu(x) = max(0, x)$$

[۱۶]



$$\sigma(y_i) = \left(\frac{e^{y_i}}{\sum_j e^{y_j}} \right) j = 1, ..., n$$

[۱۷]

x_3	x_2	x_1
$\frac{e^{x_3}}{e^{x_1}+e^{x_2}+e^{x_3}}$	$\frac{e^{x_2}}{e^{x_1}+e^{x_2}+e^{x_3}}$	$\frac{e^{x_1}}{e^{x_1}+e^{x_2}+e^{x_3}}$

۲.۳ بیهینه کننده

مدل از بهینه کننده ADAM استفاده میکند. Adam (مخفف Adaptive Moment Estimation) یک الگوریتم بهینه سازی است که به طور گسترده برای آموزش شبکه های عصبی عمیق استفاده می شود. این روش برپایه گرادیان نزولی است و مزایای SGD و RMSprop را با هم دارا میباشد. تکنیک اصلی در این بهینه کننده حفظ میانگین حرکات پیشین خود است. در زیر نحوه محاسبه آن را میبینیم[۱۸].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (۱.۳)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (۲.۳)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (۳.۳)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (۴.۳)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (۵.۳)$$

[۱۹]

فصل ۴

الگوریتم

۱.۴ تاریخچه یادگیری افزایشی

در مطالعات اخیر، استفاده از یادگیری عمیق با استفاده از شبکه‌های عصبی مصنوعی (ANNs) برای پردازش داده‌های بزرگ مورد ترجیح قرار گرفته شده است. یادگیری عمیق با استفاده از ANNs یکی از الگوریتم‌های محبوب برای یادگیری اطلاعات از مجموعه داده های پیچیده است. یادگیری عمیق با استفاده از ANNs می‌تواند مدل‌های پیچیده‌تری نسبت به تکنیک‌های سنتی یادگیری احتمالاتی ایجاد کند. بنابراین، این روش‌ها به طور گسترده‌ای برای سیستم‌های شناسایی نفوذ (IDSs) استفاده می‌شوند[۱].

با وجود اینکه این روش‌ها به طور گسترده‌ای استفاده شده اند اما اکثر مطالعات یادگیری عمیق در IDSها به یادگیری افزایشی تمرکز نمی‌کنند. به جای آن، تمرکز آن‌ها بر بهبود عملکرد طبقه‌بندی با استفاده از الگوریتم‌های یادگیری عمیق است. در مقابل، یادگیری افزایشی با استفاده از الگوریتم‌های یادگیری عمیق در حوزه پردازش تصویر در حال رشد بوده. به عنوان مثال، از شبکه‌های عصبی پیچشی عمیق (CNNs) برای ساختن یک مدل سلسله‌مراتبی برای یادگیری افزایشی استفاده کرده اند. مدل پیشنهادی آن‌ها تصاویر به‌دست‌آمده به‌صورت افزایشی را براساس ویژگی‌ها به چندین بالانس فرعی تقسیم می‌کند. در فرآیند آموزش، کلاس‌های جدید تصاویر به مدل سلسله‌مراتبی به‌عنوان زیرکلاس‌ها اضافه می‌شوند. فرآیند بازآموزش در بالانس‌های متأثر شده از سوپرکلاس‌ها محدود شده تا هزینه محاسباتی کاهش یابد. یا در مطالعه ای دیگر الگوریتم یادگیری افزایشی با استفاده از CNNها پیشنهاد شده. اما برخلاف متد قبلی، این متد در روش یادگیری افزایشی خود از روش مشارکت شبکه جزئی (partial network sharing) استفاده کردند.با الهام‌گرفتن از تکنیک‌های یادگیری انتقالی (transfer learning)، این روش لایه‌های CNN را به لایه‌های مشترک و طبقه‌بندی تقسیم می‌کند. لایه‌های ابتدایی تبدیل به لایه‌های مشترک می‌شوند و بقیه به لایه‌های طبقه‌بندی تبدیل میشوند. هنگامی که مدل با استفاده از داده‌های تصویر جدید

بازآموزش داده می‌شود، لایه‌های طبقه‌بندی برای طبقه‌بندی تصاویر جدید کلون می‌شوند. نتیجه این فرآیند کلون‌سازی، مدلی با ساختار درختی از لایه‌های مشترک و طبقه‌بندی است. هر دو روش از رویکردهای مختلفی برای ایجاد یک مدل یادگیری افزایشی استفاده می‌کنند. با این حال، ساختار هر دو مدل به شکلی شبیه به یک ساختار درختی است. علاوه بر این، هر دو روش فرآیند بازآموزش را در شاخه‌های جدید ساختار درخت محدود می‌کنند تا هزینه محاسباتی کاهش یابد. در این مطالعه ایده استفاده از یک مدل ساختار درختی برای یادگیری افزایشی انتخاب شده است[۲۰].

ما از مدل‌های DFNN را در الگوریتم T-DFNN استفاده کردیم. مدل DFNN یک نوعی از ANNs است که برای یادگیری عمیق استفاده می‌شود. ما یک ساختار درختی و مدل‌های DFNNرا ترکیب کردیم تا یک الگوریتم یادگیری افزایشی ایجاد کنیم که بتواند دانش آموزش‌های قبلی را حفظ کند در حالی که هزینه بازآموزش را کاهش دهد. چندین گره از مدل‌های DFNN در مدل T-DFNN برای طبقه‌بندی داده‌های ورودی داده شده است. برخلاف رویکردی که توضیح دادیم ، ما کلاس‌های حمله مشابه را به یک سوپرکلاس ترکیب نکردیم. در عوض، ما از یک مدل قبلی آموزش‌داده‌شده استفاده کردیم تا کلاس‌های قدیمی و جدید را به عنوان یک برچسب خروجی مشترک طبقه‌بندی کند[۱].

در آزمایش، ما روش پیشنهادی خود را با یک الگوریتم یادگیری افزایشی دیگر با ساختار درختی مقایسه کرده ایم. یک الگوریتم شناخته‌شده تصمیم‌گیری افزایشی، به نام الگوریتم درخت هوفدینگ. الگوریتم درخت هوفدینگ می‌تواند از داده‌های افزایشی با حجم بالا آموزش ببیند. این الگوریتم از این ویژگی بهره می‌برد که بخش کوچکی از داده معمولاً کافی است تا ویژگی تقسیم‌بندی بهینه مجموعه‌داده انتخاب شود. بنابراین، این الگوریتم به طور معمول برای پردازش داده‌های افزایشی استفاده می‌شود. الگوریتم درخت هوفدینگ در چند کتابخانه محبوب یادگیری ماشینی مانند scikit-learn و Weka پیاده‌سازی شده است[۱].

۲.۴ شبکه عصبی عمیق رو به جلو با ساختار درخت

الگوریتم T-DFNN هم فرآیندهای آموزش و هم طبقه‌بندی را پوشش می‌دهد. فرآیند آموزش الگوریتم T-DFNN یک مدل یادگیری فزاینده تولید می‌کند. این مدل یادگیری فزاینده سپس در فرآیند طبقه‌بندی برای طبقه‌بندی داده‌های ورودی استفاده می‌شود. هدف الگوریتم T-DFNN، طبقه‌بندی نفوذهای شبکه به صورت کارآمد است. بنابراین، فرآیند آموزش در الگوریتم T-DFNN به گونه‌ای طراحی شده است که دانش یادگرفته‌شده توسط مدل قبلی را حفظ می‌کند در حالی که تعداد داده‌های آموزشی قدیمی استفاده‌شده در فرآیند آموزش کاهش یابد.

مدل یک گره ریشه و چندین گره برگ و داخلی دارد. وقتی یک مدل T-DFNN مجدداً آموزش داده می‌شود، گره‌های آموزش دیده شده تغییر نمی‌کنند تا دانش یادگرفته‌شده قبلی حفظ شود. به جای آن، گره‌های جدید ایجاد می‌شوند. این گره‌های تازه ایجادشده سپس با گره‌های موجود نگاشت می‌شوند تا یک مدل با ساختار درخت ایجاد کنند. نوآوری الگوریتم T-DFNN پیشنهادی این مقاله، توانایی توزیع داده‌های آموزشی برای هر گره و همچنین محدود کردن تعداد داده‌های آموزشی قدیمی مورد استفاده برای آموزش این گره‌های جدید است، که این امر زمان لازم برای آموزش مدل را کوتاه‌تر می‌کند.

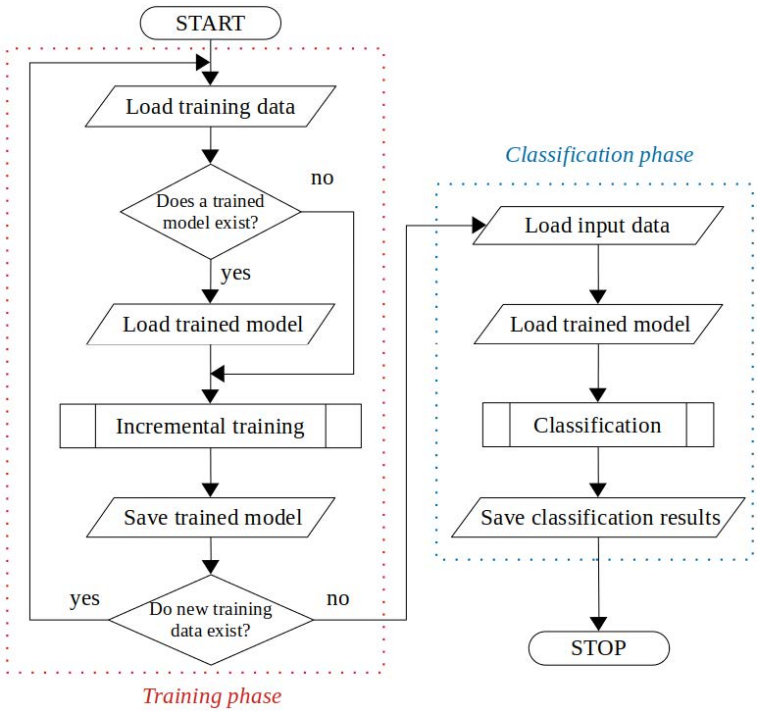
شکل ۱.۴ جریان آموزش و طبقه‌بندی الگوریتم T-DFNN را نشان می‌دهد. ویژگی اصلی الگوریتم T-DFNN در فرآیند آموزش، مکانیزم استفاده مجدد از مدل آموزش‌دیده قبلی برای یادگیری داده‌های آموزشی جدید است. بنابراین، مدل آموزش‌دیده پس از هر فرآیند آموزش فزاینده ذخیره می‌شود. به جز برای اولین فرآیند آموزش، مدل ذخیره‌شده همراه با داده‌های آموزشی جدید بارگذاری می‌شود. سپس این مدل ذخیره‌شده دوباره آموزش داده می‌شود تا داده‌های ورودی جدید را طبقه‌بندی کند.

یکی از اجزای اساسی در مدل T-DFNN گره T-DFNN است. دو مورد مهم در گره T-DFNN وجود دارد: یک مدل DFNN و یک نگاشت از برچسب‌های خروجی. یک مدل DFNN داده‌های ورودی را پردازش می‌کند و آن‌ها را به چندین برچسب خروجی طبقه‌بندی می‌کند. این برچسب‌های خروجی می‌توانند با گره‌های دیگر T-DFNN با استفاده از یک نگاشت مرتبط شوند. نگاشت یک ساختار داده‌ای است که از جفت‌های کلید_مقدار تشکیل شده است. در نگاشت برچسب خروجی، برچسب‌های خروجی به عنوان کلیدها ظاهر می‌شوند و مقادیر این کلیدها یا گره‌های دیگر یا مقادیر NULL هستند. مقدار NULL نشان‌دهنده این است که برچسب خروجی با هیچ گره ارتباط ندارد. الگوریتم ۱ پیاده‌سازی یک گره T-DFNN را نشان می دهد.

همانطور که پیش‌تر اشاره کردیم، نوآوری الگوریتم T-DFNN مکانیزم توزیع داده‌های آموزشی به چندین گره و محدود کردن تعداد داده‌های آموزشی قدیمی مورد استفاده در فرآیند آموزش است. در فرآیند تجدید آموزش، چندین گره جدید ممکن است ایجاد شود. علاوه بر داده‌های آموزشی جدید، داده‌های آموزشی قدیمی نیز برای آموزش این گره‌های جدید استفاده می‌شوند. با این حال،

Algorithm 1 T-DFNN NODE

1: **procedure** INSTANTIATION
2: $M :\leftarrow createaDFNNmodel$
3: $\mu \leftarrow emtpymap$
4: **end procedure**
5: **procedure** TRAIN(X,Y)
6: $DFNNTraining(M, X, Y)$
7: **for** $L \in output\ labels\ of\ M$ **do**
8: $\mu(L) \leftarrow NULL$
9: **end for**
10: **end procedure**
11: **procedure** CLASSIFY(X)
12: $YC \leftarrow DFNNClassification(M, X)$
13: **return** YC
14: **end procedure**
15: **procedure** GETLINKEDNODE(L)
16: $N \leftarrow \mu(L)$
17: **return** N
18: **end procedure**
19: **procedure** SETLINKEDNODE(L,N)
20: $\mu(L) \leftarrow N$
21: **end procedure**



شکل ۱.۴ : فلوچارت آموزش و طبقه بندی الگوریتم[۱]

الگوریتم T-DFNN تعداد داده‌های آموزشی قدیمی مورد استفاده برای آموزش این گره‌های جدید را محدود می‌کند. هر گره جدید فقط از داده‌های آموزشی قدیمی که به عنوان برچسب خروجی گره والد خود طبقه‌بندی شده‌اند، استفاده می‌کند. جزئیات این فرآیند آموزش در بخش بعدی توصیف شده است.

عملیات طبقه‌بندی در الگوریتم T-DFNN در چندین مرحله انجام می‌شود. ابتدا داده‌های ورودی با استفاده از گره اصلی مدل T-DFNN پردازش می‌شوند. سپس خروجی‌های عملیات طبقه‌بندی گره اصلی برای تعیین ادامه عملیات طبقه‌بندی با استفاده از گره‌های فرزند یا پایان یافتن فرآیند طبقه‌بندی مورد استفاده قرار می‌گیرند. این فرآیندها شباهت‌هایی با فرآیند طبقه‌بندی در درخت تصمیم دارند. تفاوت آن این است که در الگوریتم T-DFNN، خروجی‌های مدل DFNN به عنوان قوانین تقسیم استفاده می‌شوند. ما جزئیات فرآیند طبقه‌بندی را در بخش های بعد مورد بحث قرار می‌دهیم.

۳.۴ فرایند آموزش

با استفاده از رویهٔ یادگیری فزاینده که در الگوریتم ۲ نشان داده شده است، این فرایند آموزش اولیه به صورت زیر انجام می‌شود: در فرآیند آموزش اولیه، یک گره ایجاد می‌شود. به عبارت دیگر، این گره ریشه مدل T-DFNNاست. سپس این گره به وسیله داده‌های آموزشی داده شده، آموزش داده می‌شود. در نهایت، برچسب‌های خروجی مدل DFNN را با NULL مطابقت می‌دهیم تا نشان دهیم که برچسب‌های خروجی با هیچ گره‌ای مرتبط نیستند. مراحل آموزش اولیه را در الگوریتم ۲، خطوط ۱ تا ۴ توصیف می‌کنیم.

Algorithm 2 T-DFNN Incremental Training

```
1: procedure IncrementalTraining(X,Y,N)
2:     if N = NULL then
3:         N ← Create a T − DFNN − NODE instance
4:         Call Train(X, Y)
5:     else
6:         YC ← Call Classify(X)
7:         for L ∈ unique(YC) do
8:             XL, YL ← SelectTrainingData(X, Y, YC, L)
9:             if any values in YL ∉ L then
10:                linkedN ← CallGetLinkedNote(L)
11:                if linkedN ≠ NULL then
12:                    linkedN ← IncrementalTraining(XL, YL, linkedN)
13:                else
14:                    newN ← IncrementalTraining(XL, YL, NULL)
15:                    Call SetLinkedNode(L, newN)
16:                end if
17:            end if
18:        end for
19:    end if
20: end procedure
```

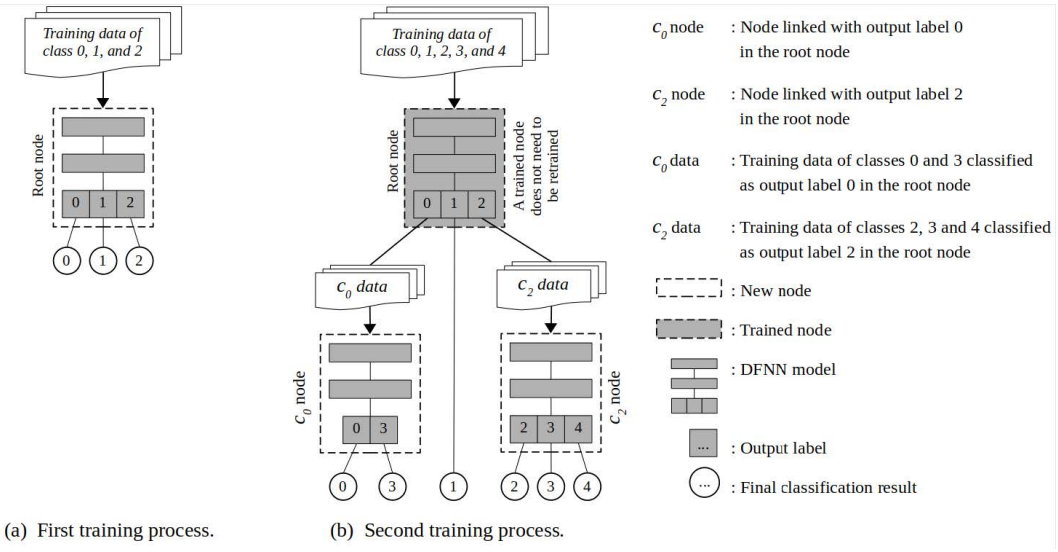
فرآیند بازآموزی با بارگذاری داده‌های آموزشی و مدل T-DFNN آموزش‌دیده شده شروع می‌شود. ما برای جلوگیری از اثر فراموشی گره‌های موجود را دوباره آموزش نمی‌دهیم. به جای آن، هر خروجی گره های موجود را به یک گره جدید نگاشت میکنیم. این گره‌های جدید بر روی داده های جدید آموزش داده میشوند تا قدرت مدل را بر روی الگوهای جدید تعمیم دهند. طریقه نگاشت در الگوریتم ۲ خطوط ۴ تا ۲۰ نشان داده شده است. این فرآیند با طبقه‌بندی داده‌های

آموزشی جدید با استفاده از مدل DFNN گره ریشه، همانطور که در الگوریتم ۲، خط ۵ نشان داده شده است، آغاز می‌شود. مدل DFNN گره ریشه داده‌های آموزشی جدید را به عنوان کلاس‌های قدیمی اشتباه طبقه‌بندی می‌کند زیرا ما مدل DFNN گره ریشه را برای طبقه‌بندی این داده‌های آموزشی جدید آموزش نداده‌ایم. این فرایند در شکل ۲.۴ نشان داده شده است. در شکل ۲.۴، مدل DFNN گره ریشه داده‌های آموزشی کلاس ۴ را به عنوان برچسب خروجی ۲ اشتباه طبقه‌بندی می‌کند. همچنین ممکن است مدل DFNN گره ریشه یک کلاس آموزشی جدید را به عنوان دو یا چند کلاس قدیمی اشتباه طبقه‌بندی کند. به عنوان مثال، در شکل۲.۴، مدل DFNN گره ریشه بخشی از داده‌های آموزشی کلاس ۳ را به عنوان برچسب خروجی ۰ و بخش دیگر را به عنوان برچسب خروجی ۲ اشتباه طبقه‌بندی می‌کند. هنگامی که چندین کلاس به عنوان یک برچسب خروجی یکسان طبقه‌بندی می‌شوند، دو شرط ممکن وجود دارد:

- برچسب خروجی به یک گره نگاشت می‌شود.
- برچسب خروجی به هیچ گره‌ای نگاشت نمی‌شود.

در شرط اول، زمانی که یک برچسب خروجی به یک گره نگاشت شد، ما فرآیند آموزش را با استفاده از گره جدید به عنوان ریشه جدید اجرا می‌کنیم. این فرآیند آموزش بازگشتی تا زمانی ادامه پیدا می‌کند که برچسب های خروجی بدون گره نگاشت شده باقی بمانند. به عبارت دیگر، شرط دوم اتفاق می‌افتد. در این شرط دوم، ما یک گره جدید ایجاد و آموزش می‌دهیم. تنها داده‌های آموزشی طبقه‌بندی شده به عنوان برچسب خروجی گره والدین در فرآیند آموزش استفاده می‌شوند. در نهایت، ما برچسب خروجی گره والدین را با این گره جدید آموزش داده شده مرتبط می‌کنیم با نگاشت برچسب خروجی گره والدین به گره جدید. این مراحل را در الگوریتم ۲، خطوط ۹ تا ۱۵ توصیف می‌کنیم.

یکی از اجزای کلیدی الگوریتم آموزش تدریجی T-DFNN انتخاب داده‌های آموزشی مناسب است که در هر گره استفاده می‌شود. این فرآیند انتخاب تعداد داده‌های آموزشی در هر گره را محدود می‌کند. ما فرآیند انتخاب داده‌های آموزشی را در الگوریتم ۳ توصیف می‌کنیم. داده‌های آموزشی در هر گره قبلاً به عنوان برچسب خروجی گره والدین طبقه‌بندی شده‌اند. به عنوان مثال، در شکل۲.۴ داده‌های آموزشی استفاده شده در گره c شامل کلاس ۰ و بخشی از داده‌های آموزشی کلاس ۳ است زیرا به عنوان برچسب خروجی ۰ در گره ریشه طبقه‌بندی شده‌اند. به همین ترتیب، داده‌های آموزشی استفاده شده در گره c2 شامل کلاس ۲، بخشی از کلاس ۳ و داده‌های آموزشی کلاس ۴ هستند زیرا به عنوان برچسب خروجی ۲ در گره ریشه طبقه‌بندی شده‌اند. داده‌های آموزشی یک کلاس ممکن است به چندین بخش تقسیم شود و با استفاده از گره‌های مختلف آموزش داده شوند. به عنوان مثال، داده‌های آموزشی کلاس ۳ استفاده شده در گره c0 قبلاً به عنوان برچسب خروجی ۰ در گره ریشه طبقه‌بندی شده‌اند. در مقابل، داده‌های آموزشی کلاس ۳ استفاده شده در گره c2 قبلاً به عنوان



شکل ۲.۴: فرایند آموزش اولیه و بازآموزش های بعدی[۱]

برچسب خروجی ۲ در گره ریشه طبقه‌بندی شده‌اند. بنابراین، هیچ اشتراکی در داده‌های آموزشی استفاده شده در این دو گره وجود ندارد. ما نیاز نداریم که از تمام داده‌های آموزشی کلاس‌های قدیمی در فرآیند آموزش داده‌های آموزشی جدید استفاده کنیم. داده‌های آموزشی برخی از کلاس‌های قدیمی فقط زمانی مورد نیاز هستند که کلاس‌های قدیمی و جدید به عنوان همان برچسب خروجی طبقه‌بندی می‌شوند. به عنوان مثال، در شکل ۴.۲ ما نیازی به استفاده از داده‌های آموزشی کلاس ۱ نداریم زیرا هنگامی که ما داده‌های آموزشی جدید را با استفاده از گره ریشه طبقه‌بندی می‌کنیم، هیچ کلاس جدیدی به عنوان برچسب خروجی ۱ طبقه‌بندی نمی‌شود. این روش تعداد داده‌های آموزشی قدیمی مورد استفاده برای فرآیند آموزش کلاس‌های جدید را کاهش می‌دهد.

۴.۴ فرایند طبقه بندی

الگوریتم طبقه‌بندی T-DFNN یک فرآیند بازگشتی است. همانطور که در الگوریتم ۴ نشان داده شده است، گام اول طبقه‌بندی داده‌های ورودی با استفاده از مدل DFNN گره ریشه است. سپس، برای هر برچسب خروجی، گره متصل شده را بررسی می‌کنیم. اگر برچسب‌های خروجی مدل DFNN گره ریشه با گره‌های دیگر نگاشت شده باشند، الگوریتم طبقه‌بندی را به صورت بازگشتی با استفاده از گره‌های متصل‌شده اجرا می‌کنیم. این فرآیند بازگشتی زمانی متوقف می‌شود که برچسب خروجی مدل DFNN در گره آخر با هیچ گره‌ای مرتبط نباشد.

وقتی مدل DFNN داده‌های ورودی را طبقه‌بندی می‌کند، دو شرایط ممکن در مورد هر برچسب

Algorithm 3 Training Data Selection

```
1: procedure SelectTrainingData(X, Y, YC, L)
2:     XL, YL ← emty array
3:     j ← 0
4:     num ← number of row in X
5:     for i ← 0 ... (num − 1) do
6:         if row(YC, i) = L then
7:             row(XL, j) = row(X, i)
8:             row(YL, j) = row(Y, i)
9:             j = j + 1
10:        end if
11:    end for
12:    returnXL, YL
13: end procedure
```

خروجی وجود دارد:

- برچسب خروجی به هیچ گره‌ای نگاشت نمی‌شود.
- برچسب خروجی به یک گره نگاشت می‌شود.

در شرط اول، داده‌های ورودی طبقه‌بندی شده با این برچسب خروجی به مرحله بعدی پردازش نمی‌شوند. بنابراین، این برچسب خروجی نتیجه طبقه‌بندی نهایی می‌شود. ما این شرط را در برچسب خروجی ۱ گره ریشه در شکل ۲.۴ نشان دادیم. در شکل۲.۴، گره ریشه داده‌های ورودی را به عنوان برچسب خروجی ۱ طبقه‌بندی می‌کند. از آنجا که هیچ گره‌ای با برچسب خروجی ۱ مرتبط نیست، نتیجه طبقه‌بندی نهایی کلاس ۱ است.

در شرط دوم، الگوریتم طبقه‌بندی را با استفاده از یک گره متصل به برچسب خروجی به صورت بازگشتی اجرا میکنیم. در این فرآیند بازگشتی، تنها داده‌های ورودی طبقه‌بندی شده به عنوان برچسب خروجی گره متصل انتخاب می‌شوند. ما این فرآیند انتخاب داده‌ها را در الگوریتم ۵ توصیف می‌کنیم. این فرآیند طبقه‌بندی بازگشتی زمانی متوقف می‌شود که برچسب خروجی بدون گره متصل پیدا می‌شود. بنابراین، آخرین برچسب خروجی بدون گره متصل نتیجه طبقه‌بندی نهایی می‌شود. برای انجام این کار، نیاز داریم که نتیجه طبقه‌بندی گره فعلی را با استفاده از خروجی فرآیند بازگشتی به‌روزرسانی کنیم که یعنی تنها نتایج طبقه‌بندی داده‌های ورودی مورد استفاده در فرآیند بازگشتی به‌روزرسانی می‌شوند. فرآیند به‌روزرسانی نتایج طبقه‌بندی را در خط ۷ از الگوریتم ۴ توصیف شده است، که در جزئیات بیشتری در الگوریتم ۶ توضیح داده شده است.

Algorithm 4 Classification

```
1: procedure CLASSIFICATION(X, N)
2:     YC ← Call classify(X) procedure of N
3:     for L ∈ unique(YC) do
4:         linkedN ← Call GetLinkedNode(L) procedure of N
5:         if linkedN ≠ NULL then
6:             XL, YI ← SelectData(X, YC, L)
7:             newYC ← Classification(XL, linkedN)
8:             YC ← UpdateLabel(YC, newYC, YI)
9:         end if
10:    end for
11:    return YC
12: end procedure
```

روند دسته‌بندی بازگشتی را در شکل ۲.۴ نشان داده شده است. در شکل ۲.۴، مدل DFNN گره اصلی داده‌های ورودی را به عنوان برچسب خروجی ۲ دسته‌بندی می‌کند. به دلیل ارتباط برچسب خروجی ۲ در گره اصلی با گره c2، ما فرآیند دسته‌بندی را به صورت بازگشتی با استفاده از گره c2 اجرا می‌کنیم. سپس، گره c2 داده‌های ورودی را به عنوان برچسب خروجی ۴ دسته‌بندی می‌کند. به دلیل عدم وجود گره‌ای مرتبط با برچسب خروجی ۴ در گره c2، نتیجه نهایی دسته‌بندی داده‌های ورودی برابر با کلاس ۴ است.

مدل T-DFNN ممکن است یک کلاس جدید را با استفاده از دو یا چند گره دسته‌بندی کند. به عنوان مثال، در شکل ۲.۴، مدل DFNN گره اصلی داده‌های ورودی را به عنوان برچسب‌های خروجی ۰ و ۲ دسته‌بندی می‌کند. هر دو برچسب خروجی با گره‌های مختلف مرتبط هستند. برچسب‌های خروجی ۰ و ۲ در گره اصلی به ترتیب با گره‌های c0 و c2 مرتبط هستند. در این شرایط، داده‌های ورودی به دو گروه تقسیم می‌شوند. گروه اول داده‌های ورودی دسته‌بندی شده به عنوان برچسب خروجی ۰ توسط مدل DFNN گره اصلی هستند، در حالی که گروه دوم داده‌های ورودی دسته‌بندی شده به عنوان برچسب خروجی ۲ توسط مدل DFNN گره اصلی هستند. این دو گروه با استفاده از گره‌های مختلف به صورت جداگانه پردازش می‌شوند. گروه اول توسط گره c0 پردازش می‌شود، در حالی که گروه دیگر توسط گره c2 پردازش می‌شود. در نهایت، هر دو مدل DFNN گره‌های c0 و c2 داده‌های ورودی را به عنوان برچسب خروجی ۳ دسته‌بندی می‌کنند. به دلیل عدم وجود گره‌ای مرتبط با برچسب خروجی ۳ در هر دو گره، نتیجه نهایی دسته‌بندی داده‌های ورودی برابر با کلاس ۳ است.

Algorithm 5 Data Selection

```
1: procedure SELECTDATA(X, YC, L)
2:     XL, YI ← empty array
3:     j ← 0
4:     num ← number of row in X
5:     for i ← 0 ... (num − 1) do
6:         if row(YC, i) = L then
7:             row(XL, j) = row(X, i)
8:             row(YI, j) = i
9:             j = j + 1
10:        end if
11:    end for
12:    return XL, YI
13: end procedure
```

فرآیندهای دسته‌بندی داده‌های ورودی در مدل T-DFNN شامل چندین گره در سطوح مختلف درخت هستند. بنابراین، زمان مورد نیاز برای دسته‌بندی داده‌های ورودی، جمع دسته‌بندی‌های گره‌ها از ریشه تا گره برگ است. به طور دقیق، زمان دسته‌بندی یک مدل T-DFNN با محاسبه زمان دسته‌بندی گره ریشه با استفاده از معادله زیر تخمین زده می‌شود.

$$Time(N) = \begin{cases} t, & d(N) = 0 \\ t + max(\{time(n) | n \in child(N)\}), & d(N) > 0 \end{cases} \quad (۱.۴)$$

- N یک گره در مدل T-DFNN است.
- t زمانی که گره n .داده های ورودی خود را دسته بندی میکند.
- $d(n)$ عمق گره n است.
- $child(N)$ مجموعه فرزندهای گره N است.

با توجه به اینکه درجه گره N برابر با ۰ است، یعنی گره N هیچ گره فرزندی ندارد، زمان طبقه‌بندی گره N برابر با زمان طبقه‌بندی مدل DFNN آن است. در غیر اینصورت، زمان طبقه‌بندی گره N مجموع زمان طبقه‌بندی مدل DFNN آن و زمان طبقه‌بندی بلندترین گره‌های فرزندش است.

فصل ۵

ارزیابی

۱.۵ پیکربندی

در بخش دادگان به صورت مفصل در مورد داده صحبت شده است. در اینجا به صورت اختصار مراحل پیش پردازش را ذکر میکنیم.

- جاگذاری مقادیر گم شده با میانگین آن حمله برای آن ویژگی.
- جاگذاری مقادیر بینهایت با دوبرابر بیشینه آن حمله برای آن ویژگی.
- جاگذاری مقادیر منفی هر ویژگی با کمینه آن حمله.
- عادی سازی مقادیر با تکنیک MinMax.
- گروه کردن داده ها به قسمت های مختلف.
- تقسیم داده ها به داده آموزش و ارزیابی.

در این مطالعه برای پیاده سازی این مقاله از کتابخانه keras بر روی چهارچوب TensorFlow2 استفاده کرده ام.

بهینه کننده مدل ADAM با نرخ یادگیری ۰.۰۰۱ است که در بخش مفاهیم به آن پرداخته شده است.

در آزمایش، ما مدل T-DFNN را با دو مدل DFNN دیگر مقایسه کردیم. این دو مدل DFNN را DFNN-batch و DFNN-all نامگذاری کردیم. مدل‌های DFNN-batch و DFNN-all ساختار شبکه و هایپرپارامترهای یکسانی با مدل DFNN مورد استفاده در گره‌های T-DFNN دارند. با این حال، این دو مدل DFNN و T-DFNN از داده‌های آموزشی به روش‌های مختلفی استفاده می‌کردند. مدل DFNN-batch فقط از داده‌های آموزشی مربوط به دسته فعلی

استفاده میکند، در حالی که مدل DFNN-all از داده‌های آموزشی مربوط به دسته‌های قدیمی و فعلی استفاده میکند. مشابه مدل DFNN-all، مدل T-DFNN نیز از داده‌های آموزشی مربوط به دسته‌های قدیمی و فعلی استفاده میکند. با این حال، همه داده‌های قدیمی در مدل T-DFNN استفاده نمیشوند. داده‌های آموزشی قدیمی فقط زمانی استفاده می‌شوند که برای آموزش گره‌های جدید نیاز باشند،

به اضافه، ما مدل پیشنهادی T-DFNN خود را با مدل درخت هوفدینگ مقایسه کردیم. همانطور که در بخش دوم بحث شد، مدل درخت هوفدینگ یک الگوریتم شناخته‌شده درخت تصمیم به‌روزرسانی‌پذیر است که قادر به یادگیری از داده‌های افزایشی بزرگ می‌باشد. برای پیاده‌سازی الگوریتم درخت هوفدینگ، از متد HoeffdingTreeClassifier ارائه‌شده توسط کتابخانه scikit-multiflowاستفاده کردیم.

ما از دقت Precision، Recall و F1-score به عنوان معیارهای دسته‌بندی برای مقایسه عملکرد مدل‌ها استفاده کردیم. این معیارها را برای هر کلاس ارزیابی مورد استفاده قرار دادیم. با استفاده از این معیارها، توانستیم عملکرد دسته‌بندی مدل‌ها را برای هر کلاس اندازه‌گیری کنیم. برای اندازه‌گیری عملکرد مدل‌ها در هر دسته ارزیابی، میانگین ماکرو و میانگین وزنی از دقت، بازخوانی و امتیاز F1 محاسبه شده است. فرمول‌های ۱.۵ و ۲.۵ نشان‌دهنده‌ی روش محاسبه‌ی میانگین ماکرو (MA) و میانگین وزنی (WA) معیارها هستند.

$$MA_m = \frac{1}{C}\sum_{i=1}^{C} m_i \qquad (۱.۵)$$

$$WA_m = \sum_{i=1}^{C} \frac{n_i}{N} * m_i \qquad (۲.۵)$$

- m معباری که در حال محاسبه وزنی آن هستیم.
- m_i مقدار معیار در حال محاسبه برای کلاس i.
- C تعداد کلاس.
- n_i تعداد داده از کلاس i.
- N تعداد کل دیتا ها.

سیستمی که بر روی آن تست گرفته ایم به شرح زیر است.

- CPU: AMD Ryzen 9 6900HS (8 cores/16 threads, up to 4.9GHz, 16MB cache)
- GPU: GeForce RTX™ 3070
- RAM: 32Gb DDR4
- OS: Windows 11 Enterprise

۲.۵ نتایج

به دلیل محدودیت در منابع محاسباتی نتایج بر پایه یک بار اجرا و با ۲۰ تکرار میباشند. نتایج زیر عملکرد مدل بر روی دسته اخر را نشان میدهد.

۳.۵ میانگین Percision

Avarage Percision				
Evaluation	DFNN-batch	DFNN-ALL	Howffding tree	T-DFNN
macro avg	0.084	0.823	0.751	0.823
weighted avg	0.532	0.874	0.812	0.858

۴.۵ میانگین Recall

Avarage Recall				
Evaluation	DFNN-batch	DFNN-ALL	Howffding tree	T-DFNN
macro avg	0.064	0.754	0.634	0.742
weighted avg	0.554	0.913	0.893	0.887

۵.۵ میانگین F1

Avarage F1 Score				
Evaluation	DFNN-batch	DFNN-ALL	Howffding tree	T-DFNN
macro avg	0.072	0.793	0.689	0.785
weighted avg	0.544	0.903	0.847	0.867

۶.۵ زمان اجرا

Duration				
Evaluation	DFNN-batch	DFNN-ALL	Howffding tree	T-DFNN
Training	12.3	4,452.54	0.15	1,743.98
Evaluation	32.22	27.35	170.56	127.54

۷.۵ تفاوت نتایج با نتایج مقاله

به طور میانگین نتایج بدست آمده از آزمایش مدل پیاده سازی شده چیزی حدود ۱۰ درصد دقت کمتری نسبت به نتایج مقاله دارد. به نظر این تفاوت دقت به علت تعداد تکرار کمتر مدل (۲۰) نسبت به تعداد تکرار مقاله (۵۰) است. اگرچه نسبت و ترتیب دقت مدل های مختلف ثابت مانده است و میتوان همان نتیجه گیری مقاله را انجام داد.

در بخش زمان اجرا هم به دلیل منابع محاسباتی ضعیف تر زمان بیشتری برای آموزش و ارزیابی برای هر مدل مصرف شده است که با این وجود ترتیب حفظ مانده است.

فصل ۶

نتیجه گیری

یادگیری تدریجی در سیستم‌های تشخیص نفوذ یک مسئله چالش‌برانگیز است. اصلی‌ترین مشکلی که در یادگیری تدریجی مواجه می‌شویم، تنوع مستمر نفوذها و مشکلات فراموشی اطلاعات قبلی هستند. ما این دو مشکل را با ارائه الگوریتم T-DFNN حل کردیم که یک ساختار داده‌ی درختی و مدل‌های DFNN را ترکیب می‌کند. نتایج آزمایشی نشان داد که مدل تولیدشده توسط الگوریتم پیشنهادی T-DFNN قادر است به صورت تدریجی یاد بگیرد و تفاوت چندانی در کارآمدی به دلیل اثر فراموشی نداشته باشد. علاوه بر این، الگوریتم T-DFNN قادر است زمان آموزش را کاهش دهد. با این حال، این الگوریتم نیاز به مراحل محاسباتی بیشتری دارد که زمان طبقه‌بندی را افزایش می‌دهد.

عوامل دیگری که بر دقت، بازخوانی و امتیاز F1 مدل تأثیر می‌گذارند، شباهت بین دسته‌ها و کمبود داده‌ها هستند. این عوامل نه تنها بر مدلT-DFNN بلکه بر مدل‌های دیگر به طور کلی تأثیر می‌گذارند. بنابراین، پیشنهاد می‌شود که تحقیقات جامع‌تری در مورد این عوامل به عنوان کارهای آتی انجام شود تا عملکرد الگوریتم T-DFNN بهبود یابد.

واژه‌نامه فارسی به انگلیسی

آدرس آی پی ..... IP address
اسکریپت ..... Script
اوبونتو ..... Ubuntu
بازآموزی ..... ReTraining
بافر ..... buffer
بهینه ساز ..... Optimizer
پردازش تصویر ..... Image Processing
تداخل فاجعه بار ..... Catastrophic Forgetting
جریان شناسه ..... Flow ID
حملات انکار سرویس ..... denial-of-service
درخت هافدینگ ..... Hoeffding Tree
درخت تصمیم بسیار سریع ..... Very Fast Decision Tree algorithm
دیتافریم ..... Data frame
روتر ..... Router
سوپرکلاس ..... Super class
سوئیچ ..... switch
سیستم پیشگیری از نفوذ ..... IPS
سیستم تشخیص نفوذ ..... Intrusion detection system
شبکه عمیق ..... Deep neural network
ضرر ..... Loss
فایروال ..... Firewall
کلون ..... Clone
گرادیان نزولی ..... gradient Descent
لیبل ..... label
مودم ..... modem
هایپرپارامتر ..... Hyper parameters
یادگیری افزایشی ..... Incremental learning
یادگیری انتقال ..... Transfer learning

یادگیری ماشین .... machine learning

واژه‌نامه انگلیسی به فارسی

برآورد لحظه تطبیقی ..... ADAM
شبکه های عصبی مصنوعی ..... ANNs
پس انتشار ..... Back-propagation
ترافیک معمولی ..... Benign traffic
نیروی بی رحم نوعی حمله است ..... Brute Force
درخت های تصمیم سنتی ..... CART
فراموشی فاجعه بار ..... Catastrophic forgetting
ضبط ..... Capture
شبکه های پیچشی ..... CNN
حملات انکار سرویس توزیع شده ..... Distributed Denial of Service
روبه جلو ..... Feed-forward
مخفف زبان نشانه گذاری فرامتن ..... HTML
سیستم تشخیص نفوذ ..... IDS
تزریق ..... Injection
کتابخانه یادگیری ماشین ..... Keras
نوعی حمله است ..... Man-in-the-Middle
سیستم عامل مک ..... Mac OS
فراداده ..... MetaData
نوعی عادی ساز کننده داده ..... Min MAx
آینه. نوعی پورت ..... mirror
هیچ ..... Null
مشارکت شبکه جزیی ..... Partial network sharing
دقت.نوعی معیار ..... Precision
طبقه بندی جنگل تصادفی ..... RandomForestClassifier
بازخوانی.نوعی معیار ..... Recall
اتابع فعال ساز خطی ..... Relu
ریشه میانگین مربعات انتشار ..... RMSprop
کتابخانه یادگیری ماشین ..... Scikit-learn

اتابع فعال ساز بر پایه نمایی ..... Soft max
نوعی پایگاه داده ..... SQL
ضربه زدن. نوعی معماری سرور ..... Tapping
کتابخانه یادگیری ماشین ..... TensorFlow
متدی برای تقسیم داده ..... Train test split
شبکه عصبی رو به جلو مبتنی بر درخت تصمیم ..... T-DFNN
مخفف درخت تصمیم بسیار سریع ..... VFDT
کتابخانه یادگیری ماشین ..... Weka

کتاب‌نامه

[1] M. Data and M. Aritsugi, "T-DFNN: An Incremental Learning Algorithm for Intrusion Detection Systems," in IEEE Access, vol. 9, pp. 154156-154171, 2021, doi: 10.1109/ACCESS.2021.312798

[2] Martellini, Maurizio; Malizia, Andrea (2017-10-30). Cyber and Chemical, Biological, Radiological, Nuclear, Explosives Challenges: Threats and Counter Efforts. Springer. ISBN 9783319621081.

[3] Mohammed, Mohssen; Rehman, Habib-ur (2015-12-02). Honeypots and Routers: Collecting Internet Attacks. CRC Press. ISBN 9781498702201.

[4] Vacca, John R. (2009-05-04). Computer and Information Security Handbook. Morgan Kaufmann. ISBN 9780080921945.

[5] David M. Chess; Steve R. White (2000). "An Undetectable Computer Virus". Proceedings of Virus Bulletin Conference. CiteSeerX 10.1.1.25.1508.

[6] Lunt, Teresa F., "IDES: An Intelligent System for Detecting Intruders," Proceedings of the Symposium on Computer Security; Threats, and Countermeasures; Rome, Italy, November 22–23, 1990, pages 110–121

[7] McCloskey, Michael; Cohen, Neal J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychology of Learning and Motivation. Vol. 24. pp. 109–165. doi:10.1016/S0079-7421(08)60536-8. ISBN 978-0-12-543324-2.

[8] Ratcliff, Roger (1990). "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions". Psychological Review. 97 (2): 285–308. doi:10.1037/0033-295x.97.2.285. PMID 2186426. S2CID 18556305

[9] Hebb, Donald Olding (1949). The Organization of Behavior: A Neuropsychological Theory. Wiley. ISBN 978-0-471-36727-7. OCLC 569043119

[10] Schlimmer, J. C., Fisher, D. A case study of incremental concept induction. Fifth National Conference on Artificial Intelligence, 496-501. Philadelphia, 1986

[11] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep Learning". Nature. 521 (7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539. PMID 26017442. S2CID 3074096.

[12] Ciresan, D.; Meier, U.; Schmidhuber, J. (2012). "Multi-column deep neural networks for image classification". 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3642–3649. arXiv:1202.2745. doi:10.1109/cvpr.2012.6248110. ISBN 978-1-4673-1228-8. S2CID 2161592.

[13] https://www.tutorialspoint.com/what-is-hoeffding-tree-algorithm

[14] https://www.unb.ca/cic/datasets/ids-2017.html

[15] https://www.kaggle.com/code/athena21/cicids2017-visualization-smote-underbalancing

[16] Brownlee, Jason (8 January 2019). "A Gentle Introduction to the Rectified Linear Unit (ReLU)". Machine Learning Mastery. Retrieved 8 April 2021.

[17] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180–184. ISBN 978-0-26203561-3.

[18] Kingma, Diederik; Ba, Jimmy (2014). "Adam: A Method for Stochastic Optimization". arXiv:1412.6980 [cs.LG].

[19] Dozat, T. (2016). "Incorporating Nesterov Momentum into Adam". S2CID 70293087.

[20] A. Fonseca and B. Cabral, "Prototyping a GPGPU neural network for deep-learning big data analysis," Big Data Res., vol. 8, pp. 506

Abstract

Recently, intrusion detection systems (IDSs) have witnessed a surge in popularity, with machine learning algorithms playing a pivotal role in their reliability. However, the prevailing challenge lies in the static nature of most models, as they are trained on datasets comprising all known intrusions. Consequently, when new intrusions emerge, the models necessitate retraining using both old and new data to maintain accurate intrusion classification. Given the continuous emergence of novel threats in real-world scenarios, it is imperative for IDS machine learning algorithms to possess the ability to learn incrementally as these new intrusions manifest.

Addressing this issue, we introduce T-DFNN, a groundbreaking algorithm designed to enable incremental learning of emerging intrusions. The T-DFNN model is a composition of multiple deep feedforward neural network (DFNN) models interconnected in a tree-like structure. To evaluate the efficacy of our proposed algorithm, we conducted experiments utilizing the widely-used CICIDS2017 network intrusion dataset, encompassing benign traffic and the most prevalent network intrusions.

The experimental results underscore the efficacy of the T-DFNN algorithm in incrementally learning new intrusions while mitigating the detrimental effects of catastrophic forgetting. Across multiple retraining processes, the T-DFNN model achieved an impressive macro average F1-score exceeding 0.85. Moreover, our proposed T-DFNN model offers several advantages over alternative models. In comparison to DFNN and Hoeffding tree models trained solely on the latest targeted intrusions, our T-DFNN model exhibits higher F1-scores. Additionally, it significantly reduces training times compared to a DFNN model trained on a dataset containing all targeted intrusions. Although various factors influence training duration, the T-DFNN algorithm demonstrates promising results in addressing the challenge of ever-evolving network intrusion variants.Recently, intrusion detection systems (IDSs) have witnessed a surge in popularity, with machine learning algorithms playing a pivotal role in their reliability. However, the prevailing challenge lies in the static nature of most models, as they are trained on datasets comprising all known intrusions. Consequently, when new intrusions emerge, the models necessitate retraining using both old and new data to maintain accurate intrusion classification. Given the

continuous emergence of novel threats in real-world scenarios, it is imperative for IDS machine learning algorithms to possess the ability to learn incrementally as these new intrusions manifest.

Addressing this issue, we introduce T-DFNN, a groundbreaking algorithm designed to enable incremental learning of emerging intrusions. The T-DFNN model is a composition of multiple deep feedforward neural network (DFNN) models interconnected in a tree-like structure. To evaluate the efficacy of our proposed algorithm, we conducted experiments utilizing the widely-used CICIDS2017 network intrusion dataset, encompassing benign traffic and the most prevalent network intrusions.

The experimental results underscore the efficacy of the T-DFNN algorithm in incrementally learning new intrusions while mitigating the detrimental effects of catastrophic forgetting. Across multiple retraining processes, the T-DFNN model achieved an impressive macro average F1-score exceeding 0.85. Moreover, our proposed T-DFNN model offers several advantages over alternative models. In comparison to DFNN and Hoeffding tree models trained solely on the latest targeted intrusions, our T-DFNN model exhibits higher F1-scores. Additionally, it significantly reduces training times compared to a DFNN model trained on a dataset containing all targeted intrusions. Although various factors influence training duration, the T-DFNN algorithm demonstrates promising results in addressing the challenge of ever-evolving network intrusion variants.

College of Science
School of Mathematics, Statistics, and Computer Science

T-DFNN: An Incremental Learning Algorithm for Intrusion Detection Systems

Rayan Forsat

Supervisor: Hedie Sajedi

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

2023