

اللهم اغفر لي

چکیده

با توسعه صنعت نرم افزار و افزایش کیفیت و قابلیت اتکای محصولات نرم افزاری، آزمون نرم افزار مورد توجه مهندسين قرار گرفت. اما از آنجايي که آزمون تمام ورودی‌ها حتی برای یک تابع ساده مشکل است و آزمون تمام واحدهای نرم افزاری نیز اغلب زمان‌بر و پرهزینه است، معمولاً از انجام آن صرف نظر می‌شود. به این ترتیب برای کاهش هزینه کلی آزمون و کشف خطاهای احتمالی پیش از وقوع آن‌ها، روش پیش‌بینی خطای نرم افزار مورد استفاده قرار می‌گیرد. یکی از چالش‌های موجود در زمینه پیش‌بینی خطای نرم افزار این است که مدل‌های پیش‌بینی قبلی، مدل‌های قدیمی‌ای بودند یا مجموعه داده‌هایی را به کار می‌گرفتند که روی پروژه‌های کوچک، محدود و قدیمی بودند؛ که به دلیل تغییر روند توسعه نرم افزارها، این مدل‌ها کارایی چندانی را برای نرم افزارهای امروزی ارائه نمی‌دهند. در این تحقیق برای کمک به فرایند پیش‌بینی خطا در مقابل روش‌های قدیمی، مجموعه داده جدیدی به نام باگ‌هانتز مورد استفاده قرار گرفته است. این مجموعه داده ارزشمند و نسبتاً بزرگ، متشکل از ۱۵ برنامه مرسوم و منبع باز جاوا است که امروزه به طور گسترده‌ای توسط مهندسين نرم افزار استفاده می‌شوند. تاکنون تکنیک‌های گوناگون یادگیری ماشین نظیر بیز ساده، رگرسیون لجستیک، جنگل تصادفی و غیره جهت کشف خطاهای نرم افزاری برای این مجموعه داده مورد استفاده قرار گرفته است. چالش دیگری که وجود دارد این است که دقت مدل‌های پیش‌بینی موجود، برای این مجموعه داده پایین است. این تحقیق باهدف افزایش عملکرد مدل پیش‌بینی خطا، تکنیک یادگیری عمیق و تکنیک ترکیبی را به کار می‌گیرد و با ترکیب شبکه عصبی و جنگل تصادفی دقت مدل پیش‌بینی را افزایش می‌دهد. شبکه عصبی مورد استفاده در این پروژه پرسپترون چندلایه و طبقه‌بندی کننده مورد استفاده در آن، مدل ترتیبی به همراه چندین لایه متراکم و حذف تصادفی است. روش ترکیبی پیشنهادی نیز از ترکیب یک شبکه عصبی و جنگل تصادفی تشکیل شده است. برای ایجاد مدل‌های پیش‌بینی قدرتمند و جلوگیری از جانب‌دار شدن مدل‌ها، تکنیک نمونه‌برداری تصادفی رو به بالا مورد استفاده قرار گرفته است. در نهایت روش پیشنهادی بر روی مجموعه داده باگ‌هانتز اعمال شد و نتایج بهبود روند پیش‌بینی خطا را به خصوص برای تکنیک ترکیبی پیشنهادی نشان می‌دهد. به طوری که مقدار معیار امتیاز اف-۱ به دست آمده توسط تکنیک شبکه عصبی پیشنهادی ۸۱٪/۷۴ است و توسط تکنیک ترکیبی پیشنهادی ۸۴٪/۸۱ است که نسبت به بهترین روش قبل، به ترتیب افزایش ۸٪ و ۱۱٪ داشته است.

کلمات کلیدی: پیش‌بینی خطای نرم افزار، یادگیری عمیق، یادگیری ماشین، یادگیری ترکیبی، مجموعه داده BugHunter، عدم تعادل کلاس

فهرست مطالب

۱	فصل اول: مقدمه
۲	۱-۱ پیش گفتار.....
۵	۲-۱ هدف تحقیق.....
۵	۳-۱ چالش‌ها و راه‌حل‌های آن.....
۷	۴-۱ سؤال‌های تحقیق.....
۸	۵-۱ فرضیه‌های تحقیق.....
۸	۶-۱ نوآوری‌های تحقیق.....
۹	۷-۱ ساختار پایان‌نامه.....
۱۱	فصل دوم: ادبیات موضوع و پیشینه تحقیق
۱۲	۱-۲ تکنیک‌های یادگیری ماشین و تکنیک‌های ترکیبی.....
۱۹	۲-۲ رویکرد شبکه عصبی.....
۲۱	۳-۲ تکنیک‌های یادگیری عمیق.....
۲۹	فصل سوم: ضروریات تحقیق
۳۰	۱-۳ معیارهای نرم‌افزار.....
۳۱	۱-۱-۳ معیارهای محصول.....
۳۱	۲-۱-۳ معیارهای فرایند.....
۳۲	۲-۳ انواع مجموعه داده خطا.....
۳۳	۳-۳ مجموعه داده خطا باگ‌هانتز و نحوه ایجاد آن.....
۳۷	۱-۳-۳ معیار پیچیدگی سیکلوماتیک مک کیب.....

- ۳۹ ۲-۳-۳ معیار تعداد خطوط کد
- ۴۰ ۳-۳-۳ معیارهای تعداد دستورات برنامه، خطوط کامنت کد، تعداد پارامترها
- ۴۰ ۴-۳ پروژه‌های مورد استفاده در مجموعه داده باگ‌هانتر
- ۴۰ ۱-۴-۳ Elasticsearch
- ۴۱ ۲-۴-۳ Hazelcast
- ۴۱ ۳-۴-۳ Orientdb
- ۴۲ ۴-۴-۳ Netty
- ۴۲ ۵-۴-۳ Neo4j
- ۴۲ ۶-۴-۳ ANTLR v4
- ۴۲ ۵-۳ جزئیات مجموعه داده باگ‌هانتر
- ۴۴ ۶-۳ پیش پردازش مجموعه داده باگ‌هانتر
- ۴۵ ۱-۶-۳ گام‌های اساسی پیش پردازش مجموعه داده hazelcast
- ۴۸ ۲-۶-۳ پیش پردازش سایر مجموعه داده‌ها
- ۵۰ ۷-۳ معیارهای ارزیابی کارایی
- ۵۱ ۸-۳ ماتریس سردرگمی
- ۵۲ ۹-۳ رویکرد یادگیری عمیق
- ۵۳ ۱۰-۳ شبکه عصبی مصنوعی و نحوه عملکرد آن
- ۵۴ ۱۱-۳ پیش‌بینی خطای نرم‌افزار با استفاده از شبکه‌های عصبی
- ۵۴ ۱۲-۳ پرسپترون چندلایه
- ۵۵ ۱۳-۳ مدل شبکه عصبی
- ۵۶ ۱۴-۳ طبقه‌بندی کلاس برای ویژگی هدف
- ۵۶ ۱۵-۳ پارامترهای شبکه عصبی

۱۶-۳ پیش متناسب و کم متناسب شدن مدل یادگیری..... ۵۹

۱-۱۶-۳ راه حل رفع کم متناسب شدن مدل..... ۶۰

۲-۱۶-۳ راه حل رفع بیش متناسب شدن مدل..... ۶۱

فصل چهارم: روش پیشنهادی ۶۵

۱-۴ پیاده سازی روش پیشنهادی..... ۶۷

۲-۴ گام های اساسی ساخت مدل پیش بینی خطای پیشنهادی..... ۶۷

۱-۲-۴ خواندن مجموعه داده..... ۶۷

۲-۲-۴ ساخت مدل پیش بینی خطا با استفاده از شبکه های عصبی عمیق..... ۶۷

۳-۲-۴ کامپایل کردن مدل..... ۶۸

۴-۲-۴ مرحله یادگیری شبکه..... ۶۸

۵-۲-۴ نمودار ارزیابی کارایی برای آموزش مدل..... ۶۹

۶-۲-۴ مرحله پیش بینی..... ۷۰

۷-۲-۴ ارزیابی مدل پیش بینی..... ۷۰

۳-۴ عدم تعادل کلاس..... ۷۱

۱-۳-۴ عدم تعادل کلاس در مجموعه داده باگ هانتر..... ۷۱

۲-۳-۴ راه حل های موجود برای ایجاد تعادل کلاس..... ۷۲

۳-۳-۴ ایجاد تعادل در کلاس های مجموعه داده باگ هانتر..... ۷۳

۴-۴ روند کلی روش پیشنهادی با استفاده از شبکه عصبی عمیق..... ۷۳

۵-۴ روند کلی روش پیشنهادی با استفاده از رویکرد ترکیبی..... ۷۵

فصل پنجم: نتایج ۷۹

۱-۵ جلوگیری از بیش متناسب شدن مدل..... ۸۰

۱-۱-۵ روش حذف تصادفی..... ۸۰

۸۱max-norm روش منظم‌سازی	۲-۱-۵
۸۱مجموعه‌داده به کلاس‌های	۳-۱-۵
۸۲پیشنهادی روش نتایج	۲-۵
۸۲broadleafcommerce مجموعه‌داده برای پیشنهادی	۱-۲-۵
۸۸netty مجموعه‌داده برای پیشنهادی	۲-۲-۵
۸۹hazelcast مجموعه‌داده برای پیشنهادی	۳-۲-۵
۹۰elasticsearch مجموعه‌داده برای پیشنهادی	۴-۲-۵
۹۰neo4j مجموعه‌داده برای پیشنهادی	۵-۲-۵
۹۱orientdb مجموعه‌داده برای پیشنهادی	۶-۲-۵
۹۲ceylon-ide-eclipse مجموعه‌داده برای پیشنهادی	۷-۲-۵
۹۲روش‌های دیگر	۳-۵
۹۷	فصل ششم: نتیجه‌گیری و پیشنهاد	
۹۸نتیجه‌گیری و جمع‌بندی	۱-۶
۹۹کارهای آینده	۲-۶
۱۰۱	پیوست	
۱۰۲خطا در توابع	خودکارسازی ایجاد مدل‌های شبکه عصبی برای پیش‌بینی
۱۰۵	مراجع	

فهرست جداول

جدول ۱-۳ مقایسه مجموعه داده‌های خطا [۴۳]	۳۳
جدول ۲-۳ معیارهای شی گرا مورد استفاده در مجموعه داده باگ‌هانتر [۴۳]	۳۷
جدول ۳-۳ معیارهای مرتبط با کلون پایه	۳۷
جدول ۴-۳ ماتریس سردرگمی	۵۲
جدول ۱-۵ تعداد داده‌های آموزشی و آزمایشی پس از پیش‌پردازش مجموعه داده	
.....broadleafcommerce	۸۳
جدول ۲-۵ تعداد داده‌های آموزشی و آزمایشی پس از اعمال تکنیک نمونه‌گیری مجدد روی	
مجموعه داده broadleafcommerce	۸۳
جدول ۳-۵ تعداد نمونه‌های آزمایشی مربوط به کلاس صفر و یک مجموعه داده	
.....broadleafcommerce	۸۳
جدول ۴-۵ تعداد داده‌های آموزشی مربوط به کلاس صفر و یک، پیش از اعمال تکنیک نمونه‌گیری	
مجدد روی broadleafcommerce	۸۴
جدول ۵-۵ تعداد داده‌های آموزشی مربوط به کلاس صفر و یک، پس از اعمال تکنیک نمونه‌گیری	
مجدد روی broadleafcommerce	۸۵
جدول ۶-۵ نتایج رویکرد پیشنهادی برای مجموعه داده broadleafcommerce	۸۷
جدول ۷-۵ نمودار هزینه مجموعه داده broadleafcommerce	۸۸
جدول ۸-۵ ماتریس سردرگمی مدل ۱۵	۸۸
جدول ۹-۵ نتایج رویکرد پیشنهادی برای مجموعه داده netty	۸۹
جدول ۱۰-۵ نتایج رویکرد پیشنهادی برای مجموعه داده hazelcast	۸۹
جدول ۱۱-۵ نتایج رویکرد پیشنهادی برای مجموعه داده elasticsearch	۹۰
جدول ۱۲-۵ نتایج رویکرد پیشنهادی برای مجموعه داده neo4j	۹۱
جدول ۱۳-۵ نتایج رویکرد پیشنهادی برای مجموعه داده orientdb	۹۱
جدول ۱۴-۵ نتایج رویکرد پیشنهادی برای مجموعه داده ceylon-ide-eclipse	۹۲
جدول ۱۵-۵ بهترین نتایج معیار امتیاز اف-۱	۹۶

فهرست اشکال

- شکل ۱-۱ نمونه خطای نرم افزار [۱] ۴
- شکل ۱-۳ قطعه فایل تفاوت یکپارچه [۴۳] ۳۵
- شکل ۲-۳ کد نمونه برای محاسبه معیار پیچیدگی سیکلوماتیک ۳۸
- شکل ۳-۳ گراف کنترل جریان ۳۹
- شکل ۴-۳ کد نمونه برای معیار تعداد خطوط کد ۴۰
- شکل ۱-۴ روند کلی رویکرد پیشنهادی توسط تکنیک شبکه عصبی ۷۴
- شکل ۲-۴ روند کلی روش پیشنهادی توسط مدل ترکیبی ۷۷
- شکل ۱-۵ نمودار هزینه مربوط به روش حذف تصادفی ۸۰
- شکل ۲-۵ نمودار هزینه مربوط به روش max-norm ۸۱
- شکل ۳-۵ نمودار هزینه مربوط به روش وزن دهی به کلاس های مجموعه داده ۸۲
- شکل ۴-۵ تعداد نمونه های آزمایشی مربوط به کلاس صفر و یک مجموعه داده
broadleafcommerce ۸۴
- شکل ۵-۵ تعداد داده های آموزشی مربوط به کلاس صفر و یک، پیش از اعمال تکنیک نمونه گیری مجدد
روی broadleafcommerce ۸۴
- شکل ۶-۵ تعداد داده های آموزشی مربوط به کلاس صفر و یک، پس از اعمال تکنیک نمونه گیری مجدد
روی broadleafcommerce ۸۵

فصل اول: مقدمه

۱-۱ پیش گفتار

پیشرفت‌های اخیر تکنولوژی، رشد شدید صنعت نرم‌افزار را به همراه داشت. با افزایش تقاضای نرم‌افزار برای کاربردهای مختلف، به‌مرور زمان پیچیدگی و اندازه نرم‌افزارها گسترش یافت. از این‌رو توسعه نرم‌افزار با کیفیت، قابلیت اتکا و امنیت در دنیای مدرن جایگاه ویژه‌ای یافت. در این حیطه مهندسين نرم‌افزار نقش اساسی را ایفا می‌کنند.

وجود خطاهای یکی از اصلی‌ترین مسائلی است که کیفیت و قابلیت اتکای نرم‌افزار را به چالش می‌کشد. برنامه‌ها و کاربردهای نرم‌افزاری مختلف توسط برنامه‌نویسان و با استفاده از کدها ایجاد می‌شوند تا نیازمندی‌های کاربر نهایی از محصول موردنظر را برآورده سازند. حتی با توسعه‌ی دقیق محصولات و برنامه‌های نرم‌افزار، ممکن است کدها دارای خطاهای پیش‌بینی‌نشده باشند که از دید برنامه‌نویسان پنهان بمانند؛ بنابراین خروجی‌ها و عملکردهای مورد انتظار کاربر با خروجی‌ها و عملکردهای واقعی سیستم نرم‌افزاری در تناقض خواهد بود و کیفیت نرم‌افزار سلب می‌شود. از این‌رو پیش‌بینی و برطرف کردن خطاهای احتمالی و تضمین کیفیت نرم‌افزار قبل از عرضه به مشتری امری حیاتی در حوزه‌ی مهندسی نرم‌افزار است. خطای نرم‌افزار به یک مرحله، فرایند، یا تعریف نادرست داده در برنامه گفته می‌شود که باعث می‌شود برنامه به روشی پیش‌بینی‌نشده انجام شود و نتایج مطلوب و مورد انتظار آن فراهم نشود. خطاهای نرم‌افزار ممکن است در هر یک از مراحل توسعه به وجود آید و سیستم را با شکست مواجه کند. در ادامه چند تعریف پر استفاده در این مبحث ارائه شده است.

نقص^۲: زمانی اتفاق می‌افتد که وظایف و نیازمندی‌های سیستم ارائه‌شده دارای کاستی و انحراف باشد و به‌درستی برآورده نشود. می‌توان گفت نقص نرم‌افزار نتیجه‌ی عدم انطباق بین نتایج مورد انتظار و واقعی سیستم‌های نرم‌افزاری است. بسته به اندازه و پیچیدگی نرم‌افزار، یافتن و برطرف کردن این نقص‌ها

^۱Bugs

^۲Defect

می‌تواند به آسانی و با سرعت بالایی انجام شود یا بسیار سخت و زمان‌بر باشد. به‌طور کلی این نقص‌ها توسط برنامه‌نویسانی که فرآیندهای طراحی شده را پیاده‌سازی می‌کنند، ایجاد می‌شوند.

خطا^۱: یک مشکل ایستا در برنامه، یا خطای واضح و آشکاری است که در محصولات نرم‌افزاری رخ داده است و مهندس آزمون باید آن را پیدا کند. خطاهای نرم‌افزاری توسط استفاده نادرست برنامه‌نویس از عملگرها یا عملوندهای برنامه به وجود می‌آیند و توسط ناظر کشف می‌شوند.

شکست^۲: مشکل یا خطای موجود در برنامه منجر به یک شکست می‌شود. این شکست ممکن است منجر به توقف عملکرد مورد انتظار سیستم نرم‌افزاری شود، یا نتایج نادرست را برای ورودی‌های تعیین‌شده توسط مشتریان محصول نرم‌افزاری به دنبال داشته باشد. به‌عبارت‌دیگر با اجراشدن کد نرم‌افزاری دارای خطا، شکست اتفاق می‌افتد و ممکن است عملکرد سیستم نرم‌افزاری را مختل کند.

اشتباه^۳: این امر اشتباه انسان است که نتایج نادرست را به دنبال دارد و منجر به خطا می‌شود.

برنامه‌ی شکل ۱-۱، دارای خطای $i > 0$ است [۱]. در صورت رخداد المان موردنظر در جایگاه صفر و مشاهده آن تنها یک‌بار در برنامه، کد موردنظر وجود آن را تشخیص نمی‌دهد که این خطا است.

از آنجایی که پرهزینه‌ترین فعالیت‌های توسعه نرم‌افزار، هزینه کشف و اصلاح خطاها است به حداقل رساندن این خطاها و ساخت نرم‌افزاری عاری از خطا موضوعی پراهمیت در مهندسی نرم‌افزار است. برطرف کردن خطاهای نرم‌افزار با به‌کارگیری منابع آزمون امری پرهزینه است و اغلب از آن صرف‌نظر می‌شود. از سوی دیگر خطاهای ناشناخته و پیش‌بینی‌نشده موجود در اغلب نرم‌افزارها، هیچ خطایی در زمان کامپایل ندارند. لذا توسعه نرم‌افزار بدون خطا، حتی با به‌کارگیری بسیار دقیق روش‌های توسعه بسیار دشوار است. روند پیش‌بینی خطا، مسئولان و مدیران نرم‌افزار را قادر به تخصیص مقرون‌به‌صرفه منابع و کاهش هزینه‌های مربوط به آزمون می‌کند.

^۱Fault
^۲Failure
^۳Error

```

/**
 * Find last index of element
 *
 * @param x array to search
 * @param y value to look for
 * @return last index of y in x; -1 if absent
 * @throws NullPointerException if x is null
 */
public int findLast (int[] x, int y)
{
    for (int i=x.length-1; i>0; i--)
    {
        if (x[i] == y)
        {
            return i;
        }
    }
    return -1;
}
//test: x = [2, 3, 5]; y=2; Expected=0
//Book website: FindLast.java
//Book website: FindLastTest.java

```

شکل ۱-۱ نمونه خطای نرم‌افزار [۱]

کشف خطاهای نرم‌افزار در مراحل اولیه توسعه هزینه‌های کمتری را در پی دارد، بنابراین پیش‌بینی اولیه آن‌ها بسیار پراهمیت است. یافتن و از بین بردن تعداد خطاهای بیشتر سطوح بالاتری از کیفیت و قابلیت اتکای سیستم را فراهم می‌کند و تلاش برای نگهداری کیفیت را بهبود می‌دهد.

نرم‌افزار مطلوب نرم‌افزاری ایمن و باکیفیت است که به‌درستی به نیاز کاربر پاسخ بدهد، خطاهای پنهان در یک دوره زمانی مشخص و شرایط محیطی داده‌شده در آن وجود نداشته باشد و در زمان اجرای نرم‌افزار سیستم دچار شکست نشود. روند پیش‌بینی خطای نرم‌افزار^۱ یکی از روش‌های بهبود کیفیت است که از آموزش یک تکنیک یادگیری برای یک پروژه مشخص استفاده می‌کند [۲].

تشخیص اولیه و دقیق خطا و حذف زود هنگام آن در طول توسعه نرم‌افزار و قبل از عرضه به مشتری، توسعه یک سیستم بسیار قابل‌اعتماد و باکیفیت، کاهش تلاش و هزینه کلی آزمون، کاهش تأثیر خطاها و بهبود فرآیند توسعه اقتصادی بدون اجرا کردن کد نرم‌افزاری و پیش از انجام عمل آزمون از مزایای پیش‌بینی خطای نرم‌افزار است.

تکنیکی که در روند پیش‌بینی خطای نرم‌افزار مورد استفاده قرار می‌گیرد از رویکرد ساخت یک مدل پیش‌بینی خطا برای یک مجموعه داده در پروژه مشخص، پیروی می‌کند. سپس این مدل بر اساس

اطلاعاتی که در رابطه با نسخه‌های قبلی پروژه دارد آموزش می‌بیند و در نهایت در پیش‌بینی خطا برای پروژه‌های نامشخص مورد استفاده قرار می‌گیرد.

۱-۲ هدف تحقیق

پیش‌بینی خودکار خطاها و کشف دقیق آن‌ها یکی از چالش‌ها در حوزه کیفیت و قابلیت اتکای نرم‌افزار است. روش‌های بسیاری برای پیش‌بینی خطای نرم‌افزار به کار گرفته شده است که یکی از آن‌ها استفاده از شبکه‌های عصبی عمیق است. اخیراً اهمیت یادگیری عمیق، در مسائل پیش‌بینی خطا افزایش یافته است. این تکنیک به‌عنوان یک ابزار فرآیند پیش‌بینی را تسریع می‌کند. شبکه‌های عصبی عمیق به‌طور خودکار اطلاعات معنایی و ساختاری برنامه‌ها را از ویژگی‌های ورودی استخراج می‌کنند و در مقابل تغییرات ورودی مقاوم‌اند. علاوه بر تکنیک‌های مبتنی بر شبکه‌های عصبی عمیق، تکنیک‌های ترکیبی نیز برای پیش‌بینی خطای نرم‌افزار مؤثر واقع می‌شوند. هدف کلی این تحقیق بهبود دقت و عملکرد روش‌های پیش‌بینی خطا با استفاده از تکنیک یادگیری عمیق و روش ترکیبی است. به‌گونه‌ای که مدل پیش‌بینی با انجام خودکار فرآیند یادگیری، برای داده‌هایی که تاکنون ندیده است روند پیش‌بینی خطا را با دقت بالایی انجام دهد. همچنین استفاده از یک مجموعه داده بسیار خوب، به‌روز و منبع باز که در دسترس همه قرار دارد، ورودی‌های آن ساختار خوبی دارد و به فرآیند پیش‌بینی خطای نرم‌افزار در مقابل روش‌های قدیمی‌تر کمک می‌کند، دارای اهمیت ویژه‌ای برای انجام این کار است.

۱-۳ چالش‌ها و راه‌حل‌های آن

امروزه، با پیشرفت نرم‌افزارها، افزایش اندازه و پیچیدگی آن‌ها و فراهم آمدن شرایطی مطلوب برای رشد و ایجاد خطاهای نرم‌افزاری، رخداد این‌گونه خطاها غیرقابل اجتناب است. همچنین پیش‌بینی خودکار تعداد خطاها در واحدهای نرم‌افزار از اهمیت ویژه‌ای برخوردار است و در تخصیص منابع محدود مؤثر

است [۳]. روش‌های مختلفی برای کشف خطاها و سپس از بین بردن آن‌ها ارائه شده است. اما با این حال، دقت و عملکرد این رویکردها به پیشرفت قابل توجهی نیاز دارد.

تکنیک‌ها و الگوریتم‌های مختلف یادگیری ماشین^۱ که پیش‌از این جهت اعمال روند پیش‌بینی خطا به‌طور گسترده‌ای استفاده می‌شد، دقت‌ها و نتایج متفاوتی را ارائه داده است. از طرفی در مورد نرم‌افزارهای بزرگ، آزمایش تمام سناریوهای ممکن از طریق روش سنتی بسیار سخت و زمان‌بر است. یادگیری عمیق یک روش مناسب برای حل این مسئله است و عملکرد مدل‌های مبتنی بر یادگیری عمیق با افزایش داده، افزایش می‌یابد.

دقت و قابلیت شبکه یادگیری عمیق برای حل مسائل پیچیده به‌طور پیوسته افزایش می‌یابد. شبکه‌های عمیق عملکرد خوبی را در مسائل مشکل یادگیری و یادگیری ماشین ارائه می‌دهند. یادگیری عمیق^۲ به‌عنوان زیرشاخه‌ای از یادگیری ماشین، در حوزه‌ی پیش‌بینی خطای نرم‌افزار نتایج مطلوب‌تری را نسبت به تکنیک‌های سنتی ارائه می‌دهد. یادگیری عمیق همچنین عملکرد قابل توجهی را در زمینه‌های مختلف نظیر بینایی کامپیوتری، پردازش زبان طبیعی، تشخیص گفتار و سایر زمینه‌ها داشته است [۴].

اخیراً یادگیری عمیق برای افزایش دقت تکنیک‌های پیش‌بینی خطا مورداستفاده قرار می‌گیرد. مدل‌های یادگیری عمیق با استفاده از شبکه عصبی مصنوعی ساخته می‌شوند و معماری این شبکه‌ها متشکل از حداقل سه لایه است. افزایش تعداد لایه‌ها نشان‌دهنده پیچیدگی بیشتر مدل است. مدل‌هایی که توسط یادگیری عمیق ایجاد می‌شوند این توانایی را دارند که به‌صورت خودکار ویژگی‌های لازم را از داده استخراج کنند و در مقابل تغییرات ورودی مقاوم‌اند. در تکنیک‌های سنتی پیش‌بینی خطا که ویژگی‌ها به‌صورت دستی طراحی می‌شوند، استخراج اطلاعات ساختاری و معنایی برنامه‌ها اغلب امکان‌پذیر نیست و مدل‌های قدرتمند ایجاد نمی‌شوند. شبکه عصبی با کسب خودکار دانش از اطلاعات معنایی و ساختاری برنامه‌ها که آن‌ها را مستقیماً از ویژگی‌های مجموعه داده به دست می‌آورد، روند پیش‌بینی خطا را بهبود

^۱Machine Learning

^۲Deep Learning

می‌بخشد و مدل‌های دقیق پیش‌بینی را ایجاد می‌کند. تکنیک‌های یادگیری عمیق برای پیش‌بینی نیازی به استخراج ویژگی‌ها به صورت دستی ندارد [۵] و همواره شکاف بین معنای برنامه‌ها و ویژگی‌های پیش‌بینی خطا را پر می‌کند [۶]. این عمل پروسه‌ی پیش‌بینی خطا را بشدت تحت تأثیر قرار می‌دهد. جهت بهبود روند پیش‌بینی خطا وجود چنین رویکردی که پروسه پیش‌بینی را راحت‌تر، سریع‌تر، بهتر و با دقت بالاتر انجام دهد، الزامی است.

علاوه بر تکنیک‌های یادگیری عمیق، تکنیک‌های یادگیری ترکیبی نیز برای پیش‌بینی خطا بسیار مؤثر واقع می‌شوند و دقت مدل‌های پیش‌بینی خطا را نسبت به روش‌های منفرد بهبود می‌دهند. مدل‌های ایجادشده توسط این روش، با استخراج ویژگی‌های مختلف می‌توانند دقت مدل‌های پیش‌بینی را افزایش دهند و فرایند پیش‌بینی خطا را بهبود بخشند.

مسئله دیگری که در بحث پیش‌بینی خطا وجود دارد عدم وجود مجموعه‌داده‌های به‌روز و مرتبط با برنامه‌های نرم‌افزاری امروزی در بین مجموعه‌داده‌های فعلی پیش‌بینی خطای نرم‌افزار است. مجموعه‌داده باگ‌هانتز^۱ یک مجموعه‌داده جدید و بزرگ است که شامل برنامه‌های پرستفاده امروزی است. هریک از برنامه‌های این مجموعه‌داده دارای معیارهای نرم‌افزاری در سطح توابع است که می‌تواند پیش‌بینی خطای نرم‌افزار را بهبود دهد. چالشی که در این مجموعه‌داده وجود دارد عدم تعادل کلاس در نمونه‌های مجموعه است. جهت برقراری تعادل کلاس‌ها تکنیک رایج نمونه‌برداری تصادفی رو به بالا^۲ مورد استفاده قرار می‌گیرد که در نتایج پیش‌بینی خطا توسط شبکه‌های عصبی عمیق بسیار مؤثر است.

۴-۱ سؤال‌های تحقیق

آیا استفاده از یادگیری عمیق می‌تواند در پیش‌بینی خطای نرم‌افزار در سطح توابع مؤثر واقع شود؟

آیا می‌توان از روی معیارهای نرم‌افزاری، وجود خطا در یک تابع را به‌درستی پیش‌بینی کرد؟

^۱BugHunter

^۲Random Over Samping

آیا ایجاد مدل‌های پیش‌بینی با استفاده از یادگیری عمیق و یادگیری ترکیبی عملکرد مدل پیش‌بینی خطا را نسبت به رویکردهای پیشین بهبود می‌بخشد؟

۱-۵ فرضیه‌های تحقیق

با کمک معیارهای نرم‌افزاری موجود در مجموعه‌داده باگ‌هانتز و وجود داده‌های کافی می‌توان پیش‌بینی خطا را برای توابع انجام داد. مدل‌های یادگیری عمیق می‌توانند عملکرد فرایند پیش‌بینی خطای نرم‌افزار را نسبت به رویکردهای سنتی یادگیری ماشینی بهبود بدهند. در کنار رویکردهای یادگیری عمیق، رویکردهای ترکیبی می‌توانند در روند پیش‌بینی خطای نرم‌افزار مؤثر واقع شوند. مدل‌های پیش‌بینی خطا با استفاده از روش‌های یادگیری عمیق و روش‌های ترکیبی، قابلیت تشخیص خطا برای داده‌های نامشخص را با عملکرد بالا خواهند داشت.

۱-۶ نوآوری‌های تحقیق

به‌طور کلی، لیست نوآوری‌ها در این پایان‌نامه به شرح زیر است:

- به‌کارگیری مجموعه‌داده خطای نوین باگ‌هانتز برای پیش‌بینی خطای نرم‌افزار که شامل برنامه‌های متنوع و محبوب است و امروزه توسط توسعه‌دهندگان و مهندسين نرم‌افزار بشدت مورد استفاده قرار می‌گیرد.
- متعادل‌سازی کلاس‌های مجموعه‌داده باگ‌هانتز توسط روش نمونه‌گیری تصادفی رو به بالا
- ارائه مدل خودکار پیش‌بینی خطا بر اساس شبکه عصبی پرسپترون چندلایه
- ارائه دو نوع مدل پیش‌بینی خطا جدید: یکی مبتنی بر شبکه عصبی پرسپترون چندلایه و یکی مبتنی بر مدل ترکیبی
- افزایش دقت مدل پیش‌بینی خطا نسبت به روش‌های قبلی