A mesterséges intelligencia néhány biztonsági vetülete

Vidács László[1*], Jelasity Márk[2], Tóth László[3], Hegedűs Péter[1], Ferenc Rudolf[3]

[1]MTA-SZTE Mesterséges Intelligencia Kutatócsoport, Szeged
[2]Szegedi Tudományegyetem Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszék, Szeged
[3]Szegedi Tudományegyetem Szoftverfejlesztési Tanszék, Szeged

Beérkezett: 2020. szeptember 22.; Elfogadva: 2020. november 2.

Összefoglalás

A mély mesterséges neuronhálók elterjedése az ipari alkalmazásokban évekkel azok megbízhatóságával, értelmezhetőségével, és biztonságával kapcsolatos szakterületek fejlődését megelőzően történt. Az egyik, gyakorlatban is jelentős területen, a képfelismerésben például a megvalósult megoldások szinte már emberi teljesítményre képesek, de ezzel együtt célzott zajjal ezek a rendszerek félrevezethetők, megzavarhatók. Jelen kéziratban ismertetünk néhány tipikus biztonsági problémát, valamint rámutatunk arra, hogy a hagyományos szoftverfejlesztés területén alkalmazott minőségbiztosítási módszerekkel rokon megoldásokra szükség van az MI-re épülő rendszerek fejlesztésében, akár a mesterséges neuronhálók biztonságát, akár az MI rendszerek hagyományos komponenseinek fejlesztését tartjuk szem előtt.

Kulcsszavak: mesterséges neuronhálók, mélytanulás, biztonság, szoftverfejlesztés

On some security aspects of AI systems

László Vidács[1*], Márk Jelasity[2], László Tóth[3], Péter Hegedűs[1], Rudolf Ferenc[3]

[1]University of Szeged and Hungarian Academy of Sciences,
MTA-SZTE Research Group on Artificial Intelligence, Szeged
[2]University of Szeged, Department of Algorithms and AI, Szeged
[3]University of Szeged, Department of Software Engineering, Szeged

Summary

Research on the trustworthiness, interpretability and security of deep neural networks lags behind the widespread application of the technology in industrial applications. For example, in image recognition, modern solutions are capable of nearly human performance. However, with targeted adversarial noise, these systems can be arbitrarily manipulated. Here, we discuss some of the security problems and point out that quality assurance methods used in traditional software development should also be adapted when developing AI-based systems, whether in the security of artificial neural networks or traditional components of AI systems. One of the main concerns about neural networks today that – to the best of our knowledge – affects all deep neural networks is the existence of adversarial examples. These examples are relatively easy to find and according to a recent experiment, a well-chosen input can attack more networks at the same time. In this paper we also present a wider perspective of security of neural architectures borrowed from the traditional software engineering discipline. While in traditional development several methods are widely applied for software testing and fault localization, there is a lack of similar well-established methods in the neural network context. In case of deep neural networks, systematic testing tools and methods are in the early stage, and a methodology to test and verify the proper behavior of the neural networks is highly desirable. Robustness testing of machine learning algorithms is a further issue. This requires the generation of large random input data using fuzz testing methods. The adaptation of automatic fault localization techniques has already started by defining notions like code coverage to neural networks. Lastly, we argue that the effective development of high quality AI-based systems need well suited frameworks that can facilitate the daily work of scientists and software developers – like the Deep-Water framework, presented in the closing part of the paper.

Keywords: artificial neural networks, deep learning, security, software engineering

Bevezetés

Az elmúlt évtized során a mesterséges intelligencia kutatásokban jelentős áttöréseket értek el, s ezek az eredmények számos ipari alkalmazásban is megjelentek. Ilyen széles körben felhasználható eredmények születtek a jelfeldolgozás területén, ezen belül a hangfelismerés, a képfelismerés, valamint a természetes nyelvek feldolgozása kiemelkedő kutatási és alkalmazási terület. Ma már természetes nyelven lehet társalogni mesterséges személyi asszisztensekkel, a nyelvek közötti fordítás rutinfeladatnak számít, az arcfelismerés szinte tökéletes pontossággal működik, és még sorolhatnánk a példákat tovább.

A legtöbb alkalmazás hátterében mély neuronhálókból álló architektúrák működnek, amelyek rendkívül nagyméretű, adott feladattípus ellátására tervezett, mintapéldák útján a működésüket a feladat által meghatározott elvárásokhoz adaptívan módosító (tanuló) számítási egységek. Ahhoz, hogy a neuronhálók működésüket adekvát módon, kielégítő pontossággal tudják a feladat követelményeihez igazítani, megfelelő számosságú mintapéldát kell számukra biztosítani. Habár a neuronhálók teljesítménye figyelemre méltó, azonban hátrányaikról sem szabad megfeledkeznünk. Ezek közül az egyik legfontosabb, hogy az automatikus tanításnak köszönhetően az adaptálódott számítási egység működési részletei nem értelmezhetők, ezért rendkívül nehéz garantálni, hogy az minden körülmények között (minden elképzelhető inputra) megfelelően és megbízhatóan működik. A megbízhatóság, illetőleg a biztonsági szempontok figyelembe vétele pedig rendkívül fontos, hiszen MI rendszereket kritikus rendszerekben is találunk, akár az egészségügyről, közlekedésről, vagy katonai alkalmazásokról van szó.

Az MI rendszerek felderítetlen biztonsági réseit rosszindulatú támadók ki is használhatják azáltal, hogy a rendszert véletlenszerű, vagy akár előre eltervezett helytelen viselkedésre bírják. Az Európai Bizottság a napokban tette közzé az MI stratégiájával foglalkozó dokumentumot *(European Commission 2020)*, amelyben szintén kiemelt szerepet kap a technológia megbízhatóságának a követelménye.

Jelen kéziratban vázoljuk az MI rendszerek néhány ismert sérülékenységét, illetve kitérünk olyan technológiai megoldásokra, amelyek segíthetnek ezeket kezelni.

Ellenséges inputok az MI rendszerekben

A matematikai formalizmust mellőzve, ebben a fejezetben egy rendkívül fontos és érdekes problémát vázolunk, amely az eddigi ismereteink szerint minden mély neuronhálóra épülő MI rendszert érint.

A mély neuronhálók számítási egységek, vagy függvények, amelyek adott inputra (ami lehet például egy kép) kiszámítanak egy adott outputot (jellemzően az értékkészlet elemeinek valószínűségi eloszlását), amely feladattól függően például egy igen/nem döntésben („a képen látható-e macska?") vagy akár egy bonyolultabb szöveges képaláírás generálásában realizálódik. A neuronhálók, megfelelő mennyiségű és minőségű tanító adatot feltételezve, automatikusan betaníthatók ilyen feladatok a gyakorlatban elvárt pontosságon történő végrehajtására. Fontos kiemelni, hogy a neuronhálók végső architektúrája és így az általuk reprezentált szimuláció is emberi beavatkozás nélkül, a mintapéldák felhasználásával történő tanítás során jön létre automatikusan, működésük jellemzően „fekete doboz", azaz részleteikben nem, vagy nehezen interpretálhatók.

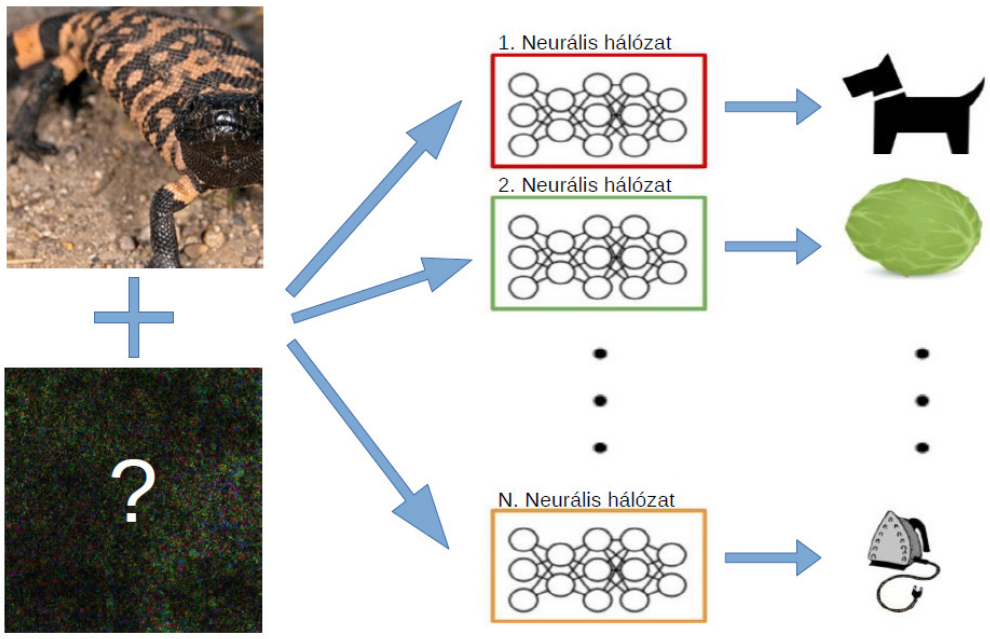
Ellenséges inputon azt értjük, hogy a támadó választ egy olyan inputot, amelyre a neuronháló helyes döntést hoz, majd megfelelő matematikai módszerekkel olyan láthatatlan zajt ad az inputhoz, amelynek hatására a neuronháló már teljesen más döntést fog hozni, miközben az emberi érzékelés számára a zajjal terhelt input változatlannak tűnik. Ilyen támadások minden ismert neuronhálózat ellen sikerrel, és meglepően könnyen kivitelezhetők. Ennek a típusú támadásnak az első leírását adja *(Szegedi et al. 2014)*. A vonatkozó szakirodalom óriási, egy friss összefoglalót ad *(Xu et al. 2019)*. A területen saját eredményekkel is rendelkezünk, egyik kutatásunk során megmutattuk, hogy olyan ellenséges input is generálható, amely egyszerre több neuronhálót is félre tud vezetni, mégpedig előre megadott mintázat szerint *(Megyeri–Hegedűs–Jelasity 2020)*. Az *1. ábra* ezt a feladatot szemlélteti, ahol a kérdőjellel jelölt perturbációt kell megtalálni úgy, hogy azt a képhez adva a különböző neuronhálók előre megadott kimenetet produkálnak.

Az ellenséges példák a neuronhálók „optikai illúziójának" tekinthetők. Bár az emberek esetében is ismert a jelenség, a humán illúziók jellemzőiben jelentősen eltérnek az MI rendszereken fenti módon értelmezettektől. Annak ellenére, hogy számos módszerrel lehetséges némileg javítani a neuronhálózatok robosztusságán az ellenséges példákkal szemben, egyelőre még messze vagyunk az olyan megoldásoktól, amely az emberi döntésekkel elfogadható konzisztenciát mutatna.

MI rendszerek tesztelése, verifikációja

A komplex MI rendszerek egyik alapvető dilemmája, hogy számos esetben csak viselkedésük megfigyelhető jellemzőit, valamint a bennük alkalmazott heurisztikákat és algoritmikus megoldásokat ismerjük, a pontos részletek, azaz például miért történik egy adott döntés úgy, ahogy az adott esetben történik, rejtve maradnak. Tudjuk azonban azt is, hogy célzott eljárásokkal és megfelelő adatok felhasználásával egy ilyen rendszert is könnyen helytelen működésre lehet kényszeríteni. Ez a tény az MI rendszerek megbízhatóságára és alkalmazására vonatkozólag jelentős kockázatot jelent *(Gleave et al. 2020)*.

Az MI rendszerek verifikációjára módszertani szempontból rendkívül változatos eszközkészlet áll rendelkezésre. A mesterséges neuronhálózatok, mint matemati-



1. ábra | Ellenséges input generálása

kai objektumok vizsgálata kiemelt kutatási terület, ahol az olyan módszerek, mint az intervallum aritmetika, dinamikus rendszerek stabilitásvizsgálata, lineáris programozás, kombinatorikus optimalizálás, globális optimalizálás, vagy az automatikus tételbizonyítás módszerével történő tulajdonságok igazolása, nyújtanak eszközkészletet ezeknek az objektumoknak vizsgálatához *(Fischetti–Jo 2018; Forti–Nistri–Papini 2005; Hamm– Brorsen–Hagan 2007)*. Az említett módszerek kombinációjával a komplexebb felépítésű mély neuronhálók vizsgálatára alkalmas eljárások kidolgozására is lehetőség nyílik.

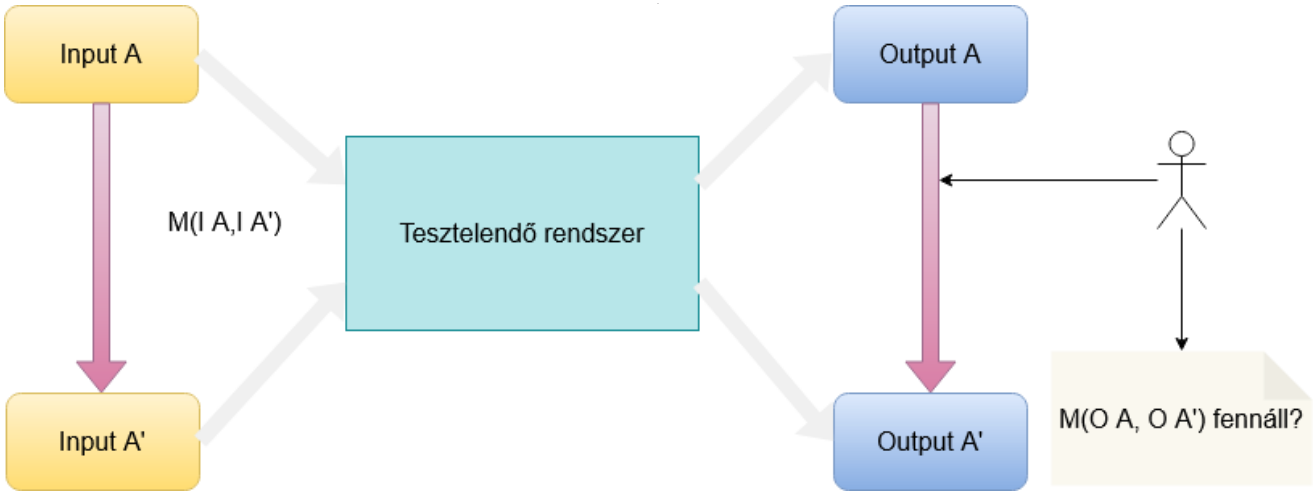
Az MI alapú rendszerek (így a mély neuronhálók is) elvárt működése jelenleg csak empirikusan ellenőrizhető, alkalmas tesztkészletek megválasztása útján *(Sun et al. 2018)*. A hagyományos felépítésű szoftverrendszerek szisztematikus teszteléséhez hasonló módszerek és eszközök ma még nem állnak rendelkezésre. Annak érdekében, hogy az e tényből adódó problémákat elkerüljük, az MI rendszerek teszteléséhez adekvát módszertan kidolgozására van szükség. Ez magában foglalja az ún. black-box (specifikáció alapú) hagyományos módszerek adaptálását (pl. input tér partícionálása vagy kombinatorikai módszerek), de a white-box (struktúra alapú) megközelítések kutatását is *(Ehmer–Khan 2012)*. Utóbbi a különböző teszt-lefedettségi kritériumok kidolgozását és azok megvalósítását jelenti, amelyre példa lehet az egyes neuronok aktiválási szintjeinek mérése *(Rauber et al. 2017)*, de egyéb más módszerek is elképzelhetők.

További kutatási terület a rendszerek robosztusságát célzó tesztelési módszerek kidolgozása, amely számos bemenet véletlenszerű generálása útján teszi próbára a vizsgált rendszer stabilitását, helyességét. Ilyen megoldások például a fuzzing *(Xie et al. 2019)*, vagy egyéb Monte Carlo jellegű tesztelési eljárások *(Gandy–Scott 2020)*. Ezeknek a módszereknek egyik fontos jellemzője, hogy adott célfüggvény által vezérelve (pl. neuron lefedettség, input adat lefedettség) automatikusan képesek a tanuláshoz szükséges adatok előállítására, ami segíti a tesztlefedettség elérését, ezáltal a teszt megbízhatóságát is növeli. A fuzz tesztelés területén jelentős eredményeket értek el a Szegedi Tudományegyetem kutatói *(Hodovan–Kiss 2018; Hodovan–Vince–Kiss 2019)*, amelyek mind klasszikus, mind biztonsági tesztelés teszteseteinek létrehozásában hasznosnak bizonyultak.

Az említett, input vezérelt esetekben a vizsgált rendszer elvárt működése, kimenete is meghatározandó feladat, amely újabb kihívásokat támaszt a tesztelő felé. Az így jelentkező orákulum problémának nevezett dilemma csökkentésére a hagyományos tesztelési eljárásokban alkalmazott metamorf módszerek vizsgálata az utóbbi években a mesterséges neuronhálók körére is kiterjedt *(Chen et al. 2018)*. A metamorf tesztelés során az input és output között ún. metamorf relációt határozunk meg és a rendszer kimenetét ezen reláció ellenében vizsgáljuk (ld. *2. ábra*). Hasonlóan a metamorf teszteléshez az adattranszformációs módszerek alkalmazása *(Zhu et al. 2019)* is segítséget nyújthat a tesztesetek generálása során. Ez utóbbi az eljárás során alkalmazott adattranszformációt modellezi makroszkopikus úton, s veti össze az így létrejött elvárt eredményt a kapott kimenettel.

Hibakeresés MI alapú rendszerekben

A tesztelés során az MI alapú rendszerekben lévő hibák jelenlétét tudjuk kimutatni, de azok pontos helye és javítási módja nem mindig egyértelmű. A komplex, mély neuronhálós modellekben való hibakeresés jelenleg is



2. ábra | Metamorf tesztelés

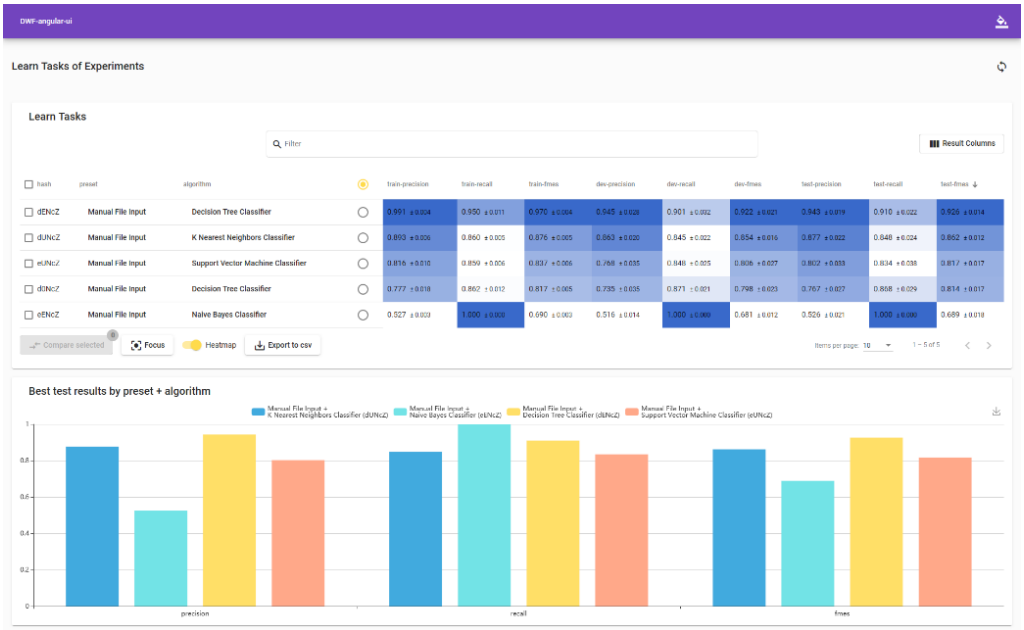
nyitott probléma. A kezdeti kutatásokon túlmutató, jól megszokott interaktív/támogatott hibakeresési módszer jelenleg nem áll rendelkezésünkre, amely egyébként a hagyományos szoftverfejlesztés bevett napi eszközkészletébe tartozik. Amennyiben a modell egy bemenetre nem az elvárt kimenetet produkálja, szükség van egy módszerre, melynek segítségével a modell működését lépésről lépésre végig elemezhetjük, és a megfelelő hibapontokat azonosíthatjuk. A hagyományos szoftverfejlesztésben az automatikus hibalokalizációs módszerek statisztikai megközelítése a program futási nyomait feldolgozva és azokat a tesztekre vetítve a program soraihoz ún. gyanússági értéket rendel, amely megmutatja, hogy az adott sor mennyire valószínű, hogy hibát tartalmaz *(Jones et al. 2002)*. Egy friss kutatásban megmutattuk, hogy a hibalokalizációs eljárások pontossága javítható, ha a program futásából nem csak az ún. lefedettségi értékeket, hanem a hívási láncolatokat is figyelembe vesszük, amelyhez azonban sokkal nagyobb mennyiségű adat feldolgozására van szükség *(Beszédes et al. 2020)*. Természetes módon felmerül, hogy hasonló módszerek a mesterséges neuronhálók területén is alkalmazhatók, adaptálhatók. Egy úttörő munkaként a teszt lefedettség fogalmát Sun és munkatársai *(Sun et al. 2019)* ültették át neuronhálókra, a teszt lefedettség számításához szükséges teszteket automatikusan generálva. Szintén 2019-ben jelent meg egy tanulmány, mely a korábban említett gyanússági értékeket neuronokhoz rendelve kísérletet tesz a neurális háló azon neuronjainak azonosítására, amelyek esetében a súlyok nem megfelelően kalibráltak *(Eniser et al. 2019)*.

A hibalokalizációt követően az automatikus hibajavító módszerek (automatic program repair) is alkalmazhatóvá válnak. Ez a terület jelenleg a hagyományos fejlesztésben is aktuális kutatási téma, amelyben a fő módszer a hibajavító foltok változatos generálása és a jelöltek teszteléssel történő ellenőrzése (generate and test). Itt a mesterséges intelligencia leginkább megoldási módszerként van jelen *(Tarlow et al. 2019)*, például a generált hibajavító jelöltek jóságának eldöntéséhez egy friss munkánkban demonstráltuk a népszerű szóbeágyazási módszerek (doc2vec, BERT) forráskódon történő alkalmazhatóságát *(Csuvik et al. 2020)*. Kimondottan neurális hálók, illetve azok megvalósítását biztosító rendszerek hibajavítására készült módszerek a közeljövőben várhatóak *(Islam et al. 2020)*.

MI fejlesztést támogató keretrendszerek

A mesterséges intelligencia, különösen az egyes gépi tanuló algoritmusok alkalmazása mára nemcsak a szoftverfejlesztők, hanem más szakemberek és kutatók egy széles rétege számára is mindennapossá vált. A modellek tanítása gyakori feladattá nőtte ki magát, amely magában foglalja a legjobb tanuló algoritmus kiválasztását, annak legjobb paraméterezését egy adott feladatra, az egyes modellek által elért eredmények nyomon követését és vizualizálását, stb. Mindezek mellett a tanulás alapját képező jellemzők kinyerése, amely önmagában is számos paramétertől függ (például a szó vagy forráskód beágyazások) is komoly kihívást jelent. Ezek a feladatok egyértelműen megfelelő szoftvertámogatásra szorulnak, jelenleg viszont a mesterséges intelligencia modell fejlesztési folyamatának csak bizonyos elemei támogatottak többnyire specifikus eszközök alkalmazása útján.

Természetesen léteznek olyan eszközök *(Polyaxon 2019)*, *(Studio.ml Community 2017)*, *(Microsoft 2018)*, *(Databrick Inc. 2020)*, *(Greff et al. 2018)*, *(Comet)*, amelyek lehetővé teszik gépi tanuló modellek létrehozását, ám ezek közös jellemzői, hogy i) elsősorban a mély neurális hálókat használó keretrendszerekre fókuszálnak (pl. PyTorch, Tensorflow, Keras) és ii) feltételezik, hogy felhasználójuk jártas ezeknek a keretrendszereknek a használatában, valamint mély tudással rendelkeznek a gépi tanulás területén. Azonban egyik feltételezés sem feltétlenül igaz minden potenciális felhasználóra, hiszen sokszor a jól bevált, klasszikus algoritmusok (döntési fák,



3. ábra | DeepWater adat kiértékelő dashboard

regresszió, SVM, stb.) sokkal hatékonyabbak tudnak lenni egy-egy probléma megoldásában. Ráadásul sok nem informatikus kutató vagy szakember is szeretné kihasználni a gépi tanulásban rejlő potenciált, akik rengeteg adattal rendelkeznek, de nem feltétlenül jártasok a gépi tanulás elméletében, illetőleg az ismert keretrendszerek kezelésében.

Figyelembe véve a fentieket, a mesterséges intelligencia alapú szoftverek létrehozását támogató, a hagyományos szoftverfejlesztés során már régóta alkalmazott úgynevezett konfigurációmenedzsment jellegű szoftverek tulajdonságaival bíró eszközökre lenne szükség, amely megfelelő módon támogatja a különböző tanuló modellek létrehozásának teljes folyamatát. Ilyen támogatást jelenthet például a jellemző kinyeréshez használt stratégiák integrálása, paramétereinek keresése, megfelelő gépi tanuló algoritmus ajánlása, a modellek eredményeinek tárolása, lekérdezése, vizualizálása, de akár az erőforrás igényes tanítási folyamat párhuzamosítása is. Egy ilyen rendszer nemcsak a mesterséges intelligencia területén jártas szakemberek, hanem egy szélesebb réteg kezébe (legyen az nyelvész, biológus vagy informatikus kutató, esetleg szakember) adna olyan eszközt, amellyel viszonylag kis erőbefektetéssel, megismételhető és visszakereshető módon tudnának tanuló modelleket előállítani. Az SZTE-n zajlik a DeepWater keretrendszer *(Ferenc et al. 2020; University of Szeged Department of Software Engineering 2019)* fejlesztése, amely a fenti jellemzőkkel bír, és megoldást nyújt a felvázolt problémák többségére, akár a gépi tanulásban kevésbé jártas szakemberek számára is. Az eszköz egyszerű grafikus felület segítségével, számos javasolt alapbeállítással támogatja a gépi tanuló modellek széles skálájával (a klasszikus algoritmusoktól kezdve a sekély neurális hálókon át a mély neurális hálókig) történő kísérletezést, optimális paraméterezés megtalálását, az eredmények verziókezelt tárolását, valamint az eredmények kiértékelését támogató adatvizualizációt (lásd *3. ábra*). Sikeresen alkalmaztuk a keretrendszert szoftverhibák, valamint biztonsági sérülékenységek előrejelzésére alkalmas modellek kialakításához és kiértékeléséhez. Továbbá több jelenleg is aktív kutatás során hasznosítjuk az eszközt, ahol természetes nyelvű szövegekből, illetve forráskódból ún. beágyazásokkal készítünk predikcióhoz használható jellemző vektorokat.

Irodalomjegyzék

Beszédes, Á., Horváth, F., Di Penta, M. & Gyimóthy, T. (2020) Leveraging Contextual Information from Function Call Chains to Improve Fault Localization. *Accepted for presentation at the 27th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2020)*. London, Ontario, Canada.

Chen, T.Y., Kuo, F-C., Liu, H., Poon, P-L., Towey, D., Tse, T.H. & Quan Zhou, Z. (2018) Metamorphic Testing: A Review of Challenges and Opportunities. A*CM Comput. Surv.,* Vol. 51, No. 1, Article 4. DOI:10.1145/3143561

Comet. *Comet website https://www.comet.ml/*

Csuvik, V., Horváth, D., Horváth, F. & Vidács, L. (2020) Utilizing Source Code Embeddings to Identify Correct Patches. *Proceedings of the Second International Workshop on Intelligent Bug Fixing (IBF 2020)*, pp. 18–15 IEEE.

Databricks Inc. (2020) *Mlflow, an open source platform for the machine learning lifecycle*. https://mlflow.org/

Eniser, H.F., Gerasimou, S., Sen, A. Hähnle, R., van der Aalst, W. (eds) (2019) DeepFault: Fault Localization for Deep Neural Networks. *Fundamental Approaches to Software Engineering. FASE 2019. Lecture Notes in Computer Science*, Vol 11424. Springer, Cham.

European Commission (eds) (2020) On Artificial Intelligence – A European approach to excellence and trust. *White paper. COM(2020)*

65, Brussels 2020.02.19. https://ec.europa.eu/info/sites/info/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf [Letöltve: 2020.03.04]

Fischetti, M. & Jo, J. (2018) Deep neural networks and mixed integer linear optimization. *Constraints,* Vol 23. pp. 296–309. DOI:10.1007/s10601-018-9285-6

Forti, M., Nistri, D. & Papini, D. (2005) Global exponential stability and global convergence in finite time of delayed neural networks with infinite gain. *IEEE Transactions on Neural Networks*, Vol. 16, No. 6, pp. 1449-1463. DOI:10.1109/TNN.2005.852862

Gandy, A., & Scott, J. (2020) Unit Testing For Mcmc And Other Monte Carlo Methods. arXiv:2001.06465

Greff, K., Klein, A., Chovanec, M., Hutter, F. & Schmidhuber, J. (2018) *The Sacred Infrastructure for Computational Research* DOI:10.6084/m9.figshare.5813412.v1.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S. & Russell, S. (2020) Adversarial Policies: Attacking Deep Reinforcement Learning. *International Conference on Learning Representations*

Hamm, L., Brorsen, B.W. & Hagan, M.T. (2007) Comparison of Stochastic Global Optimization Methods to Estimate Neural Network Weights. *Neural Process Lett,* Vol. 26**,** pp**.** 145–158. DOI: 10.1007/s11063-007-9048-7

Harrold, M. J. & Stasko, J. (2002) Visualization of test information to assist fault localization, 24th International Conference on Software Engineering, ACM, 2002, pp. 467–477.

Hodován, R. & Kiss, Á. (2018) Fuzzinator: An Open-Source Modular Random Testing Framework. *Proceedings of the 11th IEEE International Conference on Software Testing, Verification and Validation (ICST 2018)*, pp. 416-421,DOI: 10.1109/ICST.2018.00050

Hodován, R., Vince, D. & Kiss, Á. (2019) Fuzzing JavaScript Environment APIs with Interdependent Function Calls. *Integrated Formal Methods – 15th International Conference, IFM 2019, Bergen, Norway. Lecture Notes in Computer Science (LNCS)* Vol. 11918, pp 212–226. DOI: 10.1007/978-3-030-34968-4_12

Islam, J., Pan, R., Nguyen, G. & Rajan, H. (2020) Repairing Deep Neural Networks: Fix Patterns and Challenges. *ICSE'20: The 42nd International Conference on Software Engineering (May 2020).*

Jones, J. A., Harrold M. J., & Stasko J. (2002) Visualization of test information to assist fault localization, 24th International Conference on Software Engineering, ACM, 2002, pp. 467–477.

Megyeri, I., Hegedűs, I. & Jelasity, M. (2020) Adversarial Robustness of Model Sets, *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*J. A. Jones,

Microsoft (2018) *Azure Machine Learning Studio.* https://studio.azureml.net/

Khan, M.e. & Khan, F. (2012) A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 3, No. 6. DOI: 10.14569/IJACSA.2012.030603

Polyaxon (2019) *Polyaxon an open source platform for reproducible machine learning at scale.* https://polyaxon.com/.

Rauber, P.E., Fadel, SG., Falcão, A.X. & Telea, A.C. (2017) Visualizing the Hidden Activity of Artificial Neural Networks. *IEEE* Microsoft *Transactions on Visualization and Computer Graphics*, Vol. 23, No. 1, pp. 101–110. DOI:10.1109/TVCG.2016.2598838

Rudolf, F., Viszkok, T., Aladics, T., Jász, J., Hegedűs, P. (2020) Deepwater framework: The Swiss army knife of humans working with machine learning models, SoftwareX, Vol 12, 7 pages, DOI: https://doi.org/10.1016/j.softx.2020.100551

Studio.ml Community (2017) *Studio.ml.* https://studioml.readthedocs.io/en/latest/

Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M. & Ashmore, R. (2019) Structural Test Coverage Criteria for Deep Neural Networks. *ACM Trans. Embed. Comput. Syst.,* Vol. 18, No. 5, Article 94 , 23 pages. DOI:10.1145/3358233

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., D. Erhan, D., Goodfellow, I.J. & Fergus R. (2014) Intriguing properties of neural networks. 2nd International Conference on Learning Representations (ICLR). arXiv:1312.6199

Tarlow, D., Moitra, Rice, A., Chen, Z., Manzagol, P-A., Sutton, C. & Aftandilian, E. (2019) Learning to Fix Build Errors with Graph-2Diff Neural Networks. CoRR abs/1911.01205

University of Szeged Department Of Software Engineering (2019) *Deep Water Framework https://github.com/sed-inf-u-szeged/Deep-WaterFramework*

Zhu, H., Liu, D., Bayley, I., Harrison, R. & Cuzzolin, F. (2019) „Datamorphic Testing: A Method for Testing Intelligent Applications," *IEEE International Conference On Artificial Intelligence Testing (AITest)*, Newark, CA, USA, pp. 149-156. DOI: 10.1109/AITest.2019.00018

Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J. & See, S. (2019) DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019)*. ACM, New York, NY, USA, pp. 146–157. DOI:10.1145/3293882.3330579

Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J. & Jain. A.K (2019) Adversarial attacks and defenses in images, graphs and text: A review. Technical Report 1909.08072, arxiv.org.