تاریخ دریافت: ۱۳۹۷/۱۰/۱۷
تاریخ پذیرش: ۱۳۹۷/۱۲/۲۷

بهبود پیش‌بینی خروجی مورد انتظار در تولید خودکار اوراکل آزمون

مریم هاشم‌زاده
کارشناسی ارشد نرم‌افزار، گروه مهندسی نرم‌افزار – دانشکده مهندسی کامپیوتر– دانشگاه اصفهان– اصفهان– ایران
پست الکترونیکی: m.hashemzade@eng.ui.ac.ir

احمد براآنی دستجردی*
استاد، گروه مهندسی نرم‌افزار – دانشکده مهندسی کامپیوتر– دانشگاه اصفهان– اصفهان– ایران
پست الکترونیکی: ahmadb@eng.ui.ac.ir

علیرضا خلیلیان
مربی و دانشجوی دکتری نرم‌افزار، گروه مهندسی نرم‌افزار – دانشکده مهندسی کامپیوتر– دانشگاه اصفهان– اصفهان– ایران
پست الکترونیکی: khalilian@eng.ui.ac.ir

چکیده:

آزمون نرم‌افزار از فعالیت‌های مهم چرخه حیات نرم‌افزار است که برای بهبود کیفیت نرم‌افزار مورد استفاده قرار می‌گیرد. برای کاهش هزینه‌های آزمون، محققان سعی کرده‌اند که آن‌را خودکار نمایند. اما آزمون به اوراکل نیاز دارد و خودکارسازی تولید اوراکل نیز لازم می‌شود. اوراکل روالی برای تولید خروجی مورد انتظار و تعیین اجرای موفق نرم‌افزار حین آزمون است. برای تولید خودکار اوراکل آزمون باید رفتار نرم‌افزار مدل‌سازی شود و دامنه ورودی به خروجی نگاشت یابد. این کار معمولاً ازطریق ساخت مدل یادگیری روی نمونه‌های آزمایشی صورت می‌گیرد. اما گاهی داده‌ها جزئیات کافی ندارد تا مدل، رفتار دقیق را یاد بگیرد و دقت تولید اوراکل کاهش مییابد. برای حل این مشکل، در این مقاله، روشی بهبودیافته برای تولید خودکار اوراکل آزمون ارائه می‌شود. این روش اطلاعاتی از پوشش کد نرم‌افزار تحت آزمون استخراج می‌کند. سپس آن‌را به مدل یادگیر تزریق می‌نماید تا جزئیات بیش‌تری در اختیار مدل قرار گیرد و دقت پیش‌بینی افزایش یابد. روش پیش‌نهادی بااستفاده از برنامه‌های تی‌کَس و اِسکِجول از مجموعه محک زیمنس مورد ارزیابی قرار گرفته است. در آزمایش‌ها روش پیشنهادی روی برنامه تی‌کَس دقت ۹۷/۹۵ و روی برنامه اِسکِجول دقت ۶۰/۲۷ نشان داد. روش پیش‌نهادی به مشخصات کامل نرم‌افزار تحت آزمون نیاز ندارد. پس کافی است اطلاعات مربوط به نوع و تعداد متغیرهای ورودی برنامه تحت آزمون مهیا باشد. به این ترتیب میزان دخالت انسانی در تولید خودکار اوراکل آزمون کاهش می‌یابد.

واژه‌های کلیدی: آزمون نرم‌افزار، آزمون خودکار، اوراکل آزمون، پوشش کد، آزمایه.

* نویسندهٔ مسئول

۱- مقدمه

آزمـــون نرم‌افـــزار از فعالیت‌های مهـــم چرخه حیات نرم‌افزار است که برای بهبود کیفیت نرم‌افزار مورد استفاده قرار می‌گیرد [۱، ۱٤]. آزمون فرایند پرهزینه‌ایست و منابع محاسباتی زیادی می‌طلبد. به‌علاوه آزمون زمان‌گیر است و این موضـــوع تحویل نرم‌افزار را به‌تأخیـــر می‌اندازد یا تأثیـــر منفی بر کیفیت آن می‌گذارد. برای کاهش هزینه‌های آزمون، محققان سعی کرده‌اند که آن‌را خودکار نمایند [۲]. برای خودکارسازی کامل آزمون، همه مراحل و اجزای آن باید خودکار شـــوند [۳]. چون آزمون به اوراکل نیاز دارد، خودکارســـازی تولید اوراکل برای خودکارســـازی کامل آزمون لازم می‌شود.

اوراکل روالی اســـت برای تولیـــد خروجی موردانتظار و تعییـــن این‌که اجرای نرم‌افزار حیـــن آزمون موفق بوده اســـت یا خیر. ســـاخت اوراکل ســـه چالش اساسی دارد. چالش اول تولید خودکار دامنه خروجی است. ارائه دستی خروجی‌های موردانتظار می‌تواند دشـــوار و گران باشـــد. به‌طور کلی، خروجی‌های موردانتظار براساس مشخصات نرم‌افـــزار، دانش خبـــره دامنه و اطلاعات برنامه‌ســـاز از رفتاری که نرم‌افزار باید داشـــته باشـــد، به‌صورت دستی تولیـــد می‌گردد [۲]. یـــک اوراکل خـــودکار آزمون نیاز به خودکارســـازی تولید دامنـــه خروجـــی دارد. چالش دوم نگاشت دامنه ورودی به دامنه خروجی به‌صورت خودکار اســـت. یک اوراکل آزمون باید نتایج موردانتظار برای هر ترکیب ذکرشده در مشـــخصات نرم‌افزار از ورودی‌ها را ارائه نماید. علاوه‌براین، ارائه اوراکل خودکار بدون نگاشت خودکار امکان‌پذیر نمی‌باشـــد [۷]. چالش سوم مقایسه‌گر خودکار اســـت. گاهی اوقات انجام مقایسه مستقیم نقطه به نقطه کافی نیست و لازم اســـت مقایسه‌گر تعدادی آستانه برای مقایسه نتایج واقعی و موردانتظار در نظر بگیرد [۷]. درواقع، ممکن است خروجی‌های واقعی و موردانتظار کمی با یکدیگر متفاوت باشـــند اما بتوان آن‌ها را یکسان درنظر گرفت چراکه نرم‌افزار تحت آزمون به بالاترین سطح دقت نیاز نـــدارد. هم‌چنین، مقداری حد تحمل در طول مقایســـه ممکن است قابل قبول باشد.

برای تولید خودکار اوراکل آزمون باید رفتار نرم‌افزار مدل‌ســـازی شـــود و دامنه ورودی به خروجی نگاشـــت یابد. این‌کار معمولاً ازطریق ســـاخت مـــدل یادگیری روی نمونه‌های آزمایشـــی صورت می‌گیرد. امـــا گاهی داده‌ها جزئیات کافی ندارد تا مدل، رفتار دقیق را یاد بگیرد و دقت تولید اوراکل کاهش می‌یابد. برای حل این مشـــکل، در این مقاله، روشی بهبودیافته برای تولید خودکار اوراکل آزمون ارائه می‌شـــود. این روش اطلاعاتی از پوشش کد نرم‌افزار تحت آزمون اســـتخراج می‌کند. سپس آن‌را به مدل یادگیر تزریق می‌نماید تا جزئیات بیش‌تـــری در اختیار مدل قرار گیرد و دقت پیش‌بینی افزایش یابد. بنابراین هدف این مقاله اختصاصاً بهبود چالش دوم است.

روش پیش‌نهـــادی بااســـتفاده از برنامه‌های تی‌کَس و اِســـکِجول از مجموعه محک زیمنس مـــورد ارزیابی قرار گرفته اســـت. در آزمایش‌ها روش پیش‌نهادی روی برنامه تی‌کَس دقت ۹۷/۹۵ و روی برنامه اِســـکِجول دقت ٦۰/۲۷ نشـــان داد. روش پیش‌نهادی به مشخصات کامل نرم‌افزار تحت آزمون نیاز ندارد. پس کافی است اطلاعات مربوط به نوع و تعداد متغیرهـــای ورودی برنامه تحت آزمون مهیا باشد. به این ترتیب میزان دخالت انسانی در تولید خودکار اوراکل آزمون کاهش می‌یابد.

به‌طور خلاصه، نوآوری‌های این مقاله عبارتند از:

- روشـــی بهبودیافته برای تولید خـــودکار اوراکل آزمون مبتنی بر شبکه عصبی مصنوعی و اطلاعات پوشش
- ارزیابی روش پیش‌نهادی روی دو برنامه محک زیمنس به‌نام تی‌کَس و اِسکِجول و بحثی در نتایج به‌دست آمده و محدودیت‌ها

ساختار ادامه مقاله به این شرح است: بخش دوم به بیان روش پیش‌نهادی همراه با مفاهیم لازم می‌پردازد. بخش سوم جزئیات ارزیابی را ارائه می‌کند. بخش چهارم کارهای مرتبط را مرور می‌کند و نتیجه‌گیری در بخش پنجم بیان می‌شود.

۲- روش پیش‌نهادی

سه چالش برای ارائه اوراکل آزمون خودکار وجود دارد [۱]: تولید دامنه خروجی، نگاشت دامنه ورودی به دامنه خروجی و مقایسه‌گر. مدل اوراکل پیش‌نهادی در این تحقیق سه بخش دارد. در هر بخش ابزاری برای خودکارسازی یکی از چالش‌ها ارائه شده است. بخش اول تولید خودکار مجموعه داده آموزشی حاوی اطلاعات مفید از کد منبع می‌باشد. در این بخش علاوه بر تولید آزمایه‌ها که شامل ورودی‌های برنامه و خروجی‌های متناظرشان هستند، داده‌های کمکی از کد منبع استخراج می‌شود و مجموعه داده کامل، با معنا و گسترده‌ای برای حل چالش تولید دامنه خروجی ایجاد می‌گردد.

با وجود روش‌های مختلف برای بررسی چالش نگاشت دامنه، به دلیل پیاده‌سازی آسان و به‌صرفه بودن در صورت وجود داده‌های مورد نیاز، کاربرد شبکه‌های عصبی مصنوعی بیش‌تر در این زمینه مورد توجه است [۲، ۳-۵، ۶، ۸، ۹]. اما اوراکل‌های مبتنی‌بر شبکه‌های عصبی مصنوعی که قبلاً ارائه شده‌اند در آزمون نرم‌افزارهای پیچیده با دامنه ورودی و خروجی وسیع به اندازه کافی اثربخش نیستند، به طوری‌که آن‌ها دامنه ورودی و خروجی و محدوده مقادیرشان را بسیار کوچک فرض کرده و تعداد ترکیبات مجاز ورودی‌ها را کاهش داده‌اند. بهعبارت دیگر مقادیر ورودی و خروجی در محدوده بسیار کوچک و از پیش شناخته شده‌ای تغییر می‌کند. هم‌چنین خبره دامنه‌ای وجود دارد که از نحوه تغییرات ورودی‌ها و اثرگذاری آن‌ها بر خروجی‌های مختلف برنامه دانش کافی دارد. اما در عمل، در نرم‌افزارهای واقعی دامنه مقادیر ورودی و خروجی‌های حاصل از آن‌ها نمی‌تواند محدود باشد و تعداد ترکیبات مجاز مقادیر ورودی‌ها بسیار گسترده خواهد بود و این مسئله باید در تولید اوراکل آزمون خودکار جهت استفاده در آزمون خودکار مورد توجه ویژه قرار بگیرد.

در شرایطی که نرم‌افزار تحت آزمون پیچیده و بزرگ باشد و محدوده تغییرات ورودی‌ها و خروجی‌ها وسیع باشد، شبکه‌های عصبی مصنوعی نمی‌توانند به راحتی رفتار نرم‌افزار تحت آزمون را مدل کنند [۷] و خروجی‌های مطلوبی را پیش‌بینی نمایند و این هم در مورد اوراکل‌های تک شبکه‌ای قبلی [۳-۵] و هم در مورد اوراکل‌های چند شبکه‌ای [۹] ارائه شده در کارهای قبلی صدق می‌کند. بنابراین در این تحقیق با تأمین داده‌های بیش‌تر برای اوراکل‌های مبتنی‌بر شبکه‌های عصبی مصنوعی، به جای استفاده از چندین شبکه عصبی مصنوعی، می‌توان به‌صورت کامل‌تر و دقیق‌تر به کنترل چالش نگاشت دامنه ورودی به خروجی و برطرف کردن محدودیت‌های انواع قبلی پرداخت. در آخر، یک مقایسه‌گر خودکار که توسط آن دقت اوراکل مشخص می‌شود برای پرداختن به چالش مقایسه استفاده می‌گردد.

۲-۱- پیش‌فرض‌ها و محدودیت‌ها

یکی از پیش‌فرض‌های اساسی این تحقیق به‌کارگیری روش‌های یادگیری ماشینی در تولید اوراکل است. از آن‌جایی‌که شبکه‌های عصبی [۹] مصنوعی در تولید اوراکل موفق بوده‌اند، هدف ارائه مدلی از نرم‌افزار تحت آزمون با استفاده از شبکه عصبی مصنوعی است که رفتار نرم‌افزار تحت آزمون را شبیه‌سازی کند و قادر به تولید دامنه خروجی موردانتظار باشد. سپس چنین مدلی به‌عنوان اوراکل آزمون عمل می‌کند و در خودکارسازی فرآیند آزمون نرم‌افزار به کار رود. از طرفی با در نظر گرفتن این پیش‌فرض‌ها، محدودیت‌هایی مطرح می‌شوند:

(۱) چنین چهارچوبی تنها در آزمون برنامه‌های داده‌محور قابل استفاده است و نمی‌توان از آن برای آزمون حالتی که جریانی از رخدادها باید مشاهده شود، استفاده کرد. برای مثال اگر نتایج واقعی، داده‌های عددی نباشند و یا نتوان آن‌ها را به داده‌های عددی تبدیل کرد و نرمال‌سازی نمود وارسی نتایج واقعی به‌روش ذکر شده امکان‌پذیر نخواهد بود.

(۲) کیفیت راه‌کار پیش‌نهادی مستقیماً با کیفیت ورودی‌ها و شرایطی که براساس آن نتایج تولید می‌شود در ارتباط

است. برای رسیدن به بهترین دقت نباید هیچ ناسازگاری و ابهامی در منطق تولید نتایج نرم‌افزار تحت آزمون باشد. به‌علاوه، ساختار شبکه عصبی مصنوعی و روال آموزشی بسیار مهم هستند، زیرا هر آموزش نامناسبی می‌تواند بطور چشم‌گیری کیفیت اوراکل را کاهش دهد.

۲-۲- توصیف چهارچوب مدل‌سازی اوراکل پیش‌نهادی

شکل (۱) نمای کلی راه‌کار پیش‌نهادی را نمایش می‌دهد. در ابتدا بهکمک مجموعه داده تولید شده شبکه عصبی مصنوعی باید عملکرد نرم‌افزار تحت آزمون را مدل‌سازی نماید. آموزش شبکه‌های عصبی مصنوعی به نمونه‌های آموزشی کاملی که تمام عمل کرد نرم‌افزار تحت آزمون را پوشش می‌دهند نیاز دارد. علاوه براین، اغلب در نرم‌افزارهای پیچیده‌ای که دامنه مقادیر ورودی و خروجی گسترده‌ای دارند، تنها به‌کارگیری روش‌های جعبه سیاه در تولید داده‌های آموزشی، نمی‌تواند به شبکه عصبی در یادگیری رفتار برنامه تحت آزمون و ایجاد مدلی از آن کمک کند. لازم است با روش‌های جعبه سفید به کد منبع دسترسی یافت و اطلاعات و داده‌های بیش‌تری به شبکه در مرحله آموزش تزریق کرد. چنین اطلاعاتی در زمان اجرای برنامه تحت آزمون و با انجام فراکدگذاری در کد برنامه، به ازای هر دسته از ورودی‌ها به‌دست می‌آید و استفاده از این داده‌ها در کنار هر دسته از ورودی‌ها و خروجی‌های متناظرشان، مجموعه داده آموزشی کامل‌تر و مفیدتری را ایجاد می‌کند که در یادگیری رفتار نرم‌افزار تحت آزمون به شبکه عصبی مصنوعی کمک می‌نماید. زیرا این اطلاعات می‌تواند نحوه تغییرات در خروجی‌ها را در اثر تغییرات در ورودی‌ها نمایان سازد.

از آن‌جایی‌که شبکه‌های عصبی مصنوعی تنها با داده‌های عددی کار می‌کنند بعد از این‌که مجموعه داده آموزشی تکمیل شد، لازم است همه داده‌ها شامل ورودی‌های برنامه، خروجی‌های متناظرشان و داده‌های کمکی به داده‌های عددی تبدیل و نرمال‌سازی شوند. سپس داده‌های عددی حاصل بعنوان نمونه‌های آموزشی در آموزش شبکه عصبی مصنوعی به‌کار می‌روند. بعد از آموزش شبکه عصبی می‌توان از آن بعنوان اوراکل استفاده نمود.

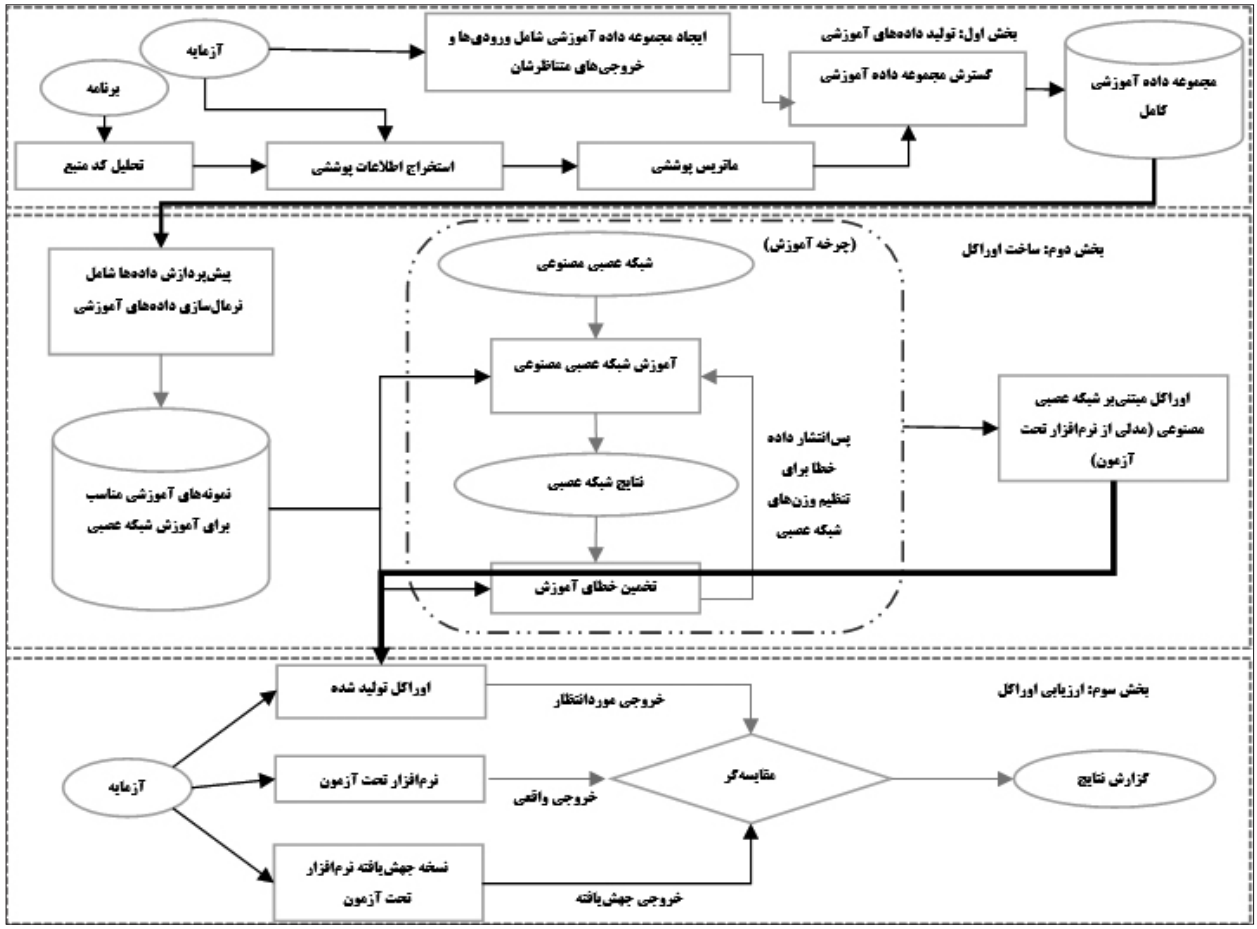
خروجی‌های موردانتظار توسط شبکه عصبی آموزش دیده تولید می‌شوند. بعد از آموزش شبکه عصبی مصنوعی، کیفیت اوراکل از طریق انجام آزمون جهش سنجیده می‌شود. در این مرحله به دو نسخه از برنامه‌های مورد مطالعه نیاز داریم: یک نسخه بدون خطا (نسخه طلایی) به‌منظور تولید خروجی‌های صحیح و مورد انتظار از برنامه و یک نسخه جهش‌یافته از برنامه که خطاهایی به آن تزریق شده است.

در گام آخر ارزیابی اوراکل انجام می‌شود. هدف، اندازه‌گیری و سنجش کیفیت اوراکل و توانایی آن در یافتن خطاهای تزریق شده است. بنابراین اوراکل برای آزمون نسخه جهش‌یافته به‌کار می‌رود و نتایج حاصل با نتایج نسخه طلایی مقایسه می‌شود. همان‌طور که در شکل (۱) مشخص شده است چهارچوب پیش‌نهادی از سه بخش اصلی تشکیل شده است.

۲-۳- تولید مجموعه داده آموزشی

تولید مجموعه داده آموزشی شامل تولید آزمایه‌ها و تولید داده‌های کمکی در کنار هر آزمایه می‌شود. در استاندارد آی‌تریپِل‌ای، آزمایه این چنین تعریف شده است: مجموعه‌ای از ورودی‌های آزمون، اجرا و نتایج موردانتظار که برای هدف خاصی نظیر آزمودن مسیر خاصی از برنامه یا بررسی انطباق با یک نیازمندی خاص، توسعه داده شده است. خروجی آزمایه می‌تواند قبول یا رد باشد. در روش‌های قبلی [۳-۵، ۶، ۸، ۹] مجموعه داده آموزشی تنها از طریق تولید آزمایه‌ها ایجاد می‌شود. سپس این مجموعه داده آموزشی برای آموزش شبکه عصبی مصنوعی و مدل‌سازی رفتار نرم‌افزار تحت آزمون به‌کار می‌رود.

در نرم‌افزارهای بزرگ‌تر که عملیات و محاسبات پیچیده‌ای دارند اغلب تنها به‌کارگیری همه آزمایه‌ها که



شکل ۱: نمای کلی راه‌کار پیش‌نهادی

کل عملکرد نرم‌افزار را پوشش دهند، منجر به آموزش موفق شبکه عصبی مصنوعی نمی‌شود و شبکه نمی‌تواند رفتار کامل و صحیح نرم‌افزار تحت آزمون را یاد بگیرد. از طرفی، در چنین شرایطی تنها استفاده از چندین شبکه عصبی مصنوعی نیز کافی نیست.

از این‌رو، لازم است فرآیند آموزش شبکه‌های عصبی مصنوعی تقویت شود، که در این راستا علاوه بر تغییر در پارامترهای شبکه عصبی می‌توان نمونه‌های آموزشی فراهم شده را غنی‌تر کرد.

رویکرد به‌کارگرفته شده در این تحقیق در زمره روش‌های جعبه سفید قرار دارد. در روش‌های جعبه سفید آزمون‌گر نیاز دارد به کد منبع برنامه تحت آزمون دسترسی داشته باشد. در این‌جا فرض بر این‌است که چنین امکانی فراهم است. به این‌منظور لازم است کد منبع مورد تحلیل قرار گیرد و درون کد برنامه فراکدگذاری صورت گیرد تا در زمان اجرا اطلاعات بیشتری در مورد برنامه تحت آزمون به‌ازای هر آزمایه به‌دست آید و این اطلاعات به‌همراه اطلاعات مربوط به هر آزمایه، نمونه‌های آموزشی شبکه عصبی را فراهم نمایند.

در مرحله اول از فرآیند تولید مجموعه داده آموزشی، ابتدا با وجود داشتن دانش اندکی از نوع و ماهیت ورودی‌ها و خروجی‌های برنامه تحت آزمون می‌توان بخشی از ورودی‌های برنامه را مشخص نمود و در واقع بخشی از داده‌های آزمون را تولید کرد. این کار می‌تواند با کمک ابزارهای تولید آزمایه[1] به‌صورت خودکار انجام شود. سپس باتوجه به نوع زبان برنامه‌سازی و مشخصات دردسترس از برنامه تحت آزمون توابعی را ایجاد نمود تا به‌طور خودکار بتوان خروجی‌های مورد انتظارشان را نیز تولید کرد. در راه‌کار پیش‌نهادی تمرکز اصلی بر روی

1- Test Case Generation Tools

نوع اطلاعات کمکی و نحوه اســتخراج اطلاعات بیش‌تر و مفیدتر از اجرای هر آزمایــه بر روی برنامه تحت آزمون می‌باشد که با وجود تغییر نسخه‌های مختلف برنامه تحت آزمون همواره قابل کاربرد باشند. به‌عبارت دیگر نوعی از اطلاعات انتخاب شود که در آزمون هر نسخه‌ای از برنامه به‌خوبــی رفتار برنامــه و نحوه تغییــر خروجی‌ها در اثر تغییر در ورودی‌ها را نشان دهد. بعد از استخراج اطلاعات کمکی، این اطلاعات با آزمایه‌های متناظرشان ترکیب شده و نمونه‌های آموزشــی کامل‌تری را ایجــاد می‌کنند که به آموزش بهتر شــبکه عصبی مصنوعی و مدل‌سازی رفتار نرم‌افزار تحت آزمون کمک می‌نمایند.

۲-٤- جمع‌آوری اطلاعات پوشش آزمون

به‌کمــک ابزارهــای تحلیل کد منبع گوناگــون می‌توان اطلاعــات متنوعی از کــد منبع به‌ازای هر اجرا اســتخراج کرد. اطلاعات پوشش آزمون که به‌ازای اجرای هر آزمایه بــر روی برنامه مورد آزمون بهدســت می‌آیــد از جمله این اطلاعات اســت. اطلاعات پوشــش آزمایه‌های برنامه توســط ابزار جمع‌آوری پوشش کد محاسبه می‌شود. این ابــزار آزمایه‌ها را بــر روی کد برنامه اجــرا می‌کند و در خروجی، به‌ازای آزمایه‌های مختلف، میزان پوشش عناصر مختلف برنامه را مشــخص می‌کند. بدین معنا که به ازای هر آزمایه مشخص می‌شــود، چه عناصری از برنامه در سطوح مختلف رده و روش و هم‌چنین در سطح کل برنامه پوشش داده شده است. در استخراج اطلاعات پوشش، باید مشخص شود چه نوع اطلاعاتی در این فایل‌ها برای ایجاد مجموعه داده آموزشی و متعاقباً در آموزش شبکه عصبی مصنوعی برای یادگیری رفتار مورد انتظار برنامه مفیدتر هستند.

از آن‌جایی‌که هدف استخراج اطلاعاتی است که در بین نسخه‌های مختلف برنامه، به خوبی رفتار برنامه را درازای اجرای آزمایه‌های مختلف نمایان ســازند، باید مشــخص کرد در فایل‌هــای خروجی حاصل کدام بخش از اطلاعات تولید شــده این هدف را ارضا می‌کنند. ابتدا فایل‌های تولید شــده توسط ابزار جمع‌آوری پوشــش کد تجزیه می‌شود و ســپس مشخص می‌شــود هر عنصر از برنامه از جمله توابع، بلوک‌های کد مشــخص، کل خطــوط برنامه، تعداد فراخوانی‌هــای تابــع و غیره به چه میزان توســط اجرای هر آزمایه پوشــش داده شده اســت. میزان اجرای هریک از توابع موجود در ســاختار برنامه تحت آزمون و تعداد فراخوانی‌های هر تابع از جمله اطلاعاتی هستند که در بین نســخه‌های مختلف برنامه که ممکن است حاوی خطاهای برنامه‌نویسی از جمله دستورات شرطی نادرست، شمارنده حلقه نادرســت، تعریف و مقداردهی نادرست یک متغیر و غیره باشند، به‌ازای اجرای آزمایه‌های یکسان، مقادیرشان تغییرات چشم‌گیری نخواهد داشت. از این‌رو برای استخراج و به کارگیری در مجموعه داده آموزشی در کنار هر آزمایه مفید خواهد بود. استخراج اطلاعات پوشش آزمون از جمله میزان پوشــش هر تابع و تعــداد فراخوانی‌های هر تابع با کمک امکانات موجود در زبان سی‌شــارپ، برای خواندن و پردازش فایل‌ها و اســتخراج الگوهای مشخص شده از درون هر فایل متنی از طریق نوشتن روش‌های لازم انجام می‌گیرد. داده‌های استخراج شــده را می‌توان در قالب یک ماتریس به نام ماتریس پوشش ذخیره نمود.

در واقــع، غنی‌ســازی نمونه‌های آزمایشــی از طریق افــزودن داده‌های کمکی در کنار هر آزمایه، زمانی حاصل می‌شود که این اطلاعات کمکی به‌ازای آزمایه‌های مختلفی که مســیرهای متفاوتی را پوشــش و مــورد آزمون قرار می‌دهند و مقادیر خروجی مختلفــی تولید می‌کنند، دارای تغییرات زیادی باشــند. در این‌صورت در آموزش شــبکه عصبی مصنوعی اطلاعاتی به کار می‌رود که نحوه تغییرات خروجی را به‌ازای تغییرات در ورودی‌ها به طور واضح‌تری منعکس می‌کنــد و می‌تواند یادگیری رفتــار برنامه مورد آزمون توسط شبکه عصبی مصنوعی را ارتقا بخشد.

در این تحقیق بر اســتخراج داده‌های مربوط به میزان پوشــش هر تابع از برنامه و تعداد فراخوانی‌های هر تابع بهــازای اجرای هر آزمایه و هم‌چنین میزان اجرای خطوط

برنامه، میزان اجرای بلوک‌های برنامه و تعداد فراخوانی‌های انجام شده در هر بار اجرا تمرکز شده است.

۲-۵- ابزار تحلیل کد منبع جی‌کاو

ابزار جی‌کاو[۲] برای تحلیل پوشش کد منبع استفاده می‌شود. این ابزار به‌طور دقیق مشخص می‌کند هر عبارت در برنامه چند بار اجرا شده است. ابزار جی‌کاو به‌عنوان یک ابزار استاندارد در کنار جی‌سی‌سی[۳] استفاده می‌شود. در واقع جی‌کاو اطلاعاتی را در مورد چگونگی اجرای بخش‌های کد برنامه ارائه می‌دهد. با استفاده از جی‌کاو یک نسخه از فایل منبع ایجاد می‌شود که حاوی فرکانس‌های اجرا است. ابزار جی‌کاو هیچ داده‌ای مبتنی‌بر زمان تولید نمی‌کند و تنها بر روی کدهای کامپایل شده با جی‌سی‌سی قابل اعمال است. ابزار جی‌کاو در انتها یک فایل سابقه تولید می‌کند که در آن مشخص شده است در هر اجرا با آزمایه‌های مختلف هر خط چند بار اجرا شده است.

۲-۶- ساخت اوراکل مبتنی‌بر شبکه عصبی مصنوعی

منظور از ساخت اوراکل ایجاد شبکه عصبی مصنوعی است که به‌عنوان اوراکل استفاده می‌شوند. در اوراکل‌های تک شبکه‌ای کل رفتار نرم‌افزار تحت آزمون توسط یک شبکه عصبی مدل‌سازی می‌شود و اگر برنامه چندین خروجی داشته باشد، همه خروجی‌ها به‌وسیله همان یک شبکه پیش‌بینی و تولید می‌شوند که طبق آن‌چه که قبلاً در مورد اوراکل‌های تک شبکه‌ای گفته شد، این اوراکل‌ها دقت بالایی ندارند. در اوراکل‌های چندشبکه‌ای [۹] به‌ازای هر خروجی برنامه تحت آزمون یک شبکه عصبی مصنوعی ساخته می‌شود که ممکن است شبکه‌های عصبی مصنوعی ساختارهای مختلفی داشته باشند اما لایه ورودی همه آن‌ها یکسان است زیرا مجموعه ورودی برای همه آن‌ها یکسان است. در صورتی‌که از شبکه‌های پرسپترون چند لایه استفاده شود از آن‌جایی‌که هر شبکه عصبی مصنوعی وظیفه تولید یکی از خروجی‌ها را دارد، شبکه‌های عصبی مصنوعی در لایه آخر یعنی لایه خروجی یک نورون دارند، در حالی‌که در لایه اول یعنی لایه ورودی به تعداد ورودی‌های برنامه تحت آزمون به‌علاوه تعداد ورودی‌های کمکی که از کد برنامه به‌ازای هر دسته از ورودی‌ها استخراج شده است نورون وجود دارد.

هر شبکه عصبی مصنوعی براساس آزمون و خطا می‌تواند یک یا بیش‌تر از یک لایه پنهان داشته باشد که تعداد نورون‌ها در لایه‌های پنهان نیز با آزمون و خطا به‌دست می‌آید. باتوجه به تعداد خروجی‌های برنامه تحت آزمون ممکن است یک یا چند شبکه عصبی مصنوعی مورد نیاز باشد که ساختار لایه ورودی و لایه خروجی همه شبکه‌های عصبی مصنوعی یکسان است اما پارامترهای لایه پنهان، تابع فعال‌سازی، نرخ یادگیری، ضریب مومِنتوم و تعداد دورهای آموزشی هر شبکه می‌تواند متفاوت باشد. بعد از تزریق داده‌های ورودی و خروجی مورد نیاز از مجموعه داده آموزشی ایجاد شده به شبکه عصبی پارامترهای نرخ یادگیری، ضریب مومنتوم، تعداد چرخه‌های آموزشی، تعداد لایه‌های پنهان و تعداد نورون‌های لایه پنهان باید به‌طور دستی در مرحله آموزش شبکه توسط آزمون‌گر تنظیم شود. سایر پارامترهای مورد نیاز شبکه عصبی مصنوعی توسط الگوریتم آموزش و ابزار پیاده‌سازی مورد استفاده به‌صورت خودکار تنظیم می‌شود. با این وجود، در صورتی‌که در پایان آموزش شبکه عصبی مصنوعی آموزش دیده مورد رضایت نباشد می‌توان مقدار هریک از پارامترها را تغییر داد. علاوه‌بر این، در صورت تغییر نمونه‌های آموزشی ناشی از هر گونه تغییرات برنامه تحت آزمون، فرآیند آموزش باید تکرار شود.

ابزارهای مختلفی برای پیاده‌سازی شبکه‌های عصبی مصنوعی وجود دارد از قبیل مؤلفه‌های مربوط به شبکه عصبی مصنوعی در نرم‌افزار مَتلب، کتابخانه‌های نورون‌دات‌نِت[۴] و اِنکاگ[۵] در ویژوال استودیودات‌نت. در این تحقیق از کتابخانه اِنکاگ استفاده شده است. در طول

2- Gcov
3- GCC
4- Neuron Dot Net
5- Encog

فرآینـد آموزش ایـن وزن‌ها پالایش می‌شـود تا خروجی مطلوب حاصل شود. از آن‌جایی‌که آموزش شبکه عصبی با مقادیر تصادفی وزن‌ها شروع می‌شود در اجراهای مختلف یک برنامه نتایج مختلفی به‌دسـت می‌آیـد. گاهی وزن‌ها از نقاط خوبی شـروع می‌شوند اما گاهی به‌قدری انتخاب‌های تصادفی دور از هم اتفاق می‌افتد که آموزش شبکه شکست می‌خورد و باید دوباره فرآیند تکرار شود. تابع فعال‌سازی انتخاب شده اَکتیوِیشن‌سیگموئید[6] است. این تابع فعال‌سازی تنها در لایه پنهان و لایه خروجـی تأثیرگذار خواهد بود و تنظیم آن برای لایه ورودی بی‌تأثیر می‌باشد.

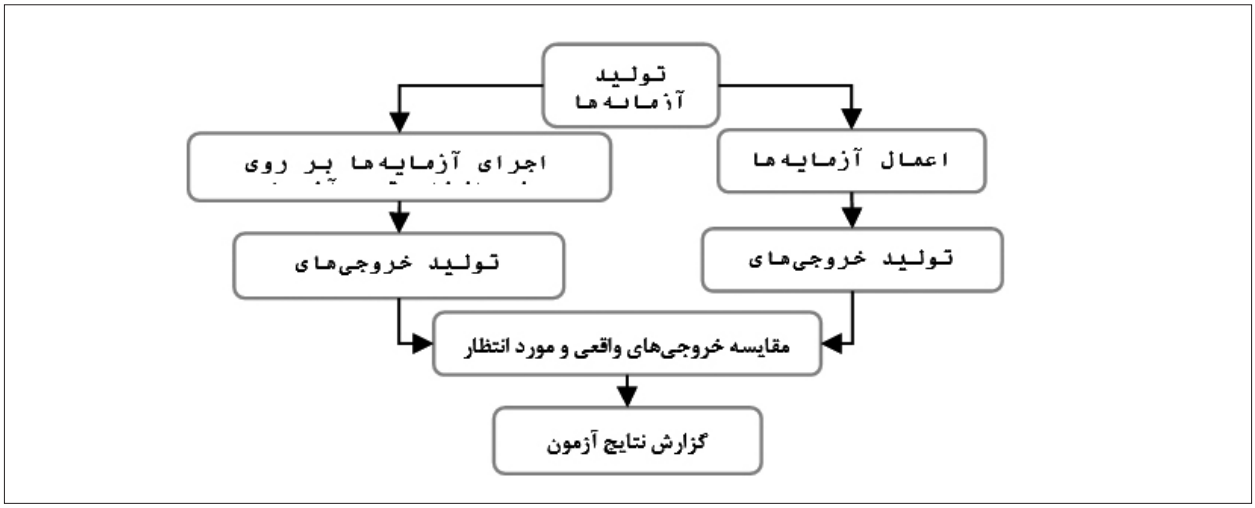
تنظیـم نورون بایاس در هر لایـه از طریق تنظیم یک مقدار بولی در زمان ساخت لایه صورت می‌گیرد. نورون بایاس هر لایه به لایه بعدی آن متصل می‌شـود. به‌همین دلیـل این مقدار تنها برای لایه ورودی و لایه پنهان مقدار بولی درسـت و لایه خروجی به آن نیـازی ندارد. بعد از سـاخت و تنظیم لایه‌هـا، مجموعه داده آموزشـی برای شـبکه مشـخص می‌گردد. الگوریتم آموزش پس‌انتشار انتخاب می‌شـود که نیاز به تنظیم نرخ یادگیری و ضریب مومنتـوم دارد. در انتها تعداد دورهای آموزشـی برای تکـرار آموزش تا رسـیدن به میزان خطـای مورد نظر مشخص می‌گردد. نرخ یادگیری مشخص می‌کند وضعیت فعلی چقدر روی گام بعـدی اثر بگذارد. ضریب مومِنتوم مشـخص می‌کند گام‌های قبلی چقدر روی گام بعدی اثر بگذارند. این پارامتر به هموار کردن تغییرات کمک می‌کند. در بخـش ایجـاد مجموعه داده آموزشـی نحـوه ایجاد مجموعه داده کاملی که می‌توان از آن به عنوان نمونه‌های آموزشی استفاده نمود شرح داده شد. اما قبل از استفاده از مجموعـه داده لازم اسـت تمام مقادیـر غیرعددی به مقادیر عددی در بازه‌های مشـخص تبدیل شـوند، زیرا شـبکه عصبی نمی‌تواند از ورودی‌هـای غیرعددی مثل ورودی‌های رشـته‌ای یاد بگیرد. مقادیر دودویی درست[7] و نادرسـت[8] به یک و صفر تبدیل می‌شوند. مقادیر متنی به‌صـورت عدد تعریف می‌شـود و به هـر کلمه یک عدد اختصـاص داده می‌شـود. همان‌طور که گفته شـد تابع فعال‌سازی انتخاب شده سیگموئید می‌باشد. این تابع تنها بـر روی داده‌های مثبت اعمال می‌شـود و خروجی‌های مثبت تولیـد می‌کند. از این‌رو، می‌توان مقادیر موجود در مجموعه داده آموزشـی مورد نظر را طبق فرمول (۱) در بازه دلخواه نرمال‌سازی نمود.

$$u = \frac{x - x_{min}}{x_{max} - x_{min}}(h - l) + l \qquad (۱)$$

بـا اسـتفاده از این فرمول همـه مقادیر x در بـازه [l, h] نرمال‌سـازی می‌شـوند. x_{min} و x_{max} به‌ترتیـب کمترین و بیشـترین مقدار x هسـتند. نرمال‌سـازی مقادیر موجود در مجموعـه داده آموزشـی با پـردازش مقادیر و یافتن بزرگ‌تریـن و کوچک‌تریـن مقدار موجود در هر سـتون مجموعـه داده آموزشـی از طریـق ایجاد توابـع لازم و توابع کتاب‌خانه‌ای زبان سی‌شـارپ برای محاسبه فرمول نرمال‌سـازی ذکر شـده صورت می‌گیرد. بعد از تعریف شـبکه عصبی مصنوعی و انجام نرمال‌سـازی مجموعه داده آموزشـی گام بعدی آغاز فرآیند آموزش است. ابزار اِنکاگ به‌صورت خودکار فرآیند آموزش را انجام می‌دهد. آزمون‌گر فقط کافی اسـت تعداد دورهـای آموزش، نرخ یادگیری، ضریب مومنتوم، تعـداد لایه‌های پنهان و تعداد نورون‌هـای لایه پنهان را مشـخص کند. سـپس فرآیند آموزش به‌طور خودکار به‌کمک ابزار مورد استفاده انجام می‌شـود. در واقع، تنها فعالیت دسـتی تعریف هر شـبکه عصبـی مصنوعی و تنظیـم برخی پارامترها اسـت، بقیه فرآیند آموزش به‌صورت خودکار به‌وسیلهٔ ابزارهای شبکه عصبی مورد استفاده صورت می‌گیرد.

رسـیدن به شـبکه‌ای بـا بالاترین کیفیـت نیازمند فرآیند سـعی و خطا است. فرآیند سعی و خطا نیازمند تغییراتی اسـت که انسـان ایجاد می‌کند. هم‌چنین اگر نرم‌افـزار تحت آزمون تغییر کنـد این تغییرات برروی نمونه‌های آموزشـی اثرگذار خواهد بود و باید فرآیند

6- ActivationSigmoid
7- true
8- false



شکل ۲: کاربرد اوراکل آزمون در فرآیند آزمون خودکار نرم‌افزار

آمـوزش را تکـرار نمـود. فرآیند آمـوزش زمانی‌که تکـرار فرآینـد به تعـداد چرخه‌های آموزشـی تعیین شـده انجام شـود و یا نرخ خطا به مقدار تعیین شـده برسد، متوقف می‌شود و شـبکه آموزش دیده ساخته می‌شـود. برای ارزیابی دقت مدل سـاخته شده شبکه آموزش دیده برای پیش‌بینی بخشی از داده‌ها استفاده می‌شـود. هم‌چنین مدل شـبکه عصبی آمـوزش دیده بـا کمک روش ارزیابی متقاطع ۱۰گانه مورد سـنجش قـرار می‌گیـرد تا همـه نمونه‌های آموزشـی موجود در مجموعـه داده آموزشـی در دو مرحلـه آموزش و آزمـون به کار رونـد. مقادیـر اِماِس‌ای، اِماِی‌پی‌ای و دقـت نیـز بـرای مـدل سـاخته شـده محاسـبه می‌گردد.

۲-۷- مقایسه‌گر خودکار

مقایسـه‌گر وظیفـه دارد نتایج واقعـی (خروجی‌های نرم‌افزار تحت آزمون) را با نتایج مورد انتظار (خروجی‌های اوراکل) مقایسـه کند و انحراف‌ها و اختلاف‌ها را گزارش کند. علاوه‌بر این، مقایسـه‌گر نیاز دارد تعدادی آسـتانه بـرای تنظیم دقت اوراکل در نظر بگیـرد. در واقع، مقدار آسـتانه میزان اختلافی اسـت که می‌تـوان بین خروجی واقعـی و خروجی مورد انتظار نادیـده گرفت. در بخش ارزیابـی توضیحات بیشـتری در مورد آسـتانه‌ها بیان می‌شود.

۲-۸-استقرار اوراکل خودکار

بعد از ارزیابی مدل اوراکل پیش‌نهادی و رضایت‌بخش بودن نتایج حاصـل از ارزیابی، می‌توان اوراکل سـاخته شـده را در خودکارسـازی فرآیند آزمون به کار برد. در واقـع می‌توان یک راه‌انداز آزمون خـودکار ایجاد کرد که فعالیت‌هـای فرآیند آزمـون نرم‌افـزار را به‌طور خودکار انجام می‌دهد و در آن از مدل اوراکل پیش‌نهادی اسـتفاده شـود. وارسـی آزمایه‌ها (قابل قبول بودن یا نبودن نتایج آزمایه‌هـا که در اصل پذیرش یا عـدم پذیرش یک آزمون است) به‌کمک مدل اوراکل پیش‌نهادشده به‌صورت خودکار انجام می‌شود. شکل (۲) این روند را نشان می‌دهد.

راه انداز آزمون خودکار آزمایه‌ها را ایجاد می‌کند و آن‌ها را برروی نرم‌افزار تحت آزمون اجرا می‌کند. در مرحله بعد مدل اوراکل آزمون ایجاد شده براساس چهارچوب پیش‌نهادی وظیفه دارد خروجی‌های مورد انتظار را تولید کند و سپس یک مقایسه‌گر خودکار وظیفه دارد خروجی‌های واقعی به‌دست آمده از اجرای نرم‌افزار تحت آزمون را با خروجی‌های مورد انتظار مقایسه کند و در نهایت گزارشی از نتیجه آزمون انجام شده تولید کند. تمامی این مراحل خودکارند.

جدول ۱: ورودی‌های برنامه تی‌کَس

ورودی	نوع	محدوده مقادیر
Cur_Vertical_Sep	صحیح	
High_Confidence	بولی	۰، ۱
Two_of_Three_Reports_Valid	بولی	۰، ۱
Own_Tracked_Alt	صحیح	
Own_Tracked_Alt_Rate	صحیح	
Other_Tracked_Alt	صحیح	
Alt_Layer_Value	صحیح	۰، ۱، ۲، ۳
Up_Separation	صحیح	
Down_Separation	صحیح	
Other_RAC	صحیح	۰، ۱، ۲
Other_Capability	صحیح	۱، ۲
Climb_Inhibit	صحیح	۰، ۱، ۲

۳- ارزیابی

به‌منظور ارزیابی فن پیش‌نهادی آزمایش‌هایی انجام می‌شوند.

۳-۱- پیاده‌سازی

برای ارزیابی رویکرد پیش‌نهادی لازم است برنامه‌هایی در نظر گرفته شود. مجموعه برنامه‌های محک زیمنس از جمله برنامه‌هایی هستند که اغلب در تحقیقات مربوط به آزمون نرم‌افزار به کار گرفته می‌شوند. زیمنس حاوی هفت برنامه به زبان سی می‌باشد. هر برنامه بین ۷ تا ۲۱ تابع دارد و اندازه برنامه‌ها بین ۱۳۸ تا ۵۱۶ خط کد می‌باشد. در مجموعه زیمنس برای هر یک از برنامه‌ها نسخه‌های خطاداری از برنامه موجود است. هم‌چنین مجموعه آزمون‌های مختلف که معیارهای متفاوتی را مورد آزمون قرار می‌دهند، برای هر یک از هفت برنامه نیز وجود دارد. همه برنامه‌های موجود در مجموعه زیمنس جهت استفاده در پیاده‌سازی و ارزیابی مدل اوراکل پیش‌نهادی مورد بررسی قرار گرفتند. باتوجه به فرضیات و محدودیت‌های ذکر شده، در این تحقیق تمرکز تنها بر ورودی‌ها و خروجی‌های عددی و یا ورودی‌ها و خروجی‌هایی است که بتوان آن‌ها را به عدد تبدیل کرد. از این‌رو، در این تحقیق تنها امکان استفاده از دو برنامه موجود در زیمنس می‌باشد.

۳-۲- برنامه مورد مطالعه اول: تی‌کَس

یکی از برنامه‌های مجموعه زیمنس برنامه تی‌کَس است. تی‌کَس یک سیستم نرم‌افزار توکار به‌حساب می‌آید که در هواپیما قرار می‌گیرد و هدف آن جلوگیری از برخورد و تصادف هوایی است. دامنه ورودی برنامه تی‌کَس ۱۲ ورودی دارد که در جدول (۱) نشان داده شده است. دامنه خروجی این برنامه تنها شامل یک خروجی از نوع عدد صحیح می‌باشد و سه مقدار ۰، ۱ و ۲ خواهد داشت.

مقادیر ۱ و ۲ به‌ترتیب مشخص می‌کند جهت حرکت هواپیما به سمت پایین و به سمت بالا باشد. مقدار صفر جهتی را مشخص نمی‌کند و بدین معنی است که سیستم نتوانسته جهتی را تعیین کند. تی‌کَس از ۱۴۱ خط کد و ۹ تابع تشکیل شده و ۱۵۷۸ آزمایه و ۴۰ نسخه خطادار دارد.

۳-۳- ایجاد مجموعه داده آموزشی برنامه تی‌کَس

گام اول در تولید مجموعه داده‌های آموزشی، فراهم کردن مجموعه‌ای از آزمایه‌ها شامل ورودی‌ها و خروجی‌های متناظرشان است. در این‌جا از همان آزمایه‌های موجود زیمنس استفاده شده است.

گام دوم جمع‌آوری اطلاعات پوشش آزمون می‌باشد. این کار با استفاده از ابزار جمع‌آوری پوشش کد انجام شده است. برای جمع‌آوری اطلاعات پوشش یک برنامه لازم است تغییراتی در فایل برنامه اعمال شود تا ابزار بتواند اطلاعات پوشش را از برنامه جمع‌آوری کند. بعد از تحلیل جعبه سفید برنامه و به‌دست آوردن فایل‌های حاوی اطلاعات پوشش آزمون، لازم است این فایل‌ها پردازش شود و بخشی از اطلاعات که در اجراهای مختلف دارای تغییرات زیادی بوده‌اند و در آشکارساختن تغییرات در خروجی به‌ازای تغییر در ورودی‌ها، مؤثرتر بوده‌اند، استخراج شود. این‌کار با روشی متدی به زبان سی‌شارپ، برای پردازش تمامی فایل‌های تولید شده به‌ازای هر

جدول ۲: نمونه‌ای از اطلاعات پوششی کد برنامه تی‌کَس

کد آزمایه	تعداد فراخوانی‌های تابع سوم	تعداد فراخوانی‌های تابع هشتم	بلوک‌های پوشش داده شده از تابع هشتم	خطوط اجراشده	انشعاب‌های اجرا شده
T۱	۲	۱	٪۶۶	٪۸۳/۰۸	٪۷۲/۷۳
T۲	۲	۱	٪۶۹	٪۷۶/۹۲	٪۶۰/۶۱
T۳	۰	۱	٪۲۳	٪۴۶/۱۵	٪۱۵/۱۵
T۴	۰	۱	٪۲۰	٪۴۶/۱۵	٪۱۲/۱۲

آزمایه و استخراج و ذخیره اطلاعات مورد نظر انجام شد. اطلاعات پوشش آزمون اســتخراج شده عبارتند از: تعداد فراخوانی‌های هر تابع در هر اجرا، میزان بلوک‌های پوشش داده شــده از هر تابع در هر اجرا، میزان خطوط برنامه که به‌ازای هر آزمایه اجرا می‌شــوند، میزان انشعاب‌های اجرا شــده از برنامه به‌ازای هر آزمایه، میزان انشعاب‌هایی که حداقل یک‌بار در یک اجرای مشــخص از آن‌ها عبور شده اســت و میزان فراخوانی‌های صورت گرفته در یک اجرای مشخص.

قابل ذکر است که در این مرحله هر نسخه اصلاح شده و یا نسخه تحت آزمون برنامه می‌تواند در تولید اطلاعات پوششــی اســتفاده شــود. باتوجه به آن‌چه که در بخش ارزیابی راه‌کار پیش‌نهادی گفته خواهد شد، در این‌جا فرض می‌شود نسخه‌های مختلف مورد آزمون تفاوت ساختاری زیــادی با یکدیگر ندارنــد. هم‌چنین باتوجه بــه این مورد تلاش شــده اســت با انجام تحلیل‌هایی بر روی فایل‌های خروجی حاصــل از ابزار جمع‌آوری اطلاعات پوششــی، اطلاعاتی استخراج شــود که با وجود تغییرات مختلف در هر نسخه مقادیرشــان تغییرات کم و قابل چشم‌پوشی در بین نسخه‌های مختلف برنامه داشته باشند و بتوان از آن‌ها در تولید مجموعه داده آموزشی استفاده نمود.

اکنون باید ماتریس پوششــی بــا مجموعه داده حاوی اطلاعات ورودی‌های برنامه و خروجی‌های متناظرشــان ترکیب شــوند و مجموعه داده آموزشــی گسترده نهایی ســاخته شــود. برای این‌کار، کد هر آزمایه در پایگاه داده مربوطه و در ماتریس پوشش مشخص می‌گردد. سپس هر سطر ماتریس پوشش در کنار سایر اطلاعات آزمایه مورد نظر درج می‌شــود و پایگاه داده بروزرسانی می‌شود. در نهایت، پایگاه داده حاصل به‌عنوان مجموعه داده آموزشی که شامل نمونه‌های آموزشی مورد نیاز در ساخت اوراکل اســت، مورد اســتفاده قرار می‌گیرد. در جدول (۲) بخشی از اطلاعات پوششــی استخراج شــده برای چند نمونه از آزمایه‌ها نشان داده شده است. مثلاً جدول (۲) نشان می‌دهد بــا اجرای آزمایه T1 بر روی نســخه تحت آزمون برنامه، ۲ بار تابع ســوم و ۱ بار تابع هشــتم فراخوانی می‌شوند، ٦٦٪از بلوک‌های تابع هشتم پوشش داده می‌شود، ٪۸۳/۰۸ از خطوط برنامه و ٪۷۲/۷۳ از دستورهای انشعابی برنامه (مثل شرط و حلقه) در زمان اجرای این آزمایه اجرا شده‌اند.

۳-٤- ساخت اوراکل برای برنامه تی‌کَس

پس از ایجاد نمونه‌های آموزشــی، باید شبکه عصبی مصنوعی که وظیفه مدل‌سازی رفتار برنامه تحت آزمون را دارد ســاخته شــود و به کمک نمونه‌های آموزشی که در بخش قبلی آماده شــدند آموزش ببیند. شــبکه عصبی مصنوعی تعریف شــده، شــبکه پِرسِــپترون ســه لایه با الگوریتم آموزشی پس‌انتشار و تابع فعال‌سازی سیگموئید می‌باشد. مرحله آموزش با بخشی از نمونه‌های آموزشی انجام شــد. بعــد از انجام آموزش، مرحله آزمون شــبکه عصبی مصنوعی با اســتفاده از سایر نمونه‌های آموزشی که در مرحله آموزش به کار نرفتند، صورت گرفت و دقت و میزان خطای پیش‌بینی شــبکه آموزش دیده محاســبه شــد. برای رسیدن به شبکه مناسب از فرآیند سعی و خطا استفاده شد. در هر بار اجرا و آزمایش پارامترهای مختلف شبکه عصبی را تغییر داده و نتایج بررسی شد تا در نهایت

شبکه عصبی قابل قبول ایجاد شود.

کیفیت آموزش شـــبکه عصبی با خطای اِماِس‌ای نشان داده می‌شود که با محاسبه مربع فاصله بین خروجی‌های نمونه‌های آموزشـــی و خروجی تولید شده توسط شبکه، در انتهای مرحله آزمون به‌دســـت می‌آیـــد. اگر نمونه‌های آموزشـــی کل منطق برنامه را پوشـــش ندهنـــد و حاوی اطلاعاتی از نحوه عملکرد برنامه درازای تغییرات ورودی‌ها نباشد، حتی اگر اِماِس‌ای خیلی نزدیک به صفر باشد، شبکه حاصـــل به‌خوبی رفتار برنامه را یـــاد نمی‌گیرد. پس بهتر اســـت مجموعه داده آموزشـــی همه قوانینی که شبکه باید مدل‌ســـازی کند را پوشـــش دهد. اِماِس‌ای و اِماِی‌پی‌ای با فرمول‌های (۲) تا (۵) محاسبه می‌شود [۱۰].

$$E = P_i - \overline{P_i} \quad (۲)$$

$$APE = (P_i - \overline{P_i}) * 100 \quad (۳)$$

$$MSE = (\sum_{i=1}^{n} E^2)/n \quad (۴)$$

$$MAPE = (\sum_{i=1}^{n} |APE|)/n \quad (۵)$$

در فرمول‌هـــای بالا P_i ، $\overline{P_i}$ و n بـــه ترتیب مقدار واقعی، مقدار پیش‌بینی شـــده توســـط شـــبکه و تعداد نمونه‌های پیش‌بینی شده هستند. براســـاس آن‌چه که بیان شد، لازم است دقت شبکه عصبی آموزش دیده نیز سنجیده شود. به این‌منظور مقدار دقت برای مدل ساخته شده با استفاده از فرمول (٦) محاسبه می‌شود [۱۰].

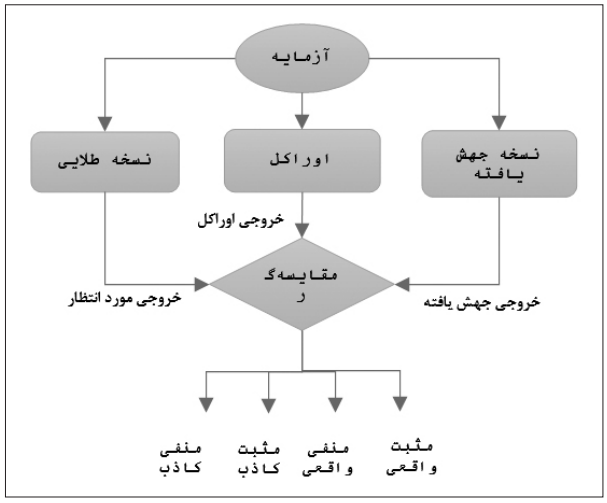
$$Precision = \frac{TruePositive}{TruePositive+FalsePositive} \quad (۶)$$

که مخرج کســـر از چپ به‌ترتیب تعـــداد مثبت‌های واقعی و مثبت‌های کاذب پیش‌بینی شـــده توســـط مدل می‌باشد. به‌منظور محاســـبه دقت مدل لازم است مقادیر مختلفی از آســـتانه و حد تحمل اختلاف بین مقادیر واقعی نمونه‌های مـــورد آزمـــون و مقادیـــر پیش‌بینی شـــده توســـط مدل، بهصورت متغیر آزمایش شـــود و مقـــداری انتخاب گردد. هرچه مقدار آســـتانه کوچک‌تر در نظر گرفته شود، تعداد کمتری از خروجی‌های پیش‌گویی شـــده توســـط اوراکل مورد پذیرش هستند و درنتیجه دقت مدل ایجاد شده کمتر خواهد بود. هرچه مقدار آســـتانه بهســـمت مقدار بزرگ‌تر مـــی‌رود، میزان اختلاف قابل قبول بین مقدار خروجی‌های واقعی و مقدار پیش‌گویی‌شده توسط مدل افزایش می‌یابد و اوراکل خروجی‌های صحیح بیش‌تری را تولید می‌کند و دقت مدل بیشتر خواهد بود.

مشخصات شـــبکه عصبی مصنوعی نهایی عبارتند از: تعـــداد نورون‌های لایه ورودی ۲۹، تعـــداد لایه‌های پنهان ۱، تعـــداد نورون‌های لایـــه پنهان ٤۰، تعـــداد نورون‌های لایه خارجـــی ۱، نرخ یادگیـــری ۰/۰۱، ضریـــب مومنتوم ۰/۰۱، تعـــداد دورهای آموزشـــی ۲۰۰۰، مقـــدار اِماِس‌ای ۰/۰۰۰۰٦۲۷، اِماِی‌پی‌ای ۰/۲۸٪. مدت زمان آموزش ۱۰٤۲ نمونه آموزشـــی توســـط این شـــبکه عصبی مصنوعی ۸ دقیقه می‌باشد. در مرحله آزمون نیز خروجی ٥۳٦ آزمایه مورد پیش‌بینی قرار گرفت. هم‌چنین دقت برای آستانه‌های ۰/۰۲، ۰/۰۲٥ و ۰/۰۳ به‌ترتیب ۹۳/۸٤٪، ۹٥/۳۳٪ و ۹۷/۹٤٪ هستند.

میزان اِماِس‌ای و اِماِی‌پی‌ای در شـــبکه عصبی پذیرفته شده نزدیک به صفر است و دقت پیش‌بینی این شبکه مقدار قابل قبولی دارد. این نشـــان می‌دهد مرحله آموزش شبکه عصبی موفقیت‌آمیز بوده و مـــدل خروجی‌های صحیحی را پیش‌بینی می‌کند. به‌منظور ســـنجش بیشـــتر دقت شبکه عصبی مصنوعی، از ارزیابی متقاطع ۱۰گانه استفاده شد.

در ارزیابـــی متقاطع ۱۰گانه همه نمونه‌های موجود در مجموعه داده آموزشـــی در طول اجراهای مختلف هم در مرحله آمـــوزش به کار می‌روند و هـــم در مرحله آزمون شـــبکه عصبی ایجاد شده استفاده می‌شـــوند. در هر اجرا دقت پیش‌بینی مدل مورد ســـنجش قـــرار می‌گیرد تا مقدار قابل اطمینانی از دقت به‌دســـت آید. تفاوت در مقادیر دقت به‌دســـت آمده از هر اجرا نشـــان دهنده میـــزان اثرگذاری متفاوتی اســـت کـــه هریک از نمونه‌ها در یادگیری شـــبکه عصبی دارد. بعد از انجام موفقیت‌آمیز مرحله‌های آموزش و آزمون شـــبکه عصبی، شـــبکه عصبی مصنوعی نهایی اوراکل را درست می‌کند و سپس اوراکل آماده است که در



شکل ۳: ارزیابی اوراکل پیش‌نهادی [۹]

فرآیند آزمون به‌کار رود.

برنامه تی‌کَس برنامه‌ای پیچیده، خطاخیز و بااهمیت است و آزمون پوششی کامل آن که همه مسیرهای ممکن را پوشش دهد مورد نیاز می‌باشد. به‌علاوه، ارائه اوراکل آزمون کامل برای این برنامه نیاز دارد آزمون‌گر تمام قوانین و نیازمندی‌ها را بداند و یا از یک خبره دامنه ماهر استفاده شود. هریک از این شرایط بسیار هزینه‌بر و یا دشوار هستند. بنابراین کاربرد اوراکل آزمون خودکار در آزمون چنین برنامه‌ای پیچیدگی آزمون و هزینه آن‌را به طور چشم‌گیری کاهش می‌دهد. راه‌کار پیش‌نهادی به‌منظور تسهیل فرآیند آزمون، اوراکل آزمون خودکار را ارائه نمود. اگرچه، لازم است داده‌های ضروری و محیط، قبل از انجام فرآیند آزمون آماده‌سازی شود.

۳-۵- ارزیابی مدل اوراکل ساخته شده برای تی‌کَس

بعد از انجام فرآیند آزمون شبکه عصبی مصنوعی و ساخته شدن اوراکل آزمون، نوبت به بخش سوم از چهارچوب پیش‌نهادی یعنی ارزیابی مدل اوراکل ساخته شده، می‌رسد. این‌کار بااستفاده از آزمون جهش انجام می‌شود. کیفیت اوراکل مبتنی‌بر شبکه عصبی پیشنهادی باید از طریق کاربرد آن در آزمون برنامه‌های نرم‌افزاری برای یافتن خطاها، بررسی و سنجیده شود. روال ارزیابی در شکل (۳) آمده است.

در آزمون جهش، تعدادی خطا به نرم‌افزار تحت آزمون تزریق می‌شود تا نسخه‌های جهش‌یافته ایجاد شوند. سپس توانایی اوراکل ایجاد شده در تشخیص این خطاها ارزیابی می‌گردد. خطاهایی از قبیل اشتباهات رایجی که معمولاً برنامه نویسان انجام می‌دهند. هم‌چنین وجود نسخه طلایی (نسخه بدون خطا) برای تولید خروجی‌های مورد انتظار صحیح ضروری است. در نهایت، نتایج حاصل از نسخه طلایی، نسخه جهش‌یافته و اوراکل آزمون ایجاد شده با کمک مقایسه‌گر خودکار با یکدیگر مقایسه می‌شود. مقایسه‌گر خودکار در این مرحله اختلاف بین نتایج را حساب کرده، سپس این فاصله را با مقدار آستانه از پیش‌تعریف شده‌ای مقایسه می‌کند و بر اساس آن در مورد نتایج تولید شده توسط اوراکل تصمیم‌گیری می‌کند. گزارش‌هایی که مقایسه‌گر ارائه می‌کند به‌صورت زیر خواهد بود [۹]:

مثبت واقعی: سه خروجی یکسان هستند. گزارش عدم وجود خطا توسط مقایسه‌گر؛ هیچ خطایی در نسخه خطادار و اوراکل وجود ندارد.

منفی واقعی: خروجی اوراکل و خروجی موردانتظار یکسانند اما خروجی نسخه جهش‌یافته با این دو متفاوت است. یعنی خروجی اوراکل صحیح است و اوراکل به درستی خطایی را در نسخه جهش‌یافته تشخیص داده است.

مثبت کاذب: خروجی‌های اوراکل و نسخه جهش‌یافته یکسانند و با خروجی مورد انتظار متفاوتند. در واقع اوراکل و نسخه جهش‌یافته خروجی نادرست تولید کرده‌اند که دلالت بر وجود خطا در نسخه جهش‌یافته و اوراکل دارد؛ اوراکل نتوانسته خطا را پیدا کند.

منفی کاذب: خروجی اوراکل متفاوت از خروجی مورد انتظار و خروجی نسخه جهش‌یافته است اما این دو خروجی یکسانند و این نشان دهنده اوراکل معیوب است.

مطالعات قبلی [۲، ٤، ٥، ۱۱] نشان می‌دهد معیارهای زیر را می‌توان برای ارزیابی اوراکل‌های آزمون به‌کار گرفت.

از این‌رو به‌منظور نشان‌دادن کیفیت مدل ساخته شده این معیارها اندازه‌گیری شد:

صحت[9]: نتایج اوراکل چقدر دقیق هستند. یعنی چند درصد از خروجی‌های مورد انتظار تولید شده توسط اوراکل دقیق است. این پارامتر از طریق مقایسه نتایج مورد انتظار صحیح که از نسخه طلایی به‌دست می‌آید، با نتایج به‌دست آمده از اوراکل محاسبه می‌شود.

دقت: یعنی دقت مقایسه‌گر در مقایسه نتایج واقعی و موردانتظار چگونه است. به‌منظور اندازه‌گیری دقت اوراکل، با درنظر گرفتن یک مقدار آستانه نتایج واقعی و مورد انتظار با یکدیگر مقایسه می‌شوند.

خطای اشکال در دسته‌بندی[10]: میزان نتایج اشتباه که مقایسه‌گر تولید کرده است.

عملی‌بودن یا استفاده‌پذیری[11]: میزان خطاهای تزریق‌شده شناسایی‌شده توسط اوراکل پیش‌نهادی از طریق این معیار اندازه‌گیری می‌شود. معیار عملی بودن متفاوت از دقت است؛ معیار دقت نتایج درست و نادرست را با هم درنظر می‌گیرد، اما این معیار فقط آزمایه‌های نادرست را در نظر می‌گیرد.

ارزیابی مدل اوراکل پیش‌نهادی به شرح زیر است: (۱) آزمایه‌ها بر روی نسخه‌های طلایی و جهش‌یافته اجرا می‌شود، هم‌چنین به اوراکل نیز اعمال می‌شود. (۲) از آن‌جایی‌که نسخه طلایی بدون خطا است، خروجی‌های به‌دست آمده از اجرای آن نیز عاری از خطا هستند و به‌عنوان خروجی‌های مورد انتظار در نظر گرفته می‌شوند. (۳) همه خروجی‌های به‌دست آمده از اوراکل، نسخه طلایی و نسخه جهش‌یافته با یکدیگر مقایسه شده و میزان اختلاف آن‌ها با یکدیگر محاسبه می‌شود. میزان اختلاف بیشتر از مقدار آستانه در نظر گرفته شده حاکی از وجود خطا می‌باشد. (٤) خروجی‌های اوراکل و خروجی‌های مورد انتظار (حاصل از نسخه طلایی) با یکدیگر مقایسه شده و مربع فاصله بین آن‌ها به‌عنوان خطای مطلق اوراکل محاسبه می‌شود.

آستانه: از آن‌جایی‌که مقدار اِم‌اِس‌ای دقیقاً صفر نیست، یک آستانه برای هر خروجی در نظر گرفته می‌شود که مشخص می‌کند تا چه اندازه فاصله بین خروجی‌های مورد انتظار و خروجی‌های اوراکل قابل چشم‌پوشی است. درصورتی‌که مقدار خروجی مورد انتظار به‌علاوه/منهای آستانه برابر با خروجی اوراکل باشد این دو خروجی یکسان درنظر گرفته می‌شود. در غیر این‌صورت اوراکل صحیح نیست و خطای مثبت یا منفی کاذب گزارش می‌شود. باتوجه به این‌که اکنون آزمون جهش برای ارزیابی اوراکل انجام می‌شود مقایسه‌گری که در بخش قبل ایجاد شد باید کمی تغییر داده شود تا خروجی‌های به‌دست آمده از نسخه طلایی را به‌جای خروجی‌های اوراکل، به‌عنوان خروجی مورد انتظار درنظر بگیرد و نتایج اوراکل را به‌همراه نتایج جهش‌یافته ارزیابی کند. در واقع، اختلاف بین خروجی مورد انتظار و خروجی اوراکل محاسبه می‌شود. هم‌چنین فاصله بین خروجی جهش‌یافته و اوراکل نیز محاسبه می‌گردد. سپس این فاصله‌های به‌دست آمده با آستانه مورد نظر مقایسه می‌شود و نتایج مقایسه نمایش داده می‌شود.

رابطه بین دقت و صحت: آستانه‌ها دقت را مشخص می‌کنند و به‌طور مستقیم بر روی صحت تأثیر می‌گذارند. هرچه مقدار آستانه بزرگ‌تر باشد، باعث می‌شود خروجی‌های نادرست اوراکل صحیح درنظر گرفته شود. در مقابل، مقدار کوچک آستانه منجر می‌شود خروجی‌های صحیح اوراکل نادرست به‌حساب آید. به‌علاوه، هرچه مقدار آستانه کوچک‌تر باشد، اوراکل دقیق‌تر است زیرا مقادیر اختلاف بسیار کوچک بین خروجی‌های مورد انتظار و خروجی‌های واقعی قابل چشم‌پوشی است. از طرف دیگر، اگر مقدار آستانه بزرگ باشد اوراکل صحت بیش‌تری دارد زیرا مقدار مجاز اختلاف بین خروجی‌های مورد انتظار و خروجی‌های واقعی افزایش می‌یابد و تعداد بیشتری از خروجی‌های اوراکل قابل قبول خواهند بود. اما این باعث می‌شود خروجی‌های نادرست به‌عنوان خروجی

9- Accuracy
10- Misclassification
11- Practicality

صحیح درنظر گرفته شـــود. بنابراین، صحت و دقت رابطه عکس دارند: هرچه دقت بیش‌تر شود صحت اوراکل کاهش می‌یابد و برعکس، هرچـــه دقت کاهش یابد صحت اوراکل افزایش پیدا می‌کند. نحوه انتخاب آســـتانه بســـتگی به نوع نرم‌افـــزار تحت آزمون خواهد داشـــت. قطعاً نرم‌افزارهای حیاتی بـــه قابلیت اطمینان بالایی نیاز دارند و بهتر اســـت مقدار آستانه در وارسی آن‌ها کوچک درنظر گرفته شود. زیرا وجود هر خطایی در آن‌ها منجر به خرابی و شکست سیستم می‌شود.

۳-۶- ارزیابـــی مدل اوراکل پیش‌نهـــادی برای برنامه تی‌کَس

ارزیابی به‌وســـیله آزمون جهش و با ایجاد دو نســـخه از برنامه‌های مورد مطالعه انجام می‌شـــود. اولین نسخه، نســـخه طلایی است که نســـخه بدون خطای برنامه مورد مطالعه است و برای ایجاد خروجی‌های مورد انتظار به‌کار می‌رود. نســـخه دوم نســـخه خطادار (جهش‌یافته) برنامه اســـت که خطاهای رایج برنامه‌نویســـی به آن تزریق شده است. از آن‌جایی‌که شبکه عصبی تنها تقریبی از برنامه‌های مورد مطالعه اســـت، برخی از خروجی‌های آن ممکن است نادرست باشـــد. به‌عبارت دیگر، رده‌های جهش‌یافته خود عامل ایجاد خطاها هســـتند که یافتـــن خطاهای آن‌ها دلیل اصلی انجام فرآیند آزمون است.

۳-۷- آزمون جهش برای برنامه تی‌کَس

در مجموعه زیمنس ٤٠ نســـخه خطـــادار برای برنامه تی‌کَس فراهم شده است که در آزمون جهش برنامه مورد مطالعـــه اول به کار گرفته شـــد و اوراکل برای پیش‌گویی خروجی برنامه و تشـــخیص وجود خطا در برنامه به‌کار گرفته شـــد. مجموعه آزمایه‌های درنظرگرفته شـــده برای آزمایش اوراکل پیش‌نهادی برروی ٤٠ نسخه خطادار اجرا شدند و خروجی‌های آن‌ها به‌عنوان خروجی‌های جهش‌یافته ذخیره شـــد و در مقایســـه با خروجی‌های پیش‌بینی شده توســـط اوراکل اســـتفاده شدند. براســـاس تعاریفی که از پارامترهای کیفیتی راه‌کار پیش‌نهادی شده‌است، هریک از این پارامترها براساس نتایجی که مقایسه‌گر تولید می‌کند، به‌صورت زیر قابل محاسبه هستند:

دقـــت: میزان دقت اوراکل پیش‌نهادی با مقدار آســـتانه در نظر گرفته‌شده برای مقایسه خروجی‌های مورد انتظار، خروجی‌هـــای جهش‌یافتـــه و خروجـــی اوراکل در روال مقایسه‌گر برابر اســـت. در جدول (۳) نتایج ارزیابی اعمال اوراکل پیش‌نهادی برروی برنامه تی‌کَس روی ۲۰ نســـخه خطادار اول آن نشان داده شده است.

صحت

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} * 100 \quad (۷)$$

میزان خطای اشکال در دسته‌بندی

$$MissClassificationError = \frac{FalsePositive + FalseNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} * 100 \quad (۸)$$

میزان عملی‌بودن

$$Practicality = \frac{TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} * 100 \quad (۹)$$

میزان صحت اوراکل پیش‌نهادی ۹۷/۹۵٪ می‌باشـــد. یعنی اوراکل پیش‌نهادی در ۹۷/۹۵٪ از موارد وارســـی آزمایه‌ها توانســـته خروجی‌های مورد انتظار را تولید کند و عملکرد اوراکل پیش‌نهادی بـــر روی برنامه مورد مطالعه موفقیت آمیز بوده اســـت. در واقع، این مقدار نشـــان‌دهنده میزان پیش‌گویی‌های صحیحی است که توســـط اوراکل آزمون پیش‌نهادی انجام شده اســـت. باتوجه به این‌که بخش‌های قابل توجهـــی از فرآیند تولید و پیاده‌ســـازی مدل اوراکل پیش‌نهـــادی به‌صورت خودکار انجام می‌شـــود، از این‌رو می‌تـــوان اوراکل آزمون خودکار ارائه شـــده را در آزمون خودکار برنامه مـــورد مطالعه، با اطمینان به خروجی‌های پیش‌بینی شده، به‌کار برد.

۳-۸- ساخت اوراکل برای برنامه اِسکِجول

برنامه دیگری که جهت ارزیابی عملکرد اوراکل آزمون پیش‌نهادی استفاده شده است، برنامه اِسکِجول از مجموعه

جدول ۳: نتایج ارزیابی عملکرد اوراکل پیش‌نهادی بر روی برنامه تی‌کَس؛ ستون‌های جدول به‌ترتیب از سمت راست: نسخه خطادار، عملی‌بودن، خطای اشکال در دسته‌بندی، مثبت واقعی، منفی واقعی، مثبت کاذب، و منفی کاذب.

ن. خ.	ع. ب.	خ. ا. د. ط.	مثبت و.	منفی و.	مثبت ک.	منفی ک.
V۱	٪۱۹/۵۹	٪۲/۰۵	۴۲۰	۱۰۵	۰	۱۱
V۲	٪۱۱/۱۹	٪۲/۰۵	۴۶۵	۶۰	۰	۱۱
V۳	٪۰/۷۵	٪۲/۰۵	۵۲۱	۴	۰	۱۱
V۴	٪۱/۳۱	٪۲/۰۵	۵۱۸	۷	۰	۱۱
V۵	٪۰/۱۹	٪۲/۰۵	۵۲۴	۱	۰	۱۱
V۶	٪۱/۶۸	٪۲/۰۵	۵۱۶	۹	۰	۱۱
V۷	٪۲/۹۹	٪۲/۰۵	۵۰۹	۱۶	۰	۱۱
V۸	٪۰/۰۰	٪۲/۰۵	۵۲۵	۰	۰	۱۱
V۹	٪۰/۹۳	٪۲/۰۵	۵۲۰	۵	۰	۱۱
V۱۰	٪۱/۸۷	٪۲/۰۵	۵۱۵	۱۰	۰	۱۱
V۱۱	٪۱/۸۷	٪۲/۰۵	۵۱۵	۱۰	۰	۱۱
V۱۲	٪۰/۹۳	٪۲/۰۵	۵۲۰	۵	۰	۱۱
V۱۳	٪۰/۰۰	٪۲/۰۵	۵۲۵	۰	۰	۱۱
V۱۴	٪۸/۰۲	٪۲/۰۵	۴۸۲	۴۳	۰	۱۱
V۱۵	٪۰/۱۹	٪۲/۰۵	۵۲۴	۱	۰	۱۱
V۱۶	٪۱۱/۳۸	٪۲/۰۵	۴۶۴	۶۱	۰	۱۱
V۱۷	٪۲/۹۹	٪۲/۰۵	۵۰۹	۱۶	۰	۱۱
V۱۸	٪۴/۶۶	٪۲/۰۵	۵۰۰	۲۵	۰	۱۱
V۱۹	٪۲/۹۹	٪۲/۰۵	۵۰۹	۱۶	۰	۱۱
V۲۰	٪۲/۹۹	٪۲/۰۵	۵۰۹	۱۶	۰	۱۱

زیمنس می‌باشد. برنامه اِسکِجول اولویت‌دهی وظایف و کارها را انجام می‌دهد. دامنه ورودی برنامه اِسکِجول شامل سه ورودی به‌صورت عدد می‌باشد که شماره فرآیندهای ورودی را مشخص می‌کند. هم‌چنین یک فایل حاوی دستورات نیز مشخص می‌شود که در آن دستوراتی جهت اعمال برروی هر فرآیند وجود دارد.

مجموعه آزمایه‌های درنظرگرفته شده برای آزمایش اوراکل پیش‌نهادی بر روی نُه نسخه خطادار اجرا شدند و خروجی‌های آن‌ها به‌عنوان خروجی‌های جهش‌یافته ذخیره شد و با خروجی‌های پیش‌بینی شده توسط اوراکل مقایسه شدند. روال ارزیابی این برنامه نیز کاملاً مانند برنامه قبلی است ولی به‌خاطر کمبود فضا تنها نتیجه نهایی ارزیابی ارائه می‌شود. در جدول (٤) نتایج ارزیابی اعمال اوراکل پیشنهادی بر روی برنامه مورد مطالعه دوم، اِسکِجول، به‌همراه محاسبه پارامترهای کیفی (صحت، دقت، عملی‌بودن و خطای اشکال در دسته‌بندی) نشان داده شده است.

محدوده مقادیر خروجی برنامه اِسکِجول بسیار گسترده است و با نرمال‌سازی مقادیر خروجی در مجموعه داده‌های آموزشی بیش از ۱۵۰۰ حالت خروجی مختلف ایجاد شد. بدیهی است تولید اوراکل با دقت قابل قبول، جهت پیش‌گویی هر یک از این خروجی‌ها دشوار می‌باشد.

مدل اوراکل پیش‌نهادی در آزمون برنامه اِسکِجول و وارسی آزمایه‌های این برنامه در ٦۰/۲۷٪ از موارد نتایج صحیح را گزارش کرد. باتوجه به تنوع در مقادیر خروجی این برنامه این مقدار از صحت مدل پیش‌نهادی قابل قبول می‌باشد. بنابراین، مقادیر خروجی مورد انتظار پیش‌گویی شده توسط اوراکل ساخته شده قابل اطمینان خواهد بود.

۳-۹ - مقایسه مدل اوراکل پیش‌نهادی با اوراکل ارائه شده قبلی

به‌منظور مقایسه اوراکل پیش‌نهادی با کارهای قبلی، اوراکل‌های ایجاد شده در کار قبلی [۹] نیز برروی برنامه‌های مورد مطالعه اعمال شدند و مقایسه‌ای بین آن‌ها صورت گرفت. از جمله محدودیت‌هایی که در اوراکل‌های مبتنی‌بر شبکه‌های عصبی مصنوعی قبلی اعم از تک‌شبکه‌ای و چند شبکه‌ای، وجود دارد لزوم فراهم ساختن مجموعه داده آموزشی حاوی تمام آزمایه‌های ممکن است که همه ترکیبات ممکن از مقادیر ورودی و خروجی‌های متناظرشان را در برگیرد. فرآیند آموزش شبکه عصبی مصنوعی مدل‌سازی شده براساس شرایط

جدول ۴: نتایج ارزیابی عملکرد اوراکل پیش‌نهادی بر روی برنامه اِسکِجول؛ ستون‌های جدول به‌ترتیب از سمت راست: نسخه خطادار، عَملی‌بودن، خطای اشکال در دسته‌بندی، مثبت واقعی، منفی واقعی، مثبت کاذب، و منفی کاذب.

ن. خ.	ع. ب.	خ. ا. د. ط.	مثبت و.	منفی و.	مثبت ک.	منفی ک.
V۱	۰	٪۳۹/۷۳	۵۷۵	۰	۱	۳۷۸
V۲	٪۲/۹۴	٪۳۹/۷۳	۵۴۷	۲۸	۴۴	۳۳۵
V۳	٪۱/۴۷	٪۳۹/۷۳	۵۶۱	۱۴	۳۴	۳۴۵
V۴	٪۳/۰۴	٪۳۹/۷۳	۵۴۶	۲۹	۵۶	۳۲۳
V۵	٪۱/۹۹	٪۳۹/۷۳	۵۵۶	۱۹	۱۷	۳۶۲
V۶	۰	٪۳۹/۷۳	۵۷۵	۰	۱	۳۷۸
V۷	٪۱/۰۵	٪۳۹/۷۳	۵۶۵	۱۰	۱۶	۳۶۳
V۸	٪۰/۴۲	٪۳۹/۷۳	۵۷۱	۴	۴	۳۷۵
V۹	۰	٪۳۹/۷۳	۵۷۵	۰	۰	۳۷۹

درنظر گرفته شـــده برای ایجاد اوراکل در [۹] بر روی دو برنامه تی‌کَس و اِسکِجول تکرار شد. بالاترین دقت به دست آمده در پیش‌بینی خروجی توسط مدل اوراکل پیش‌نهادی بـــا در نظر گرفتن مقـــدار ۰/۰۳ به عنوان آســـتانه، ۹۷/۹۴ درصد می‌باشـــد. در حالی‌که، اوراکل ایجاد شده با روش قبلی برای برنامه مورد مطالعه اول با در نظرگرفتن مقدار آستانه ۰/۰۸، دقتی معادل ۴۳/۴۷ درصد دارد.

۴- کارهای مرتبط

لاست و همکارانش [۱۲] یک اوراکل آزمون‌گر پس‌نمایی جعبه ســـیاه کاملاً خودکار را به‌کمک شبکه فازی اطلاعات معرفـــی کردنـــد. زی[12] و همکارانـــش [۱۳] برنامه‌ریـــزی هـــوش مصنوعی را به‌عنوان یـــک اوراکل آزمون خودکار رابـــط گرافیکی کاربر به‌کاربردنـــد. ونمالی[13] و همکارانش [۵] اوراکل مبتنی‌بـــر شـــبکه عصبی مصنوعـــی که تنها از یک شبکه عصبی [۱۱] تشکیل شـــده بود، پیشنهاد کردند. آگروال[14] و همکاران [٤] راه‌کار مشابهی را برای حل مسئله دسته‌بندی مثلث بررسی کردند. در [٦] کار آن‌ها دنبال شد. اوراکل‌های تک‌شـــبکه‌ای برای آزمون ســـاختارهای ایجاد تصمیم [۱۱] و وارسی ماژول‌های منطقی پیچیده [۲] به‌کار گرفته شدند. مائو[15] و همکارانش [۳] یک اوراکل تک‌شبکه‌ای برای مدل‌سازی و آزمون توابع پیوسته ایجاد کردند. در این روش نیازی به تولید دستی همه خروجی‌های مورد انتظار نبود، اما اثربخشی مدل برای حالتی که رابطه بین ورودی و خروجی بسیار پیچیده باشد مورد بررسی قرار نگرفت. شـــاه‌امیری و همکارانش [۹] برای نگاشـــت خودکار دامنه ورودی به دامنه خروجی اوراکل‌های چندشبکه‌ای مبتنی بر شبکه‌های عصبی مصنوعی را معرفی کردند. اوراکل ارائه شده اگرچه قابل استفاده برای برنامه‌های پیچیده می‌باشد و می‌تواند برای ارزیابی عملکردهای جدید و قدیمی به‌کار رود و ارزان‌تر نیز می‌باشد اما روش تولید دامنه خروجی موردانتظار به‌کار گرفته شـــده بـــرای برنامه‌هایی با دامنه مقادیر ورودی و خروجی گسترده کارایی و اثربخشی قابل توجهی نخواهد داشت. هم‌چنین این راه‌کار نیاز به مدیریت حافظه زیادی دارد و زمان‌بر می‌باشد.

۵- نتیجه‌گیری

آزمـــون نرم‌افـــزار از فعالیت‌های مهـــم چرخه حیات نرم‌افـــزار اســـت و برای افزایـــش کیفیت نرم‌افـــزار مورد اســـتفاده قـــرار می‌گیرد. هـــدف از تحقیق حاضـــر بهبود پیش‌بینی خروجی مورد انتظـــار در تولید خودکار اوراکل آزمون اســـت. برای این‌منظور، روشـــی برمبنای شـــبکه عصبی و اطلاعات پوشش کد پیش‌نهاد شد. نتایج به‌دست آمده از ارزیابی مدل پیش‌نهـــادی، صحت اوراکل را برای برنامه مورد مطالعه اول ۹۷/۹۵ درصد و برای برنامه مورد مطالعه دوم ٦۰/۲۷ درصد نشان می‌دهد. در تحقیقات آتی برنامه‌های بیشـــتری با روش پیش‌نهادی ارزیابی خواهند شد.

مراجع

1. E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," in IEEE transactions on software engineering, vol. 41, pp. 507-525, 2015.

12- Xie
13- Vanmali
14- Aggarwal
15- Mao

networks for automated software testing,» in Information Technology Journal, vol. 6, pp. 469-474, 2007.
9. S. R. Shahamiri, M. W. Kadir, S. Ibrahim, and S. Z. Mohd-Hashim, "Artificial neural networks as multi-networks automated test oracle," in Automated Software Engineering, vol. 19, pp. 303-334, 2012.
10. J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques: Elsevier, 2011.
11. S. R. Shahamiri, M. W. Kadir, and S. Ibrahim, "An automated oracle approach to test decision-making structures," in 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), pp. 30-34, 2010.
12. S. R. Shahamiri, M. W. Kadir, and S. Z. Mohd-Hashim, "A comparative study on automated software test oracle methods," in Fourth International Conference on Software Engineering Advances (ICSEA'09), pp. 140-145, 2009.
13. Q. Xie and A. M. Memon, "Designing and comparing automated test oracles for GUI-based software applications," in ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 16, p. 4, 2007.
14. P. Ammann and J. Offutt. Introduction to software testing. Cambridge University Press, 2016.

2. S. R. Shahamiri, M. W. Kadir, and S. Ibrahim, "A single-network ANN-based oracle to verify logical software modules," in 2010 2nd International Conference on Software Technology and Engineering (ICSTE), pp. V2-272-V2-276, 2010.
3. Y. Mao, F. Boqin, Z. Li, and L. Yao, "Neural networks based automated test oracle for software testing," in International Conference on Neural Information Processing, pp. 498-507, 2006.
4. K. Aggarwal, Y. Singh, A. Kaur, and O. Sangwan, «A neural net based approach to test oracle,» in ACM SIGSOFT Software Engineering Notes, vol. 29, pp. 1-6, 2004.
5. M. Vanmali, M. Last, and A. Kandel, "Using a neural network in the software testing process," in International Journal of Intelligent Systems, vol. 17, pp. 45-62, 2002.
6. H. Jin, Y. Wang, N. W. Chen, Z. J. Gou, and S. Wang, "Artificial neural network for automatic test oracles generation," in 2008 International Conference on Computer Science and Software Engineering, pp. 727-730, 2008.
7. S. R. Shahamiri, M. W. Kadir, S. Ibrahim, and S. Z. Mohd-Hashim, "An automated framework for software test oracle," in Information and Software Technology, vol. 53, pp. 774-788, 2011.
8. Y. Lu and M. Ye, «Oracle model based on RBF neural