Electronic & Cyber Defense
Vol. 13, No. 3, 2025 (Serial No. 51). pp 71- 85
ISSN: 2322-4347, E-ISSN: 2980-8979

Malware Detection Based on Firefly Meta-Heuristic Algorithm and Long-Short-Term Memory Learning Network

Aliakbar Tajari Siahmarzkooh
Assistant Professor, Department of Computer Sciences, Golestan University, Gorgan, Iran
(Correspondence: *a.tajari@gu.ac.ir*)

ARTICLE INFO

Article history:

Article Type: Research paper

Receive: 29 June 2025
Revise: 25 August 2025
Accept: 14 October 2025
Available online: 23 October 2025

Keywords:
Malware detection
Meta-heuristic algorithm
Convolutional Neural Network
Long-Short Term Memory
Firefly algorithm

ABSTRACT

Malware detection has become one of the important research areas in recent years. Despite the efforts made in this field, it is still possible to improve the accuracy of the presented models in detecting different types of malware. In this paper, an attempt has been made to adjust the parameters of deep learning networks based on the use of four meta-heuristic algorithms including bee colony, firefly, imperialist competitive and bat algorithm. Two deep learning networks including Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) with different number of layers have been used to achieve maximum detection accuracy. The experimental results show that in the best case, using the firefly algorithm and LSTM learning network with 3 layers, batch size 40, filter size 5*5 and number of periods 300 leads to an accuracy of 99.997% in malware detection, which is significantly superior compared to other presented works. Other performance evaluation parameters also show very good values, which indicates the superiority of the proposed method over other similar works.

Cite this article: Tajari Siahmarzkooh, A. (2025). Malware Detection Based on Firefly Meta-Heuristic Algorithm and Long-Short-Term Memory Learning Network. *Electronic and Cyber Defense*, 13(3), pp. 71-85. 2025.
https://dor.isc.ac/dor/20.1001.1.23224347.1404.13.3.6.5

Publisher: Imam Hossein University

«پدافند الکترونیکی و سایبری»
سال سیزدهم، شماره ۳، پاییز ۱۴۰۴، ص ۷۱-۸۵
شاپا الکترونیکی: 4347 -2322 شاپا چاپی: 8979- 2980

تشخیص بدافزار مبتنی بر الگوریتم فرا ابتکاری کرم شب‌تاب و شبکه‌ی یادگیری حافظه طولانی کوتاه‌مدت

علی‌اکبر تجری سیاه‌مرزکوه
استادیار، گروه علوم کامپیوتر، دانشگاه گلستان، گرگان، ایران (نویسنده مسئول) a.tajari@gu.ac.ir

چکیده

تشخیص بدافزار به یکی از حوزه‌های تحقیقاتی مهم در چند سال اخیر تبدیل‌شده است. علی‌رغم کوشش‌هایی که در این زمینه صورت گرفته است، همچنان امکان بهبود دقت مدل‌های ارائه‌شده در تشخیص انواع مختلف بدافزار وجود دارد. در این مقاله، با تکیه‌بر به‌کارگیری چهار الگوریتم فرا ابتکاری زنبورعسل، کرم شب‌تاب، رقابت استعماری و خفاش، تلاش برای تنظیم پارامترهای شبکه‌های یادگیری عمیق صورت گرفته است. دو شبکه‌ی یاد گیر عمیق شامل شبکه‌ی عصبی پیچشی و حافظه‌ی طولانی کوتاه‌مدت با تعداد لایه‌های مختلف برای دستیابی به حداکثر دقت تشخیص مورداستفاده قرارگرفته‌اند. نتایج آزمایش نشان می‌دهد که در بهترین حالت، بهره‌گیری از الگوریتم کرم شب‌تاب و شبکه‌ی یاد گیر *LSTM* با 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 5*5 و تعداد دوره‌ی 300 منجر به‌دقت 99.997٪ در تشخیص بدافزار می‌شود که در مقایسه با سایر کارهای ارائه‌شده از برتری قابل‌ملاحظه‌ای برخوردار است. سایر پارامترهای ارزیابی کار آیی نیز مقادیر بسیار خوبی را نشان می‌دهند که این موضوع بیانگر برتری روش پیشنهادی نسبت به سایر کارهای مشابه است.

مشخصات مقاله

تاریخچه مقاله:
نوع مقاله: علمی- پژوهشی

دریافت: 08 تیر 1404
بازنگری: 23 شهریور 1404
پذیرش: 22 مهر 1404
ارائه آنلاین: 01 آبان 1404

کلیدواژه‌ها:
تشخیص بدافزار
الگوریتم فرا ابتکاری
شبکه‌ی عصبی پیچشی
حافظه‌ی طولانی کوتاه‌مدت
الگوریتم کرم شب‌تاب

استناد:
تجری سیاه مرزکوه، علی اکبر. (۱۴۰۴). تشخیص بدافزار مبتنی بر الگوریتم فراابتکاری کرم شب تاب و شبکه یادگیری حافظه طولانی کوتاه مدت. *پدافند الکترونیکی و سایبری*، ۱۳(۳)، ص ۷۱-۸۶. https://dor.isc.ac/dor/20.1001.1.23224347.1404.13.3.6.5

ناشر: دانشگاه جامع امام حسین(ع).

۱. مقدمه

افـزایش تعـداد و پیچیـدگی بـدافزارها بـه یکـی از جـدی‌تـرین تهدیدات امنیت سایبری تبدیل‌شده است و از طرفـی نیـز امکـان آلودگی دستگاه‌های متصل به اینترنت توسط کـدهای مخـرب در حال افزایش است [1]. اگر ایـن دسـتگاه‌هـا آلـوده شـوند در آن صورت مهاجمان می‌توانند از آن‌هـا بـرای کنتـرل سـامانه‌های در معرض خطـر بـرای مـدت طـولانی اسـتفاده کننـد [2]. بـدافزار، اطلاعات محرمانه را می‌دزدد، داده‌های مهم را از بـین مـی‌بـرد و گاهی از داده‌ها به‌عنوان وثیقه برای باج‌خواهی، پول‌شویی و سـایر جرائم بسیار مخرب اسـتفاده مـی‌کنـد. در سـال‌های اخیـر، یـک حمله‌ی منع سرویس توزیع‌شده در مقیاس بزرگ با بهره‌برداری از دستگاه‌های آلوده به یک برنامه‌ی بدافزار به نام Mirai انجام شـد [3]. در سال 2017، تعداد زیادی از رایانه‌ها در سراسر جهان بـه باج افزاری به نام WannaCry آلوده شدند [3]. علاوه بـر ایـن، در سال 2022، یک حمله‌ی هدفمند توسط یک برنامه‌ی بدافزار بـه نام Emotet در ایمیل‌های فیشینگ پیوست شد [3].

اگرچه صنعت امنیت سایبری بـه‌طـور مـداوم در حـال نظـارت از طرق مختلف بر بدافزار است، اما مهاجمان سایبری هیچ نشـانه‌ای از توقف یا کنـد کـردن حمـلات از خـود نشـان نمـی‌دهنـد [4]. گروه‌های هکـری مخـرب از روش‌هـای بـدافزاری ماننـد درهـای پشـتی، کاوشـگرها و جاسـوس‌افزارهـا کـه بـه‌طـور گسـترده مورداسـتفاده‌ی مهاجمـان اسـت بـرای آلـوده کـردن دسـتگاه‌ها استفاده می‌کنند [4].

Malspam حاوی پیوست‌ها یا URLهـای آلـوده اسـت. این‌گونـه بدافزارها می‌توانند اطلاعات حساس را از سـازمان‌هـا جمـع‌آوری کنند. نمونه‌های دیگر بدافزارها عبارت‌اند از WannaCry، Kovter، ZeuS، Dridex، IcedID، Gh0st و Mirai. طراحـی سـامانه‌های تشخیص بدافزار کارآمد هنـوز در یـک چـالش بـین مهاجمـان و تحلیلگران امنیتی به سر می‌برد [5].

کشف کدهای مخرب جدید امری مستمر بـوده و تعـداد کـدهای مخرب نیز به‌سرعت در حال افزایش است. ازاین‌رو، تجزیه‌وتحلیل و طبقـه‌بنـدی تمـام کـدهای مخـرب موجـود بـا اسـتفاده از اشکال‌زدایی و قوانین کار پیچیده‌ای است [6].

یافتن راهکارهای دفاعی در هر زمان که یـک کـد مخـرب جدیـد کشف شود، روشی به‌اندازه کافی سـریع بـرای تشـخیص کـدهای مخرب محسوب نمی‌شود. برای پاسخ به یک کـد مخـرب جدیـد، باید راهکاری جهت تجزیه‌وتحلیل و طبقـه‌بنـدی سـریع آن پیـدا کنیم. در حالت کلی، کدهای مخـرب از کتابخانـه‌هـا و APIهـای مشابه استفاده می‌کنند. بنـابراین، شـباهت‌هـایی در رفتـار آن‌هـا وجود دارد که اگر بتوانیم کدهای مخرب جدید را شناسایی کرده و آن‌ها را در خانواده‌های کدهای مخرب موجود طبقه‌بندی کنیم، می‌توانیم نوع کد مخرب جدید را تأیید کرده و روش‌های دفـاعی مناسب در مقابل آن ارائه کنیم [7].

اگرچه شناسایی دقیق و سریع بدافزار امری ضروری است، اما یک روش شناسـایی واحـد مـی‌توانـد حملـه را نادیـده بگیـرد. طبـق گزارش‌ها تعداد حملات بدافزار جدیـد هرسـاله و بـه‌طـور مـداوم مشاهده می‌شود. این گزارش‌ها نشان می‌دهد که روش تشـخیص بدافزار باید اطلاعات مربوط به بـدافزار جدیـد را در زمـان واقعـی جمع‌آوری کند تا به فعالیت آن پاسخ دهد. بسیاری از سامانه‌های تشخیص بدافزار به‌طور گسـترده موردمطالعـه قرارگرفته‌انـد، امـا همچنان می‌توانند بسته به نوع، بدافزار را نادیده بگیرند [7]. حتی با افزایش دقت روش‌های یادگیری ماشـین، ایـن روش همچنـان ازنظر قابلیت استفاده و ایمنی مشکلاتی دارد.

درحالی‌که تشخیص بدافزار به‌طور گسترده برای مقابله با حملات موردمطالعه قرارگرفته است، آن‌ها هنوز بـرای شناسـایی بـدافزار جدید کافی نیستند. بسیاری از روش‌های موجود بر اساس نتـایج تشخیص به‌دست‌آمده آمده از اجرای یک روش مبتنی بر قـوانین از پیش مشخص‌شده بر روی چند نمونـه آزمایشـی بـرای تعیـین مخرب بودن نمونه استفاده می‌کنند. درحالی‌کـه مطالعـات نشـان می‌دهد که اتکا به قوانین از پـیش تعیین‌شـده بـر دقـت قضـاوت تأثیر منفی می‌گذارد. به‌عنوان‌مثال، حتی اگر موتورهای شناسایی قابل‌اعتماد با دقت بالا تشخیص دهنـد کـه یـک فایـل مشـکوک بدافزار است، زمانی که طبقه‌بنـدی کننـده‌های مبتنـی بـر امضـا تشخیص دهند که فایل مذکور خوش‌خیم اسـت، بـدافزار نادیـده گرفته می‌شود. بنابراین، این روش‌های مبتنی بر امضا ممکن است انواع جدید بـدافزارها را از دسـت بدهنـد، زیـرا اکثـر موتورهـای ضدویروس فاقد اطلاعات کافی برای شناسـایی بـدافزار جدیـد در آن زمان هستند [8].

در این مقاله، برای در نظر گرفتن قابلیت اطمینان تصمیم نهـایی در تشخیص بدافزار، پارامترهای شبکه‌ی یاد گیر بـرای تشـخیص بدافزار را با اجرای چنـد روش هـوش مصـنوعی مـورد ارزیـابی و تحلیل قرار مـی‌دهـیم تـا بتـوانیم بهتـرین مقـادیر را بـرای ایـن پارامترها به دست آوریم. می‌توان انتظار داشت کـه اعمـال چنـد روش با کاهش مثبت‌ها و منفـی‌هـای کـاذب، دقـت تشـخیص را بهبود بخشد. ساختار مطالب در ادامه این مقاله به شرح زیر است. در بخش 2 پیشینه‌ی حوزه‌ی تحقیقاتی تشخیص بدافزار را شـرح می‌دهیم. بخش 3 مفاهیم کلیدی و اجزای سیستم پیشـنهادی را ارائه می‌کند. در بخش 4 بـه ارائـه‌ی نحـوه‌ی پیـاده‌سـازی روش پیشنهادی پرداخته می‌شود. در بخش 5، نتـایج ارزیـابی سیسـتم پیشنهادی را گزارش می‌کنیم و در بخش 6، نتیجه‌گیری را ارائـه می‌دهیم و در مورد آن بحث می‌کنیم. نوآوری اصـلی ایـن مقالـه، ترکیب الگوریتم‌های یادگیری ماشین با چنـد روش فـرا ابتکـاری

برای بهبود دقت سیستم تشخیص بدافزار است که در بخش راهکار پیشنهادی به آن پرداخته می‌شود.

۲. کارهای پیشین

تاکنون بررسی‌های تحقیقاتی قابل‌توجهی در قالب تجزیه‌وتحلیل و تشخیص بدافزار انجام‌شده است. این بخش بررسی روش‌های مختلف مورداستفاده برای طبقه‌بندی بدافزارها را ارائه می‌دهد.

روش‌های یادگیری عمیق بر چندین لایه‌ی انتزاعی متمرکز هستند، به‌گونه‌ای که لایه‌های بالاتر اطلاعات داده‌های انتزاعی بیشتری را نشان می‌دهند. شبکه‌های عصبی جایگزین روش‌های یادگیری ماشین معمولی به‌عنوان جایگزینی در شناسایی بدافزارها شدند. مزایای مدل‌های شبکه‌ی عصبی شامل توانایی یادگیری افزایشی، لایه‌های آموزشی در صورت لزوم و غیره است. یادگیری عمیق به توسعه‌ی مدل‌های خودکار و تعمیم‌یافته برای شناسایی و طبقه‌بندی بدافزارهای شناخته‌شده و ناشناخته کمک می‌کند. شبکه CNN، شبکه‌های عصبی پیش‌خور [۹] هستند که به‌طور خاص برای مشکلات طبقه‌بندی استفاده می‌شوند. با توجه به توانایی یادگیری ویژگی‌های قوی، سامانه‌های تشخیص بدافزار پیشرفته از مدل CNN برای یادگیری الگوهای دودویی در شناسایی بدافزار استفاده می‌کنند. مدل‌های گروهی می‌توانند چندین مدل یادگیری ماشین و یادگیری عمیق را ترکیب کنند. در بسیاری از تحقیقات، مدل CNN برای تشخیص بدافزار پیشنهادشده است. در ادامه به تشریح چند روش شناسایی بدافزار اشاره می‌کنیم.

ایمران و همکاران [۱۰]، یک رویکرد طبقه‌بندی بدافزار بر اساس شباهت را پیشنهاد کردند. درروش پیشنهادی آن‌ها، توالی فراخوانی API با استفاده از مدل‌های پنهان مارکوف به دست آمد و امتیازات شباهت برای طبقه‌بندی بدافزار محاسبه شد. رویکرد آن‌ها با داده‌های کمتر به‌خوبی کار می‌کند و نیاز به سربار محاسباتی بالایی دارد. تجزیه‌وتحلیل پویا ناکارآمد است، زیرا بدافزار ممکن است رفتار خود را در محیط‌های مجازی در طول اجرا تغییر دهد. روش‌های ترکیبی از ویژگی‌های مشتق شده از روش‌های ایستا، پویا یا یادگیری ماشین برای طبقه‌بندی بدافزار استفاده می‌کنند. ریک و همکاران [12]، ویژگی‌های فراخوانی API پویا را استخراج کردند و از ماشین بردار پشتیبان برای شناسایی بدافزار استفاده کردند.

لی و همکاران [۱۳]، یک راهکار تشخیص بدافزار تجزیه‌وتحلیل مبتنی بر مجوز برای توسعه‌ی بدافزار به نام SIGPID را توسعه دادند. در این راهکار، مقادیر معیارهای ارزیابی شامل دقت، فراخوانی، صحت و اندازه‌گیری F با استفاده از طبقه‌بندی‌کننده‌ی ماشین بردار پشتیبان، بالای 90٪ به‌دست‌آمده است و مدت‌زمان تجزیه‌وتحلیل 4 تا 32 برابر سریع‌تر از استفاده از مجوزهای کامل است و دلیل این امر، این واقعیت است که ماشین بردار پشتیبان می‌تواند روی نمونه‌هایی که در آن مجوزهای کمتری به آن داده‌شده است آموزش ببیند.

لی [۱۴]، طبقه‌بندی‌کننده‌ی جدید و قابل‌اعتمادی را برای شناسایی بدافزار اندروید بر اساس معماری ماشین و استخراج ویژگی‌های برنامه از فایل‌ها و کد منبع پیشنهاد کرد. بر اساس یافته‌های این مقاله، ویژگی‌های عددی یک برنامه اغلب یک بردار بسیار پراکنده است و تعاملات بین ویژگی‌های مختلف برای افشای الگوهای رفتار مخرب بسیار حیاتی است.

ژائو [۱۵]، یک سیستم تشخیص بدافزار مبتنی بر فراخوانی API حساس برای اندروید ارائه کرد. فراخوانی‌های API مربوط به برنامه‌های مختلف در طول فرآیند آموزش با استفاده از تحلیل معکوس بازیابی می‌شوند. یافته‌های آزمایش نشان می‌دهد که دقت و صحت طرح ارائه‌شده دارای دقت 93٪ است.

وانگ و همکاران [۱۶]، از شبکه‌ی عصبی پیچشی (CNN) و رمزگذار خودکار عمیق (DAE) به‌عنوان دو جزء اصلی مدل ترکیبی خود استفاده کردند. برای دستیابی به‌دقت بالاتر هنگام شناسایی نرم‌افزارهای مخرب، از چندین CNN برای جستجوی برنامه‌های مخرب اندروید و بازسازی ویژگی‌های برنامه‌های اندروید استفاده‌شده است. در مقایسه با روش‌های یادگیری ماشین سنتی، CNN-S پیشرفت قابل‌توجهی در توانایی خود در شناسایی نرم‌افزارهای مخرب نشان می‌دهد. به‌طور مشخص، کار آیی مدل CNN-S ۵٪ بهتر از ماشین بردار پشتیبان است، درحالی‌که زمان آموزش مدل ترکیبی DAE-CNN ۸۳٪ بهتر از CNN-S است.

کیم و همکاران [۱۷]، یک معماری منحصربه‌فرد برای شناسایی بدافزارهای مخرب ارائه کردند. درروش پیشنهادی آن‌ها، یک رویکرد استخراج ویژگی مبتنی بر وجود یا شباهت برای اصلاح ویژگی‌ها و به دست آوردن ویژگی‌های مؤثرتر در تشخیص بدافزار اعمال‌شده است. چارچوب پیشنهادی آن‌ها از ویژگی‌های زیاد و نقطه نظرات مختلف برای به تصویر کشیدن ویژگی‌های برنامه‌های اندروید استفاده می‌کند. چندین آزمایش بر روی 41260 نمونه به‌منظور ارزیابی عملکرد آموزش صورت گرفته است.

ژو و همکاران [۱۸]، یک رویکرد کلی برای کاهش میزان نویز در داده‌های آموزشی برای سامانه‌های تشخیص بدافزار اندروید مبتنی بر هوش مصنوعی ارائه کردند. در راهکار ارائه‌شده توسط آن‌ها، قبل از فراهم نمودن داده‌های آموزشی برای هر روش تشخیص بدافزار مبتنی بر یادگیری ماشین، ابتدا الگوریتم‌های شناسایی داده‌های پرت برای تمامی ویژگی‌ها اعمال می‌شود تا نویز کاهش یابد.

رن و همکاران [۱۹]، راهکارهای مبتنی بر یادگیری عمیق را برای شناسایی دو ویروس اندرویدی انتها به انتها[1] ارائه کردند. در مقایسه با راهکارهای تشخیص که تاکنون مورداستفاده قرار گرفتند، رویکرد پیشنهادی آن‌ها مزایای یک فرآیند یادگیری فراگیر را فراهم می‌کند. نتایج آزمایش نشان داد که روش ارائه‌شده دارای دقت تشخیص ۹۵.۸٪ است، از منابع کمتری استفاده می‌کند. همچنین نیازی به مهندسی ویژگی‌های انسانی ندارد و محدودیت‌هایی در اندازه‌ی فایل ورودی ندارد.

وانگ و همکاران [۲۰]، برای افزایش کار آیی تشخیص بدافزار پیشنهاد کردند تا ابتدا ویژگی‌های برنامه‌های اندروید با استفاده از چند CNN و یک مدل ترکیبی از CNN و DAE ساخته شود. تحت این شرایط، CNN-S دارای قابلیت بالا در استخراج ویژگی و شناسایی بدافزار است.

در جدول 1، مقایسه‌ی بین چند نمونه روش تشخیص در مجموعه داده‌های مختلف و مزایا و معایب آن‌ها نشان داده‌شده است.

۳. مفاهیم و اجزای سیستم پیشنهادی

در این بخش به ارائه‌ی مفاهیم استفاده‌شده درروش پیشنهادی و اجزای آن می‌پردازیم.

3-1. یادگیری عمیق

یادگیری عمیق زیرشاخه‌ای از یادگیری ماشین است که ورودی را در سطوح مختلف یاد می‌گیرد تا بازنمایی دانش بهتری به دست آورد. پیشرفت‌ها در علم کامپیوتر با یادگیری عمیق عمدتاً از طریق شبکه‌های عصبی پیچشی (CNN) توسعه‌یافته است [31].

مدل‌های یادگیری عمیق، ویژگی‌های پیچیده را یاد می‌گیرند و یک مدل پیچیده با لایه‌های کانولوشن زیاد که به میلیون‌ها پارامتر نیاز دارند آموزش می‌دهند. این امر درنهایت منجر به برازش بیش‌ازحد تنها در چند دوره می‌شود و مدل به‌خوبی تعمیم نمی‌یابد و درنتیجه مدل عملکرد ضعیفی دارد. استفاده از مفهوم CNN که به‌طور کامل بر روی یک مجموعه داده‌ی گسترده آموزش‌دیده شده می‌تواند برای شناسایی و طبقه‌بندی بدافزار مؤثر واقع شود. ایده‌ی کلیدی این است که دانش به‌دست‌آمده در یادگیری یک مدل می‌تواند به تقویت یک کار متفاوت در یادگیری کمک کند [۲۱].

این شبکه‌ها به‌طور فزاینده‌ای عمیق‌تر ساخته می‌شوند و ورودی از لایه‌های زیادی عبور می‌کند. اطلاعات ورودی ممکن است قبل از رسیدن به لایه‌ی نهایی شبکه ناپدید شوند. ResNet و سایر CNNها این مشکل را برطرف می‌کنند، اما مسیرهای کوتاه‌تری از لایه‌های قبلی به لایه‌های بعدی ایجاد می‌کنند.

[1] Android end-to-end

جدول (1). بررسی چند روش تشخیص بدافزار با استفاده از یادگیری عمیق و یادگیری ماشین

روش	مزایا	محدودیت‌ها
جنگل تصادفی و شبکه عصبی مصنوعی [21]	سرعت‌بالا	دقت پایین تشخیص چند کلاسه
رمزگذار خودکار {22}	دقت بالای تشخیص چند کلاسه	سرعت متوسط
شبکه عصبی پیچشی و شبکه طولانی کوتاه‌مدت [23]	دقت بالای دو کلاسه	سرعت‌پایین در تشخیص چند کلاسه
k نزدیک‌ترین همسایگی ترکیبی [24]	همگرایی سریع	دقت پایین در تشخیص چند کلاسه
شبکه عصبی و درخت تصمیم [25]	دقت بالا در داده‌های نامتوازن	همگرایی کند و ضعف شناسایی برخی حملات
ماشین بردار پشتیبان و شبکه عصبی کم‌عمق [26]	سرعت‌بالای تشخیص	مشکلات مربوط به داده-های پرت
بیزین و شبکه باور عمیق [27]	پیش‌پردازش قوی داده‌ها	دقت پایین در داده‌های نامتوازن
الگوریتم پرندگان و شبکه بازگشتی [28]	دقت بالای تشخیص دو کلاسه	سرعت‌پایین در همگرایی و رسیدن به نتیجه نهایی
شبکه عصبی عمیق [29]	دقت بالا در تشخیص چند کلاسه	همگرایی زودرس و بهینه محلی
رگرسیون لجستیک و گرادیان تقویتی [30]	شناسایی داده‌های پرت با دقت بالا	دقت پایین در داده‌های نامتوازن

نوع دیگر الگوریتم یادگیری عمیق، حافظه‌ی طولانی کوتاه‌مدت (LSTM) است [۲۲] که نوعی شبکه‌ی عصبی بازگشتی (RNN) است که باهدف مقابله با مشکل گرادیان ناپدید شدن موجود در RNNهای سنتی ارائه‌شده است. مزیت آن نسبت به سایر RNNها، مدل‌های پنهان مارکوف و سایر روش‌های یادگیری توالی، عدم حساسیت نسبی آن به طول شکاف، است. هدف آن ارائه‌ی یک حافظه‌ی کوتاه‌مدت برای RNN است که می‌تواند هزاران گام دوام بیاورد، بنابراین حافظه‌ی طولانی کوتاه‌مدت برای طبقه‌بندی، پردازش و پیش‌بینی داده‌ها بر اساس سری‌های زمانی قابل‌استفاده است [۳۲].

یک واحد LSTM از یک سلول، یک دروازه‌ی ورودی، یک دروازه‌ی خروجی و یک دروازه‌ی فراموشی تشکیل‌شده است. سلول مقادیر را در بازه‌های زمانی دلخواه به خاطر می‌آورد و سه دروازه جریان اطلاعات را به داخل و خارج از سلول تنظیم می‌کنند. دروازه‌های فراموشی تصمیم می‌گیرند چه اطلاعاتی را از حالت قبلی با نگاشت وضعیت قبلی و ورودی فعلی به مقداری بین 0 و 1 حذف کنند. مقدار (گرد شده) یک به معنای حفظ اطلاعات و مقدار صفر به معنای دور انداختن آن است. گیت‌های ورودی تصمیم می‌گیرند که با استفاده از سیستم مشابه دروازه‌های فراموشی، کدام بخش از اطلاعات جدید را در وضعیت سلول فعلی ذخیره کنند. گیت‌های خروجی با در نظر گرفتن

حالت‌های قبلی و فعلی، کنترل می‌کنند که کدام بخش از اطلاعات در سلول فعلی با اختصاص مقداری از صفرتا یک به اطلاعات، خروجی بگیرند. خروجی انتخابی اطلاعات مربوطه از وضعیت فعلی به شبکه LSTM این امکان را می‌دهد تا وابستگی‌های مفید و طولانی‌مدت را برای پیش‌بینی‌ها در مراحل زمانی فعلی و آینده حفظ کند.

3-2. الگوریتم‌های فرا ابتکاری

الگوریتم فرا ابتکاری، یک‌رویه یا اکتشاف سطح بالاتر است که برای یافتن، تولید، تنظیم و یا انتخاب یک اکتشاف طراحی‌شده است که ممکن است راه‌حل مناسبی برای یک مسئله‌ی بهینه-سازی یا یک مسئله‌ی یادگیری ماشین با اطلاعات ناقص ارائه دهد. فرا ابتکاری زیرمجموعه‌ای از راه‌حل‌ها را نمونه‌برداری می‌کند که در غیر این صورت آن‌قدر بزرگ است که نمی‌توان آن‌ها را به‌طور کامل برشمرد یا به نحو دیگری کاوش کرد. فرا ابتکاری ممکن است مفروضات نسبتاً کمی در مورد حل مسئله‌ی بهینه‌سازی داشته باشد و بنابراین ممکن است برای مسائل مختلف قابل‌استفاده باشد.

در مقایسه با الگوریتم‌های بهینه‌سازی و روش‌های تکرارشونده، فرا ابتکاری تضمین نمی‌کند که بتوان یک راه‌حل بهینه را برای برخی از کلاس‌ها پیدا کرد. بسیاری از فرا ابتکاری‌ها نوعی بهینه-سازی تصادفی را اجرا می‌کنند، به‌طوری‌که راه‌حل یافت شده به مجموعه متغیرهای تصادفی تولیدشده وابسته است. فرا ابتکاری اغلب می‌تواند با جست‌وجوی مجموعه‌ی بزرگی از راه‌حل‌های امکان‌پذیر، راه‌حل‌های خوبی را با تلاش محاسباتی کمتر نسبت به الگوریتم‌های بهینه‌سازی، روش‌های تکراری یا اکتشافی ساده پیدا کند. به‌این‌ترتیب، آن‌ها رویکردهای مفیدی برای مسائل بهینه‌سازی هستند. در این مقاله از چهار الگوریتم فرا ابتکاری زنبورعسل مصنوعی [33]، کرم شب‌تاب [34]، رقابت استعماری [35] و خفاش [36] استفاده‌شده است.

3-2-1. الگوریتم زنبورعسل مصنوعی

زنبورعسل مصنوعی (ABC) یک الگوریتم مبتنی بر جمعیت است که در آن موقعیت منبع غذایی، یک راه‌حل ممکن برای مسئله‌ی بهینه‌سازی را نشان می‌دهد و مقدار شهد یک منبع غذایی باکیفیت راه‌حل تخصیصی ارتباط مستقیم دارد [33]. تعداد زنبورهای شاغل برابر با تعداد راه‌حل‌های جمعیت است. در مرحله‌ی اول، یک جمعیت اولیه‌ی تصادفی که همان موقعیت‌های منابع غذایی هستند تولید می‌شود. پس از مقداردهی اولیه، جمعیت به‌این‌ترتیب در معرض چرخه‌ی فرآیندهای جست‌وجوی زنبورهای شاغل، تماشاگر و پیشاهنگ قرار می‌گیرد. یک زنبور شاغل به‌شرط که مقدار شهد منبع جدید بیشتر از منبع قبلی باشد، در حافظه‌ی خود تغییری در موقعیت منبع ایجاد کرده و موقعیت منبع غذایی جدید را کشف می‌کند. در این حالت، زنبور موقعیت منبع جدید را به خاطر می‌سپارد و شهد قدیمی را فراموش می‌کند. بعدازاینکه تمامی زنبورهای استخدام‌شده فرآیند جست‌وجو را کامل کردند، اطلاعات موقعیت منابع را با تماشاگران در منطقه‌ی رقص به اشتراک می‌گذارند. زنبور به شرطی که شهدش از شهد قبلی بالاتر باشد، موقعیت جدید را حفظ و شهد قبلی را فراموش می‌کند.

در الگوریتم ABC، نیمه‌ی اول جمعیت شامل زنبورهای شاغل و نیمه‌ی دوم زنبورهای تماشاگر است. تعداد زنبورهای شاغل یا زنبورهای تماشاگر برابر با تعداد راه‌حل‌های موجود در جمعیت است. ABC یک جمعیت اولیه‌ی توزیع‌شده‌ی تصادفی از راه‌حل‌ها (منابع غذایی) تولید می‌کند. فرض کنید $X_i = \{x_{i,1}, x_{i,2}, \dots, x_{i,n}\}$ نشان‌دهنده‌ی i-امین راه‌حل در جمعیت و n اندازه‌ی بعد باشد [۳۳].

هر زنبور به کار گرفته‌شده‌ی X_i یک راه‌حل کاندید جدید V_i را در همسایگی موقعیت فعلی خود به‌صورت رابطه‌ی ۱ تولید می-کند:

$$v_{i,k} = x_{i,k} + \Phi_{i,k} * (x_{i,k} - x_{j,k}) \qquad (۱)$$

به‌گونه‌ای که X_j یک راه‌حل کاندید تصادفی، k یک شاخص بعد تصادفی انتخابی از مجموعه‌ی $\{1,2, \dots, n\}$ و $\Phi_{i,k}$ یک عدد تصادفی در بازه‌ی $[-1,1]$ است.

زمانی که راه‌حل کاندید جدید V_i تولید می‌شود، یک انتخاب حریصانه استفاده می‌شود. اگر ارزش تناسب V_i بهتر از والد خود یعنی X_i باشد، در آن صورت X_i با V_i به‌روزرسانی می‌شود، در غیر این صورت X_i بدون تغییر می‌ماند. پس‌ازآنکه تمامی زنبورهای استخدام‌شده، فرآیند جست‌وجو را کامل کردند، اطلاعات منابع غذایی خود را از طریق رقص با زنبورهای ناظر به اشتراک می‌گذارند. زنبور ناظر، اطلاعات شهد گرفته‌شده از زنبورهای شاغل را ارزیابی می‌کند و منبع غذایی را با احتمال مرتبط با مقدار شهد آن انتخاب می‌کند. این انتخاب احتمالی بر مبنای انتخاب چرخ رولت است که به‌صورت رابطه‌ی ۲ نشان داده می‌شود:

$$P_i = \frac{fit_i}{\sum_j fit_j} \qquad (۲)$$

که در آن fit_i مقدار تناسب i-امین راه‌حل در جمعیت است. هرچه راه‌حل بهتر باشد احتمال انتخاب آن به‌عنوان منبع غذایی انتخاب‌شده بیشتر است. اگر موقعیتی را نتوان پس از تعدادی چرخه بهبود بخشید منبع غذایی آن موقعیت رها می‌شود. فرض کنید که منبع رهاشده X_i باشد، در آن صورت زنبور پیشاهنگ منبع غذایی جدیدی را کشف می‌کند [۳۳] که باید با رابطه‌ی ۳ جایگزین شود:

$$x_{i,k} = lb_k + \Phi_{i,k} * (ub_k - lb_k) \quad (۳)$$

به‌گونه‌ای که $\Phi_{i,k} = rand(0,1)$ یک عدد تصادفی در بازه‌ی $[-1,1]$ و lb_k و ub_k به‌این‌ترتیب نشان‌دهنده‌ی حد پایین و بالای بعد k-ام هستند.

مراحل اصلی الگوریتم در ادامه آورده شده است:

۱- منابع غذایی اولیه برای تمام زنبورهای شاغل تولید می‌شود.

۲- مراحل زیر را تکرار کن:

الف- هر زنبور شاغل به خاطر خود به سراغ منبع غذایی می‌رود و نزدیک‌ترین منبع را تعیین می‌کند، سپس مقدار شهد آن را ارزیابی کرده و در کندو می‌رقصد.

ب- هر بیننده‌ی رقص زنبورهای شاغل را تماشا و بسته به رقص یکی از منابع آن‌ها را انتخاب می‌کند و سپس به آن منبع می‌رود. پس از انتخاب همسایه در اطراف آن، مقدار شهد آن را ارزیابی می‌کند.

ج- منابع غذایی رهاشده مشخص‌شده و با منابع غذایی جدید کشف‌شده توسط پیشاهنگان جایگزین می‌شوند.

د- بهترین منبع غذایی یافت شده تاکنون ثبت‌شده است.

۳- تا زمانی که (شرایط موردنیاز برآورده شود)

3-2-2. الگوریتم کرم شب‌تاب

تعاملات اجتماعی کرم شب‌تاب به‌عنوان مبنایی برای این روش بهینه‌سازی فرا ابتکاری عمل می‌کند که در سال 2007 در دانشگاه کمبریج توسط دکتر ژین‌شه یانگ ایجاد شد.

این روش بر اساس رفتار جمعیت دیده‌شده در طبیعت برای موجوداتی مانند ماهی، حشرات و پرندگان بناشده است [۳۴]. الگوریتم‌های هوش ازدحامی مانند الگوریتم پرندگان، بهینه‌سازی کلونی زنبورهای مصنوعی و الگوریتم‌های جست‌وجوی باکتری شباهت‌های زیادی با الگوریتم کرم شب‌تاب دارند.

درروش کرم شب‌تاب از اعداد تصادفی واقعی استفاده می‌شود که بر اساس توانایی ذرات ازدحامی در برقراری ارتباط بنانهاده شده است و ازلحاظ بهینه‌سازی چندهدفه مؤثرتر عمل می‌کند. این الگوریتم از سه قانون بر پایه ویژگی‌های چشمک‌زن شب‌تاب‌های واقعی تشکیل‌شده است که در زیر توضیح داده‌شده است:

۱- همه کرم‌های شب‌تاب تک جنسیتی هستند. آن‌ها صرف‌نظر از جنسیت خود به سمت افراد جذاب‌تر و درخشان تر حرکت می‌کنند.

۲- میزان جذب کرم شب‌تاب با درخشندگی آن متناسب است. این امر با افزایش فاصله از سایر کرم‌های شب‌تاب کاهش می‌یابد، زیرا هوا نور را جذب می‌کند. اگر کرم شب‌تاب درخشان تر یا جذاب‌تر از یک کرم شب‌تاب وجود نداشته باشد، به‌طور تصادفی حرکت می‌کند.

۳- روشنایی یا شدت نور یک کرم شب‌تاب با مقدار تابع هدف یک مسئله مشخص تعیین می‌شود.

در مسائل بهینه‌سازی حداکثر سازی، روشنایی I یک کرم شب‌تاب در یک مکان خاص x را می‌توان به‌صورت $I(x) \propto f(x)$ نشان داد. جذابیت β بافاصله‌ی $r(i,j)$ بین کرم شب‌تاب i و کرم شب‌تاب j تغییر می‌کند. علاوه بر این، شدت نور بافاصله از منبع کاهش می‌یابد. بنابراین جذابیت با درجه‌ی جذب تغییر می‌کند.

شدت نور به‌صورت رابطه‌ی ۴ است که در آن I_0 شدت نور در منبع، r فاصله‌ی بین دو کرم شب‌تاب، و γ ثابت جذب نور برای یک محیط مشخص است [۳۴].

$$I(r) = I_0 e^{-\gamma r^2} \quad (۴)$$

میزان جذب بین دو کرم شب‌تاب به‌صورت رابطه‌ی ۵ است که در آن β_0 میزان جذب برای $r = 0$ است.

$$\beta = \frac{\beta_0}{1+\gamma r^2} \quad (۵)$$

فاصله‌ی بین دو کرم شب‌تاب را می‌توان بافاصله‌ی دکارتی به‌دست‌آمده آورد به‌گونه‌ای که $x(i,k)$، k-امین جزء مختصات فضایی x_i مربوط به i-امین کرم شب‌تاب است (رابطه‌ی ۶).

$$r_{i,j} = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \quad (۶)$$

حرکت یک کرم شب‌تاب به سمت کرم شب‌تاب دیگر به شرح رابطه‌ی ۷ است. جمله‌ی دوم مربوط به جذابیت کرم شب‌تاب و جمله‌ی سوم مربوطه به تصادفی سازی است که در آن α پارامتر تصادفی سازی و ε_i بردار اعداد تصادفی است که برگرفته از توزیع گاوسی یا توزیع یکنواخت است.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right) + \alpha \varepsilon_i^t \quad (۷)$$

3-2-3. الگوریتم رقابت استعماری

الگوریتم رقابت استعماری نوعی روش محاسباتی است که برای حل انواع مختلف مسائل بهینه‌سازی استفاده می‌شوند. این روش نیز مانند بسیاری از روش‌ها در حوزه‌ی محاسبات تکاملی، در فرآیند بهینه‌سازی خود نیازی به گرادیان تابع ندارد. از یک دیدگاه خاص، رقابت استعماری را می‌توان به‌عنوان همتای اجتماعی الگوریتم ژنتیک در نظر گرفت. رقابت استعماری، مدل ریاضی و شبیه‌سازی کامپیوتری تکامل اجتماعی انسان است، درحالی‌که الگوریتم ژنتیک بر اساس تکامل بیولوژیکی گونه‌ها است.

این الگوریتم با تولید مجموعه‌ای از راه‌حل‌های تصادفی کاندید در فضای جست‌وجوی مسئله بهینه‌سازی شروع می‌شود. نقاط تصادفی تولیدشده را کشورهای اولیه می‌نامند. کشورها در این الگوریتم همتای کروموزوم‌ها در الگوریتم ژنتیک و همانند ذرات

در بهینه‌سازی ازدحام ذرات بوده و عملاً آرایه‌ای از مقادیر یک راه‌حل به‌عنوان نامزد مسئله بهینه‌سازی هستند. تابع هزینه‌ی مسئله‌ی بهینه‌سازی، قدرت هر کشور را تعیین می‌کند که بر اساس این قدرت، برخی از بهترین کشورهای اولیه (کشورهایی با کمترین ارزش تابع هزینه)، استعمارگر می‌شوند و شروع به کنترل سایر کشورها (به نام مستعمره‌ها) می‌کنند و امپراتوری-های اولیه را تشکیل می‌دهند [35].

دو عملگر اصلی این الگوریتم جذب و انقلاب هستند. جذب باعث می‌شود که مستعمره‌های هر امپراتوری در فضای ویژگی‌های سیاسی-اجتماعی یا همان فضای جست‌وجوی بهینه‌سازی به دولت استعمارگر نزدیک شوند. انقلاب، تغییرات تصادفی ناگهانی را در موقعیت برخی از کشورها در فضای جست‌وجو ایجاد می-کند. در طول جذب و انقلاب، یک مستعمره ممکن است به موقعیت بهتری برسد و این شانس را داشته باشد که کنترل کل امپراتوری را در دست بگیرد و جایگزین دولت استعمارگر کنونی در امپراتوری شود.

رقابت استعماری بخش دیگری از این الگوریتم است. همه‌ی امپراتوری‌ها سعی می‌کنند در این بازی پیروز شوند و مستعمره-های امپراتوری‌های دیگر را در اختیار بگیرند. در هر مرحله از الگوریتم، همه‌ی امپراتوری‌ها بر اساس قدرت خود این شانس رادارند که کنترل یک یا چند مستعمره‌ی ضعیف‌ترین امپراتوری را در دست بگیرند.

الگوریتم رقابت استعماری به‌طور دقیق‌تر شامل مراحل زیر است:

۱- تعداد N_{imp} امپراتوری را بر روی N_{pop} کشور به‌طور تصادفی مقداردهی اولیه کنید. وضعیت تمام کشورها را ارزیابی کنید و بهترین امپراتوری را به‌عنوان استعمارگر معرفی کنید [35].

۲- برای هر امپراتوری موارد زیر را تکرار کنید.

الف- برای هر مستعمره در امپراتوری فعلی، مراحل ب و ج را انجام دهید.

ب- مرحله‌ی اول راه رفتن: مستعمره‌ی کنونی را در جهت بین مستعمره و استعمارگر مربوطه‌اش مطابق رابطه‌های ۸ و ۹ حرکت دهید.

$$X_{c,new} = X_c + X_{new} \quad (۸)$$

$$V_{new} = (\beta * rand - 1) + (X_{imp} - X_c) \quad (۹)$$

به‌گونه‌ای که در آن، X_c و $X_{c,new}$ به ترتیب بیانگر موقعیت فعلی و جدید مستعمره هستند. β پارامتر کنترلی بزرگ‌تر از یک و X_{imp} موقعیت امپراتوری است.

ج- استعمارگر را در هر امپراتوری با شناسایی مستعمره‌ی آزمایشی آن به‌روزرسانی کنید.

۳- قدرت کل و قدرت هر امپراتوری را با استفاده از رابطه‌های ۱۰ الی ۱۲ محاسبه کنید.

$$T_{imp} = fitness(X_{imp}) + k * \frac{\sum_{i=1}^{N_{imp}} fitness(X_{imp})}{N_{imp}} \quad (10)$$

$$NT_{imp} = T_{imp} - max(T_{imp}) \quad (11)$$

$$P_{imp} = \left| \frac{NT_{imp}}{\sum_{i=1}^{N_{imp}} NT_{imp}} \right| \quad (12)$$

4- بدترین مستعمره را از ضعیف‌ترین امپراتوری به امپراتوری که بیشترین احتمال را بر اساس P_{imp} دارد منتقل کنید.

5- امپراتوری‌های ناتوان را از بین ببرید.

6- حلقه را از مرحله‌ی 2 تا زمانی که معیار خاتمه برآورده شود، یعنی تکمیل تعداد تعیین‌شده‌ی ارزیابی عملکرد تکرار کنید.

3-2-4. الگوریتم خفاش

این الگوریتم از رفتار پژواک یابی خفاش‌ها با نرخ‌های مختلف پالس انتشار و بلندی صدا الهام گرفته‌شده است. ایدئال‌سازی پژواک خفاش‌ها را می‌توان به‌صورت زیر خلاصه کرد: هر خفاش مجازی به‌طور تصادفی با سرعت v_i در موقعیت (راه‌حل) x_i با فرکانس یا طول‌موج و بلندی صدای متغیر A_i پرواز می‌کند. خفاش با جست‌وجو و یافتن طعمه‌ی خود، فرکانس، بلندی صدا و نرخ انتشار پالس r را تغییر می‌دهد. جست‌وجو با یک پیاده‌روی تصادفی محلی تشدید می‌شود. انتخاب بهترین‌ها تا زمانی که معیارهای توقف مشخصی برآورده شود ادامه می‌یابد. این روش اساساً از یک راهکار تنظیم فرکانس برای کنترل رفتار پویای گروهی از خفاش‌ها استفاده می‌کند و تعادل بین اکتشاف و بهره-برداری را می‌توان با تنظیم پارامترهای وابسته در الگوریتم خفاش کنترل کرد. با این توضیحات، الگوریتم خفاش با سه قانون زیر توسعه داده‌شده است [36]:

1- تمام خفاش‌ها از پژواک برای حس کردن فاصله استفاده می‌کنند و همچنین تفاوت بین مواد غذایی یا شکار و موانع پس‌زمینه را به روشی جادویی می‌دانند.

2- خفاش‌ها به‌طور تصادفی با سرعت v_i در موقعیت x_i با فرکانسf_{min} ، طول‌موج متغیر λ و بلندی صدای A برای جست-وجوی طعمه پرواز می‌کنند. آن‌ها می‌توانند به‌طور خودکار طول‌موج (یا فرکانس) پالس‌های ساطع‌شده‌ی خود را تنظیم کنند و نرخ ساطع شدن پالس را بسته به نزدیکی هدفشان تنظیم کنند.

3- اگرچه بلندی صدا می‌تواند از بسیاری جهات متفاوت باشد، ما فرض می‌کنیم که بلندی صدا از یک مقدار بزرگ مثبت تا یک مقدار ثابت حداقل A_{min} متغیر است.

برای تشریح بهتر الگوریتم خفاش باید گفت این الگوریتم با مقداردهی اولیه‌ی جمعیت خفاش‌ها در فضای جست‌وجوی n بعدی شروع می‌شود که در آن موقعیت خفاش i با x_i^t و سرعت آن با v_i^t در زمان نشان داده می‌شود. بنابراین، موقعیت‌های جدید x_i^{t+1} و سرعت‌های جدید v_i^{t+1} در مرحله‌ی زمانی t+1 را می‌توان به‌صورت زیر تعیین کرد [36]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (13)$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) \quad (14)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (15)$$

به‌گونه‌ای که $\beta \in [0,1]$ یک بردار تصادفی است که از توزیع یکنواخت به‌دست‌آمده است. در اینجا x_* بهترین مکان سراسر جاری (راه‌حل) است که پس از مقایسه‌ی تمام راه‌حل‌ها در بین کل n خفاش محاسبه می‌گردد. f_i نشان‌دهنده‌ی فرکانس پالس ساطع‌شده توسط خفاش i در لحظهd جاری است و f_{min} و f_{max} به ترتیب حداقل و حداکثر مقدار فرکانس پالس را نشان می‌دهند.

پیاده‌روی تصادفی را می‌توان به‌عنوان فرآیندی از جست‌وجوی محلی تشریح کرد که راه‌حل جدیدی را توسط راه‌حل انتخاب‌شده ایجاد می‌کند. یک خفاش به‌طور تصادفی از جمعیت خفاش‌ها انتخاب می‌شود. سپس موقعیت مربوط به این خفاش مطابق رابطه‌ی 16 به‌روزرسانی می‌شود.

$$x_{new} = x_{old} + \varepsilon A^{(t)} \quad (16)$$

به‌گونه‌ای که در آن $\epsilon\varepsilon[0,1]$ یک عدد تصادفی است، درحالی‌که $A^{(t)}$ میانگین بلندی صدای تمام خفاش‌ها در مرحله‌ی زمانی t است.

بلندی A_i و نرخ r_i مربوط به انتشار پالس باید مطابق با تکرارها به‌روزرسانی شوند که این موارد مسئول متعادل کردن ترکیب بین حرکات محلی و سراسری هستند. در ابتدا، بلندی صدا قوی است و ضربان نبض کم است. هنگامی‌که خفاش طعمه‌ی خود را به‌دست‌آمده می‌آورد صدا کاهش می‌یابد، درحالی‌که ضربان نبض افزایش می‌یابد (رابطه‌های 17 و 18).

$$A_i^{t+1} = \alpha A_i^t \quad (17)$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \quad (18)$$

به‌گونه‌ای که α و γ برای هر $0 < \alpha < 1$ و $\gamma > 0$ ثابت هستند. می‌توانیم فرض کنیم که $A_i = A_{min} = 0$ به این معنی است که خفاش به‌تازگی طعمه‌ای پیداکرده و به‌طور موقت صدا را متوقف کرده است (رابطه‌ی 19).

$$A_i^t \to 0, r_i^t \to r_i^0, as\ t \to \infty \quad (19)$$

۴. راهکار پیشنهادی

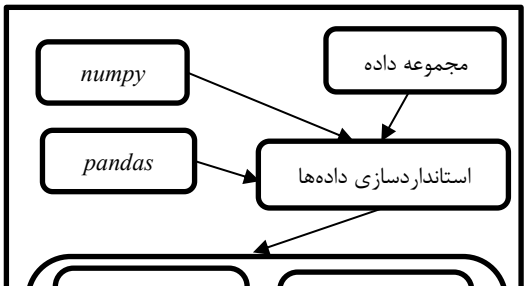
در این بخش به نحوه‌ی استفاده از روش‌های بیان‌شده در این مقاله به‌عنوان راهکار پیشنهادی برای تشخیص بدافزار می‌پردازیم. سیستم پیشنهادی شامل دو بخش اساسی است: بخش اول سیستم شامل استفاده از الگوریتم‌های فرا ابتکاری است که بر روی مجموعه داده اعمال‌شده است و مقادیر پارامترهای کلیدی شبکه‌ی یادگیر عمیق را برای آموزش و آزمایش فراهم می‌کند. بخش دوم شامل سیستم اصلی تشخیص بدافزار است که بر مبنای شبکه یادگیری عمیق طراحی‌شده است. در این بخش هر دو روش یادگیری عمیق شبکه‌ی عصبی پیچشی و حافظه‌ی طولانی کوتاه‌مدت برای به‌دست‌آمده آوردن بهترین نتیجه استفاده‌شده‌اند. عملیات پیاده‌سازی با استفاده از کتابخانه‌های معروف پایتون به نام‌های TensorFlow و ScikitLearn و بر روی سامانه‌ای با پردازنده 5.3 گیگاهرتز و رم 8 گیگابایت انجام‌شده است که در بخش نتایج به آن می‌پردازیم.

ساختار و نمای کلی روش پیشنهادی در شکل 1 قابل‌مشاهده است. همان‌گونه که در این شکل می‌بینید ابتدا الگوریتم‌های فرا ابتکاری بر روی مجموعه داده اعمال می‌شوند تا پارامترهای دقیق شبکه یادگیر عمیق به‌دست‌آمده بیابند. در ادامه با اعمال دو روش یادگیری عمیق، نتایج مربوط به سیستم تشخیص بدافزار به‌عنوان خروجی به‌دست‌آمده می‌آیند.

همان‌گونه که در بالا ذکر شد روش پیشنهادی از دو بخش اصلی تشکیل‌شده است. بنابراین، در این بخش به تشریح این دو بخش می‌پردازیم.

4-1. بخش اول: الگوریتم فرا ابتکاری

عملکرد کلی روش‌های فرا ابتکاری به این صورت است که ابتدا یک جمعیت اولیه‌ی تصادفی تولید می‌شود. سپس تابع برازش بر روی آن‌ها اعمال می‌شود تا بهترین‌ها از این جمعیت انتخاب شوند. نکته‌ی قابل‌توجه این است که بهتر بودن در اینجا به معنای خطای کمتر درروش یادگیری عمیق است؛ یعنی هر چه یک فرد از جمعیت منجر به خطای کمتری در تابع یاد گیر شود آن فرد ازلحاظ برازش در مرتبه‌ی بهتری قرار دارد و در اولویت انتخاب برای مراحل بعدی قرار می‌گیرد. پس از انتخاب بهترین‌ها، جمعیت وارد فازهای بعدی یعنی بازترکیب، جهش، حرکت به‌سوی نقاط بهینه و موارد دیگر می‌شود. این روند ادامه می‌یابد و نسل جدید تولید می‌شود و بهترین‌های آن جایگزین نسل قبلی می‌شود. این فرآیند آن‌قدر تکرار می‌شود تا دیگر نتیجه‌ی بهتری حاصل نشود.



شکل (1). فلوچارت روش پیشنهادی

در پایان این مرحله، بهترین فرد از جمعیت به‌عنوان ورودی (پارامتر تنظیمی) برای مرحله‌ی دوم که یادگیری عمیق است ارسال می‌گردد. شایان‌ذکر است که با توجه به اینکه دو روش مختلف یادگیری عمیق در این مقاله استفاده‌شده است، بنابراین دو مقدار به‌عنوان بهترین مقدار جمعیت انتخاب می‌شود.

4-2. بخش دوم: یادگیری عمیق

با توجه به اینکه عملکرد شبکه‌ی عمیق وابستگی زیادی به مقادیر ابر پارامترها دارد، بنابراین وجود مرحله‌ی قبلی برای دستیابی به بهترین نتایج در مرحله‌ی یادگیری الزامی است. در این مرحله، عملیات یادگیری با توجه به مقادیر متفاوت ابر پارامترها در دو روش شبکه‌ی عصبی پیچشی و حافظه‌ی طولانی کوتاه‌مدت با در نظر گرفتن نمونه داده‌های ورودی و مقادیر به‌دست‌آمده صورت می‌گیرد.

در اینجا در مورد ابر پارامترهای کلیدی که باید در طراحی شبکه‌های یادگیر عمیق در نظر گرفته شوند بحث می‌کنیم. یکی از این ابر پارامترها تعداد لایه‌ها است. تعداد لایه‌ها در CNN یک ابر پارامتر حیاتی است که عمق شبکه را تعیین می‌کند. یک شبکه‌ی عمیق‌تر می‌تواند ویژگی‌ها و الگوهای پیچیده‌تری را از داده‌ها بیاموزد، اما از طرفی دیگر، مستعد بیش‌ازحد برازش است. بنابراین، ایجاد تعادل بین تعداد لایه‌ها و پیچیدگی مسئله‌ی بسیار مهمی است. یک نقطه‌ی شروع خوب، استفاده از تعداد کمی لایه و افزایش تدریجی عمق آن‌ها تا رسیدن به عملکرد مطلوب است. این موضوع با آزمایش روش‌های فرا ابتکاری بررسی می‌شود.

اندازه‌ی فیلتر، یکی دیگر از ابر پارامترهای مهم است که میدان هر لایه‌ی کانولوشن را تعیین می‌کند. فیلتر بزرگ‌تر می‌تواند اطلاعات بیشتری را از ورودی بگیرد، اما تعداد پارامترهای شبکه را نیز افزایش می‌دهد. فیلتر کوچک‌تر می‌تواند تعداد پارامترها را کاهش دهد، اما ممکن است نتواند تمام ویژگی‌های مربوطه را ثبت کند. بنابراین، آزمایش با اندازه‌های مختلف فیلتر و انتخاب فیلتری که بهترین عملکرد را دارد مهم است.

اندازه‌ی دسته یک ابر پارامتر است که تعداد نمونه‌هایی را که توسط شبکه در هر تکرار آموزشی پردازش می‌شوند تعیین می‌کند. دسته‌ی بزرگ‌تر می‌تواند واریانس برآوردهای گرادیان را کاهش دهد و پایداری آموزش را بهبود بخشد. بااین‌حال، نیاز به حافظه را نیز افزایش می‌دهد و ممکن است منجر به همگرایی کندتر شود. دسته‌ی کوچک‌تر می‌تواند نیاز به حافظه را کاهش دهد و سرعت همگرایی را بهبود بخشد، اما ممکن است منجر به تخمین گرادیان نویزی شود. بنابراین، مهم است که اندازه‌های مختلف دسته آزمایش شوند و از بین آن‌ها بهترین مبادله بین پایداری و سرعت مشخص گردد.

```
Phase 1: Data preprocessing with pandas and numpy (normalization and noise removing)
Phase 2: Meta-Heuristic Functions
1- initialize population (100 cases)
2- Fitness evaluation
3- for i from 1 to 100
        Fitness evaluation
        If new_fit > best_fit
            Best_fit = new_fit
Phase 3: CNN / LSTM
Tunning Phase: Train the network with different hyperparameters on the train section of dataset using scikitLearn to obtain best values
Testing Phase: Test on the test section of dataset and calculate evaluation metrics such as accuracy
```

شکل 2. شبه کد روش پیشنهادی

ابر پارامتر دیگر یعنی تعداد دوره‌ها تعیین می‌کند که چند تکرار کامل از مجموعه داده اجرا شود. درحالی‌که ازنظر تئوری، این عدد را می‌توان روی یک عدد صحیح بین یک و بی‌نهایت تنظیم کرد، این عدد باید تا زمانی افزایش یابد که دقت اعتبارسنجی شروع به کاهش کند، حتی اگر دقت آموزش افزایش یابد که در این صورت خطر بیش‌ازحد برازش وجود خواهد داشت. یک حرکت حرفه‌ای این است که از روش توقف زودهنگام استفاده کنیم تا ابتدا تعداد زیادی دوره‌ی آموزش را مشخص کنیم و زمانی که عملکرد مدل با یک آستانه‌ی از پیش تعیین‌شده در

مجموعه داده‌ی اعتبارسنجی بهبود نمی‌یابد آموزش را متوقف کنیم.

شبه کد مربوط به روش پیشنهادی در شکل 2 قابل‌مشاهده است. همان‌گونه که در این شکل مشاهده می‌شود در ابتدا الگوریتم‌های فرا ابتکاری انجام می‌شوند و الگوریتم وارد مرحله‌ی بعدی می‌شود. در این مرحله، یادگیری با توجه به پارامترهای مشخص‌شده در مرحله‌ی فرا ابتکاری به‌صورت تکراری صورت می‌پذیرد تا مدل با کمترین خطا برای دستیابی به بالاترین دقت ایجاد شود.

۵. نتایج آزمایش

در این بخش به ارائه‌ی نتایج شبیه‌سازی می‌پردازیم.

مجموعه داده‌ای که برای ارزیابی نتایج مورداستفاده قرارگرفته است UNSW-NB15 نام دارد. بسته‌های شبکه‌ی خام این مجموعه داده توسط ابزار IXIA PerfectStorm در آزمایشگاه Cyber Range در UNSW Canberra برای ایجاد ترکیبی از فعالیت‌های عادی واقعی و رفتارهای حمله‌ی مصنوعی ایجادشده است. ابزار tcpdump برای ضبط 100 گیگابایت از ترافیک خام (به‌عنوان‌مثال، فایل‌های Pcap) استفاده‌شده است. این مجموعه داده دارای 9 نوع حمله است که عبارت‌اند از Fuzzers، Analysis، Backdoors، DoS، Exploits، Generic، Reconnaissance، Shellcode و Worms. از ابزارهای Argus و Bro-IDS استفاده‌شده و دوازده الگوریتم برای تولید 49 ویژگی با برچسب کلاس توسعه داده‌شده است.

تعداد کل رکوردها 254044 است که در چهار فایل CSV ذخیره می‌شوند. بخشی از این مجموعه داده به‌عنوان یک مجموعه آموزشی و بخش دیگر در قالب مجموعه آزمایشی پیکربندی می‌شود. تعداد رکوردهای مجموعه‌ی آموزشی 175341 رکورد و مجموعه‌ی آزمایشی 82332 رکورد از انواع مختلف حمله و عادی است.

پارامترهایی که در الگوریتم‌ها فرا ابتکاری مورداستفاده قرارگرفته‌اند به همراه مقادیر آن به این شرح است: جمعیت اولیه 100، تعداد تکرار 100 مرتبه و تعداد بیت برای هر جمعیت 30 است. ابر پارامترها در شبکه‌های یادگیری عمیق نیز تعداد لایه‌ها شامل مقادیر 2 و 3، اندازه‌ی فیلتر بین 3*3 الی 5*5، تعداد دسته‌ها 20، 30 و 50 و تعداد دوره‌ها مقادیر 200، 300، 500 و 1000 در نظر گرفته‌شده است.

عملیات پیاده‌سازی روی سیستم با پردازنده Core-i7 و 3.5 گیگاهرتز و رم 8 گیگابایتی انجام‌شده است. برای انجام عملیات پیش‌پردازش مانند نرمال‌سازی و استانداردسازی از کتابخانه‌های پایتون به نام pandas و numpy استفاده‌شده است. برای آموزش و ارزیابی نتایج نیز از کتابخانه‌های scikit-learn و keras استفاده‌شده است. در گزارش‌گیری‌ها برای دستیابی به بهترین نتایج، میانگین 10 بار اجرای برنامه و همچنین میزان داده‌های آموزش از 60 الی 90 درصد و به همین ترتیب داده‌های آزمایش از 10 الی 40 درصد تغییر پیداکرده است.

۵-۱. ارزیابی نتایج

معیارهایی که در این مقاله به محاسبه و مقایسه آن‌ها پرداخته‌ایم به شرح زیر است:

الف) دقت (Accuracy): این معیار با استفاده از عملیات تقسیم تعداد نمونه‌هایی که به‌درستی طبقه‌بندی‌شده‌اند بر تعداد کل نمونه داده‌ها در مجموعه داده محاسبه می‌شود که در رابطه‌ی 20 نشان داده‌شده است.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (20)$$

ب) امتیاز F1 (F1 Score): میانگین هارمونیک پارامتر فراخوانی (Recall) و درستی (Precision) با استفاده از این معیار محاسبه می‌شود که در رابطه‌ی 21 نشان داده‌شده است.

$$F1\ Score = \frac{2*Precision*Recall}{Precision+Recall} \quad (21)$$

ج) نرخ هشدار نادرست (FAR): این معیار نشان‌دهنده‌ی تعداد نمونه‌های عادی است که اشتباهاً به‌عنوان داده‌ی حمله شناسایی‌شده‌اند و در رابطه 22 نشان داده‌شده است.

$$FAR = \frac{FP}{TN+FP} \quad (22)$$

د) نرخ منفی نادرست (FNR): تعداد نمونه داده‌های حمله که به‌طور نادرست به‌عنوان نمونه‌های عادی توسط مدل برچسب‌گذاری شده‌اند. رابطه‌ی 23 این معیار را نشان می‌دهد.

$$FNR = \frac{FN}{TP+FN} \quad (23)$$

ه) خاص بودن (Specificity): این معیار بیانگر نرخ منفی واقعی (TNR) بوده و نشان‌دهنده‌ی داده‌های عادی است که به طرز صحیح توسط مدل شناسایی‌شده‌اند و در رابطه‌ی 24 نشان داده‌شده است.

$$Specificity = \frac{TN}{TN+FP} \quad (24)$$

و) ضریب همبستگی ماتیو (MCC): برای ارزیابی تفاوت بین مقادیر پیش‌بینی‌شده و مقادیر واقعی از این معیار استفاده می‌شود (رابطه‌ی 25)

$$MCC = \frac{TN*TP-FN*FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (25)$$

ازآنجایی‌که هر نوع حمله‌ای رفتار متفاوتی با سایر حمله‌ها دارد و ازاین‌رو ترافیک متفاوتی در شبکه ایجاد می‌کند، بدیهی است که

برای دستیابی به‌دقت تشخیص بالا نیاز است تا برای هر یک از حملاتی که در شبکه اتفاق می‌افتد مقدار پارامترهای تنظیم‌شده و روش استفاده‌شده برای تشخیص آن نیز متفاوت باشد. در این بخش نیز با تغییر مقادیر پارامترها و درواقع به‌دست‌آمده آوردن دقیق‌ترین مقدار پارامترهای شبکه یادگیر عمیق از طریق الگوریتم‌های فرا ابتکاری به بالاترین دقت ممکن دست پیدا می‌کنیم.

از طرفی، معیار ارزیابی روش پیشنهادی می‌تواند بر پایه‌ی هر یک از 6 پارامتر معرفی‌شده در این بخش نیز باشد اما در اغلب موارد پارامتر دقت، مهم‌ترین معیار در اغلب مقالات پژوهشی در نظر گرفته می‌شود، هرچند که در این مقاله روی نتایج به‌دست‌آمده آمده از معیارهای دیگر نیز تمرکز می‌کنیم.

جدول 2 نتایج به‌دست‌آمده آمده از اعمال روش‌های مختلف فرا ابتکاری معرفی‌شده در مرحله‌ی اول و استفاده از شبکه‌ی یادگیر CNN را برای مقادیر مختلف ابر پارامترها (2 لایه، اندازه‌ی دسته 30، اندازه‌ی فیلتر 4*4 و تعداد دوره 500) نشان می‌دهد. همان‌طور که در این جدول مشاهده می‌شود بهترین مقادیر برای پارامتر دقت زمانی به‌دست‌آمده می‌آید که الگوریتم خفاش به‌عنوان تنظیم‌کننده‌ی پارامترهای شبکه‌ی یادگیر استفاده‌شده باشد. از طرفی، بهترین مقادیر معیار F1 Score و FAR زمانی حاصل‌شده است که الگوریتم زنبورعسل به کار گرفته‌شده است. همچنین ضریب همبستگی ماتیو نیز در حالت استفاده از الگوریتم کرم شب‌تاب منجر به نتایج بهتری نسبت به سایر روش‌ها شده است.

جدول (2). نتایج آزمایش با استفاده از CNN شامل 2 لایه، اندازه‌ی دسته 30، اندازه‌ی فیلتر 4*4 و تعداد دوره 500

معیار	زنبورعسل	رقابت استعماری	کرم شب‌تاب	خفاش
دقت	98.438	97.743	98.037	99.984
امتیاز F1	96.467	95.994	96.293	95.284
نرخ هشدار نادرست	0.063	0.074	0.085	0.076
نرخ منفی نادرست	0.047	0.051	0.044	0.034
خاص بودن	99.382	98.274	99.483	99.692
ضریب همبستگی ماتیو	93.274	94.938	95.263	91.883

نکته‌ی مهمی که از این نتایج حاصل می‌شود این است که اولاً، این روش ترکیبی می‌تواند نقطه‌ی اتکای بسیار خوبی برای تعیین اینکه از کدام نوع الگوریتم فرا ابتکاری استفاده کنیم باشد، زیرا بسته به اینکه کدام معیار ارزیابی برای ما اهمیت بیشتری دارد (اینکه صرفاً به‌دقت روش بسنده کنیم و یا تعداد برچسب‌گذاری‌های نادرست داده‌های عادی مدنظر باشد) تنوع و حق انتخاب برای مقابله با حملات بدافزار امکان‌پذیر است. ثانیاً، نتایج به‌دست‌آمده آمده در حالت کلی برای تمام روش‌های استفاده‌شده در پارامترهای کلیدی و اثرگذار مانند دقت، همگی نتایج بسیار خوبی را تولید کرده‌اند و این موضوع نشان از قابلیت اعتماد روش پیشنهادی دارد.

جدول (3). نتایج آزمایش با استفاده از CNN، 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 3*3 و تعداد دوره 300

معیار	زنبورعسل	رقابت استعماری	کرم شب‌تاب	خفاش
دقت	99.904	99.915	99.920	99.978
امتیاز F1	96.489	95.045	94.004	96.310
نرخ هشدار نادرست	0.074	0.083	0.081	0.093
نرخ منفی نادرست	0.088	0.083	0.081	0.076
خاص بودن	97.403	98.381	98.541	96.329
ضریب همبستگی ماتیو	93.746	93.279	95.309	96.427

جدول 3، نتایج ارزیابی را با در نظر گرفتن 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 3*3 و تعداد دوره 300 برای همان الگوریتم استفاده‌شده در جدول 2 نشان می‌دهد. نتایج به‌دست‌آمده آمده، تغییرات چندانی را نسبت به جدول 3 نشان نمی‌دهد و تقریباً همان نتایج با اختلاف اندک را نمایش می‌دهد. سایر نتایج شبکه‌ی عصبی پیچشی برای حالت‌های مختلف مقادیر ابر پارامترها به دلیل کسب نتایج ضعیف‌تر نشان داده نشده است.

مشابه همین موضوع در جدول 4 برای شبکه‌ی یادگیر LSTM با در نظر گرفتن 2 لایه، اندازه‌ی دسته 50، اندازه‌ی فیلتر 4*4 و تعداد دوره 500 مشاهده می‌شود. همان‌طور که در این جدول مشاهده می‌کنید بهترین میزان دقت برای شبکه‌ی LSTM زمانی به‌دست‌آمده آمده است که از الگوریتم فرا ابتکاری زنبورعسل استفاده می‌شود. همچنین مقدار معیارهای خاص بودن و نرخ منفی نادرست، زمانی که از رقابت استعماری استفاده می‌شود بهترین نتایج به‌دست‌آمده می‌آید.

اما در جدول 5 که نتایج حاصل از همان الگوریتم‌های به کار گرفته‌شده در جدول 4 با 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 5*5 و تعداد دوره 300 را نشان می‌دهد نتایج بسیار بهتری نسبت به‌کل جداول 1 الی 3 به‌دست‌آمده آمده است. همان‌طور که در جدول 5 مشاهده می‌کنید به‌جز پارامتر امتیاز F1، تمامی معیارها در حالتی که الگوریتم کرم شب‌تاب به‌عنوان الگوریتم فرا ابتکاری استفاده‌شده است نسبت به سایر روش‌ها نتایج بهتری را تولید می‌کند. نتایج به‌دست‌آمده از کلیه حالات دیگر برتری دارد.

با مقایسه‌ی نتایج به‌دست‌آمده آمده از جداول 1 الی 4، این نتیجه حاصل می‌شود که بالاترین میزان دقت مربوط به زمانی است که از الگوریتم فرا ابتکاری کرم شب‌تاب و شبکه‌ی یادگیر LSTM با 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 5*5 و تعداد دوره 300 استفاده‌شده است.

جدول 6، مقایسه‌ای بین بهترین نتایج به‌دست‌آمده آمده از دقت روش پیشنهادی با سایر روش‌های ارائه‌شده درزمینه‌ی تشخیص بدافزار را نشان می‌دهد. نتایج این جدول نشان می‌دهد که روش پیشنهادی، بهترین عملکرد را ازلحاظ دقت نسبت به سایر روش‌های مطرح‌شده دارد.

جدول (4). نتایج آزمایش با استفاده از LSTM شامل 2 لایه، اندازه‌ی دسته 50، اندازه‌ی فیلتر 4*4 و تعداد دوره 500

معیار	زنبورعسل	رقابت استعماری	کرم شب‌تاب	خفاش
دقت	99.992	99.833	99.281	99.103
امتیاز F1	98.537	98.258	97.830	97.372
نرخ هشدار نادرست	0.017	0.022	0.031	0.028
نرخ منفی نادرست	0.018	0.011	0.019	0.021
خاص بودن	98.285	99.831	99.733	99.385
ضریب همبستگی ماتیو	98.629	97.844	98.205	99.284

جدول (5). نتایج آزمایش با استفاده از *LSTM* شامل 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 5*5 و تعداد دوره 300

معیار	زنبورعسل	رقابت استعماری	کرم شب‌تاب	خفاش
دقت	99.840	99.292	99.997	99.937
امتیاز *F1*	99.387	99.382	98.474	99.103
نرخ هشدار نادرست	0.006	0.007	0.002	0.005
نرخ منفی نادرست	0.008	0.010	0.004	0.008
خاص بودن	99.837	99.933	99.983	99.968
ضریب همبستگی ماتیو	99.731	98.994	99.735	99.039

جدول (6). مقایسه‌ی دقت روش پیشنهادی با برخی روش‌های دیگر

روش	دقت
کاهش ویژگی‌ها و بیز ساده [14]	86
خوشه‌بندی و درخت تصمیم [16]	97
شبکه باور عمیق [18]	98
جنگل تصادفی و شبکه عصبی مصنوعی [21]	98.94
شبکه عصبی پیچشی و شبکه طولانی کوتاه‌مدت [23]	99.958
k نزدیکترین همسایگی ترکیبی [24]	97.36
شبکه عصبی و درخت تصمیم [25]	95.39
ماشین بردار پشتیبان و شبکه عصبی کم‌عمق [26]	97.62
الگوریتم پرندگان و شبکه بازگشتی [28]	97.06
شبکه CNN [31]	99.4
شبکه عصبی بازگشتی عمیق [19]	99.5
روش پیشنهادی (کرم شب‌تاب و *LSTM*)	99.997

۶. نتیجه‌گیری

تشخیص بدافزار با دقت بالا، یکی از چالش‌های اساسی در حوزه‌ی تشخیص نفوذ و امنیت به‌شمار می‌رود. به‌طور جزئی‌تر، تنظیم دقیق پارامترهای شبکه‌ی یادگیر عمیق، موضوعی است که به‌سادگی قابل حل نیست. استفاده از الگوریتم‌های فرا ابتکاری و تنظیم پارامترهای شبکه‌ی یادگیر با استفاده از آن‌ها راه‌حل مناسبی است که اگر به‌درستی به کار گرفته شود می‌تواند نتایج بسیار خوبی را تولید نماید. در این مقاله، از چهار روش فرا ابتکاری زنبورعسل، رقابت استعماری، خفاش و کرم شب‌تاب برای تنظیم پارامترها و همچنین در نظر گرفتن مقادیر مختلف ابر پارامترها برای دو شبکه‌ی یادگیر *CNN* و *LSTM* استفاده‌شده است. نتایج شبیه‌سازی‌ها نشان می‌دهد که استفاده از الگوریتم‌های ذکرشده می‌تواند برای شناسایی بدافزارها نتایج خوبی را فراهم نماید که از بین تمام حالت‌های بررسی‌شده، شبکه‌ی یادگیر *LSTM* شامل 3 لایه، اندازه‌ی دسته 40، اندازه‌ی فیلتر 5*5 و تعداد دوره‌ی 300 و استفاده از الگوریتم کرم شب‌تاب بهترین نتیجه برای معیار دقت یعنی ۹۹.۹۹۷ درصد را به دنبال دارد. همچنین، سایر پارامترهای ارزیابی کار آیی روش پیشنهادی ازجمله امتیاز *F1*، خاص بودن و ضریب همبستگی ماتیو به ترتیب دارای مقادیر ۹۹.۳۸۲، ۹۹.۹۸۳ و ۹۹.۷۳۵ است که نشان‌دهنده‌ی برتری روش پیشنهادی نسبت به سایر کارهای موجود است.

مراجع

[1] M .Khrram, and M. Rahmanimanesh, “DDoS attack detection using ensemble method classification and active learning approach,” Electronic and Cyber Defense, vol. 11, pp. 101-118, 2024, [In Persian]dor: 20.1001.1.23224347.1402.11.3.10.5.

[2] H. Tanha, and M. Abbasi, “Identify malicious traffic on IoT infrastructure using neural networks and deep learning” Electronic and Cyber Defense, vol. 11, pp. 1-13, 2023, [In Persian]. dor: 20.1001.1.23224347.1402.11.2.1.4

[3] M. Ghanavati Nasab, M. Ghazvini, and F. Ghasemian, “Mobile botnets detection using deep learning techniques,” Electronic and Cyber Defense, vol. 11, pp. 31–43, 2023, dor: 20.1001.1.23224347.1402.11.2.3.6 [In Persian].

[4] A. Thakkar, and R. Lohiya, “A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges,” Archives of Computational Methods in Engineering, vol. 4, pp. 3211-3243, 2020. https://doi.org/10.1007/s11831-020-09496-0.

[5] R. Atefinia, and M. Ahmadi, “Network intrusion detection using multi-architectural modular deep neural network,” Journal of Supercomputing, vol. 4, pp. 3571–3593, 2021. https://doi.org/10.1007/s11227-020-03410-y.

[17] K. Kancherla, and S. Mukkamala, “Image visualization based malware detection,” In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, pp. 40–44, 2013. DOI: 10.1109/CICYBS.2013.6597204.

[18] L. Liu, and B. Wang, “Malware classification using gray-scale images and ensemble learning,” In Proceedings of the 3rd International Conference on Systems and Informatics (ICSAI), China, pp. 1018–1022, 2016. DOI:10.1109/ICSAI.2016.7811100.

[19] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, “Image-Based malware classification using ensemble of CNN architectures (IMCEC),” Computer Security, vol. 92, 101748, 2020. https://doi.org/10.1016/j.cose.2020.101748 .

[20] A.F. Agarap, and F.J.H. Pepito, “Towards building an intelligent anti-malware system a deep learning approach using support vector machine (SVM) for malware classification,” arXiv 2017, arXiv:1801.00318, 2017. https://doi.org/10.48550/arXiv.1801.00318.

[21] S.A. Roseline, S. Geetha, S. Kadry, and Y. Nam, “Intelligent vision-based malware detection and classification using deep random forest paradigm,” IEEE Access, vol. 8, pp. 206303-206324., 2024, DOI: https://doi.org/10.1109/ACCESS.2020.3036491.

[22] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, “Malware detection with deep neural network using process behavior,” in: 2016 IEEE 40th annual computer software and applications conference (COMPSAC), vol. 2, pp. 577-582, 2016, DOI: 10.1109/COMPSAC.2016.151.

[23] M. Akhtar, and T. Feng, “Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time,” Symmetry, vol. 14, pp. 2308, 2022, DOI: https://doi.org/10.3390/sym14112308.

[24] A. Al-Saaidah, M. Abualhaj, Q. Shambour, A. Abu-Shareha, L. Abualigah, S. Al-Khatib, and Y. Alraba’nah, “Enhancing malware detection performance: leveraging K-Nearest Neighbors with Firefly Optimization Algorithm,” Multimedia Tools and Applications, vol. 3, pp. 1-24, 2024, . DOI: https://doi.org/10.1007/s11042-024-18914-5.

[25] R. Damaševičius, A. Venčkauskas, J. Toldinas, and S. Grigaliūnas, “Ensemble-based classification using neural networks and machine learning models for windows PE malware detection,” Electronics, vol. 10, pp. 485, 2021, DOI: https://doi.org/10.3390/electronics10040485.

[26] I. Pranaou, S. Christy, and T. Poovizhi, “Implementation of ML Algorithm for Spyware Detection System Using SVM with KNN Algorithm for Comparison of Accuracy,” in: 2024 9th International Conference on Applying New Technology in Green Buildings (ATiGB), pp. 1-5, 2024, DOI: 10.1109/ATiGB63471.2024.10717756.

[27] Yerima, S. Sezer, and G. McWilliams, “Analysis of Bayesian classification-based approaches for Android malware detection,” IET Information Security, vol. 8, pp. 25-36, 2014, DOI: https://doi.org/10.1049/iet-ifs.2013.0095.

[6] S.I. Popoola, A. Bamidele, A. Ruth, M. Hammoudeh, A. Kelvin, and A. Aderemi, “SMOTE-DRNN: A deep learning algorithm for botnet detection in the Internet-of-Things networks,” Sensors, vol. 9, pp. 2851-2861, 2021. https://doi.org/10.3390/s21092985.

[7] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El-Fadili, “Toward a deep learning-based intrusion detection system for IoT against botnet attacks,” IAES International Journal of Artificial Intelligence, vol. 1, pp. 110-122, 2021. DOI:10.11591/ijai.v10.i1.pp110-120.

[8] M. Abdel-Basset, L. Abdel-Fatah, and A. Sangaiah, “Metaheuristic algorithms: A comprehensive review,” In Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Academic press, pp. 185-231, 2018. https://doi.org/10.1016/B978-0-12-813314-9.00010-4.

[9] A.S. Bozkir, A.O. Cankaya, and M. Aydos, “Utilization and Comparison of Convolutional Neural Networks in Malware Recognition,” In Proceedings of the 27th Signal Processing and Communications Applications Conference (SIU), Sivas, Turkey, pp. 1–4, 2019. DOI**:** 10.1109/SIU.2019.8806511.

[10] S.A. Roseline, S. Geetha, S. Kadry, and Y. Nam, “Intelligent Vision-based Malware Detection and Classification using Deep Random Forest Paradigm,” IEEE Access, vol. 8. pp. 206303–206324, 2020. DOI**:** 10.1109/ACCESS.2020.3036491.

[11] M. Imran, M.T. Afzal, and M.A. Qadir, “Similarity-based malware classification using hidden Markov model,” In Proceedings of the Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), Jakarta, Indonesia, pp. 129–134, 2015. DOI**:** 10.1109/CyberSec.2015.33.

[12] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic analysis of malware behavior using machine learning,” Journal of Computer Security, vol. 19, pp. 639–668, 2011. DOI:10.3233/JCS-2010-0410.

[13] S.O. Subairu, J. Alhassan, S. Misra, O. Abayomi-Alli, R. Ahuja, R. Damasevicius, and R. Maskeliunas, “An experimental approach to unravel effects of malware on system network interface,” In Lecture Notes in Electrical Engineering; Springer: Singapore, pp. 225–235, 2020. https://doi.org/10.1007/978-981-15-0372-6_17.

[14] H. Yan, H. Zhou, and H. Zhang, “Automatic malware classification via PRICoLBP,” Chinese Journal of Electronics, vol. 27, pp. 852–859, 2018. DOI:10.1049/cje.2018.05.001.

[15] H. Naeem, F. Ullah, M.R. Naeem, S. Khalid, D. Vasan, S. Jabbar, and S. Saeed, “Malware detection in industrial internet of things based on hybrid image visualization and deep learning model,” Ad Hoc Networks, pp. 105-118, 2020. https://doi.org/10.1016/j.adhoc.2020.102154.

[16] K. Han, B. Kang, E.G. Im, “Malware analysis using visualized image matrices,” The Scientific World Journal, 2014. doi: 10.1155/2014/132713. Epub 2014 Jul 16.

detection and variant classification," Computational Intelligence, vol. 38, pp. 1748–1771, 2020. https://doi.org/10.1111/coin.12532

[33] D. Karaboga, and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, LNCS: 789-798, Springer, Berlin, 2007. https://doi.org/10.1007/978-3-540-72950-1_77.

[34] M. Abualhaj, M. Al-Zyoud, A. Alsaaideh, A. Abu-Sharcha, and S. Al-Khatib, "Enhancing Malware Detection through Self-Union Feature Selection Using Firefly Algorithm with Random Forest Classification," International Journal of Intelligent Engineering & Systems, vol. 4, pp. 376-389, 2024. DOI: 10.22266/ijies2024.0831.29.

[35] A. Kaveh, "Imperialist competitive algorithm," Advances in Metaheuristic Algorithms for Optimal Design of Structures, pp. 353-373, 2017. https://en.wikipedia.org/wiki/Imperialist_competitive_algorithm.

[36] R. Penmasta, S. Mallidi, K. Jhansi, and D. Latha, "Bat optimization algorithm for wrapper-based feature selection and performance improvement of android malware detection," IET Networks, vol. 3, 2021. DOI: 10.1049/ntw2.12022.

[28] M. Al-Andoli, S. Tan, K. Sim, C. Lim, and P. Goh, "Parallel Deep Learning with a hybrid BP-PSO framework for feature extraction and malware classification," Applied Soft Computing, vol. 131, pp. 109756, 2022, DOI: https://doi.org/10.1016/j.asoc.2022.109756.

[29] S. Imtiaz, S. ur-Rehman, A. Javed, Z. Jalil, X. Liu, and W. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network," Future Generation computer systems, vol. 115, pp. 844-856, 2021, DOI: https://doi.org/10.1016/j.future.2020.10.008.

[30] L. Suhuan, and H. Xiaojun, "Android malware detection based on logistic regression and XGBoost," in: 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pp. 528-532, 2019, DOI: 10.1109/ICSESS47205.2019.9040851.

[31] S. Kumar, and K. Panda, "SDIF-CNN: Stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification," Applied Soft Computing, vol. 146, 110676, 2023. https://doi.org/10.1016/j.asoc.2023.110676.

[32] P. Yadava, N. Menonb, V. Ravic, S. Vishvanathand, and D.T. Phame, "A two-stage deep learning framework for image-based android malware