DOI: https://doi.org/10.24297/ijct.v21i.9009

An Empirical Model For Validity And Verification Of Ai Behavior: Overcoming Ai Hazards In Neural Networks

Ayse K. Arslan

[1]*Association of Oxford Alumni, Northern California, USA*

Abstract

Rapid progress in machine learning and artificial intelligence (AI) has brought increasing attention to the potential impacts of AI technologies on society. This paper discusses hazards in machine learning systems, defined as unintended and harmful behavior that may emerge from poor design of real-world AI systems with a particular focus on ANN. The paper provides a review of previous work in these areas as well as suggesting research directions with a focus on relevance to cutting-edge AI systems with a focus on neural networks. Finally, the paper considers the high-level question of how to think most productively about the safety of forward-looking applications of AI.

Key Words: AI, machine learning, research, algorithms, neural networks, software development, engineering

1.INTRODUCTION

There is now a broad consensus that AI research is progressing steadily, and that its impact on society is likely to increase. The last few years have seen rapid progress on long-standing, difficult problems in machine learning (ML) and artificial intelligence (AI), in diverse areas which brought excitement about the positive potential for AI to transform medicine [12], science [9], and transportation [6], along with concerns about the privacy [7], security [1], fairness [3], economic [32], and military [16] implications of autonomous systems, as well as concerns about the longer-term implications of powerful AI [27, 17].

The aim of this paper is to catalogue some of the various possible ways in which AI, especially within the context of ANN (artificial neural networks) can cause harm. The aim is not to determine how common and serious these harms are or how they stack up against the many benefits of information—questions that would need to be engaged before one could reach a considered position about potential policy implications, yet rather to enlighten the reader on potential threats caused by this technology.

2. EXISTING WORK

Artificial Intelligence (AI) refers to the art of creating machines that are able to think and act like humans; or think and act reasonably [4, 7]. In order to build an agent that can think and act as so, the agent must be able to learn new things. To learn means that the agent should improve its performance on future tasks taking its past experience into account [20, 7]. Making an agent able to learn is an area of study called Machine Learning (ML).

Artificial Neural Network or ANN is a software structure developed and based on concepts inspired by biological functions of brain; it aims at creating machines able to learn like a human-being [2, 7]. Thus, ANN is part of ML. Interestingly, ANN has many other names in AI field including parallel distributed processing, neural computation and connectionism [12, 11]. Most ANN types are supervised learning network. That is, both an input and the correct output should be given to a network where the network should learn a function that maps inputs to outputs.

ANN may refer to two levels of abstraction:

(1) ANN as a person's brain and

(2) ANN as a group of learners.

Thus, network architecture refers first to a learner's inner abilities and mental capacities and; second, refers to a way in which designers of learning-environment arrange a network of learners. It is worth noting that ANN is a universal modeling system. Universality means that ANN can learn any given function no matter what neuron type is used. It has been proved that with few neurons and by changing biases and weights only, ANN can compute any zigzag-shaped function [2]. The question now is how we arrange neurons in ANN to make it easier for a learning algorithm to find those biases and weights.

For clarity and simplicity, the paper divides the most common ANN architectures based on three criteria: (1) number of layers, (2) flow of information and (3) neuron connectivity.

Learning Algorithm

Designing network architectures is a difficult task but training and teaching these networks are surely more difficult. To understand how ANN has been trained, it is better to start with a very simple one neuron example [26]. The principles which are used to teach a single neuron are also used to teach a whole network. However, a network level adds extra complexity which requires an additional step. Suppose you have a very simple neuron with one input and one output. You want to teach this neuron to do a certain task (for example to memorize a multiplication table for number 5). To teach this neuron, ANN researchers usually give it a so-called training set. A training set contains a number of different input values (1, 2, 3, 4, 5, 6 ...) paired with the correct output (5, 10, 15, 20, 25, 30 ...).

One note in ANN model of learning is how AI researchers are setting the value of learning rate. Actually, learning rate is one of many other parameters which are left free for human and outside of ANN's control. For example, (1) the number of layers, (2) the number of neurons in each layer, (3) the size of training set, (4) the activation function type, and (5) regularization parameter as well as (6) the learning rate are some of those free parameters which are called hyperparameters [24, 18] Choosing the right values of hyper-parameters is left for a person who manages the ANN.

Bandura [18] criticizes those views of human learning which concentrate merely on neural patterns to interpret learning and argues that such views strip humans of agentic capabilities and a self-identity. In contrary, Bandura [18] conceives consciousness as an emergent property of brain activities which is not reducible solely to the property of neurons activity. In other words, the consciousness is higher-level force which is a result of lower-level neural activities but its properties are not limited to them. As clarified in this study, ANN design shows the need for consciousness force to manage and regulate ANN learning but this force does not occur as an emergent property of neural activity as Bandura proposes. Rather, it is a completely distinct entity which uses, guides and manages the neural activity and does not result from it. Therefore, overcoming hazards in the field AI becomes crucial to maximize societal benefit of AI given its significant expansion.

3. RESEARCH METHOD

Research has been developed and constructed based on a review of various books focusing on Russell and Norvig (2016), Tinholt, et al. (2017), Tito (2017), and Zhang and Dafoe (2019). This research identifies various concepts that are very helpful in formulating final questions. These simple but effective methods are useful to achieve the purpose of exploratory research.

How can one enable meaningful human control over an AI system after it begins to operate? ("Ok, I built the system wrong, can I fix it?")

RESULTS

User-Centered Design (UCD) is a systematic approach that is used to overcome AI hazards during software development. UCD is typically divided into 5 main phases:

- Phase 1 – User Research
- Phase 2 – High-level design
- Phase 3 – Detailed design
- Phase 4 – Development and development support
- Phase 5 – Testing and Installation support

User centered design focuses on software development using a top-down holistic approach. The main goal of UCD is to start by building a user navigation model first using multiple UCD tools and techniques in alignment with ANN development goal. The second major step will be to use this navigation model to define and design the ANN application structure using multiple prototyping techniques (Figure 1).

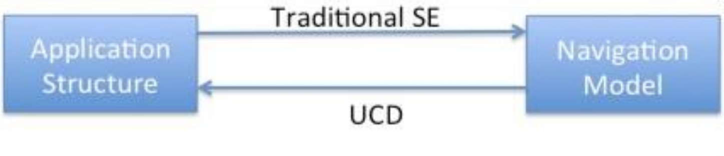

Figure 1: Traditional SE (software engineering) vs UCD

As it can be seen in Figure 1, given the main departure from the traditional software development starting with application structure, the main focus of UCD is on the following different points:

1) Who are the users for ANN?
2) What do the users currently do?
3) What do the users want from the new software based on ANN?

The first step of identifying the users starts by meeting with different stakeholders affiliated with the training domain. Two main aspects need to be considered:

i. Different types of users are researched and identified as "User Roles".

ii. Users can have different backgrounds, different experience, and different context to teach in. These differences are looked at using "User Personas" as a common UCD tool that is growing in popularity.

The second part, "what the users currently do" is done via different UCD tools. Several users can be met during brainstorming sessions or design workshops to get their input on their daily work practices. A series of interviews and questionnaires could also supplement the work resulting in potential categories for data analysis.

The third part, "what the users want" can be conducted with deeper analysis on the findings extracted in the previous part. This is where the navigation modeling is used as an innovative technique to envision the user needs in terms of a navigation structure that can be translated into an actual application structure.

After successful completion of the User Research phase, the high-level and detailed design phases can start. The following sections provide an overview of the various phases of UCD.

Phase 1 – User Research

During the User Research phase, focus groups with engineering and computing systems staff can be organized to understand the course design process used by the participants. Participants can also be asked to fill an electronic questionnaire about software design tools that they currently use to create and manage their software

Data collected from the focus groups about software design process can be categorized as inputs, processing and decision-making, and output artifacts.

Phase 2 – High-level Design

Once the user research provided a relatively clear idea and understanding of domain- and user needs, this initial design phase provides a high-level design with concepts identification, conceptual modeling and early prototyping for ANN. The main goal of high-level design is to plot down schematic ideas and steps into visual graphs and models; an early blueprint of ANN. This can be done by investigating different options and provide design alternatives to ensure a broad view before identifying a good design. Doing this early on, at high-level, sketchy, paper-based only, and without going into details could help provide several solution alternatives at a very low cost. The high- level design sketches can be discussed with the users to make sure what they said in unstructured dialogs and vague ideas and imaginations can now be concretely captured in design artifacts for further validation and clarifications. At this stage, there are 2 tools that are most suitable for the development stage:

Tools:

a) **Navigation Model** is one of the essential methods of design. A significant challenge in complex software is not the contents of each screen, but how the user mentally builds a mental view of how all screens are connected (like a city road map), and how to navigate between hundreds of screens to accomplish their task. In this regard, an effective technique- elastic prototyping- can be used an implementation of a participatory design to help designers and users build a navigation model together, greatly reducing time and effort needed.

b) **Prototyping (PT)** is extensively used in UCD to visualize and validate all otherwise vague ideas and unclear expectations at low cost and high effectiveness. There exist three main categories of prototyping: Paper (low-level) PT, low-fidelity electronic (medium level) PT, and high-fidelity, detailed PT. Paper prototypes are very inexpensive and help capture several initial ideas and concepts, and validate them. After explaining their needs, users often change their minds when they see them on paper. Therefore, multiple paper PT sessions gives a head start in validating what users actually mean and need. After initial concepts, once design ideas and directions were identified, a medium fidelity prototyping stage can start where a sketchy visualization of key screens without contents are provided to be gradually validated them and added with initial contents.

Phase 3 – Detailed Design

At this stage, the focus is on the main high level solution, including details from different perspectives such as main application features, auxiliary features, concrete navigation models, menu options, visual and interaction consistency across all screens, exceptions and error massages and recovery, reliability assurances and, help. This phase can proceed in parallel with development phase as more details are uncovered and technical problems arise. User interface mockups can be created with details of various user inputs that will be solicited through the course design process.

Phase 4 – Development and Development Support

As implementation of essential features starts, close collaboration between designers and software engineers (software architects and developers) is essential to ensure the consistency of design and to prevent any deviations.

Several technical problems require careful reconsideration of detailed design and even high-level design options. Iteration is a fundamental design approach that is extensively being used across the UCD process. Therefore, UCD is highly iterative and most of its phases are heavily overlapping to ensure design and development decisions are aligned at all times with the actual user needs.

This phase of the project includes identifying appropriate technologies to be used for the development of the ANN application, design of the back-end database schema, installation and configuration of the server-side and client-side technologies, and development of the user interface screens for login, registration, index, and creation of an instructional module and the connectivity of these web pages with the backend database.

Analysis of Technologies

The purpose of analyzing various technologies during this phase of the project is to ensure rapid development with the latest technologies in the field of software development and use open source technologies wherever feasible. Towards this end, an analysis of web application frameworks, version control systems, server side technologies and client side technologies can be performed.

System Architecture

A Model–View–Controller (MVC) architecture is suggested as the underlying web application framework. MVC is a software architecture pattern which separates the representation of information from the user's interaction with it. The recommended architecture can be described as follows:

- The foundation is the Java Virtual Machine (JVM).
- There is a separation between the Java language and the JVM.
- The final layer of the architecture is the application layer. This layer follows the Model-
- View-Controller (MVC) pattern.
- A controller handles requests and creates or prepares the response. A controller can generate the response directly or delegate to a view.
- A controller can have multiple public action methods, each of which maps to a URI.

Table 1 shows some exemplary technologies to inform the final selection.

Key Architecture Functions	**Possible Solutions**	**Analysis Comments**	**Final Solutions**
Framework	Django, Grails, WebApp2	• Django and WebApp2 are written in Python and have Google App Engine support • Django has request handler, template engine and form processor • WebAp 2 has request handler	
Version Control	Bitbucket, Github, Gitlab, Gitlolite, SVN	• Bitbucket - free private repositories • Github - free public repos + paid private repos • SVN is centralized, Git is decentralized • All work with Unix, Linux and Windows systems	
Databases	SQL, NoSQL, JSON, Google Datastore, MySQL, PostGreSQL	• App Engine Datastore provides a NoSQL schema-less object datastore, with a query engine and atomic transactions • ANN data is expected to have numerous relations and hence schema-less store is not being chosen	
Client side scripting	ExtJS, jQuery, JavaScript, CoffeeScript, AngularJS, BackboneJS, HTML5, CSS3, Twitter Bootstrap	• Backbone.js requires more Boilerplate code, but is smaller than Angular.js • ExtJS does not provide good support	
Semantic Web Technologies	OWLite, Protege, Apache Jena	Apache Jena • API for reading, processing and writing RDF data in XML, N-triples and Turtle formats • Rule-based inference engine for reasoning with RDF and OWL data sources	

		• Stores to allow large numbers of RDF triples to be efficiently stored on disk • Query engine compliant with the latest SPARQL	
Licensing		• GPL, MIT, BSD - It is not permissible under the GPL to use GPL in proprietary software while keeping that software closed source • MIT and BSD: Because you cannot restrict others from simply obtaining the source code, selling open source licensed software as is makes for a difficult proposition	
Cloud/web technologies	Google App Engine, Amazon Web Services (AWS)	• Google App Engine: Free within quota, help and tutorial • AWS: Free usage for a year	

Table 1: Analysis of Technologies for ANN development

A model is a Map that the view uses when rendering information on the web page. The keys within that Map correspond to variable names accessible by the view.

DISCUSSION

There are many ways of responding to information hazards. In many cases, the best response is no response, i.e., to proceed as though no such hazard existed. The benefits of information may so far outweigh its costs that even when information hazards are fully accounted for, we still under-invest in the gathering and dissemination of information. Moreover, ignorance carries its own dangers which are oftentimes greater than those of knowledge. Information risks might simply be tolerated.

When mitigation is called for, it need not take the form of an active attempt to suppress information through measures such as bans, censorship, disinformation campaigns, encryption, or secrecy. One response option is simply to invest less in discovering and disseminating certain kinds of information. Somebody who is worried about the spoiler hazard of learning about the ending of a movie can simply refrain from reading reviews and plot summaries.

At the same time, however, we should recognize that knowledge and information frequently have downsides. Future scientific and technological advances, in particular, may create information which, misused, would cause tremendous harm—including, potentially, existential catastrophe.

It can also be hoped that new information technologies will bring about a vastly more transparent society, in which everybody (the watchmen included) are under constant surveillance; and that this universal transparency will prevent the worst potential misuses of the new technological powers that humanity will develop.

CONCLUSION

Even if our best policy is to form an unyielding commitment to unlimited freedom of thought, virtually limitless freedom of speech, an extremely wide freedom of inquiry, we should realize not only that this policy has costs but that perhaps the strongest reason for adopting such an uncompromising stance would itself be based on an information hazard; namely, norm hazard: the risk that precious yet fragile norms of truth-seeking and truthful reporting would be jeopardized if we permitted convenient exceptions in our own adherence to them or if their violation were in general too readily excused.

It is said that a little knowledge is a dangerous thing. It is an open question whether more knowledge is safer. Even if our best bet is that more knowledge is on average good, we should recognize that there are numerous cases in which more knowledge makes things worse.

REFERENCES

1. Rakesh Agrawal and Ramakrishnan Srikant. "Privacy-preserving data mining". In: ACM Sigmod Record 29.2 (2000), pp. 439–450.
2. Rajeev Alur. "Formal verification of hybrid systems". In: Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on. IEEE. 2011, pp. 273–278.
3. Kenneth Anderson, Daniel Reisner, and Matthew C Waxman. "Adapting the Law of Armed Conflict to Autonomous Weapon Systems". In: International Law Studies 90 (2014).
4. Susan Leigh Anderson and Michael Anderson. "A Prima Facie Duty Approach to Machine Ethics Machine Learning of Features of Ethical Dilemmas, Prima Facie Duties, and Decision Principles through a Dialogue with Ethicists". In: Machine Ethics (2011), p. 476.
5. David Andre and Stuart J Russell. "State abstraction for programmable reinforcement learning agents". In: Eighteenth national conference on Artificial intelligence. American Association for Artificial Intelligence. 2002, pp. 119–125.
6. Stuart Armstrong, Nick Bostrom, and Carl Shulman. "Racing to the precipice: a model of artificial intelligence development". In: (2013).
7. Stuart Armstrong, Kaj Sotala, and Se ́an S O hEigeartaigh. "The errors, insights and lessons of famous AI predictions–and what they mean for the future". In: Journal of Experimental & Theoretical Artificial Intelligence ahead-of-print (2014), pp. 1–26. url: http://www.fhi.ox.ac.uk/wp- content/uploads/FAIC.pdf.
8. Gustaf Arrhenius. "The impossibility of a satisfactory population ethics". In: Descriptive and normative approaches to human behavior (2011).
9. Peter M Asaro. "What should we want from a robot ethic?" In: International Review of Information Ethics 6.12 (2006), pp. 9–16.
10. Peter Asaro. "How just could a robot war be?" In: Current issues in computing and philosophy (2008), pp. 50–64.
11. Karl J Astr ̈om and Bj ̈orn Wittenmark. Adaptive control. Courier Dover Publications, 2013.
12. Silvia Bellezza, Anat Keinan, and Neeru Paharia. Conspicuous Consumption of Time: When Busyness at Work and Lack of Leisure Time Become a Status Symbol. 2014. url: http://www.hbs.edu/ faculty/Pages/item.aspx?num=47139.
13. M Boden et al. "Principles of robotics". In: The United Kingdom's Engineering and Physical Sciences Research Council (EPSRC). web publication (2011).
14. Nick Bostrom. "Infinite ethics". In: Analysis and Metaphysics 10 (2011), pp. 9–59.
15. Nick Bostrom. Moral Uncertainty–Towards a Solution? 2009. url: http://www.overcomingbias. com/2009/01/moral-uncertainty-towards-a-solution.html.
16. Nick Bostrom. Superintelligence: Paths, dangers, strategies. Oxford University Press, 2014.
17. Nick Bostrom. "The superintelligent will: Motivation and instrumental rationality in advanced arti- ficial agents". In: Minds and Machines 22.2 (2012), pp. 71–85.
18. Ju ̈rgen Branke et al. Multiobjective optimization: Interactive and evolutionary approaches. Vol. 5252. Springer Science & Business Media, 2008.

19. Selmer Bringsjord et al. “Piagetian roboethics via category theory: Moving beyond mere formal operations to engineer robots whose decisions are guaranteed to be ethically correct”. In: Machine ethics (2011), pp. 361–374.

20. Yuriy Brun and Michael D Ernst. “Finding latent code errors via machine learning over program executions”. In: Proceedings of the 26th International Conference on Software Engineering. IEEE Computer Society. 2004, pp. 480–490.

21. Erik Brynjolfsson and Andrew McAfee. The second machine age: work, progress, and prosperity in a time of brilliant technologies. W.W. Norton & Company, 2014.

22. Erik Brynjolfsson, Andrew McAfee, and Michael Spence. “Labor, Capital, and Ideas in the Power Law Economy”. In: Foreign Aff. 93 (2014), p. 44.

23. Ryan Calo. “Robotics and the New Cyberlaw”. In: Available at SSRN 2402972 (2014).

24. Ryan Calo. “The Case for a Federal Robotics Commission”. In: Available at SSRN 2529151 (2014).

25. David Chalmers. “The singularity: A philosophical analysis”. In: Journal of Consciousness Studies 17.9-10 (2010), pp. 7–65.

26. Wei Chu and Zoubin Ghahramani. “Preference Learning with Gaussian Processes”. In: In Proc. ICML 2005. 2005, pp. 137–144.

27. Robin R Churchill and Geir Ulfstein. “Autonomous institutional arrangements in multilateral environmental agreements: a little-noticed phenomenon in international law”. In: American Journal of International Law (2000), pp. 623–659.

28. Andrew E Clark and Andrew J Oswald. “Unhappiness and unemployment”. In: The Economic Journal (1994), pp. 648–659.

29. Owen Cotton-Barratt and Toby Ord. Strategic considerations about different speeds of AI takeoff. Aug. 2014. url: http://www.fhi.ox.ac.uk/strategic-considerations-about-different- speeds-of-ai-takeoff/.

30. Andr ́e DeHon et al. “Preliminary design of the SAFE platform”. In: Proceedings of the 6th Workshop on Programming Languages and Operating Systems. ACM. 2011, p. 4.

31. Louise A Dennis et al. “Practical Verification of Decision-Making in Agent-Based Autonomous Sys- tems”. In: arXiv preprint arXiv:1310.2431 (2013).

32. Daniel Dewey. “Long-term strategies for ending existential risk from fast takeoff”. In: (Nov. 2014). url: http://www.danieldewey.net/fast-takeoff-strategies.pdf.

33. United Nations Institute for Disarmament Research. The Weaponization of Increasingly Autonomous Technologies: Implications for Security and Arms Control. UNIDIR, 2014.

34. Bonnie Lynn Docherty. Losing Humanity: The Case Against Killer Robots. Human Rights Watch, 2012.

35. Peter Eckersley and Anders Sandberg. “Is Brain Emulation Dangerous?” In: Journal of Artificial General Intelligence 4.3 (2013), pp. 170–194.

36. Beno Eckmann. “Social choice and topology a case of pure and applied mathematics”. In: Expositiones Mathematicae 22.4 (2004), pp. 385–393.

37. Benja Fallenstein and Nate Soares. Vingean Reflection: Reliable Reasoning for Self-Modifying Agents. Tech. rep. Machine Intelligence Research Institute, 2014. url: https://intelligence.org/files/ VingeanReflection.pdf.

38. Kathleen Fisher. “HACMS: high assurance cyber military systems”. In: Proceedings of the 2012 ACM conference on high integrity language technology. ACM. 2012, pp. 51–52.

39. Carl Frey and Michael Osborne. The future of employment: how susceptible are jobs to computerisa- tion? Working Paper. Oxford Martin School, 2013.

40. Edward L Glaeser. "Secular joblessness". In: Secular Stagnation: Facts, Causes and Cures (2014), p. 69.

41. Irving John Good. "Speculations concerning the first ultraintelligent machine". In: Advances in computers 6.31 (1965), p. 88.

42. Katja Grace. Algorithmic Progress in Six Domains. Tech. rep. Machine Intelligence Research Institute, 2013. url: http://intelligence.org/files/AlgorithmicProgress.pdf.

43. Katja Grace and Paul Christiano. Resolutions of mathematical conjectures. 2014. url: http://www.aiimpacts.org/resolutions-of-mathematical-conjectures.

44. [The Tauri Group. Retrospective Analysis of Technology Forecasting: In-scope Extension. Tech. rep. 2012. url: http://www.dtic.mil/get-tr-doc/pdf?AD=ADA568107.

45. Tom Gunter et al. "Sampling for inference in probabilistic models with fast Bayesian quadrature". In: Advances in Neural Information Processing Systems. 2014, pp. 2789–2797.

46. Joseph Y. Halpern and Rafael Pass. "Game Theory with Translucent Players". In: CoRR abs/1308.3778 (2013). url: http://arxiv.org/abs/1308.3778.

47. Joseph Y. Halpern and Rafael Pass. "I Don't Want to Think About it Now: Decision Theory With Costly Computation". In: CoRR abs/1106.2657 (2011). url: http://arxiv.org/abs/1106.2657.

48. Joseph Y Halpern, Rafael Pass, and Lior Seeman. "Decision Theory with Resource-Bounded Agents". In: Topics in cognitive science 6.2 (2014), pp. 245–257.

49. Kristian J Hammond, Timothy M Converse, and Joshua W Grass. "The stabilization of environ- ments". In: Artificial Intelligence 72.1 (1995), pp. 305–327.

50. Robin Hanson. "Economics of the singularity". In: Spectrum, IEEE 45.6 (2008), pp. 45–50.