Safe Learning of Intelligent Agents: The Pros and Cons for Stimulus Algorithms in Autonomous System Design

Ukponaye Desmond Eboigbe
Department of Health and Rehabilitation Sciences
Western University, Ontario, Canada
ueboigbe@uwo.ca

Abstract

Autonomous systems are intelligent systems that are able to interact with their environment and make decisions without human supervision or intervention. Intuitively, in autonomous systems, outcomes are solely achievable by implementation of learning techniques on knowledge extracted from real data, thus "safe learning" is a terminology used to emphasize the need for safety with scalability in the buildup to design and analysis of decision algorithms. Since autonomous systems tend to mimic humans and possibly act as superhumans by making intelligent decisions, it must somehow encounter problems peculiar to *Homo sapiens*, which is ability to make correct decisions at critical moments or show resilience when the chips are down; rely on experience when faced with uncertain conditions or make a guess when all possible known options are exhausted. Thus this study proposes implementation of a stimulus plan otherwise referred to as "Stimulus Algorithms", which is a special consolidated learning mechanism that uses reinforcement learning, adaptive learning, and neural networks to enhance the ability of intelligent agents in making profitable decisions during crisis. This could be likened to special instinct or discerning powers that assist humans in making certain decisions in unfamiliar conditions rather than relying on knowledge and experience alone. In addition to censoring, this study introduces a model for generating data for the stimulus algorithm called the presumptive observation or mean model. Implementing stimulus algorithm might make the programming process more cumbersome, ambiguous and voluminous but this study espouses the significance of creating a stimulus algorithm as a respite for entropy.

1.0 Background

Defining problems and proposing solutions has taken on a new dimension in the digital age. As global civilization advances rapidly toward full digitization, complexity increasingly characterizes modern technological design. Innovations in Artificial Intelligence (AI), Data Analytics, and Machine Learning (ML) have unlocked vast opportunities for designing and implementing computational systems to solve real-world problems. Simultaneously, advances in hardware – such as sensory devices and motor systems – continue to enhance innovation in autonomous systems design. An autonomous system is an independent system capable of collecting data, processing it, making decisions, and reacting to its environment based on acquired knowledge[17]. These systems are self-sufficient and operate without human supervision [22], mimicking what is often considered the defining trait of *Homo sapiens* – intelligence.

The capacity to "think" in both biological and artificial entities is limited by their ability to learn. Among primates, *Homo sapiens* are regarded as the most intelligent due to their larger brain size (cognitive capability), bipedal mobility (efficiency), lighter body mass (portability), smaller teeth (granularity), language use (communication), tool creation (innovation), and adaptability. Similarly, autonomous systems can be evaluated by analogous traits such as algorithmic processing capacity, granularity (agility), portability, scalability, interactivity, transferability, and innovation [1]. The

ultimate aim of developing such systems is to delegate tasks traditionally performed by humans to machines, thereby optimizing resources.

To achieve this, autonomous systems must be capable of learning, retaining knowledge, and executing actions independently (fig. 1) – emphasizing the importance of safe learning in intelligent agents [21]. These agents are algorithmic entities programmed to collect data, process data, analyze databases or sources, update knowledge bases, and gain experience [17] (fig. 2). They use data, learning algorithms, and reasoning to make decisions and improve their performance over time without direct human intervention [22]. They also serve as intermediaries between software and hardware, translating abstract commands into physical actions.

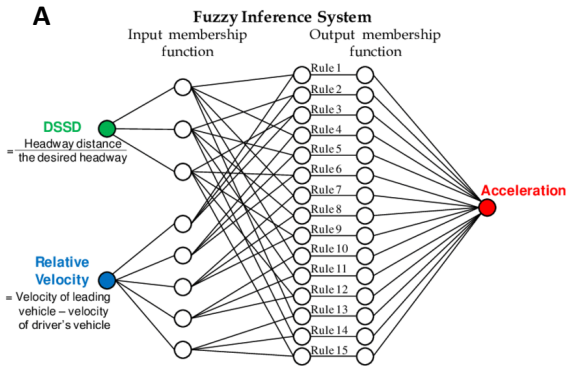

Fig. 1: Simple fuzzy inference system[25]

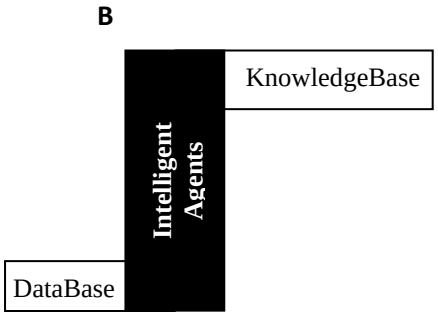

Fig. 2: Black box of the learning agent

***A** is a simple agent implemented in self-driving automobiles [25] – **B** the agent takes sensory data from the proceeding vehicle (environment) uses a rule-based fuzzy inference algorithm (black box neural network) and updates its knowledge about the environment and determine acceleration or deceleration rate to create safe distance in between [25].*

1.1 Safe learning of intelligent agents

The term *safe learning* serves as a critical reminder that must be conscientiously observed when developing learning algorithms, particularly in safety-critical environments. Unlike humans, computers cannot evaluate the consequences of their actions—they execute commands regardless of potential harm or self-destruction [21]. In reality, even humans, despite their natural intelligence, can act harmfully due to factors such as upbringing, peer pressure, misinformation, environmental conditions, or psychological issues. If intelligent human behavior is susceptible to such risks, then artificial intelligence, which lacks innate judgment and experience, may be even more vulnerable.

1.2 Stimulus algorithms

Building a fault-tolerant autonomous system necessitates *safe learning* of intelligent agents by enhancing both software and hardware architectures. For a system to dynamically adapt to unforeseen conditions, its algorithmic building blocks must be scalable to accommodate changes. A self-sufficient system requires multiple sensory organs – such as sensors, LiDAR, or radars – as well as more actuators or effectors [20]. While multiple sensors increase reliability in environmental detection and additional actuators offer greater freedom of movement, they also introduce complexity [21]. On the other hand, more components could raise the likelihood of technical failures and potential conflicts in functionality, often resulting in reduced performance. System failures may stem from hardware malfunctions or software errors – both of which can compromise the integrity of the design [20]. For an intelligent agent to learn and act safely, an intermediary mechanism is essential. In humans, instinct serves as the bridge between knowledge and experience.

Often, in the pursuit of innovation and amid intense industry competition, safety considerations are unintentionally overlooked. With growing demand for speed and increased reliance on smart technologies, decision-making is frequently delegated to machine learning algorithms and AI systems [3]. As a result, these algorithms are rarely scrutinized, and interpretability is often sacrificed for efficiency and usability [3]. Consequently, in autonomous systems, a *stimulus algorithm* is proposed as an intermediary to mitigate hardware or software failures – this is the core focus of the current research. In socioeconomics, a stimulus plan cushions the impact of economic downturns. Likewise, in scientific computing, a well-designed stimulus algorithm can function as a

safety net during failures. Though implementation may be challenging, it offers valuable insight into building safer systems.

Safe learning of safety-critical adaptive controllers demands interdisciplinary knowledge across multiple discipline . Designing such systems requires rigorous risk analysis, as algorithms built on flawed assumptions can be more dangerous than the problems they aim to solve. Hence, further research is essential to improve safety in real-world applications such as smart automobiles, aerospace, biochemical processes, health and medicine, and nuclear energy.

Moreover, machine learning is a relatively young field, still rapidly evolving by creating new problem formalizations inspired by practical needs [5]. The interplay between optimization and machine learning is one of the most significant developments in modern computational science. Optimization techniques are crucial in extracting meaningful insights from large datasets. Importantly, machine learning is not merely a consumer of optimization methods – it is also a source of novel optimization ideas [6]. Therefore, building adaptable, risk-aware autonomous systems requires strict adherence to safety protocols, especially in high-stakes domains.

1.2.1 Entropy

Entropy is the tendency of a system that is left to itself to descend into chaos [14]. It is the baneful aspect that limits artificial intelligence unlike natural intelligence. Nevertheless entropy can be delayed and this study uncovers possible ways to reduce the propensity for an autonomous system to deteriorate into entropy. As technology evolves, the need to revamp ideas and develop more robust intelligent systems with scalable algorithms that address uncertainties especially in safety critical systems/environments become pivotal to achieving dependable and trustworthy results especially in an unprecedented era of digital transformation.

2.0 Literature review

Autonomous systems are defined as those that operate without human intervention – such as self-driving cars—which require the ability to sense their environment, avoid obstacles and hazards [17, 25], perceive and interpret disparate data sources, determine appropriate actions, create plans, and act only when it is safe to do so in dynamic conditions [17]. While many current systems offer semi-autonomy, fully autonomous systems face significant challenges related to safety for widespread adoption [22]. The integration of autonomous robotic systems in hazardous environments – such as nuclear reactor maintenance and wind turbine repair – presents opportunities to perform tasks that are dirty or dangerous [17,18]. However, this also necessitates careful consideration of decision-making transparency, safety concerns, ethical implications, and rigorous verification processes to build public and regulatory trust [22].

Simultaneously, the increasing use of robots and cobots in various workplaces introduces safety hazards, including impacts and crushing [20], which demand thorough risk assessments, adherence to safety standards, and the implementation of control measures such as safeguarding devices, training [19,20], and personal protective equipment to ensure worker safety when interacting with these automated systems [19]. The broader implications of safety and ethics in autonomous systems across all sectors raise fundamental questions about public trust, regulatory frameworks, and the balance between safety, cost, performance, transparency, and innovation. This requires ongoing dialogue among the public, engineers, and policymakers to navigate complex trade-offs and ensure responsible deployment [15,16,21].

A safety-critical system is one where failure could lead to serious consequences such as death, injury, or significant damage [21]. These systems – often computer-based [21] – require stringent reliability measures [15, 22], specialized software engineering practices [22], and thorough testing and verification across various sectors, including infrastructure, medicine, nuclear engineering, and transportation [15,21,22]

2.1 The MCAS Saga

The “double tragedy” of Boeing 737 MAX aircraft was largely due to an automated software tool called the Maneuvering Characteristics Augmentation System (MCAS), designed to automatically adjust flight components to improve balance during specific flight conditions [2]. MCAS was programmed to activate when a single angle-of-attack (AOA) sensor detected a high angle, without cross-checking with the second sensor. This made the system vulnerable to faulty sensor input – a

critical oversight that proved fatal. On October 29, 2018, Lion Air Flight 610 crashed 12 minutes after takeoff from Jakarta, Indonesia, killing all on board. The MCAS had previously masked faulty speed and altitude readings on an earlier flight, leading to confusion and failure to manage the faulty input [2]. Just five months later, on March 10, 2019, Ethiopian Airlines Flight 302 crashed six minutes after takeoff from Addis Ababa under similar circumstances. The MCAS activated based on incorrect sensor data, forcing the aircraft into a nose-dive despite the pilot's efforts to regain control [2]. Investigations revealed a pattern in both crashes, raising serious concerns about the aircraft's safety. Boeing faced global grounding of the 737 MAX, reputational damage, significant financial loss, and costly compensation. The core issue stemmed from poor software design, insufficient fail-safes, and a lack of robust algorithmic thinking in safety-critical systems.

To stay competitive with Airbus, Boeing opted to modify the existing 737NG instead of designing a new aircraft. This decision introduced major design challenges – especially mounting larger, more fuel-efficient engines. Due to the 737's low ground clearance, the engines had to be repositioned, altering the aircraft's aerodynamics and increasing the risk of a nose-up stall [4]. MCAS was introduced as a software solution to counteract this stall risk. However, the system relied on a single AOA sensor, which, when faulty, led to the stabilizer forcing the aircraft's nose downward – mirroring the crash pattern in both incidents. Ironically, MCAS became the source of the very problem it was meant to prevent. This highlights a major risk in autonomous systems: when design decisions lack adequate safety considerations, the system can inadvertently cause catastrophic failures.

2.2 Scope of the stimulus algorithm

For a system to be completely autonomous, it should be able to apply some extra-ordinary measures like instinct and discernment in humans. It should be able to make a reasonable guess and rely on some external influences like inferential learning (learning from other people's experiences, as in humans), preemptive thinking (premonition for disaster, as in humans) as well as comparative thinking (imagination, as in humans), and surrender to higher order (human agents), when all options are exhausted.

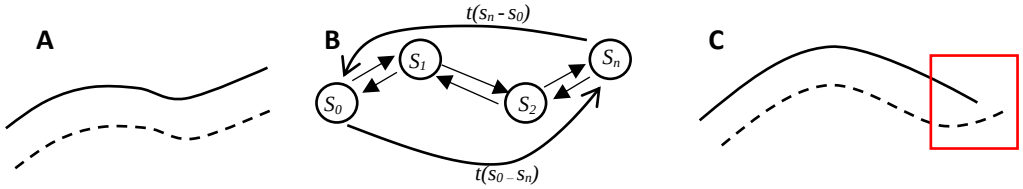

Fig. 3. Simulated route of a system running algorithm and the stimulus algorithm

Whereas the thick smooth line path (———) represent the real system while the broken line path (- - - - -) represents the stimulus algorithm running concurrently. | ***A*** *represent the trajectory of a system running in harmony with the stimulus algorithm, |* **B** *represents a Markov Chain Transition diagram showing predicted pattern and time to move from state S_0 to S_n (given that the system obeys the markov chain rule: which states that "the future state of a process depends only on the current state, not on the sequence of events that preceded it) while |* **C** *represents a process running contrary to the stimulus algorithm at a certain point (crisis zone - highlighted in red box). This could be the point where the error occurs, and the autonomous system at this instance can either rely on the stimulus algorithm to delay entropy or leverage on this extra advantage to assess the workings of the running system rather than a progression in error.*

The concept of a fully autonomous system replicating human-like qualities – such as instinct, discernment, guessing, inferential learning, preemptive thinking, imagination, and deference to human authority – is highly complex and currently beyond the full scope of existing technologies. While autonomous systems can sense, perceive, plan, and act without human intervention, their capabilities remain largely algorithmic and data-driven [17, 22]. For example:

- *Instinct and discernment* are approximated through decision-making algorithms, but lack the innate, experience-independent qualities seen in humans. Their judgments are based on logic and pattern recognition, not intuition.
- *Guessing under uncertainty* is addressed through probabilistic models like Bayesian networks, which allow systems to make decisions based on likelihoods rather than deterministic rules. However, this is a mathematical approach – not true intuition.
- *Inferential learning*, or learning from others' experiences, is not yet a strength of current systems. Machine learning models typically learn from their own data or curated datasets, not vicariously from other agents' experiences in the human sense.

- *Preemptive thinking*, such as anticipating danger, is structured through system monitoring and safety analysis tools like FMECA. While effective, this is not equivalent to the human sense of premonition.
- *Comparative thinking or imagination* is partially implemented through scenario-based training. Systems are exposed to varied situations, but lack the creative and open-ended cognitive flexibility humans use to imagine new outcomes.
- *Deference to humans* is a well-established part of autonomous design. Most systems include fail-safes, disengagement mechanisms, and protocols for human override when limits are reached, reflecting an understanding of their operational boundaries.

In essence, current autonomous systems demonstrate advanced but limited capabilities. They can manage uncertainty, prioritize safety, and learn from data, but they do not yet possess the deeper human-like faculties of intuition, creativity, or social understanding. Future progress may bridge some of these gaps, but replicating the full range of human cognition remains a major challenge. Hence the need to create a boundary for autonomous action referred to as the "safe zone" – implemented using the innovative "presumptive observation model" – outside which the AI agent must sense danger and surrender operation to the human agents as a precursor for crisis avoidance.

2.3 Transition trajectory

For a Markov Chain process, let a state be represented with S while t represent the time to move from states: origin (S_0) to destination (S_n) and the transition pattern can thus be predicted applying data from the present state. The probability of moving from state S_0 to S_n is $P(S_{0\text{-}n})$. And for a sequence of random variables X_1, X_2, X_3, … the Markov property is defined as:

$$P(X_{n+1}= x \mid X_n= x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_{n+1} = x \mid X_n= x_n)$$

Using the Markov Chain transition algorithm, it could be mathematically possible for the stimulus algorithm to predict trajectory pattern of action and preempt timing for certain actions to occur else an error signal will be generated and perhaps return the system to a safe state. With reference to the Boeing 737 MAX "double saga" – if a stimulus algorithm was implemented using Markov chain process for instance, it could perhaps determine that nose-diving eight and twelve minutes after take-off is a red flag and then coerce the MCAS to trigger a steady altitude flight pending further action from the pilot or rather deactivate the autopilot software and signal the pilot to take over the control, or perhaps cutoff the accelerating power to enable more time for safety precautions and maneuver.

3.0 Methodology

Relying solely on data from one source (sensors) could lead to chaos because the system software may not be able to determine when the sensor sends in wrong data. Thus a stimulus algorithm should be built in such a way that it is able to generate its own data by applying some statistical models by censoring the sensor data. By censorship, the stimulus algorithm is able to raise alarm when data input goes out of range and cause the system to reevaluate its database. For instance, if an airlifting plane suddenly begins to follow a downward trajectory even if the sensor is faulty a stimulus algorithm should be able to assess the situation and preempt a fault and deactivate the downward propeller and return the system to a neutral state, by this entropy is delayed until the pilot is able to manage the situation.

3.1 Evaluating trust on passing-on information using information pedigree

It is common knowledge that a vast majority of knowledge and beliefs come from communication which are indirect experiences [7]. In this regard communication is essentially the interaction between intelligent agents but may also include interaction with the environment. As in humans, to build trust there have to be some sort of communication and connection – and for autonomous system to be fully independent, it must have to evaluate communication with other components and the immediate world around to determine trustworthiness. Furthermore, the communicated information itself could come from the sender's indirect experiences as well. In such cases, the sender serves as a mediator who passes on information through from a source agent to a receiver agent [7].

Ironically, the reciprocal trust tendencies among agents is an issue that must be considered ideally. When an agent tend to ignore the importance of passing-on information perhaps due to a fault, it could hinder the receiver agent's ability to diagnose the sources of conflict in the system. This can equally affect the receiver agent's evaluation of other agents, and as a result the quality of their acquired information [7]. In such situations where conflict occurs, it is crucial for an agent to identify the original sources that are responsible for the conflicting information. The need for this is intensified in high-risk situations [7].

==========*For example, in a RoboCup scenario, the perception of a robot is limited, the robot who is far away from the ball has to rely on communication to obtain the estimated position of the ball. Suppose Robot-1 is close enough to perceive the ball position, he sends information about the position of the ball to the closest robot, say Robot-2, and Robot-2 passes this information onto Robot-3. Typically, Robot-3 will consider the received information as Robot-2's perception because the information source (Robot-1) is normally not mentioned. Suppose at the same time (the ball hasn't been moved yet), Robot-3 got a message stating a different position message from Robot-4 who is also close enough, then Robot-3 has to judge which position is more accurate before taking action.[6]* ==========

3.2 Degree of trustworthiness

This refers to how reliably an autonomous system performs its tasks, especially in uncertain or changing environments [20, 24]. It considers how well the system can make safe, consistent, and explainable decisions – key to building user confidence in automation. Autonomous systems often encounter situations where sensor data is noisy, incomplete, or ambiguous [17]. Fuzzy logic can be used to represent and reason with this uncertainty. Instead of crisp binary states (e.g., an obstacle is either present or not), fuzzy variables can represent degrees of presence (e.g., an obstacle is "somewhat present" with a membership value of 0.7) – this aligns with the need for autonomous systems to sense and perceive their environment even with imperfect data [20, 21, 23]

3.2.1 Fuzzy Logic

Fuzzy logic deals with reasoning that is approximate rather than fixed and exact. Unlike traditional binary logic (true/false), fuzzy logic allows for degrees of truth, which is useful in complex or uncertain environments. [23, 25] Fuzzy logic provides a framework for creating rule-based systems where the rules are expressed in a natural language format that can handle vague terms (e.g., "if the obstacle is close and the speed is high, then apply strong braking"). [23].

- *Fuzzy Inference System (FIS):* Uses fuzzy logic to map inputs to outputs; applies a set of *if-then* rules to make decisions based on imprecise or vague data, and common in control systems (e.g., temperature regulation, speed control) [25].
- *Fuzzy Logic in Neural Networks (FLNN):* Combines fuzzy logic with neural networks; helps neural networks handle uncertainty better; and enhances learning performance in complex, noisy, or uncertain data environments.
- *Fuzzy Logic in Reinforcement Learning (Fuzzy Q-learning):* Integrates fuzzy logic with reinforcement learning; uses fuzzy sets to represent states and actions, improving flexibility; and allows the agent to adapt more effectively in dynamic or unpredictable environments.

3.3 Data censorship

Trust is not constant nor inherently assumed; rather, it reflects the degree of alignment between facts or information shared by two or more entities. When intelligent agents interact, the consistency of the information they exchange influences the level of trust. Greater disparity in their data sources can lead to lower trust, making trust a measurable function of the similarity in shared knowledge [24]. However this can also be achieved by censorship, while the running system utilizes real data from sensors, the *Stimulus Algorithm* will utilize censored data. Censorship may be derived using least squared estimation, maximum likelihood estimation or via implementation of fine statistical model as a checkmate for real data. In this study an innovative model (the presumptive observation [φx]) will be used to generate data for the stimulus algorithm.

In general, trust serves to reduce chaos and complexity in systems. Since every system is composed of multiple interconnected parts, components often depend on one another to carry out specific tasks. In such relationships, a component (the "served") must be able to assess the reliability of another (the "server") and identify when it fails to perform correctly. In cases of failure, the "served" should be able to seek alternatives – such as the proposed *Stimulus Algorithm* – to maintain functionality. As agents interact, they adapt and learn from experience. Through generalization, agents can

develop reasoning strategies and make assumptions about other agents or aspects of the environment that they are otherwise unaware of [8]. Therefore, trust evaluation becomes an essential survival skill for agents operating in uncertain or dynamic environments [24].

3.3.1 Generating statistical data for stimulus algorithm: the presumptive observation model

The *presumptive observation* refers to the range of values expected to occur, beyond which the system is presumed to be in a state of crisis. φx_i is the set of variables that indicate the possibility of an observation occurring, with its boundaries serving as a premonition – or early indication – of potentially catastrophic events.

$$\varphi x = 2\left[\log(x+\in) \times \sqrt{\frac{1}{n+\delta}} \times e^{-\alpha(speed)}\right]^2 \times \mu$$

μ is the mean value of observed data
x is the values of recorded observations at various stratas
n is the number of stratas where observations were taken
∈ → ensures logarithm stability (>0)
δ→prevents division by zero at ground state and (0 time point)
(speed) →forces convergences as speed tens to zero

Data boundaries:
X--- sensor data value
(X – φx) --- lower boundary
(X + φx) --- upper boundary
(X+φx) – (X-φx) --- safe zone

Ultimately, the stimulus algorithm will have two boundaries of data to execute, the lower and upper limit, meanwhile the observed data falls in-between. If in the course of operation the observed (sensor) data falls beyond the presumptive data range abruptly, the stimulus algorithm signals error and neutralizes the autonomous system mechanism or trigger manual control system thereby maintaining safety and prolonging entropy. In theory, if this simple analogy as outlined here was implemented in the Boeing 737 MAX, the double tragedy could have been avoided or managed to a certain degree.

3.4 Risk categorization

Every situation in its setup is classified according to the risk factor. The category could range from no-threat to severe threat. In situations where crash is imminent, more emphasis ought to be on algorithmic robustness and error recovery rather than on the architectural framework. The risk category could simply be normal, near crash, crash, or fallback [12]. Trying to keep a hostile foreign government from stealing military secrets is quite a different matter from trying to keep students from inserting a funny message-of-the-day into the system, the amount of effort needed for security and protection clearly depends on who the enemy is thought to be [13]. The amount of risk associated with an autonomous system would determine the level of sophistication of the stimulus algorithm that will be designed to checkmate the system.

4.0 Implementation and result

To implement design, a real 3 hours flight log data was retrieved from the flightaware database (https://www.flightaware.com/live/flight/CSN8032/history/20250321/0740Z/MMMX/MMTJ/tracklog), with chart displaying the altitude and speed of the aircraft as well as the time stamps (strata) at which data was recorded (fig 4). The dataset was exported to MS Excel, stored as xlsx file as well as a comma separated value (csv), and then imported to python editing environment, for exploration and analysis. The presumptive observation model function was used to create lower and upper limit boundaries to replicate the flight trajectory (fig. 5), using data generated from the presumptive observation model. The path represented with blue broken line represents the upper boundary and the red broken line represents the lower boundary, while the green smooth line represents the real

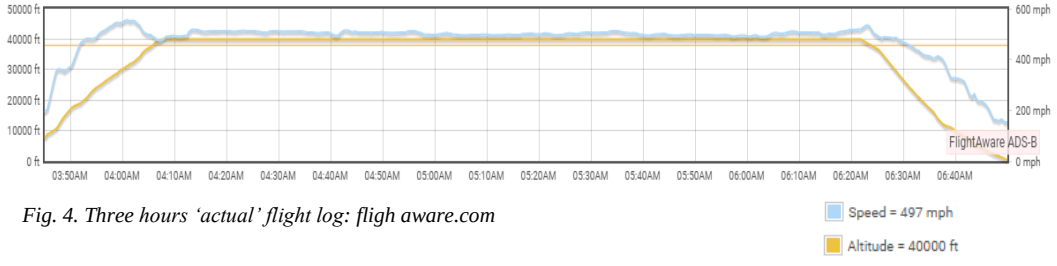

Fig. 4. Three hours 'actual' flight log: fligh aware.com

flight trajectory. The reconstructed flight path (appendix I, fig. 6) shows the safe zone for entropy delay. At every point in the flight, the safe zone is directly proportional to altitude i.e. feet above the ground (appendix I – fig.7). The stimulus algorithm implements the Fuzzy Logic Reinforcement Learning, using (Q-learning algorithm) (appendix II)

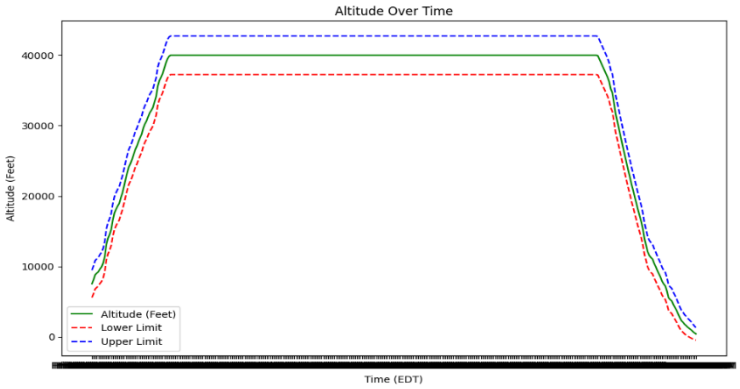

Fig. 5: Replication flight trajectory using presumptive model

After training, the agent's actions are evaluated in a test scenario where it starts at the beginning of the environment (the lower limit of the safety zone: state = -10) and tries to reach the goal state (upper limit of the safety zone: state = 10) – assuming the system abruptly plunges into chaos, and tries to avoid the crisis zone along the way. Thus, the agent delays entropy by taking a series of actions by navigating from one end of the safety zone to the other and intends to reach the goal state safely, allowing ample time for higher-order actions, rather than descending into disaster. The sequence of actions is presented in the output illustration (appendix III). For details on data (flight log), codes: presumptive observation (source_code.py) and stimulus algorithm implementation with Q_learning (q_learning.py): https://github.com/EspeeWorld/BIAI-PROJECT/tree/main

4.1 Discussion, Summary, and Future Direction

There is no doubt that the rapid pace of innovation in scientific endeavors continues to drive the pursuit of effective solutions for a safer world. These advancements span across various sectors, including healthcare, biochemical informatics, agro-economy, socio-economy, biotechnology, sustainable ecosystems, machine dependability, and cleaner energy. This trend has also sparked a growing desire for convenience, leading to increased investments in smart technologies, autonomous systems, and telecommuting.

However, amid the current wave of cyber threats, fully transitioning to a digital landscape presents new uncertainties concerning trust and safety. While there is a deliberate and strategic push toward future technologies – particularly autonomous, Internet-of-Things-powered systems – there is also an inherent challenge: in many real-world applications, data collected are often noisy [17], incomplete, or vulnerable to attacks. This underlines the need for censorship and adaptive optimization processes to enable "safe learning" for intelligent system interface controllers [21].

Similar to humans, who use heuristics or rules of thumb to solve problems and can process partial or uncertain information through approximate reasoning, intelligent agents (or expert systems) are designed to process knowledge symbolically, applying rules to solve specific problems. They, too, can handle imprecise reasoning and evaluate fuzzy data. Ironically, both humans and machines are susceptible to errors, especially when fed incorrect information [9].

Humans, however, are uniquely equipped with suprarational capabilities to assess the consequences of their actions in advance and avoid undesirable outcomes – a capacity that non-human agents lack. Thus, this study focuses on exploring ways to mitigate chaos or delay entropy when mistakes occur within autonomous systems. In light of this, the proposal for **Stimulus Algorithms** in autonomous systems becomes crucial, as trust is fundamental when assigning responsibilities – not just to humans, but even more so to machines.

Conversely, over-reliance on automated algorithms for error control and fault resolution in high impact tasks could potentially diminish human expertise. Experience is often gained by solving problems; if autonomous systems become too perfect, humans may lose critical problem-solving abilities due to limited empirical exposure. Moreover, while machines may perform consistently, they may not be capable of preserving or reproducing knowledge in the nuanced way humans do.

Although autonomous systems are intended to operate with minimal human intervention, it is recommended to retain the option for direct human oversight or intervention when necessary (e.g., through remote control or an emergency stop switch) [11].

Reference

[1] Nesnas, Issa A.D.; Rasmussen, Robert; Day, John, 2022, "Principles for Architecting Autonomous Systems", https://hdl.handle.net/2014/56153, 44th Annual AAS Guidance, Navigation and Control Conference 2022, Breckenridge, Colorado, February 3-9, 2022, JPL Open Repository. https://dataverse.jpl.nasa.gov/dataset.xhtml?persistentId=hdl:2014/56153

[2] C. Palmer, Senior Technology Writer, "The Boeing 737 Max Saga: Automating Failure". Elsevier. ScienceDirect. Engineering 6 (2020) 2-3

[3] L. S. Piano, "Ethical Principles in Machine Learning and Artificial Intelligence: Cases from the Field and Possible Ways Forward". Article of Humanities and Social Sciences Communications, 2020. *https://doi.org/10.1057/s41599-020-0501-9*

[4] J. Herkert, J. Borenstein and K. Miller, "The Boeing 737 MAX: Lessons for Engineering Ethics," Science and Engineering Ethics, 26:2957–2974, 2020. https://doi.org/10.1007/s11948-020-00252-y

[5] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science 349, 255, 2015

[6] S. Sra, S. Nowozin, and S. J. Wright, "Optimization for Machine Learning," MIT Press, Cambridge, MA, 2011,

[7] C. M. John-Jules, and M. Tambe (Eds.), "Intelligent Agents VIII: Agent Theories, Architectures, and Languages," 8th International Workshop, ATAL. Seattle, WA, USA, 2001, pp. 290-305,

[8] S. Chib and E. Greenberg, "Markov Chain Monte Carlo Simulation Methods in Econometrics," Econometric Theory Vol. 12. No.3 Pp 409-431. Cambridge University Press, 1996.

[9] M. Negnevitsky, "Artificial intelligence: A guide to intelligent systems," Second Edition. Addison Wesley. UK, 2005.

[10] W. Lamersdorf, L. Braubach, and A. Pokahr, "A Universal Criteria Catalog for Evaluation of Heterogeneous Agent Development Artifacts," Distributed Systems and Information Systems; Computer Science Department, University of Hamburg, Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany, 2008.

[11] B. Lussier, A. Lampe, R. Chatila, J. Guiochet, F.F. Ingrand, K. Marc-Olivier, D. Powell, "Fault Tolerance in Autonomous Systems: How and How Much?", Conference: 4th IARP-IEEE/RAS-EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, 2005.

[12] K. Czarnecki, "Operational Design Domain for Automated Driving Systems Taxonomy of Basic Terms," Waterloo Intelligent Systems Engineering (WISE) Lab. University of Waterloo, Canada, 2018.

[13] Andrew T. S., Bos H. (2014). Modern Operating Systems. Fourth Edition. Vrije University, Amsterdam, Netherland. Pearson. ISBN – 10:0-13-359162-X, ISBN–13: 978 -0-13-359162-0

[14] Entropy: wikitionary definition – this page was last edited on 18 July 2023, at 05:57. https://en.wiktionary.org/wiki/entropy

[15] Illiashenko, O., Kharchenko, V., Babeshko, I., Fesenko, H., & Di Giandomenico, F. (2023). Security-informed safety analysis of autonomous transport systems considering AI-powered cyberattacks and protection. *Entropy*, *25*(8), 1123. https://doi.org/10.3390/e25081123

[16] Almaskati, D., Pamidimukkala, A., Kermanshachi, S., Rosenberger, J., & Foss, A. (2024). Investigation of the impacts of the deployment of autonomous vehicles on first responders. *Smart and Resilient Transportation*, *6*(2), 150–168. https://doi.org/10.1108/SRT-05-2024-0005

[17] BlackBerry QNX. (2025, April 17). *Ultimate guide to autonomous systems*. https://blackberry.qnx.com/en/ultimate-guides/autonomous-systems

[18] European Commission. (2019, March 6). *Autonomous robots to improve disaster relief*. Research and Innovation. https://projects.research-and-innovation.ec.europa.eu/en/projects/success-stories/all/autonomous-robots-improve-disaster-relief

[19] Canadian Centre for Occupational Health and Safety. (2025, April 17). *Safety hazards: What are robots and cobots?* https://www.ccohs.ca/oshanswers/safety_haz/robots_cobots.html

[20] Ferrell, C. (n.d.). *Failure recognition and fault tolerance of an autonomous robot*. Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[21] Lyons, J. B., Clark, M. A., Wagner, A. R., & Schuelke, M. J. (2017). *Certifiable trust in autonomous systems: Making the intractable tangible*.

[22] National Engineering Policy Centre. (2020, June). *Safety and ethics of autonomous systems: Project overview – Executive summary*. Royal Academy of Engineering. https://nepc.raeng.org.uk/media/nqnhktgq/nepc-safety-and-ethics-of-autonomous-systems.pdf

[23] Zadeh, L. A. (1996). *Fuzzy logic = computing with words*. IEEE Transactions on Fuzzy Systems, 4(2), 103–111. https://doi.org/10.1109/91.493904

[24] Sabater, J., & Sierra, C. (2005). *Review on computational trust and reputation models*. Artificial Intelligence Review, 24(1), 33–60. https://doi.org/10.1007/s10462-004-0041-5

[25] Sato, T., & Akamatsu, M. (2012). Understanding driver car-following behavior using a fuzzy logic car-following model. *Fuzzy Logic-Algorithms, Techniques and Implementations*, 265-282.

Appendix I

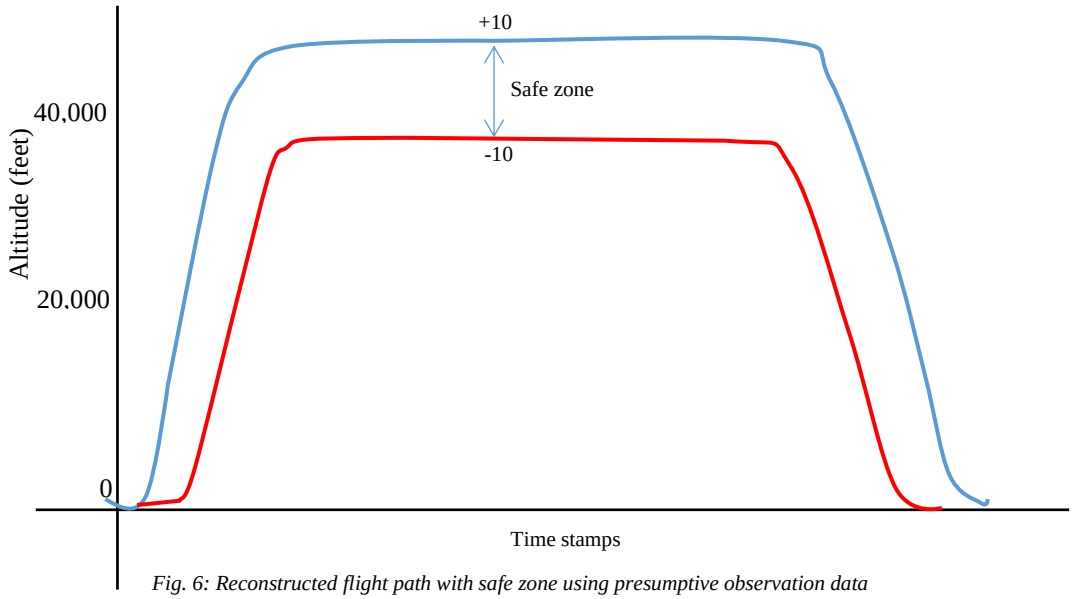

Fig. 6: Reconstructed flight path with safe zone using presumptive observation data

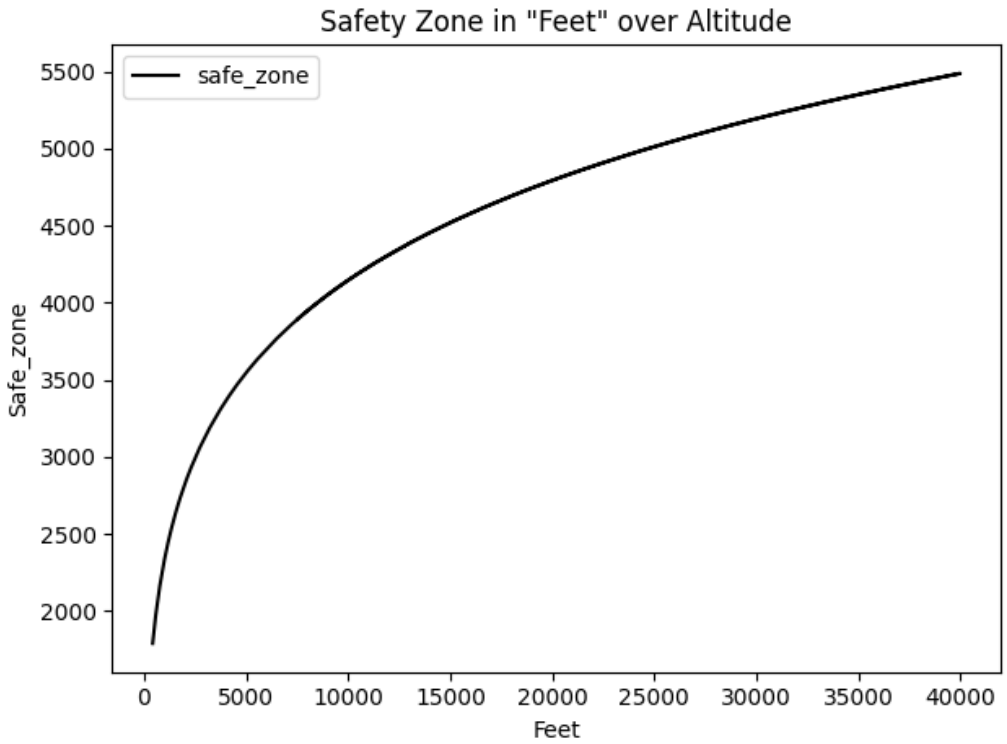

Fig. 7: Safe zone in feet over aircraft altitude.

Appendix II

Learning Algorithm (Q-Learning)

==========

1. State Space: Defined as a 1D array of states from -10 to 10. This represents the agent's position in the environment.
2. Action Space: The agent can take two actions: move left (-1) or right (+1).
3. Q-Table: The Q-table is initialized with zeros. It stores the expected rewards for state-action pairs.
4. Exploration vs. Exploitation: The agent uses an epsilon-greedy policy for balancing exploration (random action) and exploitation (choosing the best-known action).
5. Safety Constraints: During the learning process, the agent is penalized heavily if it moves into the crisis zone (<-8 and >8), simulating a crisis. The agent also gets a bonus if it reaches the safe goal state (GS).
6. Q-Learning Update: The Q-values are updated using the standard Q-learning formula:

$$Q(s, \alpha) = Q(s, \alpha) + \alpha\ r + \gamma \max Q(s, \alpha') - Q(s, \alpha)$$

Where:

- r *is the reward*
- γ *is the discount*
- α *is the learning rate*

==========

Hint

- The entire environment (-10 to 10) is valid.
- The agent starts at -10.
- Penalty is -10 if agent enters or stays in state < -8 or state > 8.
- It avoids lingering in state < -8 or state > 8 because of the -10 penalty.
- It takes action +1 repeatedly to reach the goal at 10.
- Reward is +1 in the "safe" middle zone (-8 to 8).
- Reward is +5 for reaching the goal (state == 10).
- Once learned, the policy optimizes for staying in the safe zone and reaching the goal safely.

Appendix III

```
==========================================

                  Output
==========================================
Agent's Path                    Actions:
State: -10,     ---------       Action: 1
State: -9,       ---------      Action: 1
State: -8,       ---------      Action: 1
State: -7,       ---------      Action: 1
State: -6,       ---------      Action: 1
State: -5,       ---------      Action: 1
State: -4,       ---------      Action: 1
State: -3,       ---------      Action: 1
State: -2,       ---------      Action: 1
State: -1,       ---------      Action: 1
State: 0,        ---------      Action: 1
State: 1,        ---------      Action: 1
State: 2,        ---------      Action: 1
State: 3,        ---------      Action: 1
State: 4,        ---------      Action: 1
State: 5,        ---------      Action: 1
State: 6,        ---------      Action: 1
State: 7,        ---------      Action: 1
State: 8,        ---------      Action: 1
State: 9,        ---------      Action: 1
State: 10,                      Action: Goal Reached
```