Scientific Journal of Electronical & Cyber Defence
Vol. 10, No. 2, Summer 2022, Serial No. 38

Evaluating Deep Learning Models for Test Data Generation In File Based Fuzzers

M. T. Taghavi*, M. Bagheri

* PhD student, Imam Hossein University (AS), Tehran, Iran

(Received: 11/05/2021, Accepted: 11/12/2021)

ABSTRACT

Fuzzing means repeatedly running the program being tested, by modified inputs, with the aim of finding its vulnerabilities. If the program has a complex input structure, generating modified inputs for fuzzing is not an easy task. The best solution in such cases is to use the input structure of the program under test to produce accurate test data. The problem is that the input structure documentation of program under test may not be available. Human understanding of such complex structures is also hard to achieve, costly, time consuming, and prone to errors. To overcome to above problems, this research proposes the use of machine learning and deep neural networks, which automatically learn the complex structures of program inputs and generate test data tailored to this structure. One of main challenges in this field is choosing the appropriate deep learning model which suits the intended application. In this paper, suitable deep learning models for learning and test data generation in file-based fuzzers are studied. Also, the evaluation is performed by introducing and applying the appropriate performance evaluation parameters. So the recurrent neural network and its derivations are introduced as the best deep learning models for text data. Also, effective parameters considered for performance evaluation include the training time, loss value in training and evaluation time. The loss value as the main parameter is evaluated once in various deep learning models with same structure and again in the same deep learning models with various structures and the best deep learning model is selected and proposed .

Keywords: Fuzzing, Deep Learning, Text Test Data Generation, Performance Evaluation.

* Corresponding Author Email: Ttaghavi@ihu.ac.ir

نشریه علمی "پدافند الکترونیکی و سایبری"
سال دهم، شماره ۲، تابستان ۱۴۰۱، ص ۷۳-۵۷

علمی – پژوهشی

ارزیابی مدل‌های یادگیری عمیق برای تولید داده آزمون در فازرهای مبتنی بر فایل

محمدتقی تقوی[1*]، مسعود باقری[2]

۱- دانشجوی دکترا ، ۲- استادیار، دانشگاه جامع امام حسین (ع)، تهران، ایران

(دریافت: ۱۴۰۰/۰۲/۲۱، پذیرش: ۱۴۰۰/۰۹/۲۰)

چکیده

فازینگ به معنی اجرای مکرر برنامه تحت آزمون با ورودی‌های تغییر یافته، با هدف یافتن آسیب‌پذیری است. در صورتی که ورودی‌های برنامه تحت آزمون دارای ساختار پیچیده‌ای باشند، تولید ورودی‌های تغییر یافته برای انجام فازینگ کار راحتی نیست. بهترین راه حل در این موارد، استفاده از ساختار ورودی برنامه تحت آزمون به‌منظور تولید دقیق داده آزمون است. مشکلی که وجود دارد این است که ممکن است مستندات ساختار ورودی برنامه تحت آزمون در دسترس نباشد. همچنین درک انسانی چنین ساختارهای پیچیده‌ای نیز بسیار مشکل، پرهزینه، زمان‌بر و مستعد خطای انسانی است. برای غلبه بر مشکلات فوق، استفاده از یادگیری ماشین و شبکه‌های عصبی عمیق به‌منظور یادگیری خودکار ساختارهای پیچیده ورودی‌های برنامه و تولید داده آزمون متناسب با این ساختار پیشنهاد شده است. یکی از چالش‌های اصلی در این زمینه، استفاده از مدل یادگیری متناسب با کاربرد مورد نظر است. در این مقاله، مدل‌های یادگیری عمیق مناسب برای یادگیری و تولید داده آزمون در فازرهای مبتنی بر فایل مورد بررسی قرار گرفته است. همچنین با معرفی پارامترهای مناسب برای بررسی کارایی، ارزیابی مدل‌های یادگیری عمیق انجام شده است. بر این اساس، شبکه‌های عصبی بازرخداد و مشتقات آن به‌عنوان بهترین مدل‌های یادگیری عمیق برای داده-های متنی انتخاب شده است. همچنین پارامترهای مؤثر برای ارزیابی کارایی مدل‌های یادگیری عمیق شامل زمان آموزش، میزان خطای مدل‌ها در زمان آموزش و و زمان ارزیابی درنظر گرفته شده است. پارامتر میزان خطا به‌عنوان پارامتر اصلی، یک بار در مدل‌های یادگیری عمیق مختلف با ساختار یکسان و یک بار در مدل‌های یادگیری عمیق یکسان با ساختار متفاوت مورد ارزیابی قرار گرفته و بهترین مدل یادگیری عمیق انتخاب و معرفی شده است.

کلیدواژه‌ها: فازینگ، یادگیری عمیق، تولید داده آزمون متنی، ارزیابی کارایی

۱- مقدمه

نرم‌افزارها همه جا حضور دارند، از سامانه‌های حیاتی همانند نرم‌افزار کنترل تجهیزات صنعتی گرفته تا سامانه‌های پزشکی همانند سامانه‌های تنظیم ضربان قلب بیمار و حتی نرم‌افزارهای خانگی. وجود این طیف گسترده نرم‌افزار، موجب وابستگی حیرت انگیز انسان به تجهیزات پردازش اطلاعات شده است. وابستگی زیاد به فناوری اطلاعات و پیچیدگی در حال رشد نرم‌افزارها موجب افزایش تهدیدات امنیتی همانند افزایش تعداد آسیب‌پذیری‌های قابل بهره‌برداری در نرم‌افزارها شده است.

فازینگ[1] از زمان معرفی به‌عنوان رایج‌ترین روش آزمون نرم‌افزار به‌منظور کشف خطاهای بالقوه معروف شده است [۳-۱]. دلیل این امر آن است که این روش قابلیت کشف انواع بسیار متفاوتی از خطاها و آسیب‌پذیری‌ها در برنامه‌های مختلف را دارد [۴]. فازینگ روش پویای کشف آسیب‌پذیری‌ها و نقاط ضعف در برنامه هدف با استفاده از داده‌های آزمون[2] بدشکل‌سازی شده[3] شامل فایل‌ها، پروتکل‌های شبکه و غیره است. با استفاده از فازینگ می‌توان بسیاری از خطاهای بالقوه موجود در یک برنامه کاربردی را قبل از انتشار عمومی کشف و اصلاح کرد تا در آینده توسط هکرها مورد سوء استفاده قرار نگیرد [۴].

تولید داده‌های متنی مشابه با داده‌های واقعی با استفاده از یادگیری عمیق کاربرد زیادی در حوزه‌های مختلف فناوری اطلاعات همانند ترجمه ماشینی[4]، جوابگویی به سؤالات، توسعه بات‌های پاسخگو و غیره دارد [۵]. یکی از کاربردهای جدید و مهم در این زمینه، تولید داده آزمون برای ارزیابی برنامه‌های تجاری قبل از انتشار عمومی به‌منظور کشف و برطرف کردن آسیب‌پذیری‌های[5] بالقوه موجود در آن‌ها با استفاده از آزمون فاز[1]

* رایانامه نویسنده مسئول: Ttaghavi@ihu.ac.ir

[1] Fuzzing

[2] Test Data
[3] Malformed
[4] Machine Translation
[5] Vulnerabilities

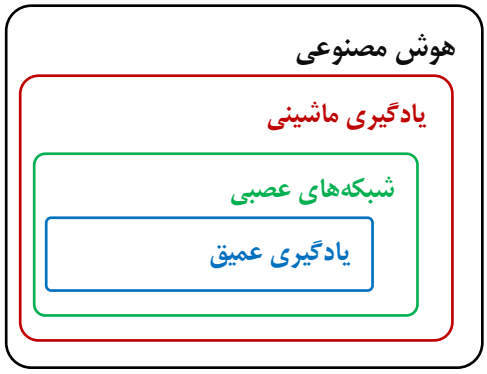
است [۱۰-۶]. به‌منظور تولید داده متنی با استفاده از یادگیری عمیق[۲]، نیاز به بررسی مقوله پردازش زبان طبیعی[۳] در حوزه یادگیری عمیق وجود دارد. پردازش زبان طبیعی حوزه مربوط به مهندسی نرم‌افزار و هوش مصنوعی است که مدیریت زبان انسانی را به عهده دارد [۱۱]. برای این منظور مدل‌های مختلفی ارائه شده است. انتخاب مدل مولد متناسب با کارکرد خاص در پردازش زبان طبیعی تأثیر زیادی در میزان دقت و کارایی خروجی نهایی دارد. به‌منظور بررسی ارزیابی کارایی و دقت مدل‌های شبکه عصبی تولید متن، کارهایی انجام شده است [۵ و ۱۵-۱۲] که در این مقاله به بررسی آن‌ها پرداخته شده است.

با توجه به اینکه انتخاب، مقایسه و ارزیابی مدل‌های شبکه عصبی عمیق باید مبتنی بر کارکرد مورد نظر باشد، در این مقاله به بررسی مدل‌های یادگیری عمیق مناسب برای یادگیری و تولید داده آزمون متنی پرداخته شده است. داده‌های متنی تولید شده با استفاده از مدل‌های یادگیری عمیق به‌عنوان داده آزمون متنی در آزمون فاز مورد استفاده قرار خواهد گرفت. آزمون فاز مورد نظر این مقاله مبتنی بر فایل‌های ساختارمند متنی است. از این جهت، بررسی نقاط قوت و ضعف هر کدام از مدل‌های یادگیری عمیق مبتنی بر کاربرد مورد نظر مقاله یعنی آزمون فاز است. علاوه بر انتخاب مدل‌های یادگیری عمیق، پارامترهای مؤثر بر مدل‌های یادگیری عمیق نیز در مقاله بررسی شده است.

ساختار مقاله به‌صورت زیر است: در بخش دو، به بررسی مفهوم یادگیری عمیق و مدل‌های مولد یادگیری مرتبط با کارکرد مورد نظر مقاله یعنی آزمون فاز مبتنی بر فایل‌های ساختارمند پرداخته شده است. در بخش سه، آزمون فاز فایل‌های ساختارمند متنی مورد بررسی قرار گرفته است. در بخش چهار، کارهای مرتبط یعنی روش‌های ارزیابی مدل‌های مولد یادگیری عمیق متنی و پژوهش‌های مرتبط با تولید داده آزمون برای انجام فازینگ مورد بررسی قرار گرفته است. در بخش پنج، راهکار پیشنهادی یعنی روش پیاده‌سازی مدل‌های یادگیری عمیق مختلف و ارزیابی آن‌ها ارائه شده است. در بخش شش، نتایج حاصل از ارزیابی مورد بررسی قرار گرفته و مدل یادگیری عمیق مناسب انتخاب شده است. بخش هفت شامل نتیجه‌گیری و کارهای آتی است.

۲- یادگیری عمیق و مدل‌های مولد داده متنی

از نظر سلسله مراتبی، یادگیری عمیق به‌عنوان یکی از شاخه‌های یادگیری ماشین است که آن هم به نوبه خود بخشی از هوش مصنوعی است. شکل (۱) جایگاه کلی یادگیری عمیق را نشان می‌دهد.



شکل (۱): جایگاه یادگیری عمیق در هوش مصنوعی

راهبردهای اصلی در یادگیری عمیق عبارتند از [۱۶ و ۱۷]:

- یادگیری با نظارت[۴]. این راهبرد یک ناظر[۵] یا راهبر را به خدمت می‌گیرد که از خود شبکه هوشمندتر است. برای مثال در تشخیص چهره، ناظر یک دسته تصویر را به شبکه نشان می‌دهد و نام هر چهره را می‌داند. شبکه، پیش‌بینی خود را انجام می‌دهد و حدس خود را با پاسخ صحیح مقایسه می‌کند، سپس بر اساس میزان خطا، پارامترهای خود را تنظیم می‌کند تا به پاسخ درست نزدیک شود. در یادگیری با نظارت هر نمونه در مجموعه داده‌ها علاوه بر بردار ویژگی‌ها[۶] با یک برچسب[۷] یا هدف[۸] همراه است که نقش ناظر را ایفاء می‌کند. دسته‌بندی[۹] نمونه‌ای از این نوع است.
- یادگیری بدون نظارت[۱۰]. این روش هنگامی که مجموعه داده نمونه با پاسخ معلوم در دسترس نباشد، استفاده می‌شود. به عبارت دیگر نمونه‌ها برچسب گذاری نشده باشند. جست‌وجو برای یافتن الگوی ناشناخته در بین یک مجموعه داده، کاربردی متداول از این روش است؛ برای مثال خوشه‌بندی[۱۱]، یعنی تقسیم یک مجموعه از عناصر به گروه‌هایی با توجه به الگوهای از قبل نامعلوم، به‌صورت بدون نظارت انجام می‌شود.
- یادگیری تقویتی[۱۲]. این راهبرد بر اساس مشاهده و نظام پاداش و جزا است. یادگیری تقویتی در رباتیک رایج است. در

[1] Fuzz Test
[2] Deep Learning
[3] Natural Language Processing

[4] Supervised Learning
[5] Supervisor (Teacher)
[6] Features
[7] Label
[8] Target
[9] Classification
[10] Unsupervised Learning
[11] Clustering
[12] Reinforcement Learning

زمان t ربات وظیفه T را انجام می‌دهد و نتیجه آن روی محیط را مشاهده می‌کند.

مهم‌ترین الگوریتم‌های یادگیری عمیق شامل موارد زیر است [۱۸]:

- شبکه‌های عصبی پیچیده[1] CNN: شامل چندین لایه بوده و بیشتر برای پردازش تصاویر کاربرد دارد.
- شبکه گیرنده‌های چند لایه[2] MLP: دارای تعداد ورودی و خروجی یکسان هستند ولی ممکن است چندین لایه پنهان داشته باشند و بیشتر برای کاربردهای تشخیص گفتار، تشخیص تصویر و ترجمه ماشینی کاربرد دارد.
- شبکه‌های عصبی بازرخداد[3] RNN: یکی از مهم‌ترین الگوریتم‌های یادگیری عمیق در حوزه پردازش و تولید متن است. شبکه‌های عصبی بازرخداد کلاسی از شبکه‌های عصبی هستند که به‌صورت یک گراف جهت‌دار دوری یا DCG[4] بیان می‌شوند. به‌عبارت دیگر ورودی هر یک از لایه(های) پنهان یا خروجی علاوه بر خروجی لایه قبل، شامل ورودی از مرحله قبل به‌صورت بازخورد نیز می‌شود. به دلیل اینکه RNN دارای حافظه داخلی است، قادر است داده‌های مرحله قبلی را به خاطر بسپارد و به همین خاطر پردازش داده‌های ترتیبی برای این مدل بسیار راحت می‌شود. این خاصیت باعث راحتی در وظایفی همانند ترجمه زبانی، تولید زبانی، تحلیل اهداف و غیره است.
- شبکه‌های حافظه کوتاه مدت بلند[5] LSTM: شبکه عصبی بازرخداد در حالت استاندارد، دارای نقطه ضعف اساسی است. در دنباله‌های طولانی که وابستگی بین ورودی‌هایی با فاصله زیاد وجود دارد، این شبکه قادر به حفظ این وابستگی‌ها نیست. به بیان دیگر گرادیان در دوره‌های زمانی بلند مدت تمایل به ناپدید شدن یا انفجار (زیاد شدن بی‌اندازه) دارد. در حالت ناپدید شدن گرادیان، یادگیری مدل متوقف می‌شود. شبکه‌های حافظه کوتاه مدت بلند برای مقابله با مسئله فوق و افزایش قدرت به خاطر سپاری شبکه عصبی بازرخداد ایجاد شده است. هسته اصلی این مدل سلول حافظه C است که در آن بر خلاف شبکه بازرخداد استاندارد که خروجی هر واحد تنها با اعمال یک تابع انگیزش ایجاد می‌شود، خروجی با محاسبات پیچیده‌تری تعیین می‌شود که به ویژه هدف آن‌ها منظم‌سازی و حفظ قدرت گرادیان است. ایده اصلی در LSTM تخصیص مسیری مجزا علاوه بر یال بازخوردی حالت پنهان برای جریان حافظه است.
- واحدهای بازرخداد دروازه‌ای[6] GRU: توسعه‌ای دیگر بر شبکه‌های عصبی بازرخداد استاندارد است [۱۹] که ساختار LSTM را با یک شبکه دروازه‌ای تغییر داده است. GRU سیگنال‌هایی تولید می‌کند که ورودی فعلی و حافظه قبلی را برای به‌روزرسانی فعالیت فعلی و وضعیت فعلی شبکه کنترل می‌کند. این شبکه ساده‌تر از LSTM بوده که در آن به‌روزرسانی پارامترها همچنین برای دروازه‌ها نیز مطابق الگوریتم استفاده می‌شود. بسیاری از معماری‌های مولد عمیق از GRU برای تولید متن استفاده کرده‌اند [۲۰ و ۲۱].
- شبکه‌های مولد متضاد[7] GAN: ایده اصلی شبکه‌های مولد متضاد، ایجاد یک بازی بین دو بازیکن یعنی تولید کننده و تشخیص دهنده است [۱۲]. وظیفه تولید کننده ایجاد نمونه-هایی است که به نظر برسد از منابع واقعی مثلاً از داده‌های آموزشی آمده است. تشخیص دهنده معمولاً یادگیری با نظارت را با استفاده از یک شبکه عصبی عمیق برای آموزش تفاوت قائل شدن میان داده‌های واقعی با داده‌های جعلی (داده‌هایی که توسط تولید کننده ایجاد شده است) انجام می‌دهد. هر دو بازیکن دارای توابع هزینه هستند که بر اساس پارامترهای هر کدام تعریف شده است. یعنی $\theta^{(D)}$ برای پارامترهای تشخیص دهنده و $\theta^{(G)}$ برای پارامترهای تولید کننده. هدف تشخیص دهنده کاهش میزان خطای خود یعنی $J^{(D)}(\theta^{(D)}.\theta^{(G)})$ است و باید این کار را با کنترل کردن $\theta^{(D)}$ انجام دهد. هدف تولید کننده نیز کاهش میزان خطای خود $J^{(G)}(\theta^{(D)}.\theta^{(G)})$ است و باید این کار را با کنترل کردن $\theta^{(G)}$ انجام دهد. در زمان آموزش شبکه GAN، شبکه‌های تولید و تشخیص پشت سر هم آموزش می‌بینند تا هر دو هم‌زمان بهبود یابند و به این صورت قادر به تولید نمونه‌های دارای کیفیت انسانی از تولید کننده هستند. کاربرد شبکه‌های مولد متضاد بیشتر در پردازش تصاویر است ولی به تازگی تلاش‌هایی برای تولید متن با استفاده از این الگوریتم شده است [۲۶-۲۲].
- شبکه‌های عملکرد شعاعی[8] RBFN: نوع خاصی از شبکه‌های عصبی رو به جلو هستند که از توابع پایه شعاعی به‌عنوان توابع فعال‌سازی استفاده می‌کنند. آن‌ها دارای یک لایه

[1] Convelutional Neural Networks
[2] Multi-Layer Perceptrons
[3] Recurrent Neural Networks
[4] Directed Cycle Graph
[5] Long Short Term Memory
[6] Gated Recurrent Units
[7] Generative Adversarial Networks
[8] Radial Basis Function Networks

ورودی، یک لایه مخفی و یک لایه خروجی هستند و بیشتر برای طبقه‌بندی[۱]، رگرسیون[۲] و پیش‌بینی سری زمانی[۳] مورد استفاده قرار می‌گیرند.

- نقشه‌های سازماندهی خودکار[۴] SOM: به‌منظور کاهش ابعاد داده‌ها از طریق شبکه‌های عصبی کاربرد دارند.
- ماشین‌های بالتزمن محدود شده[۵] RBM: شبکه‌های عصبی تصادفی هستند که می‌توانند از توزیع احتمال در مجموعه‌ای از ورودی‌ها یاد بگیرند. از این الگوریتم یادگیری عمیق برای کاهش ابعاد، طبقه‌بندی، رگرسیون، فیلتر مشترک، یادگیری ویژگی‌ها و مدل‌سازی موضوع استفاده می‌شود. همچنین به‌عنوان بلوک‌های سازنده شبکه‌های باور عمیق کاربرد دارند.
- شبکه‌های باور عمیق[۶] DBN: مدل‌های مولدی هستند که از چندین لایه متغیر تصادفی و پنهان تشکیل شده‌اند. متغیرهای نهفته مقادیر باینری دارند و اغلب واحدهای پنهان نامیده می‌شوند. DBN مجموعه‌ای از ماشین‌های بالتزمن است که بین لایه‌ها ارتباط دارد و هر لایه RBM با هر دو لایه قبلی و بعدی ارتباط برقرار می‌کند. این شبکه‌های عصبی برای شناسایی تصویر، شناسایی فیلم و داده‌های ضبط حرکت استفاده می‌شوند.
- کدگذارهای خودکار[۷] AE: کدگذارهای خودکار نوع خاصی از شبکه‌های عصبی رو به جلو هستند که در آن‌ها ورودی و خروجی یکسان است. آن‌ها شبکه‌های عصبی آموزش دیده‌ای هستند که داده‌ها را از لایه ورودی به لایه خروجی تکثیر می‌کنند. کدگذارهای خودکار برای اهدافی مانند کشف دارویی، پیش‌بینی محبوبیت و پردازش تصویر استفاده می‌شوند.

۳- آزمون فاز

امروزه یکی از رایج‌ترین روش‌های کشف آسیب‌پذیری، آزمون فاز یا فازینگ است. آزمون فاز رویکردی است که به‌منظور بهبود امنیت و قابلیت اطمینان در پیاده سازی سامانه‌ها پیشنهاد شده است. آزمون فاز شامل تحریک سامانه‌های تحت آزمون با استفاده از ورودی‌های تصادفی یا جهش یافته، به‌منظور شناسایی رفتارهای ناخواسته نظیر توقف‌های ناگهانی[۸] و یا نقض محرمانگی است [۲۷]. فازینگ یک روش آزمون نرم‌افزار است که اغلب به‌صورت خودکار یا نیمه خودکار انجام می‌شود و شامل ارائه داده‌های نامعتبر، غیر منتظره و تصادفی به‌عنوان ورودی برنامه‌ها و سپس نظارت بر برنامه به‌منظور تشخیص توقف‌های ناگهانی در برنامه، خطای موجود در قسمت اعلان کد برنامه و امکان نفوذ به حافظه است. برای فازینگ تعاریفی ارائه شده است [۲۸] که به آن‌ها اشاره می‌شود.

تعریف ۱ (فازینگ): به معنی اجرای برنامه تحت آزمون با استفاده از ورودی‌هایی که از فضای ورودی نمونه برداری شده‌اند و این فضای ورودی خاص به برنامه تحت آزمون یا PUT[۹] تحمیل می‌شود. اجرای برنامه تحت آزمون در دفعات متعدد و بسیار زیاد انجام می‌گیرد. آزمون فاز یکی از روش‌های آزمون نرم‌افزار است که از فازینگ استفاده می‌کند. تفاوت فازینگ با سایر روش‌های آزمون نرم‌افزار در این است که هدف فازینگ در آزمون نرم‌افزار، یافتن خطاهای مرتبط با امنیت در نرم‌افزار است، همانند اختلالات ناگهانی در نرم‌افزار در حین اجرا و پردازش داده آزمون.

تعریف ۲ (آزمون فاز): به معنی استفاده از فازینگ برای بررسی وجود شکست‌های امنیتی در برنامه تحت آزمون است.

تعریف ۳ (فازر): برنامه‌ای است که عملیات فازینگ را بر روی برنامه تحت آزمون انجام می‌دهد.

فازینگ را می‌توان بر اساس پارامترهای مختلف دسته‌بندی کرد. در زیر دسته‌بندی جامعی از انواع فازینگ ارائه شده است:

- بسته به نوع هدف: در این حالت فازینگ دارای دو دسته اساسی است. همان‌طور که گفته شد، هدف فازینگ کشف آسیب‌پذیری از نرم‌افزار است. معمولاً نرم‌افزار در دو حالت کد منبع[۱۰] و کد دودویی[۱۱] موجود است. بنابراین فازینگ نیز در دو حالت کد منبع و کد دودویی است.
- بسته به آگاهی از ساختار ورودی‌های برنامه هدف: این بخش تمرکز بر این دارد که آیا فازر از ساختار و گرامر ورودی‌هایی برنامه که قصد فاز کردن آن را دارد آگاه است یانه. از این جهت فازینگ دارای دو نوع تصادفی و هوشمند است.
- بسته به آگاهی از ساختار برنامه هدف: این تقسیم بندی فازینگ با توجه به آگاهی فازر از ساختار درونی برنامه تحت آزمون انجام شده است. این بخش به سه حالت تقسیم شده است: جعبه سفید[۱۲]، جعبه سیاه[۱۳] و جعبه خاکستری[۱۴].

[1] Classification
[2] Regression
[3] Time-Series Prediction
[4] Self-Organizing Maps
[5] Restricted Boltzmann Machines
[6] Deep Belief Networks
[7] Auto Encoders
[8] Crash

[9] Program Under Test
[10] Source Code
[11] Binary Code
[12] White Box
[13] Black Box
[14] Grey Box

- بسته به روش تولید داده‌های جدید: از این جهت، فازینگ دارای دو بخش است: فازینگ مبتنی بر تولید داده جدید[1] و فازینگ مبتنی بر جهش[2] (تغییر) داده قبلی.

روش‌های آزمون نرم‌افزار همانند آزمون فاز فقط توانایی نشان دادن وجود خرابی در برنامه تحت آزمون را دارند ولی نمی‌توانند نبود آن را تضمین کنند. داده‌های آزمون تولید شده در روش فازینگ به چند روش مختلف در اختیار برنامه هدف قرار داده می‌شود. البته نوع داده تولید شده وابسته به نوع ورودی‌هایی که هر برنامه هدف ممکن است قبول نماید با هم متفاوت خواهد بود. چند مورد از انواع ورودی‌هایی که برنامه‌های هدف می‌توانند آن‌ها را قبول کنند، عبارتند از: فایل‌ها، متغیرهای محلی، پارامترهای فراخوانی، داده‌های تبادل شبکه و وقایع و منابع سیستم عامل. در میان انواع ورودی‌های فوق، برنامه‌هایی که ورودی آن‌ها از نوع فایل‌های ساختارمند هستند، از اهمیت، تنوع و پیچیدگی بالایی برخوردار هستند، به دلیل اینکه نیاز به استخراج ساختار نهفته در فایل‌های ورودی برنامه هدف وجود دارد. راهکار استخراج ساختار نهفته فایل‌های ساختارمند متنی در این مقاله، استفاده از یادگیری عمیق برای یادگیری ساختار است که در بخش ۵-۱ به آن اشاره شده است.

۴- کارهای انجام شده

این بخش به بررسی کارهای انجام شده در خصوص ارزیابی مدل-های مولد یادگیری عمیق برای تولید داده متنی پرداخته است. هر کدام از مقالات بررسی شده تعدادی مدل مولد یادگیری عمیق را انتخاب کرده و بر اساس کاربرد مورد نظر خود و پارامترهای مختلف کیفیت و قدرت تولید داده متنی را مورد ارزیابی قرار داده‌اند.

مقاله ارائه شده توسط کاوتکار و همکاران [۱۲] به بررسی و ارزیابی مدل‌های مولد متن پرداخته است. مدل‌های مولد مورد بررسی این مقاله شامل شبکه‌های عصبی بازرخداد[3]، روش نمونه برداری برنامه‌ریزی شده[4] که توسط بنجیو و همکاران [۲۹] ارائه شده و همچنین راهکاری به نام SeqGAN است که مبتنی بر شبکه‌های مولد متضاد[5] بوده و توسط یو و همکاران [۲۲] ارائه شده است. پارامترهای ارزیابی در این مقاله شامل میزان خطا و میزان سرگشتگی است. طبق نتایج این مقاله آموزش شبکه‌های مولد متضاد برای ورودی‌های متنی کار سختی است و این امر به خاطر ماهیت گسسته در متون است. این مسئله در نتایج ارزیابی-های ارائه شده نیز مشخص است.

سیفکا و همکاران [۱۳] مدل‌های مولد مشتق شده از الگوریتم کدگذار خودکار به‌منظور تولید جملات را مورد ارزیابی قرار داده‌اند. مدل‌های مولد اشاره شده شامل کدگذار خودکار ساده AE[6]، متنوع VAE[7]، منظم شده متضاد ARAE[8] و متضاد AAE[9] است. بر طبق ارزیابی‌های کیفی ارائه شده در مقاله، هیچ کدام از مدل‌های مبتنی بر کدگذار خودکار تحت تمامی شرایط بهتر از مدل‌های دیگر رفتار نمی‌کنند و تمامی مدل‌های دارای ترازی[10] میان نمونه برداری خوب و کارایی بالای بازسازی جملات هستند.

در مقاله ارائه شده توسط ژو و همکاران [۱۴]، زیرساختی به نام Texygen به‌منظور ارزیابی مدل‌های مولد مشتق شده از شبکه‌های مولد متضاد (GAN) ارائه شده است. مدل‌های مولد مورد بررسی شامل SeqGAN [۲۲]، MaliGAN [۲۵]، RankGAN [۲۳]، TextGAN [۲۶] و LeakGAN [۲۴] است. بر اساس نتایج این مقاله با استفاده از زیرساخت Texygen، نمی‌توان تفاوت زیادی را بین مدل‌های مشتق شده از شبکه‌های مولد متضاد پیدا کرد زیرا هر کدام از این روش‌ها گرامرهای خاصی را یادگیری و بر اساس آن داده متنی را تولید می‌کنند.

منتهایی و همکاران [۱۵]، معیارهایی برای ارزیابی هم‌زمان کیفیت و تنوع مدل‌های عصبی مولد متن، با استفاده از تخمین فاصله مدل مولد آموزش دیده و توزیع داده واقعی ارائه کرده‌اند. به‌منظور ارزیابی معیارها، از زیرساخت ارائه شده مقاله قبلی یعنی Texygen استفاده شده است [۱۴]. تمامی مدل‌ها با استفاده از مجموعه داده یکسان آموزش دیده‌اند. بر اساس نتایج ارزیابی‌های انجام شده در این مقاله، روش محاسبه احتمال بیشینه MLE[11] بر پایه LSTM کارایی بهتری را از نظر معیارهای درنظر گیرنده هم‌زمان کیفیت و تنوع، نسبت به شبکه‌های مولد متضاد دارد. در حقیقت الگوریتم‌های مبتنی بر شبکه‌های مولد متضاد قادر به تولید متون با کارایی بالا از نظر معیارهای ترکیبی نیستند. این موضوع علاوه بر این مقاله، در نتایج گزارش مقاله کاسیا و

[1] Generation Base
[2] Mutation Base
[3] Recurrent Neural Network
[4] Scheduled Sampling
[5] Generative Adversarial Network

[6] Auto Encoder
[7] Variational Auto Encoder
[8] Adversarially Regularized Auto Encoder
[9] Adversarial Auto Encoder
[10] Trade-off
[11] Maximum Likelihood Estimation

همکاران [۳۰] که مدل‌های مشتق شده از GAN را با MLE برای تولید متن مقایسه کرده نیز ارائه شده است.

مقاله ارائه شده توسط اقبال و همکاران [۵]، به بررسی جامع تمامی مدل‌های مولد یادگیری عمیق متنی شامل RNN، CNN، VAE و GAN پرداخته و با استفاده از پارامترهایی همانند Rouge و BELU آن‌ها را با هم مقایسه کرده است. در نتایج این مقاله نیز اشاره شده است که در حوزه داده‌های گسسته همانند متن، کارایی روش‌های مبتنی بر VAE بیشتر از روش‌های مبتنی بر GAN است و کارایی روش‌های مبتنی بر GAN برای داده‌های پیوسته همانند تصاویر بسیار بیشتر است.

در ادامه برخی از کارها در خصوص یادگیری و تولید داده آزمون برای انجام عملیات فازینگ اشاره شده است.

باستانی و همکاران [۳۱] الگوریتمی برای تولید یک گرامر مستقل از متن روی یک مجموعه از ورودی‌های نمونه داده شده ارائه کرده‌اند، که درنهایت برای تولید داده‌های جدید مورد نیاز آزمون فازی استفاده می‌شود. این الگوریتم یک مجموعه از مراحل تعمیم‌پذیر[1] را با معرفی ساختارهای تکراری و متناوب برای عبارت‌های منظم به‌کار می‌بندد و غیر پایانه‌ها[2] را برای گرامر مستقل از متن[3] در هم ادغام می‌نماید که به نوبه خود یک گرامر یکنواخت از زبان ورودی به‌دست می‌دهد. الگوریتم مذکور در ابزاری به نام Glade پیاده‌سازی شده که از آن برای فازینگ برنامه‌ها استفاده شده است.

مقاله ارائه شده توسط گادفروید و همکاران [۳۲]، روشی برای تولید داده آزمون جهت استفاده در آزمون فاز بر مبنای شبکه‌های عصبی بازرخداد[4] و مدل کدگذار-کدگشا[5] ارائه کرده است. در این مقاله ساختار فایل پی‌دی‌اف[6] به‌عنوان ساختار منتخب برای یادگیری و بخش پی‌دی‌اف خوان مرورگر وب اج شرکت مایکروسافت به‌عنوان برنامه تحت آزمون انتخاب شده است. ایده اصلی مقاله، یادگیری یک مدل مولد زبان روی مجموعه‌ای از ویژگی‌های اشیای پی‌دی‌اف با داشتن مجموعه‌ای از نمونه‌های اولیه است. مدل کدگذار-کدگشا اجازه یادگیری متن با طول دلخواه را برای پیش‌بینی توالی بعدی کاراکترها، می‌دهد.

وانگ و همکاران [۳۳] یک روش تولید داده از روی داده‌های مشابه ارائه کرده‌اند که با یادگیری ساختار انبوه داده‌های مشابه فعلی، اقدام به تولید نمونه‌های بعدی مناسب به‌منظور فازینگ برنامه‌هایی می‌کند که چنین ورودی‌هایی را می‌پذیرند. در این مقاله، داده آزمون برای برنامه‌هایی تولید می‌شود که ورودی‌های به شدت ساختار یافته همانند XSL، XML، HTML و جاوا اسکریپت را پردازش می‌کنند.

پادورارو و همکاران [۳۴] یک ابزار متن باز ارائه کرده‌اند که قادر به تولید داده آزمون برای برنامه‌های تحت آزمون می‌باشد. این ابزار اقدام به دسته‌بندی فایل‌های موجود در پوشه داده نموده است و برای هر کدام از دسته‌ها مدل مولدی را تولید می‌کند. البته دسته‌بندی انواع فایل در پوشه داده با استفاده از دستور $file - l$ در سامانه عامل یونیکس انجام می‌گیرد.

فان و همکاران [۳۵] روشی برای تولید خودکار داده آزمون برای فازینگ جعبه سیاه مخصوص پروتکل‌های شبکه ارائه کرده‌اند. روش ارائه شده در مقاله از روش‌های یادگیری ماشین برای یادگیری یک مدل مولد داده آزمون ورودی مخصوص پروتکل‌های شبکه استفاده می‌کند. این کار با پردازش ترافیک مخصوص به آن پروتکل و تولید پیام‌های جدید با استفاده از مدل یادگیرنده انجام می‌شود. پیام‌های جدید می‌توانند به‌عنوان داده آزمون برای فازینگ برنامه‌های استفاده کننده از پروتکل شبکه مذکور مورد استفاده قرار بگیرد.

شی و همکاران [۳۶] اقدام به یادگیری خودکار ساختار پروتکل‌های صنعتی با استفاده از شبکه‌های عصبی مبتنی بر شبکه مولد متضاد یا GAN به‌منظور ارزیابی دقیق تابع توزیع مدل مولد پیام پروتکل‌های مربوط به شبکه‌های صنعتی نموده‌اند. بر اساس روش ارائه شده در این مقاله، یک چارچوب فازینگ هوشمند به نام GANFuzz برای آزمون پروتکل‌های شبکه‌های صنعتی ارائه شده است.

ذاکری و همکاران [۳۷] از مدل زبان عصبی[7] مبتنی بر شبکه‌های عصبی بازرخداد برای ایجاد مدل مولد به‌منظور یادگیری ساختار فایل‌های ساختارمند و سپس تولید داده آزمون استفاده کرده‌اند.

هر کدام از کارهای اشاره شده، از یکی از مدل‌های یادگیری عمیق برای یادگیری و تولید استفاده کرده‌اند که دارای مزایا و معایب مخصوص به خود هستند. در بخش ۵-۲ در مورد مدل مناسب کارکرد این مقاله بحث و نتیجه‌گیری شده است.

[1] Extendable
[2] Non Terminal
[3] Context Free
[4] Recurrent Neural Network
[5] Encoder-Decoder
[6] Portable Document Format
[7] Neural Language

۵- روش تحقیق

روش تحقیق پیشنهادی این مقاله، انتخاب مدل‌های مولد متناسب با کارکرد مورد نظر، شناسایی و انتخاب پارامترهای مؤثر در کارایی، انتخاب مجموعه داده مناسب برای آموزش مدل‌های یادگیری عمیق و درنهایت پیاده‌سازی، آموزش و ارزیابی مدل‌های مختلف به‌منظور شناسایی بهترین مدل و مناسب‌ترین پارامترها با کارایی بالا، با توجه به مسئله تحقیق مطرح شده در مقاله است که در ادامه به آن اشاره شده است.

۵-۱- مسئله تحقیق

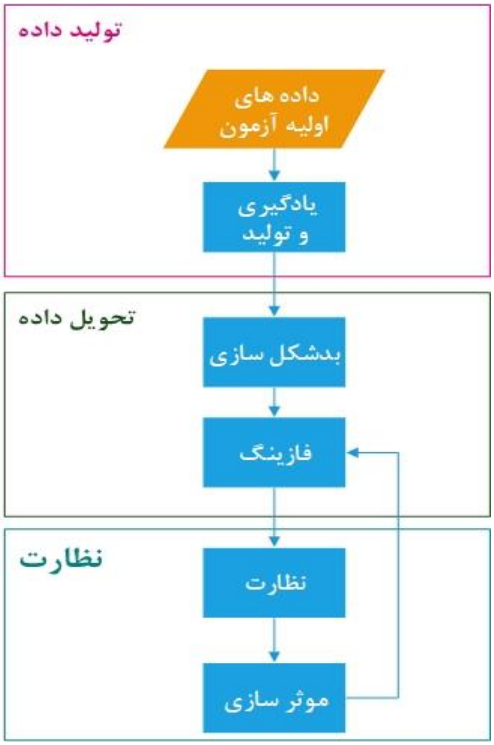
مسئله اصلی تحقیق در این مقاله، طراحی و پیاده‌سازی ابزار فازینگ به‌منظور انجام عملیات آزمون فاز برنامه‌های اجرایی پردازش کننده فایل‌های ساختارمند متنی است. شکل (۲) معماری کلان آزمون فاز فایل‌های ساختارمند متنی را نشان می‌دهد. اجزای اصلی این معماری کلان شامل موارد زیر است:

- تولید داده آزمون: این مرحله شامل سازوکارهای یادگیری و تولید داده آزمون است. در این مرحله، انباره داده‌های اولیه آزمون شامل انبوه فایل‌های جمع‌آوری شده از یک نوع فایل ساختارمند ناشناخته پردازش می‌شود (در این مقاله فرض این است که تعداد زیادی فایل با ساختار ناشناخته در دسترس است و هدف یادگیری ساختار ناشناخته و تولید خودکار داده‌های مشابه برای کشف آسیب‌پذیری از برنامه پردازش کننده آن فایل است). اولین اقدام، پردازش اولیه فایل‌های ساختارمند موجود در انباره به‌منظور تولید داده آزمون اولیه است. بر این اساس، داده‌های آزمون، از درون فایل‌های ساختارمند موجود در انباره استخراج و پردازش می‌شود. داده‌های آزمون استخراج شده در این مرحله به سه مجموعه آموزش[1]، ارزیابی[2] و تولید[3] تقسیم می‌شوند [۳۷]. از داده‌های مجموعه آموزش به‌منظور آموزش مدل مولد، از داده‌های مجموعه ارزیابی به‌منظور ارزیابی مدل مولد و از داده‌های مجموعه تولید به‌عنوان مقدمه و پیشوند برای تولید داده‌های آزمون توسط مدل مولد استفاده می‌شود. اقدام بعدی، یادگیری ساختار داده‌های آزمون و ایجاد یک مدل مولد است که بتواند پس از یادگیری، ساختار مشابه داده آزمون را به‌صورت خودکار تولید نماید. در ادامه، داده‌های آزمون جدید با استفاده از مدل مولد، تولید می‌گردند. برای تولید داده جدید، همان‌طور که اشاره شد، مقادیر پیشوندی به مدل مولد داده می‌شود که از میان مجموعه داده تولید انتخاب می‌شود تا مدل مولد داده آزمون جدید را بر اساس آن تولید نماید. صحت داده‌های آزمون تولید شده جدید نیز با مجموعه داده‌های ارزیابی مورد بررسی قرار می‌گیرد. انتخاب مدل مولد متناسب با داده‌های آزمون و با کارایی بالا در این مرحله، دارای اهمیت زیادی است. فرآیند انتخاب مدل مولد مناسب که در این مقاله به آن پرداخته شده است، برای این مرحله آزمون فاز است.
- تحویل داده آزمون: پس از پردازش، یادگیری و تولید داده‌های آزمون جدید در مرحله اول، در این مرحله فرآیند تحویل داده‌های آزمون جدید تولید شده به برنامه هدف در قالب آزمون فاز انجام می‌گیرد. قبل از انجام آزمون فاز، ابتدا فرآیند بدشکل‌سازی اولیه[4] بر روی داده‌های تولید شده جدید انجام می‌گیرد. هدف از انجام این کار این است که اگر داده‌های تولید شده جدید به‌صورت عادی در قالب آزمون فاز به برنامه هدف تزریق شود، به احتمال زیاد برنامه هدف قادر به پردازش داده‌های آزمون به‌صورت موفق خواهد بود. بنابراین نیاز است که بدشکل‌سازی اولیه به‌منظور ایجاد اختلال در برنامه هدف انجام گیرد. الگوریتم بدشکل‌سازی مورد استفاده در این مرحله شامل تغییر در مقادیر مرزی، انجام تغییرات در ساختار داده‌ها است. پس از انجام بدشکل‌سازی اولیه، داده‌های آزمون بدشکل شده در قالب آزمون فاز تحویل برنامه هدف می‌شود. نتایج آزمون فاز برای بررسی دقیق به مرحله سوم یعنی نظارت ارسال می‌گردد.
- نظارت: اقدام اصلی در این مرحله، بررسی دقیق بازخورد انجام آزمون فاز با داده‌های آزمون بر روی برنامه هدف است. همچنین داده‌های آزمون که منجر به ایجاد خطا بر روی برنامه هدف شوند برای استفاده‌های بعدی یعنی مرحله مؤثر سازی ذخیره می‌شوند. یکی دیگر از اقدامات اصلی در این مرحله، انجام عملیات مؤثر سازی داده‌های آزمون است که با استفاده از الگوریتم‌های مرتبط همانند الگوریتم ژنتیک و بازخوردهایی که از میزان تأثیر داده‌های اولیه آزمون تولیدی گرفته شده است، مورد بازنگری قرار گرفته و با تغییرات اندکی که در داده‌های آزمون داده می‌شود، سعی بر حداکثر کردن میزان تأثیر مخرب داده آزمون تولید شده بر روی برنامه تحت آزمون است [۳۸]. همان‌طور که از ساختار معماری نیز قابل مشاهده است، اطلاعاتی که از مرحله بعدی یعنی آزمون فاز به این مرحله

[1] Training
[2] Evaluation
[3] Generation

[4] Initial Malforming

می‌رسد، نقش زیادی در نوع تغییرات در بدشکل‌سازی داده آزمون دارد.



شکل (۲): معماری کلان آزمون فاز مبتنی بر فایل‌های ساختارمند متنی

۵-۲- بررسی و انتخاب مدل یادگیری عمیق

تولید داده آزمون با استفاده از روش یادگیری و تولید دارای پیچیدگی‌های زیادی است. پارامترهای زیادی برای ارزیابی مدل‌های یادگیری عمیق مبتنی بر یادگیری و تولید داده آزمون داده‌های متنی وجود دارد. هر کاربرد یادگیری و تولید داده آزمون از دیگری متفاوت بوده و ویژگی‌های مخصوص به خود را دارد. با توجه به مشخص شدن نوع استفاده از الگوریتم‌های یادگیری عمیق در کاربرد این مقاله در بخش ۳، و همچنین بررسی کارهای انجام شده در خصوص ارزیابی الگوریتم‌های یادگیری عمیق در حوزه تولید متن در بخش ۴، انتخاب مدل مولد متناسب با کاربرد مورد نظر این مقاله دارای ملاحظاتی است که در این بخش به آن اشاره شده است. معروف‌ترین مدل‌های مولد برای تولید داده متنی با توجه به بررسی‌های انجام شده شامل موارد زیر است:

- شبکه‌های عصبی بازرخداد یا RNN که مخصوص داده‌های رشته‌ای همانند متون هستند.
- شبکه‌های حافظه کوتاه مدت بلند یا LSTM به‌عنوان توسعه‌ای بر شبکه‌های عصبی بازرخداد بوده و به‌صورت گسترده در حوزه متن مورد استفاده قرار می‌گیرند.
- شبکه‌های مبتنی بر واحدهای بازرخداد دروازه‌ای یا GRU هم به‌عنوان توسعه‌ای بر شبکه‌های عصبی بازرخداد هستند.
- الگوریتم‌های مبتنی بر محاسبه احتمال بیشینه یا MLE.
- شبکه‌های مولد متضاد یا GAN که اساساً مخصوص داده‌های پیوسته همانند تصاویر هستند ولی به تازگی جزو جدیدترین روش‌های تولید داده متنی است.

هر کدام از الگوریتم‌های تولید داده متنی اشاره شده در بالا برای کاربرد مورد نظر مقاله یعنی تولید داده آزمون متنی از فایل‌های ساختارمند متنی دارای ملاحظاتی هستند که در اینجا به برخی از آن‌ها اشاره می‌شود.

- شبکه‌های مولد متضاد یا GAN با وجود تلاش‌هایی که برای تولید متن از آن‌ها شده است همچنان دارای ایراداتی در تولید متن هستند و بیشتر مناسب داده‌های پیوسته همانند تصاویر هستند.
- روش‌های مشتق شده از GAN در صورت موفق بودن، بر اساس رسالت ذاتی خود، داده‌های کاملاً واقعی تولید می‌کنند، در حالی که یکی از اهداف فازینگ، تولید داده‌های شبه واقعی بدشکل شده به‌منظور کشف آسیب‌پذیری‌های بالقوه است. بنابراین استفاده از شبکه‌های مولد متضاد با هدف اصلی فازینگ در تناقض است.
- الگوریتم‌های مبتنی بر محاسبه احتمال بیشینه یا MLE جزو روش‌های سنتی و قدیمی در حوزه پردازش متن است.

مناسب‌ترین الگوریتم‌های شبکه عصبی برای تولید متن با توجه به بررسی‌های انجام شده، شبکه‌های عصبی بازرخداد و مشتقات آن هستند که با توجه به بررسی‌های انجام شده شامل خود RNN و شبکه‌های GRU و LSTM است [۲۱]. بنابراین مدل‌های یادگیری عمیق شامل RNN، GRU و LSTM به‌عنوان مدل‌های مورد بررسی در این مقاله انتخاب شدند. علاوه بر انتخاب مدل یادگیری عمیق، پارامترهای دیگری نیز برای ارزیابی مدل‌های یادگیری عمیق تأثیر دارند. در ادامه به بررسی پارامترهای ارزیابی شبکه‌های عصبی عمیق برای تولید داده آزمون پرداخته خواهد شد.

۵-۳- شناسایی پارامترهای مهم

مبحث اصلی در طرح ارائه شده، ایجاد مدل‌های مختلف و مقایسه مدل‌های ایجاد شده از نظر کارایی است. عوامل زیادی بر یادگیری عمیق در یک وظیفه خاص تأثیر می‌گذارند که شناسایی و مدیریت مناسب آن عوامل، تأثیر زیادی بر بهبود کیفیت یادگیری عمیق در آن وظیفه خاص دارد. برای این منظور، تمامی پارامترهای مؤثر بر یادگیری عمیق در کارکرد مورد نظر مقاله به

همراه بازه مقادیر مورد نظر در جدول (۱) لیست شده است. برخی از عوامل تأثیرگذار، پارامترهایی هستند که مقدار آن‌ها به‌صورت تجربی به‌دست می‌آید و دلیل خاصی برای انتخاب آن مقدار وجود ندارد.

جدول (۱): پارامترهای مؤثر بر یادگیری عمیق داده‌های متنی

ردیف	عنوان پارامتر	نوع پارامتر	کاربرد پارامتر	بازه تغییرات
۱	نوع مدل یادگیری عمیق	مدل	یادگیری و تولید	LSTM, RNN, GRU
۲	تعداد لایه شبکه عصبی	عددی	یادگیری وتولید	۱، ۲، ۴
۳	ساختار لایه های شبکه عصبی	ساختار	یادگیری وتولید	هرمی، صاف
۴	تعداد گره لایه‌های ساختار هرمی	عددی	یادگیری وتولید	(۱۲۸) (۱۲۸،۶۴) (۱۲۸،۶۴،۳۲،۱۶)
۵	تعداد گره لایه‌های ساختار صاف	عددی	یادگیری وتولید	(۱۲۸) (۱۲۸،۱۲۸) (۱۲۸،۱۲۸،۱۲۸،۱۲۸)
۶	تعداد دوره یادگیری	عددی	یادگیری	۱۶، ۳۲
۷	اندازه دسته های پردازش	عددی	یادگیری	۶۴، ۱۲۸
۸	طول زیر رشته ها	عددی	یادگیری و تولید	۴۰، ۸۰
۹	میزان دارپات	عددی	یادگیری	۰/۳، ۰/۴
۱۰	میزان تفکیک داده‌ها	مشخصه	یادگیری و تولید	(۵۰،۲۵،۲۵) (۶۰،۲۰،۲۰) (۷۰،۱۵،۱۵)

پارامترهای مشخص شده در جدول (۱) شامل موارد زیر است:

- نوع مدل یادگیری عمیق: الگوریتم اصلی شبکه عصبی عمیق برای یادگیری و تولید است.
- تعداد لایه شبکه عصبی: تعداد لایه‌های مخفی[1] در شبکه عصبی عمیق است که به‌صورت ابرپارامتر[2] بوده و به‌صورت تجربی به‌دست می‌آید.
- ساختار لایه‌های مخفی شبکه عصبی: به معنی نوع چینش نورون‌ها[3] در لایه‌های مخفی شبکه عصبی است.
- تعداد دوره یادگیری[4]: تعداد دور آموزش مدل با داده‌های آموزش است که به‌صورت ابرپارامتر است.
- اندازه دسته‌های[5] پردازش: اندازه هر دسته داده‌ها که برای آموزش به شبکه عصبی خورانده می‌شود.
- طول زیررشته‌ها: طول بازه رشته‌های تفکیک شده برای یادگیری عمیق در مجموعه آموزش.
- میزان دارپات[6]: درصد نورون‌های هر لایه که به‌صورت تصادفی خاموش می‌شوند.
- میزان تفکیک داده‌ها: درصد تفکیک بین داده‌های آموزش، ارزیابی و تولید است.

۵-۴- پیاده‌سازی مدل‌های شبکه عصبی

در این بخش در مورد فرآیند طراحی و پیاده‌سازی مدل‌های مورد نیاز برای یادگیری توضیح داده شده است. همان‌طور که در بخش قبل ارائه شد، سه مورد از الگوریتم‌های شبکه عصبی مخصوص متن برای یادگیری انتخاب شده است. تمامی مراحل طراحی و پیاده‌سازی مدل‌های مختلف در یک سامانه رایانه‌ای با پردازنده Intel Core i7-4790K 4.0GHz و حافظه اصلی ۱۶ گیگابایت DDR3 و پردازنده گرافیکی NVIDIA GeForce GTX 1080 انجام شده است. پیاده‌سازی مراحل آموزش مدل و پیش‌بینی یا تولید داده در چارچوب Keras با استفاده از زبان پایتون انجام شده است.

داده مهم‌ترین عنصر در فرآیند یادگیری و ارزیابی شبکه‌های عصبی است. داده مورد استفاده در این مقاله، از ساختارهای مورد استفاده در انبوه فایل‌های ساختارمند متنی جمع‌آوری شده است. برای نمونه از ساختارهای اشیای درون فایل‌های ساختارمند pdf و fb2 استفاده شده است. هدف، یادگیری و تولید و بدشکل‌سازی نمونه‌های جدید این ساختارها به‌منظور استفاده در آزمون فاز است. بردار داده در این مقاله شامل رشته‌های حرفی است که تبدیل به رشته‌ای از اعداد شده است. یک جدول به نام char2int برای نگاشت حروف به اعداد استفاده شده است که مدل در زمان یادگیری از آن استفاده می‌کند. همچنین در زمان تولید داده، برای تبدیل اعداد به حروف متناظرشان، از یک جدول دیگر به نام int2char برای نگاشت تمامی اعداد واحد به حروف متناظر استفاده شده است. این جدول‌ها به‌منظور استفاده‌های بعدی به‌صورت pickle [۳۹] در فایل ذخیره می‌شوند. بردار داده در این حالت به‌عنوان ماتریس رشته استفاده شده است که باید به رشته اعداد تبدیل شود. قبل از استفاده از ماتریس رشته، پیش پردازش بر روی مجموعه داده انجام می‌شود. زمانی که رشته اعداد با کمک ماژول مجموعه داده تنسورفلو[7] تولید می‌شود، این آرایه

[1] Hidden Layer
[2] Hyper Parameter
[3] Neurons
[4] Learning Epoc

[5] Batch Size
[6] Dropout
[7] Tensorflow

تبدیل به ماتریس رشته می‌شود. رایج‌ترین روش برای انجام این کار، استفاده از بردارسازی وان-هات[1] به معنی بردار دارای یک عنصر معتبر است. در این روش یک بردار به طول اندازه تمامی حروف موجود در داده‌های آزمون برای هر حرف درنظر گرفته می‌شود که تنها یکی از عناصر آن بردار برابر یک است و بقیه عناصر بردار صفر هستند. در این صورت برای نمایش n حرف، تعداد n بردار متفاوت وجود دارد. خروجی شبکه عصبی عمیق نیز به شکل یک حرف تعریف شده که در واقع یک بردار وان-هات است. فرآیند آموزش شبکه عصبی عمیق طوری انجام می‌شود که در خروجی تنها یکی از عناصر بردار یک و بقیه صفر باشد. مجموعه داده به دو بخش مجموعه خصوصیات و مجموعه برچسب‌ها تقسیم می‌شود. سپس این مجموعه داده درهم‌ریزی[2] شده و دسته‌بندی[3] می‌شود تا مغایرت‌هایی به مجموعه داده اضافه اضافه شود تا کارایی مدل بالاتر برود.

مدل‌های شبکه عصبی طوری طراحی شده‌اند تا بتوانند با بررسی تعدادی حرف قبلی به تولید حرف جدید بپردازند. این مدل‌ها وابسته به ورودی‌هایی هستند که از میان داده‌های مجموعه آموزش انتخاب می‌شوند. رشته‌هایی برای آموزش مدل‌ها انتخاب و مورد استفاده قرار می‌گیرند که بتوانند کارایی مدل‌ها را بیشینه نمایند. ساختار مورد استفاده برای هر کدام از الگوریتم‌ها دارای یک، دو یا چهار لایه اصلی است. ساختارهای دو و چهار لایه به دو صورت هرمی و صاف طراحی شده است. در ساختار هرمی تعداد نورون‌ها در لایه‌های بعدی نصف می‌شوند ولی در ساختار صاف تعداد نورون‌ها در تمامی لایه‌ها یکسان است. هر لایه اصلی در این ساختارها شامل یک لایه از الگوریتم‌های GRU، LSTM یا RNN با تعداد مشخص شده در جدول (۱) و یک لایه دراپات [۴۰] برای جلوگیری از اثر پیش برازندگی[4] داده و پشتیبانی از عمومیت بیشتر در مدل‌ها با روشن و خاموش کردن واحدهای عصبی به‌صورت تصادفی در لایه‌های عصبی، مورد استفاده قرار گرفته است. پس از لایه‌های اصلی، داده‌های یادگیری شده به لایه Dense [۴۱] به همراه تابع انگیزش[5] پاس داده می‌شود. نورون‌های لایه Dense اطمینان می‌دهند که تمامی نورون‌ها به‌صورت کامل با هم ارتباط دارند. تعداد نورون‌های لایه Dense برابر تعداد حالت‌های خروجی شبکه عصبی عمیق است. هدف اصلی تابع انگیزش پیوسته‌سازی و خروج از حالت خطی (غیر خطی شدن) مدل است. در کاربرد این مقاله از یک تابع طبقه‌بندی کننده به نام تابع بیشینه هموار[6] استفاده شده است.

[1] One Hot
[2] Shuffle
[3] Batch
[4] Overfitting
[5] Activation Function
[6] Softmax Function

تابع بیشینه هموار یک بردار k تایی از اعداد حقیقی را به‌عنوان ورودی دریافت نموده و یک بردار k تایی از مقادیر حقیقی در بازه [۰,۱] را به‌عنوان خروجی می‌دهد به‌طوری که جمع مؤلفه‌های آن برابر یک خواهد بود:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \ (for\ i = 1\ to\ k)$$

در مرحله آموزش، شبکه عصبی به ازای هر ورودی از مجموعه آموزش یک خروجی تولید می‌کند. از آنجایی که خروجی درست برای ورودی مربوطه در زمان آموزش در دسترس است، می‌توان به‌صورت کمی اختلاف مقدار محاسبه شده $\hat{y}$ و مقدار واقعی y را توسط تابع خطای Loss محاسبه کرد. استفاده از تابع خطا برای محاسبه مقدار خطا به‌منظور بررسی رفتار مناسب یا غیر مناسب مدل‌ها پس از بهینه‌سازی، محاسبه و مورد استفاده قرار می‌گیرد. در این مقاله از تابع خطای آنتروپی متقاطع طبقه‌بندی شده[7] استفاده شده که تابع خطا در آن به‌صورت زیر تعریف شده است:

$$Loss = -\sum_{i=1}^{output\ size} y_i \log \hat{y}_i$$

به‌منظور آموزش عملی شبکه با کمینه کردن تابع خطا، الگوریتم بهینه‌سازی[8] بر شبکه عصبی اعمال می‌شود. این الگوریتم مبتنی بر گرادیان کاهشی[9] است که به‌صورت تکراری تلاش می‌کند تا کمینه محلی یک تابع را با برداشتن گام‌های کوچکی در جهت خلاف جهت گرادیان آن تابع پیدا کند. هدف از اعمال الگوریتم بهینه‌سازی، بهینه‌سازی مراحل یادگیری مدل است، به‌طوری که خطاها با نرخ بیشتری همگرا شوند. تعداد زیادی الگوریتم بهینه‌سازی مبتنی بر گرادیان کاهشی وجود دارد [۴۲]. در این مقاله از الگوریتم بهینه‌سازی Adam [۴۳] استفاده شده است.

پس از آموزش یک مدل، نتیجه در یک ساختار ترتیبی ذخیره می‌شود تا برای ارزیابی مدل با مجموعه داده ارزیابی و تولید داده جدید با پیشوندهای انتخاب شده از مجموعه تولید، مورد استفاده قرار بگیرد. وزن‌های این مدل ترتیبی در پیکربندی‌های کوچک مدل بارگذاری می‌شوند، یعنی تغییر ابعاد مدل برای پردازش فقط یک حرف به جای کل متن انجام می‌شود. برای ارزیابی مدل با استفاده از مجموعه داده ارزیابی، از تابع `model.evaluate()` استفاده می‌شود. این تابع میزان

[7] Categorical Cross Entropy
[8] Optimization Algorithm
[9] Gradient Decent

دقت و میزان خطای مدل یادگیری شده را بر روی مجموعه داده ارزیابی برمی‌گرداند. شرط انجام ارزیابی مدل این است که مجموعه داده ارزیابی قبلاً در زمان آموزش توسط مدل مشاهده نشده باشد تا ارزیابی دقیقی به‌دست آید. برای تولید داده با مدل آموزش دیده، زمانی که مدل آماده است، یک رشته ورودی به‌عنوان ورودی از مجموعه تولید انتخاب شده و به مدل داده می‌شود تا مدل بتواند کاراکتر بعدی را پیش‌بینی نماید. این کار با استفاده از تابع `model.predict()` انجام می‌شود. پس از مشخص کردن کاراکتر بعدی، ورودی قبلی بهینه‌سازی شده و به‌عنوان ورودی برای پیش‌بینی به مدل داده می‌شود. این فرآیند به تعداد مشخصی تکرار می‌شود.

۵-۵- مقایسه حالت‌های مختلف

به‌منظور مقایسه و انتخاب بهترین مدل شبکه عصبی عمیق متناسب با کارکرد مقاله، حالت‌های ممکن مدل شبکه عصبی مطرح شده در جدول (۱)، پیاده‌سازی و با هم مقایسه شده است. در جدول (۱) ویژگی‌های مشخص شده از دو نوع ساختاری و عددی هستند. ویژگی‌های ساختاری شامل ردیف‌های ۱ تا ۵ جدول و ویژگی‌های عددی شامل ردیف‌های ۶ تا ۹ جدول هستند. ردیف ۱۰ جدول به درصد تفکیک میان مجموعه‌های آموزش، ارزیابی و تولید مربوط می‌شود که برای تمامی حالات در این مقاله میزان متوسط یعنی ۶۰ درصد آموزش، ۲۰ درصد ارزیابی و ۲۰ درصد تولید انتخاب شده است.

بر اساس حالت‌های مطرح شده در ویژگی‌های ساختاری، سه حالت از الگوریتم‌های شبکه عصبی اشاره شده (GRU، LSTM، RNN) در پنج حالت (یک لایه، دولایه هرمی، دولایه صاف، چهارلایه هرمی و چهارلایه صاف) پیاده‌سازی شده است. بنابراین ۱۵ حالت ممکن برای انواع شبکه عصبی از نظر نوع و ساختار وجود دارد که همه پیاده‌سازی شده است. در مورد ویژگی‌های عددی همان‌طور که در جدول مشاهده می‌شود، برای هر ویژگی مقدار کم و زیاد پیش‌بینی شده است. در این مقاله ۱۵ حالت نوع و ساختاری اشاره شده، یک بار با مقادیر کم و یک بار با مقادیر زیاد یادگیری شده است. مقادیر کم شامل تعداد دوره ۱۶، اندازه دسته ۶۴، طول زیر رشته ۴۰ و میزان دراپات ۰/۳ و مقادیر زیاد شامل تعداد دوره ۳۲، اندازه دسته ۱۲۸، طول زیررشته ۸۰ و میزان دراپات ۰/۴ انتخاب شده است.

همان‌طور که گفته شد، میزان تفکیک داده‌ها در تمامی حالات ثابت است. با توجه به حالات اشاره شده در مجموع ۳۰ حالت از انواع شبکه عصبی پیاده‌سازی شده و بر روی یک مجموعه داده یکسان یادگیری شده‌اند تا کارایی هر کدام از آن‌ها با هم مقایسه شود. خروجی این مرحله ۳۰ مدل مختلف شبکه عصبی به همراه زمان و سابقه[1] آموزش آن‌ها است که به‌صورت فایل ذخیره می‌شود تا در مرحله بعد برای ارزیابی و تولید استفاده شود.

در مرحله ارزیابی، ۳۰ مدل مختلف شبکه عصبی بارگذاری شده و با تعداد دوره به اندازه ۱۰۲۴ و ۲۰۴۸ و با داده‌های مجموعه ارزیابی، عملیات ارزیابی انجام شده و نتایج ذخیره می‌شود. با توجه به موارد فوق انواع مقایسه‌های انجام شده شامل موارد زیر است:

- مقایسه زمان کل مرحله آموزش میان تمامی حالت‌ها؛
- مقایسه میزان خطا بین الگوریتم‌های مختلف (سه مدل انتخاب شده) با پارامترهای یکسان در مرحله آموزش؛
- مقایسه میزان خطا بین الگوریتم‌های یکسان با پارامترهای مختلف (دو حالت کمینه و بیشینه) در مرحله آموزش؛
- مقایسه میزان خطا در مرحله ارزیابی میان تمامی حالت‌ها در دو حالت (تعداد دوره ۱۰۲۴ و ۲۰۴۸).

نتایج مقایسه‌های انجام شده به‌صورت گراف و جدول در بخش بعدی ارائه شده و مورد بررسی و تحلیل قرار گرفته است.

۶- نتایج و بحث

جدول (۲) زمان مورد نیاز آموزش تمامی ۳۰ حالت مشخص شده در بخش قبل را برای مرحله آموزش نشان می‌دهد. واحد زمانی در این جدول ثانیه است. همان‌طور که از نتایج جدول مشخص است، زمان یادگیری مدل‌ها با مقادیر عددی زیاد، تقریباً به اندازه دو برابر زمان یادگیری مدل‌ها با مقادیر عددی کم است. این نتیجه تقریباً قابل پیش‌بینی بود، به دلیل اینکه پارامترهای مقادیر عددی زیاد تقریباً دو برابر پارامترهای مقادیر عددی کم هستند. نکته قابل توجه در این است که با مقادیر عددی یکسان، شبکه‌های عصبی مبتنی بر RNN بیشترین زمان را برای آموزش استفاده می‌کنند. در حالی که شبکه‌های عصبی مبتنی بر GRU و LSTM تقریباً زمان‌های مشابه و کمتری را برای آموزش استفاده می‌کنند. شبکه‌های عصبی مبتنی بر RNN چندین برابر سایر شبکه‌های عصبی برای آموزش زمان استفاده می‌کنند. بهترین زمان آموزش به ترتیب مربوط به ردیف‌های ۶، ۱، ۲ و ۳ بوده و بدترین زمان آموزش مربوط ردیف‌های ۲۹، ۳۰، ۲۷ و ۲۸ هستند.

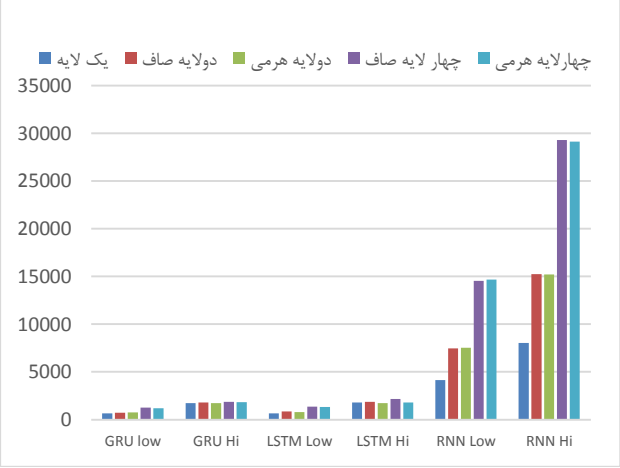
[1] Log

جدول (۲): زمان آموزش مدل‌های مشخص شده در مرحله آموزش

ردیف	مدل‌ها با مقادیر عددی کم	زمان	ردیف	مدل‌ها با مقادیر عددی زیاد	زمان
1	GRU-1Layer_Min	655	16	GRU-1Layer_Max	1722
2	GRU-2Layer-Flat_Min	739	17	GRU-2Layer-Flat_Max	1792
3	GRU-2Layer-InvPyr_Min	746	18	GRU-2Layer-InvPyr_Max	1753
4	GRU-4Layer-Flat_Min	1262	19	GRU-4Layer-Flat_Max	1858
5	GRU-4Layer-InvPyr_Min	1193	20	GRU-4Layer-InvPyr_Max	1872
6	LSTM-1Layer_Min	643	21	LSTM-1Layer_Max	1751
7	LSTM-2Layer-Flat_Min	848	22	LSTM-2Layer-Flat_Max	1771
8	LSTM-2Layer-InvPyr_Min	787	23	LSTM-2Layer-InvPyr_Max	1797
9	LSTM-4Layer-Flat_Min	1364	24	LSTM-4Layer-Flat_Max	2156
10	LSTM-4Layer-InvPyr_Min	1315	25	LSTM-4Layer-InvPyr_Max	1813
11	RNN-1Layer_Min	4146	26	RNN-1Layer_Max	8037
12	RNN-2Layer-Flat_Min	7449	27	RNN-2Layer-Flat_Max	15238
13	RNN-2Layer-InvPyr_Min	7531	28	RNN-2Layer-InvPyr_Max	15202
14	RNN-4Layer-Flat_Min	14544	29	RNN-4Layer-Flat_Max	29319
15	RNN-4Layer-InvPyr_Min	14663	30	RNN-4Layer-InvPyr_Max	29138

در شکل (۳) زمان مورد نیاز برای یادگیری حالت‌های مختلف شبکه‌های عصبی درنظر گرفته شده، بر اساس ساختار شبکه‌های عصبی با هم مقایسه شده است. مطابق شکل، زمان آموزش در شبکه‌های عصبی RNN با افزایش تعداد لایه مخفی دو برابر می‌شود. در حالی که در شبکه‌های عصبی LSTM و GRU، زمان آموزش در تمامی ساختارها تقریباً مساوی است و افزایش تعداد لایه مخفی تغییر زیادی در زمان آموزش ایجاد نکرده است.

در مرحله بعد، میزان خطا به‌عنوان یکی از پارامترهای ارزیابی کارایی بین مدل‌های مختلف شبکه عصبی با ساختارهای یکسان با هم مقایسه شده است. به این معنی که در هر نمودار، به ازای ساختار و پارامترهای یکسان، میزان تغییرات در خطای الگوریتم‌های مختلف شبکه عصبی با هم مقایسه شده است.



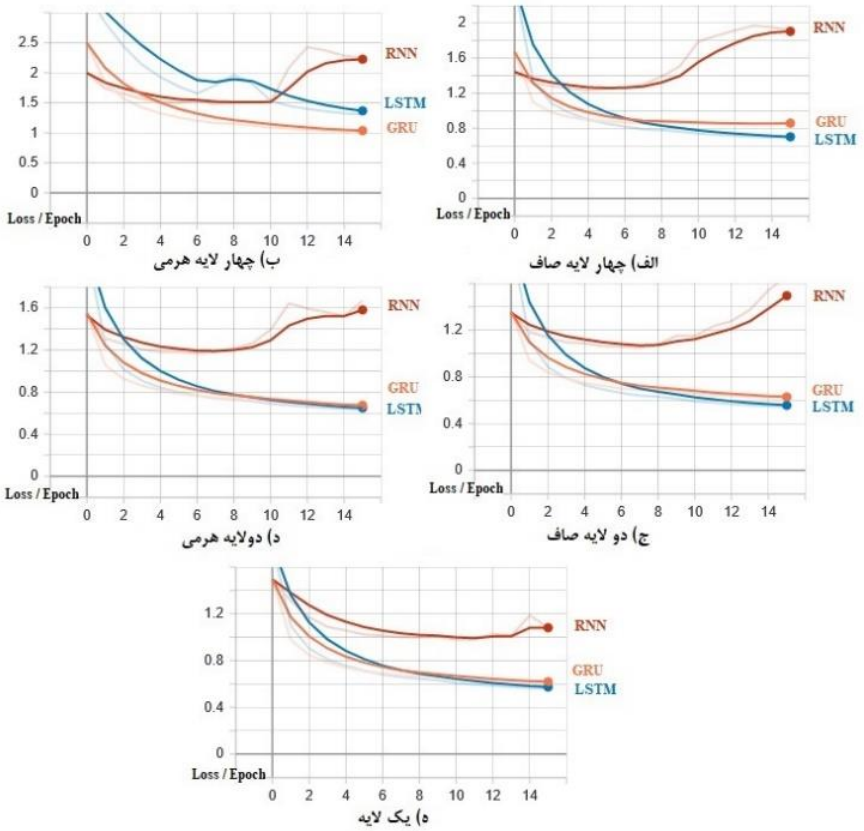
شکل (۳): دسته‌بندی زمان آموزش مدل‌های عصبی

برای این منظور پارامترهای یکسان (تعداد دوره، اندازه دسته‌ها، طول زیررشته‌ها و میزان دراپات) در ساختارهای پنج حالته (یک‌لایه، دولایه‌های هرمی و صاف، چهارلایه‌های هرمی و صاف) روی الگوریتم‌های مورد بررسی یعنی RNN، GRU و LSTM با هم مقایسه شده است.

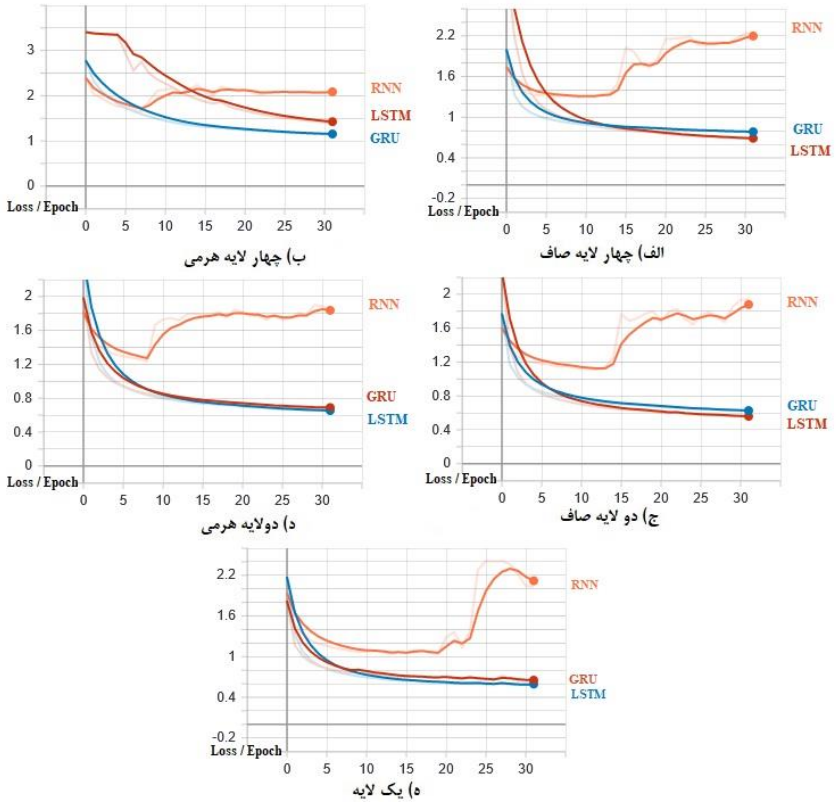
با توجه به اینکه پارامترهای یکسان در دو گروه پارامترهای کم و زیاد بررسی شده است، شکل (۴) نتایج مربوط به پارامترهای با مقدار کم و شکل (۵) نتایج مربوط به پارامترهای با مقدار زیاد را نشان می‌دهد. یکی از پارامترهای مهم تعداد دوره آموزش مدل شبکه عصبی است. در شکل (۴)، تعداد دوره آموزش مدل (خط افقی نمودارها) ۱۵ دوره و در شکل (۵) این مقدار برابر ۳۰ دوره است.

همان‌طور که از نتایج شکل (۴) و شکل (۵) قابل مشاهده است، میزان خطای شبکه عصبی RNN نسبت به LSTM و GRU در تمامی حالات با اختلاف زیادی بالاتر است. در چهار حالت از پنج حالت مربوط به ساختارهای مشخص شده، شبکه عصبی LSTM دارای میزان خطای کمتری نسبت به GRU و RNN است و فقط در بخش ب هر دو شکل یعنی ساختار چهارلایه هرمی میزان خطای GRU کمتر از بقیه است.

نکته جالب توجه دیگر این است که در تمامی حالت‌های هر دو شکل (۴) و شکل (۵) میزان خطا در شبکه عصبی RNN با افزایش تعداد دوره آموزش صید صعودی داشته و افزایش پیدا کرده است. با توجه به نمودارهای شکل (۴) و شکل (۵) شبکه عصبی LSTM کارایی بهتری را از نظر میزان خطای کمتر داشته است.



شکل (۴): نمودارهای مقایسه میزان خطای الگوریتم‌های مختلف در ساختارهای یکسان با پارامترهای عددی کم



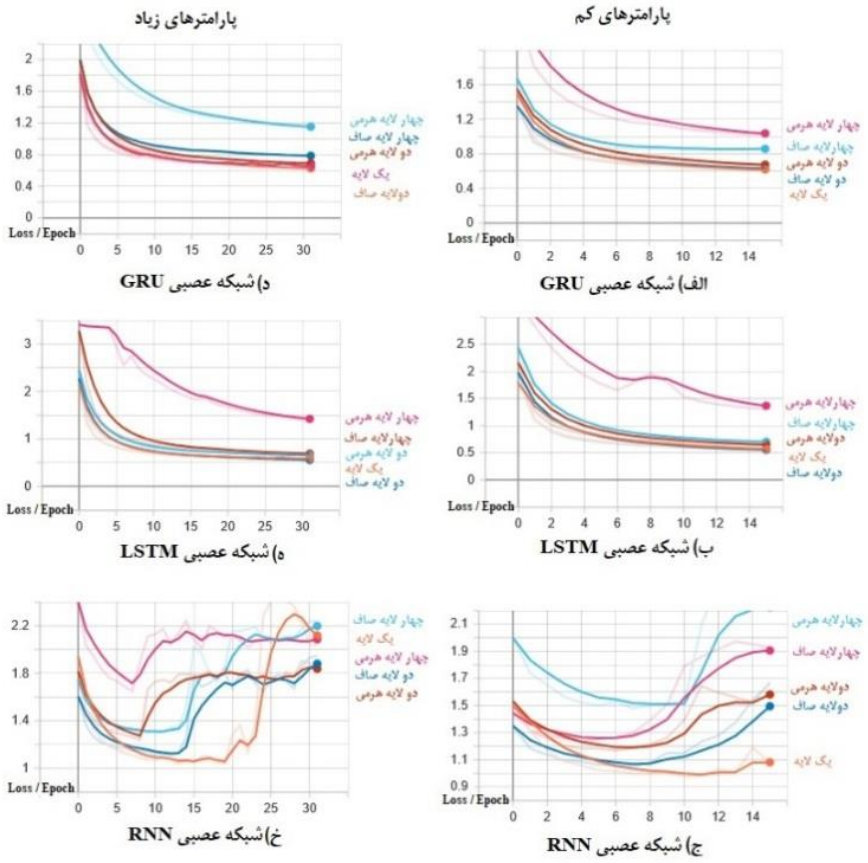
شکل (۵): نمودارهای مقایسه میزان خطای الگوریتم‌های مختلف در ساختارهای یکسان با پارامترهای عددی زیاد

یکی دیگر از ملاک‌های ارزیابی کارایی در مدل‌های مختلف شبکه‌های عصبی، مقایسه میزان خطا میان الگوریتم‌های شبکه

عصبی یکسان با پارامترهای مختلف در مرحله آموزش است. هدف از این کار، بررسی و انتخاب ساختار بهتر در بین الگوریتم-های یکسان است. همانند حالت قبلی این مقایسه در دو حالت پارامترهای کم و زیاد انجام می‌گیرد. نتایج این مقایسه در شکل (۶ ارائه شده است.

همان‌طور که از نتایج نمودارهای شکل (۶) می‌توان فهمید، ساختار یک‌لایه در بخش الف یعنی شبکه عصبی GRU با پارامترهای کم، ساختار دولایه صاف در بخش ب یعنی شبکه عصبی LSTM با پارامترهای کم، ساختار یک‌لایه در بخش ج یعنی شبکه عصبی RNN با پارامترهای کم، ساختار دولایه صاف در بخش د یعنی شبکه عصبی GRU با پارامترهای زیاد، ساختار دولایه صاف در بخش ه یعنی شبکه عصبی LSTM با پارامترهای زیاد و درنهایت ساختار دولایه هرمی در بخش خ یعنی شبکه عصبی RNN با پارامترهای زیاد میزان خطای کمتری را داشته است. با مقایسه شبکه‌های عصبی یکسان با پارامترهای کم و زیاد نتایج جالبی حاصل می‌شود.

با مقایسه نمودارهای بخش الف و بخش د بر روی شبکه عصبی GRU می‌توان نتیجه گرفت که با افزایش تعداد دوره آموزش از ۱۵ به ۳۰ دور میزان خطا تغییر چندانی ندارد. البته نتایج در شبکه عصبی GRU یکسان نبوده و در بخش الف ساختار یک‌لایه و در بخش د ساختار دولایه صاف خطای کمتری دارند. مقایسه بخش‌های ب و ه مربوط به شبکه عصبی LSTM هم نشان می‌دهد با افزایش تعداد دوره آموزش تغییر زیادی در کم شدن میزان خطا رخ نداده است ولی در هر دو نمودار ساختار دولایه صاف کمترین میزان خطا را دارد. مقایسه بخش‌های ج و خ از شبکه عصبی RNN نشان می‌دهد در هر دو بخش با افزایش تعداد دوره آموزش میزان خطا نه تنها کاهش نداشته است، بلکه افزایش هم داشته است.



شکل (۶): بخش‌های الف، ب و ج: نمودارهای مقایسه میزان خطای ساختارهای مختلف در الگوریتم‌های یکسان با پارامترهای عددی کم، بخش‌های د، ه و خ: نمودارهای مقایسه میزان خطای ساختارهای مختلف در الگوریتم‌های یکسان با پارامترهای عددی زیاد

آخرین مرحله ارزیابی بر خلاف مراحل قبلی که به بخش آموزش مدل‌های شبکه عصبی مربوط می‌شدند، در بخش ارزیابی شبکه‌های عصبی انجام شده است. همان‌طور که قبلاً هم گفته شد، مرحله ارزیابی با داده‌های مجموعه ارزیابی انجام می‌شود که قبلاً توسط مدل‌های شبکه عصبی مشاهده نشده است. هدف از انجام مرحله ارزیابی با داده‌های مجموعه ارزیابی این است که میزان دقت مدل‌های شبکه عصبی در برابر داده‌های جدید مورد سنجش و ارزیابی قرار بگیرد. به‌منظور درک بهتر میزان خطای

الگوریتم‌های مختلف در این مرحله، ارزیابی در دو حالت انجام شده است. در حالت اول داده‌های مجموعه ارزیابی به اندازه ۱۰۲۴ دور و در حالت دوم داده‌های مجموعه ارزیابی به اندازه ۲۰۴۸ دور به مدل خورانده می‌شود تا میزیان دقت و خطا مشخص شود. هدف از این کار این است که متوجه شویم با دو برابر شدن تعداد دور ارزیابی مدل، آیا میزان خطا و دقت تغییرات محسوسی دارد یا نه. بر این اساس ارزیابی انجام گرفته و نتایج در جدول (۳) ارائه شده است.

همان‌طور که نتایج در جدول (۳) قابل مشاهده است، با دو برابر شدن تعداد دوره ارزیابی مدل‌های شبکه عصبی، نه تنها تغییر زیادی در میزان خطای مدل‌ها با مجموعه داده ارزیابی مشاهده نمی‌شود، بلکه در اکثر موارد میزان خطا حتی بیشتر هم شده است. با توجه به نتایج جدول، کمترین میزان خطا مربوط به ردیف‌های ۲۲، ۵۲، ۲۱ و ۱۹ است که در میان آن‌ها شبکه عصبی LSTM دولایه صاف با پارامترهای کم و زیاد در جایگاه اول قرار دارد. بیشترین میزان خطا هم مربوط به ردیف‌های ۵۹، ۲۹، ۴۴ و ۱۴ است که شبکه عصبی RNN چهارلایه صاف با پارامترهای کم و زیاد در جایگاه اول قرار دارد.

جدول (۳): میزان خطای مدل‌های شبکه عصبی در مرحله ارزیابی

ردیف	حالت ۱۰۲۴ دور	خطا	ردیف	حالت ۲۰۴۸ دور	خطا
1	GRU-1Layer_Min	1.382	31	GRU-1Layer_Min	1.384
2	GRU-2Layer-Flat_Min	1.288	32	GRU-2Layer-Flat_Min	1.289
3	GRU-2Layer-InvPyr_Min	1.354	33	GRU-2Layer-InvPyr_Min	1.355
4	GRU-4Layer-Flat_Min	1.403	34	GRU-4Layer-Flat_Min	1.404
5	GRU-4Layer-InvPyr_Min	1.597	35	GRU-4Layer-InvPyr_Min	1.594
6	LSTM-1Layer_Min	1.386	36	LSTM-1Layer_Min	1.384
7	LSTM-2Layer-Flat_Min	1.281	37	LSTM-2Layer-Flat_Min	1.281
8	LSTM-2Layer-InvPyr_Min	1.429	38	LSTM-2Layer-InvPyr_Min	1.434
9	LSTM-4Layer-Flat_Min	1.444	39	LSTM-4Layer-Flat_Min	1.443
10	LSTM-4Layer-InvPyr_Min	1.799	40	LSTM-4Layer-InvPyr_Min	1.801
11	RNN-1Layer_Min	2.681	41	RNN-1Layer_Min	2.687
12	RNN-2Layer-Flat_Min	2.825	42	RNN-2Layer-Flat_Min	2.827
13	RNN-2Layer-InvPyr_Min	2.468	43	RNN-2Layer-InvPyr_Min	2.468
14	RNN-4Layer-Flat_Min	2.869	44	RNN-4Layer-Flat_Min	2.870
15	RNN-4Layer-InvPyr_Min	2.711	45	RNN-4Layer-InvPyr_Min	2.711
16	GRU-1Layer_Max	1.322	46	GRU-1Layer_Max	1.331
17	GRU-2Layer-Flat_Max	1.277	47	GRU-2Layer-Flat_Max	1.287
18	GRU-2Layer-InvPyr_Max	1.346	48	GRU-2Layer-InvPyr_Max	1.355
19	GRU-4Layer-Flat_Max	1.265	49	GRU-4Layer-Flat_Max	1.274
20	GRU-4Layer-InvPyr_Max	1.675	50	GRU-4Layer-InvPyr_Max	1.682
21	LSTM-1Layer_Max	1.261	51	LSTM-1Layer_Max	1.270
22	LSTM-2Layer-Flat_Max	1.219	52	LSTM-2Layer-Flat_Max	1.231
23	LSTM-2Layer-InvPyr_Max	1.281	53	LSTM-2Layer-InvPyr_Max	1.291
24	LSTM-4Layer-Flat_Max	1.393	54	LSTM-4Layer-Flat_Max	1.399
25	LSTM-4Layer-InvPyr_Max	1.889	55	LSTM-4Layer-InvPyr_Max	1.899
26	RNN-1Layer_Max	2.568	56	RNN-1Layer_Max	2.574
27	RNN-2Layer-Flat_Max	2.589	57	RNN-2Layer-Flat_Max	2.599
28	RNN-2Layer-InvPyr_Max	2.262	58	RNN-2Layer-InvPyr_Max	2.271
29	RNN-4Layer-Flat_Max	2.931	59	RNN-4Layer-Flat_Max	2.935
30	RNN-4Layer-InvPyr_Max	2.646	60	RNN-4Layer-InvPyr_Max	2.652

۷- نتیجه‌گیری

هدف این مقاله، انتخاب، بررسی و ارزیابی انواع مدل‌های شبکه-های عصبی به‌منظور استخراج بهترین شبکه عصبی از نظر نوع و ساختار برای استفاده در تولید داده آزمون متنی برای آزمون فاز فایل‌های ساختارمند متنی است. داده‌های آزمون متنی از محتوای پرکاربردترین ساختارهای موجود در فایل‌های ساختارمند متنی استخراج و به سه قسمت آموزش، ارزیابی و تولید تقسیم شده است. به‌منظور انتخاب شبکه عصبی مناسب، در بخش کارهای انجام شده، جدیدترین تحقیقات انجام شده در خصوص مقایسه و ارزیابی شبکه‌های عصبی تولید متن مورد بررسی قرار گرفت و بر اساس نتایج بررسی‌ها، مناسب‌ترین الگوریتم‌های شبکه عصبی برای تولید متن انتخاب شدند. در ادامه سایر پارامترهای تأثیرگذار بر شبکه عصبی، مورد بررسی قرار گرفت و چندین ساختار با مشخصات عددی مد نظر قرار گرفت. سپس تمامی مدل‌های مورد نظر طراحی و پیاده‌سازی شدند و بر اساس ملاک‌هایی نظیر زمان آموزش و میزان خطا در مراحل یادگیری و ارزیابی مورد بررسی قرار گرفتند. درنهایت با توجه به نتایج بررسی شده شبکه‌های عصبی مبتنی بر LSTM با ساختار دولایه صاف و پارامترهای عددی کم به‌عنوان شبکه عصبی مناسب برای کاربرد مطرح شده در این مقاله انتخاب شده است. عوامل زیادی بر روی کارایی، میزان خطای کمتر، میزان دقت بالاتر و کیفیت تولید داده متنی در مدل‌های شبکه عصبی عمیق مولد متن تأثیر دارند. به‌عنوان کارهای آتی، تحقیقات بیشتر در خصوص سایر عوامل تأثیرگذار بر کارایی مدل‌های شبکه عصبی پیشنهاد می‌شود.

۸- مراجع

[1] B. Miller and L. Fredriksen, "An Empirical Study of the Reliability of Unix Utilities," *Communication of ACM*, vol. 33, no. 12, pp. 32-44, 1990.

[2] B. Miller, D. Koski, C. Pheow, L. Maganty, R. Murthy, A. Natarjan, and J. Steidl, "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services," University of Wisconsin, Madison, 1995.

[3] P. Gogefroid, "From Blackbox Fuzzing to Whitebox Fuzzing Towards Verification," 2010. [Online]. Available: http://selab.fbk.eu/issta2010/download/slides/Godefroid-Keynote-ISSTA2010.pdf.

[4] P. Godefroid, M. Levin, and D. Molnar, "Automated Whitebox Fuzz Testing," *In Proceedings of the Network and Distributed System Security Symposium*, 2008.

[5] T. Iqbal and S. Qureshi, "The Survey: Text Generation Models in Deep Learning," *Journal of King Saud University-Computer and Information Sciences (Production and hosting by Elsevier), vol. 34, no. 6, pp. 2515-2528*, 2020.

[6] O. Bastani, A. Aiken, P. Liang, and R. Sharma, "Synthesizing Program Input Grammars," *ACM SIGPLAN Notices - PLDI '17* , vol. 52, no. 6, pp. 95-110, 2017.

vs. Bidirectional RNN for Script Generation," *arXiv:1908.04332*, 2019.

[22] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, pp. 2852-2858, 2017.

[23] K. Lin, D. Li, X. He, M.-T. Sun, and Z. Zhang, "Adversarial Ranking for Language Generation," In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[24] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, " Long Text Generation via Adversarial Training with Leaked Information," In *Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, 2018.

[25] T. Che, Y. Li, R. Zhang, R. Devon Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-Likelihood Augmented Discrete Generative Adversarial Networks," *arXiv:1702.07983*, 2017.

[26] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial Feature Matching for Text Generation," In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, NSW, Australia, 2017.

[27] D. Ceara, M. Potet, G. Ensimag, and L. Mounier, "Detecting Software Vulnerabilities Static Taint Analysis Potet," *University Politehnica Bucuresti and University Joseph Fourie*, 2009.

[28] V. Manes, H. Han, C. Han, S. Cha, and M. Egele, "The Art, Science, and Engineering of Fuzzing: A Survey," *ACM Computing Surveys*, 2019.

[29] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks," *arXiv:1506.03099*, vol. 3, 2015.

[30] M. Caccia, L. Caccia, W. Fedus, H. Larochelle, J. Pineau, and L. Charlin, "Language GANs Falling Short," *arXiv:1811.02549*, 2020.

[31] O. Bastani, A. Aiken, P. Liang, and R. Sharma, "Synthesizing Program Input Grammars," *ACM SIGPLAN Notices - PLDI '17* , vol. 52, no. 6, pp. 95-110, 2017.

[32] P. Godefroid, H. Peleg, and R. Singh, "Learn&Fuzz: Machine Learning for Input Fuzzing," *ASE (Automated Software Engineering)*, 2017.

[33] J. Wang, B. Chent, L. Wei, and Y. Liu, "Skyfire: Data-Driven Seed Generation for Fuzzing," *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[34] C. Paduraru and M. C. Melemciuc, "An Automatic Test Data Generation Tool using Machine Learning," *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pp. 472-481, 2018.

[35] R. Fan and Y. Chang, "Machine Learning for Black-Box Fuzzing of Network Protocols," *Information and Communications Security (ICICS 2017)*, pp. 621-632, 2018.

[7] P. Godefroid, H. Peleg, and R. Singh, "Learn&Fuzz: Machine Learning for Input Fuzzing," *ASE (Automated Software Engineering)*, 2017.

[8] J. Wang, B. Chent, L. Wei, and Y. Liu, "Skyfire: Data-Driven Seed Generation for Fuzzing," *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[9] C. Paduraru and M. C. Melemciuc, "An Automatic Test Data Generation Tool Using Machine Learning," *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pp. 472-481, 2018.

[10] D. She, K. Pei, D. Epstein, J. Yang, and B. Ray, "NEUZZ: Efficient Fuzzing with Neural Program Smoothing," *IEEE Symposium on Security and Privacy*, vol. 89, no. 49, pp. 38-53, 2019.

[11] Elvis, "Deep Learning for NLP: An Overview of Recent Trends," dair.ir, 24 08 2018. [Online]. Available: https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d. [Accessed 22 02 2021].

[12] P. Kawthekar, R. Rewari, and S. Bhooshan, "Evaluating Generative Models for Text Generation," *arXiv*, 2017.

[13] O. Cífka, A. Severyn, E. Alfonseca, and K. Filippova, "Eval All, Trust a Few, do Wrong to None: Comparing Sentence Generation Models," *arXiv:1804.07972*, 2017.

[14] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A Benchmarking Platform for Text Generation Models," *arXiv:1802.01886v1*, 2018.

[15] E. Montahaei, D. Alihosseini, and M. Soleymani Baghshah, "Jointly Measuring Diversity and Quality in Text Generation Models," In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation (NeuralGen)*, Minneapolis, Minnesota, USA, 2019.

[16] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Massachusett: MIT Press, 2016.

[17] D. Shiffman, S. Fry, and Z. Marsh, The Nature of Code, Daniel Shiffman, 2012.

[18] A. Biswal, "Top 10 Deep Learning Algorithms You Should Know in 2021," Simplilearn, 16 2 2021. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm. [Accessed 3 3 2021].

[19] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv:1406.1078*, 2014.

[20] M. Hong, M. Wang, L. Luo, X. Tan, D. Zhang, and Y. Lao, "Combining Gated Recurrent Unit and Attention Pooling for Sentimental Classification," In *Proceedings of the 2018 2Nd International Conference on Computer Science and Artificial Intelligence, ser. CSAI '18*, New York, NY, USA, 2018.

[21] S. Mangal, P. Joshi, and R. Modak, "LSTM vs. GRU

[36] Z. Hu, J. Shi, Y. Huang, J. Xiong and X. Bu, "GANFuzz: A GAN-based Industrial Network Protocol Fuzzing Framework," In *Proceedings of the 15th ACM International Conference on Computing Frontiers (CF18) pp. 138-145*, Ischia, Italy, 2018.

[37] M. Zakeri, S. Parsa, and A. Kalaee, "Format-aware Learn&Fuzz: Deep Test Data Generation for Efficient Fuzzing," *arXiv Prepr arXiv181209961*, 2019.

[38] T. Taghavi and M. Bagheri, "Presenting an Intelligence Test Data Generation Method to Discover Software Vulnerabilities," *Advanced Defence Science& Technology*, vol. 4, no. 37, pp. 307-322, 2019 (In Persian).

[39] "Pickle — Python Object Serialization," [Online]. Available: https://docs.python.org/3/library/pickle.html. [Accessed 18 04 2021].

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

[41] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700-4708, 2017.

[42] A. Karpathy, Connecting Images and Natural Language, Stanford University, 2016.

[43] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2017.