Simpler is Better: A Case for Limited ML Models in Safety-Critical Applications

Justin Beaurivage
justin.beaurivage@uqtr.ca
Université du Québec à Trois-Rivières
Trois-Rivières, Canada

Per Lindgren
per.lindgren@ltu.se
Luleå Tekniska Universitet
Luleå, Sweden

Théo Duville
theoduville@gmail.com
Université du Québec à Trois-Rivières
Trois-Rivières, Canada

ABSTRACT

Context-aware decision-making is essential to the effectiveness of safety-critical systems, where the same input may require different actions depending on the situation. Machine learning offers a path to capturing this nuance, enabling real-time responses that go beyond rigid, rule-based logic. While much attention has gone to deep learning, simpler models like random forests remain powerful and better suited to constrained environments such as embedded systems. In this paper, through the practical example of automatic activation devices used in skydiving, we show how deploying a limited random forest model can provide context awareness to a safety-critical system, thus improving its effectiveness at preventing injury. We show that while smaller, simpler models may sacrifice some accuracy compared to their more complex counterparts, their advantages in interpretability and lower hardware demands often outweigh this tradeoff. Importantly, we argue that in safety-critical systems, an algorithm need only perform well enough to provide a meaningful safety benefit. Pursuing maximal performance can lead to disproportionate costs, both in complexity and resources, with little or no corresponding gain in actual safety. More broadly, we advocate for smaller systems with a clear and defined purpose, where their simplicity is a clear advantage from the fact that they are much easier to analyze and verify in the context of their safety applications.

KEYWORDS

resource-limited systems, safety-critical systems, limited machine learning, skydiving, automatic activation device, aad

1 INTRODUCTION

In an era where machine learning (ML) and artificial intelligence is increasingly defined by scale, it is easy to assume that larger models are inherently better. But when deployed in safety-critical systems, where decisions must be made in real time, under strict energy and resource constraints, and where failure can result in loss of life, this assumption breaks down. In these contexts, simplicity is not a limitation, but rather a requirement. The prevailing "more is better" mindset has led to models that are harder to interpret, harder to verify, and often offer diminishing returns beyond a certain complexity. This article argues for a deliberate shift in focus: toward smaller, more efficient models that prioritize explainability, formal verifiability, and energy awareness over raw performance. In doing so, we challenge the assumption that increasing model complexity is always the path to better outcomes, and instead advocate for intelligent sufficiency over unchecked scale. Using the case study of an automatic emergency activation device (AAD) for skydiving, a system designed to deploy a reserve parachute when the skydiver cannot, we demonstrate how a compact and carefully tuned random forest can meet the strict demands of safety, without excess. Importantly, we strive to show that safety can be improved by providing context-aware decision making to existing algorithms, while limiting ourselves to the resources already available on modern AAD hardware. In doing so, we advocate for a broader rethinking of machine learning's role in safety-critical domains and, more generally, in a world increasingly defined by its ecological and computational limits.

LIMITS '25, June 26–27, 2025, Online
2025.

2 BACKGROUND

2.1 Safety-critical systems

A safety-critical system is traditionally defined as a system where failure could lead to consequences deemed unacceptable or catastrophic [10]; examples of unacceptable consequences can include loss of life, significant property damage, or environmental harm. As the world evolves and computers become more and more ubiquitous, we must carefully consider the failure modes of computing systems when they play a safety-critical role. For example, radiation is a major concern for computers in spacecraft; it can cause bit flips in memory, leading to unpredictable behavior. Similarly, in medical devices, software bugs or hardware malfunctions can result in incorrect diagnoses or treatments, endangering patients' lives. Analyses of past incidents have shown that software designs are often unnecessarily complex when it comes to safety-critical systems. Moreover, in [12] the author argues that the context in which software is used is critical to determining the safety qualities of the overall system, and the software cannot be qualified as safe in and of itself. As such, we argue that simpler and more context-aware systems are essential to guaranteeing the safety of operation in safety-critical systems.

2.2 Automatic Activation Devices

The automatic activation device is a safety device used in civilian and military parachuting, and is universally used by the vast majority of skydivers. It dramatically increases the safety of jumpers by automatically deploying their reserve parachute when it detects that an impact with the ground is imminent: for example, when the jumper is unconscious, injured, or otherwise unable or unwilling to deploy their parachute on their own.

Its mode of operations consists of measuring the barometric altitude, and evaluating whether activation is necessary based on

predefined conditions; namely, the device will activate if the skydiver is falling above a preset vertical velocity and below a preset height above ground level. According to Airtec GmbH, a leading AAD manufacturer, AADs have saved at least 5200 lives since their introduction in 1991 [9].

It is obvious that AADs very much fit the definition of a safety-critical device: their malfunction or improper operation may easily result in loss of life. Beyond their utility in sport skydiving, AADs are also used by jumpers who serve as firefighters, search and rescue teams, and military personnel, supporting critical operations that contribute directly to public safety and societal well-being.

AADs must typically be very energy efficient; some models can remain in service for 15 years on a single charge, without requiring maintenance [2]. The amount of computing power, and the sensors available on the device must therefore be quite limited in order to avoid depleting the battery.

Figure 1: An automatic activation device used in skydiving.

2.3 Machine learning in safety-critical applications

In recent years, deep neural networks, large language models, and artificial intelligence have captured much of the public and academic spotlight. While these approaches have shown impressive capabilities, a wide range of other machine learning methods continue to evolve and find application across many domains [14]. In safety-critical systems in particular, machine learning is increasingly valued for its ability to process complex sensor inputs, generalize from limited data, and support real-time decision-making. Unlike traditional rule-based systems, machine learning algorithms can adapt to subtle and high-dimensional patterns in the data, enabling automation in environments that would otherwise be difficult to model explicitly. In some structured classification tasks with safety implications, ML models have shown superior performance compared to hand-crafted logic [8].

However, this promise comes with substantial challenges. Many ML models, particularly complex ones like deep neural networks or large ensembles, are inherently opaque, and often described as "black boxes." This lack of transparency makes it difficult to understand why a specific decision was made, complicating the process of debugging, validation, and certification. In safety-critical contexts, where the consequences of a wrong decision can be severe or irreversible, this opacity is more than an inconvenience: it's often an unacceptable risk.

Moreover, ML models can exhibit non-deterministic behavior. Small changes in input can lead to disproportionately large changes in output, especially near decision boundaries. This sensitivity undermines the predictability required in systems where consistency and reliability are paramount. Compounding this issue is the risk of overfitting: a model that performs well on training data might still fail when deployed in the real world, particularly if it learned superficial correlations rather than robust causal relationships.

One of the most fundamental limitations of ML in safety-critical applications is the difficulty to formally verify the trained models. Traditional safety engineering often relies on methods like model checking or theorem proving, which require clear, well-defined logic. Most ML models, however, are too complex or too opaque to submit to such rigorous analysis. Without formal guarantees, the trust in such systems rests on empirical validation, which may be insufficient for edge cases or rare, high-risk scenarios.

Recently some work has gone into shifting the paradigm from large, general and computationally intensive models towards smaller and more specialized algorithms, making them accessible for deployment on embedded systems and on the edge [5], [3]. Deployment however presents unique challenges: ML models that run well in simulation may behave differently when deployed on embedded hardware. Resource constraints, such as limited memory, processing power, and energy, can affect performance, especially for real-time applications. Sensor noise, quantization effects, and timing jitter further complicate matters, potentially leading to behavior that was never observed during training and/or in simulation.

In light of these challenges, it becomes clear that raw model performance is only part of the equation. For ML to be used responsibly in safety-critical applications, model design must prioritize predictability, simplicity, explainability, and verifiability. In many cases, the most prudent approach is not to push for the most powerful model possible, but rather to ask: What is the smallest, simplest model that performs well enough–and can be trusted?

2.4 Random forests

Random forests (RF) are a type of ensemble learning method used for classification and other prediction tasks. They work by combining the outputs of many individual decision trees, each trained on a random subset of the data and features [6]. For classification, each tree casts a "vote," and the class with the majority vote becomes the model's prediction. This structure makes random forests robust to

overfitting and capable of capturing complex patterns in data.

In practice, each decision tree in the forest is a flowchart-like structure that splits the data based on simple rules, such as whether a sensor reading is above or below a certain threshold. Two key parameters govern the complexity of a random forest: the number of trees in the forest and the size of each tree, typically controlled by limiting the number of nodes or the depth of the tree. Increasing these parameters can improve accuracy up to a point, but also leads to larger models that require more memory, take longer to evaluate, and are harder to interpret; this tradeoff between complexity and performance becomes especially important in resource-limited systems.

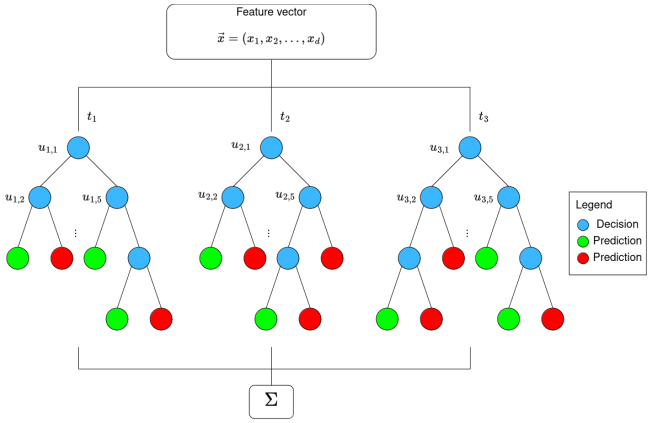

Figure 2: A typical, simplified random forest model.

3 A LIMITED RANDOM FOREST

In this section, we challenge the notion that larger, more complex machine learning models always yield better results through a practical example.

Although AADs have unquestionably enhanced the overall safety of skydiving, they are not without limitations, and in rare cases, can fail or introduce additional risks. The algorithms embedded in modern AADs, while simple and very reliable, still handle some edge cases poorly.

Commanded deployments when the reserve parachute should have remained undeployed can be highly problematic. For example, the reserve parachute may be automatically deployed near the ground during a high-performance landing–a highly complex maneuver, that requires immense skill and precision to execute safely–if the pilot sustains a vertical velocity high enough to trigger the deployment mechanism. This can easily prove fatal by interfering with the jumper's carefully timed sequence of actions. Reserve parachute deployments under a fully functional main parachute may also create entanglement situations that can put the jumper's safety at risk. Conversely, wrongly inhibited reserve parachute deployments can be equally as dangerous. For instance, an out-of-control and unconscious wingsuit pilot may sustain a rate of fall low enough as to not trigger the release mechanism.

These blind spots may be mitigated by devising algorithms that are more aware of the jumper's context. In the first example, an accident could be avoided if the algorithm would detect that the jumper is performing an high-speed maneuver under a fully functional parachute.

Using a machine learning algorithm to detect and classify the jumper's current flight phase can provide the context needed to fill the gaps where the established algorithms cannot perform adequately. We have found that random forests, an arguably extremely simple and lightweight ML method, can perform exceedingly well in predicting a jumper's flight phase. Using the predicted flight phase, the AAD can make more informed decisions, by potentially inhibiting a deployment if it detects that the jumper is in a flight phase where this would cause them harm. This configuration allows for the phase classification model to complement the existing AAD deployment logic, instead of replacing it completely. More advanced algorithms could use the model to tune its deployment criteria based on the detected flight phase, although this is beyond the scope of this study.

3.1 A simple, yet effective model

Using data collected from wearable trackers during 22 freefall skydiving jumps and the `randomForest` R package [7], we train a random forest model to predict the flight phase of a skydiver. We split the jumps into 8 distinct classes, as shown in Table 1. Each recorded data point contains 7 features:

- Horizontal speed: v_h
- Vertical speed: v_z
- Height above ground: h_{AGL}
- 3D components of the acceleration vector: a_x, a_y, a_z
- Glide Ratio: $\frac{L}{D}$

This model will serve as a starting point to show how we can reduce its complexity, while remaining sufficiently accurate to improve an AAD's effectiveness. Note that each jump does not necessarily go through all classes, or in the same order. Figure 3 illustrates how the model predicts the flight phases for 10 distinct skydives, while Table 2 shows the general parameters that define the model.

The **base** model is relatively large and complex, but captures a wide range of behaviors with excellent predictive performance. From this starting point, we will systematically reduce the model's size and complexity, aiming to find the simplest version that is still effective at providing context with the goal of improving safety, without significantly impacting resource consumption.

3.2 Diminishing returns

The most obvious method to reduce the model complexity is to simply reduce the forest's width (number of trees) and depth (number of nodes per tree). In Figure 4, we can clearly see that for the problem at hand, a forest with more than 50 trees offers no performance advantage compared to a smaller forest. By reducing both

Table 1: Flight phases

Class	Description
Airplane	Jumper is riding the aircraft to exit altitude
Freefall	Jumper is in freefall
Wingsuit	Jumper is gliding with a wingsuit
Opening	The parachute is opening and the jumper is decelerating
Canopy	Flight under a deployed canopy
Swoop	Jumper is performing a high-speed landing
Recovery arc	Jumper is transitioning from vertical to horizontal flight during a high-speed landing
Landing	Jumper is touching down on the ground

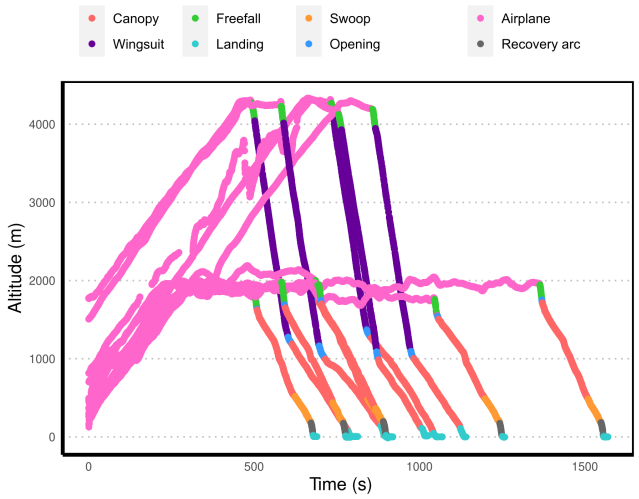

Figure 3: Flight phase classifications for 10 skydives using a random forest model.

Table 2: Base model

Model: **base**	
Number of trees	300
Number of nodes per tree	2049
Features	$v_h, v_z, h_{AGL}, a_x, a_y, a_z, \frac{L}{D}$
Overall accuracy	99.84%

the number of trees and the number of nodes per tree, the **trimmed** model (Table 3) has its complexity reduced by 91.9%, with a small loss in overall accuracy of 0.13% when compared to the **base** model.

Table 3: Trimmed model

Model: **trimmed**	
Number of trees	50
Number of nodes per tree	999
Features	$v_h, v_z, h_{AGL}, a_x, a_y, a_z, \frac{L}{D}$
Overall accuracy	99.71%

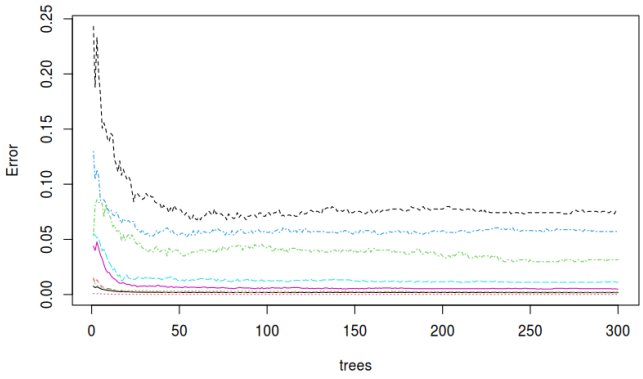

Figure 4: Prediction error per class, in function of the number of trees in the forest.

While the **trimmed** model significantly reduces the forest complexity, it requires parameters that can't be measured with current AAD hardware. In particular, the horizontal speed and glide ratio are typically measured with a GNSS receiver, which is relatively power-hungry in a context where ultra-low power consumption is a priority. We are therefore interested in stripping the model down to the bare minimum where it is still effective in aiding the AAD in making context-aware decisions, while avoiding to induce additional hardware requirements, both in terms of computing power and power consumption. Including a GNSS receiver would also induce additional cost, as they are typically relatively costly. Furthermore, including a GNSS receiver would add a variety of failure modes that each should imperatively be handled gracefully in order to guarantee safety, ranging from loss of signal, to noisy data, and even malicious actors deliberately spoofing the received coordinates [15].

3.3 Just enough is good enough

In this next experiment we train a random forest model that uses sensor readings that are already available current AAD hardware, or that could be easily integrated while maintaining a high power efficiency (ie, a MEMS accelerometer). The **minimal** model (Table 4) is 97.6% less complex than the **base** model, and 70% less complex than the **trimmed** model. However, this model shows a noticeable drop in raw overall accuracy. Notably, the model becomes less accurate at detecting a clear transition between adjacent flight phases.

This loss in accuracy can be mitigated by recombining the classes that are functionnally similar–that is, by detecting not *what* is happening, but by informing the AAD *how* it should respond if its deployment criteria are met. For instance, the ADD response should be identical whether the jumper is in the Canopy or the Opening phase. Table 5 shows the effectiveness of each model at improving the AAD's function.

While the **minimal** model does not achieve perfect prediction accuracy, it is still *sufficiently effective* at providing context to the AAD, thus improving its overall safety value by a significant margin.

Table 4: Minimal model

Model: **minimal**	
Number of trees	30
Number of nodes per tree	499
Features	v_z, h_{AGL}, a_x, a_y, a_z
Overall accuracy	99.10%

Table 5: Effectiveness of the models.

Class	Accuracy (%)	False positives (%)	False negatives (%)
Model: **base**			
Airplane	99.98	0.02	0
Freefall	96.73	0	2.29
Wingsuit	99.76	0	0
Opening	99.06	0	0
Canopy	99.57	0	0.06
Swoop	99.60	0.27	0
Recovery Arc	98.71	0	0
Landing	96.98	1.51	0
Model: **trimmed**			
Airplane	99.98	0.02	0
Freefall	97.05	0	2.29
Wingsuit	99.70	0	0
Canopy	98.97	0	0.26
Swoop	99.46	0.27	0
Recovery Arc	98.08	0	0
Landing	97.38	1.31	0
Model: **minimal**			
Airplane	99.95	0.05	0
Freefall	95.25	0	2.71
Wingsuit	99.40	0	0
Opening+Canopy	98.86	0	0.93
Recovery Arc+ Swoop+Landing	94.30	5.70	0

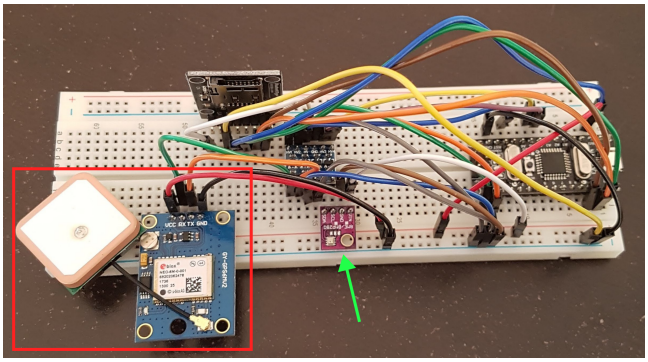

Figure 5: A prototype data acquisition setup, showing a MEMS barometer (in green), and a GNSS receiver (in red), which could be eliminated. Note that the MEMS accelerometer is missing in this image.

For example, using this model would have inhibited 94.3% of false deployments during a high performance landing, and 99.95% of false deployments inside an aircraft.

3.4 A note on deployments

In this study, we have focused on simulating our models. While we haven't implemented them on hardware, we are still interested in estimating the hardware resources required to deploy them. Some previous work has gone into reducing the amount of storage [1], memory [4], and latency [11] of random forests deployed on embedded hardware. Using the method presented in [4], we estimate the storage and memory that would be required to deploy each model on the ATSAMD21 [13], a relatively small, low-power and ubiquitous microcontroller boasting a single ARM® Cortex-M0+ core. Table 6 shows the hardware resources necessary to deploy the models; only **minimal** would fit on the onboard flash memory, which has a maximum capacity of 256 KiB. Attempting to deploy the other two models would require more powerful devices, driving up both the cost and power consumption of the system. We also notice that the memory requirements are quite negligible in each case; using the method in [4] also does not require the use of floating point arithmetics, further driving down the computational power requirements[1].

Table 6: Hardware resources needed to deploy the models on an ATSAMD21 microcontroller.

Model	**base**	**trimmed**	**minimal**
RAM usage (bytes)	128	128	104
Storage (KiB)	7203.5	585.4	175.4
Fits in flash memory?	No	No	Yes

3.5 Limitations and further work

Despite the benefits, the addition of a context-aware model introduces a critical new concern: the potential for false negatives, where valid emergency deployments are mistakenly inhibited. These cases, though rare, are far more severe in consequence than false positives. A missed deployment in a true emergency scenario represents a catastrophic failure, potentially resulting in fatal injury.

In our tests, the model misclassifies 2.71% of the data points belonging to the Freefall class, and 0.93% of the points belonging to the Opening+Canopy class–both of which should trigger an emergency deployment if the AAD's conditions are met–mistakenly labeling them as a phase where deployment should be inhibited. However, because the model is intentionally kept simple and is highly explainable, we can clearly identify the cause of these misclassifications: all of them occur in conditions at which the AAD would not have initiated a deployment. As a result, in a real-world scenario, these cases would not lead to an inhibited activation, and the safety-critical function of the AAD would remain unaffected. That being said, none of our training data contains situations where

[1]The random forest algorithm uses floating point arithmetics during the training phase, but at run-time the classification can be implemented using exclusively integer arithmetics.

the AAD would have activated. This is of course a limitation of our study; the recording of data during intentional AAD deployments would impart obvious ethical concerns, and they rarely occur naturally in the field.

Subsequent efforts can also be carried out to improve the **minimal** model's accuracy without increasing its complexity. For instance, the model has no concept of time dependence; many of the classification errors come from misclassifying a single point within a long stream of correct predictions. The addition of a simple post-processing filter could help curb those errors without requiring additional computing power.

Ultimately, this study presents a proof-of-concept and insights into how intelligent and context-aware software can help improve safety. However, proper risk assessment and software verification studies should absolutely be conducted before considering using our proof-of-concept with real-world systems.

4 CONCLUSION

In its essence this work highlights a fundamental limitation of rule-based approaches in safety-critical systems: their difficulty in capturing context. Although traditional algorithms can handle well-defined, static decision rules, they often struggle or outright fail when decisions depend on nuanced, situational factors. By introducing compact, context-aware models like a tuned random forest, we show that it's possible to bridge this gap with minimal complexity and maximal relevance to real-world conditions.

Context awareness is essential in any safety-critical system where correct action depends not just on sensor inputs, but on understanding when and why those inputs matter. Many such systems operate in dynamic, uncertain environments where rigid rules or purely model-based logic fall short. The same signal can demand opposite responses depending on timing, surroundings, or intent—subtleties that static decision boundaries often cannot capture. Incorporating even minimal context through lightweight learning models enables systems to distinguish between superficially similar situations with very different safety implications. This shift from reaction to interpretation marks an interesting step toward more resilient, human-aligned automation.

In skydiving as well as other safety-critical domains, the goal is not to maximize raw predictive performance but to achieve sufficient accuracy within strict operational constraints. Systems must be reliable, interpretable, and resource-efficient, especially when deployed on edge devices with limited compute, memory, and power. By focusing on compact, context-aware models that do just enough to support safe decision-making, we move toward solutions that are not only effective but also practical and trustworthy in the environments where they matter most.

This work shows that achieving context-aware decision-making doesn't require more resources—just smarter use of the ones we already have. By trimming away excess complexity and focusing on what truly matters, we can design systems that are both efficient and capable. In safety-critical applications, intelligent use of resources isn't just a technical choice—it's a path to more reliable, deployable, and responsible technology.

ACKNOWLEDGMENTS

We would like to thank all those who participated in the skydiving data collection campaign. We also thank Cédric Léveillé and Orig'Amis Rigging for their permission to use their photograph of a Cypres AAD.

REFERENCES

[1] A. Painsky and S, Rosset. 2016. Compressing Random Forests. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. ISSN: 2374-8486. (Dec. 2016), 1131–1136. doi:10.1109/ICDM.2016.0148.

[2] MarS a.s. 2019. FAQ. (2019). Retrieved Apr. 30, 2025 from https://www.m2aad.com/faq/.

[3] Youssef Abadade, Anas Temouden, Hatim Bamoumen, Nabil Benamar, Yousra Chtouki, and Abdelhakim Senhaji Hafid. 2023. A Comprehensive Survey on TinyML. *IEEE Access*, 11, 96892–96922. doi:10.1109/ACCESS.2023.3294111.

[4] Justin Beaurivage, Messaoud Ahmed Ouameur, and Frédéric Domingue. 2025. A Memory Representation of Random Forests Optimized for Resource-Limited Embedded Devices. (Apr. 2025). https://files.beaurivage.io/forest_compression_v2.0-rc2%2Bpreprint.pdf.

[5] Sérgio Branco, André G. Ferreira, and Jorge Cabral. 2019. Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey. en. *Electronics*, 8, 11, (Nov. 2019), 1289. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute. doi:10.3390/electronics8111289.

[6] Leo Breiman. 2001. Random Forests. en. *Machine Learning*, 45, 1, (Oct. 2001), 5–32. doi:10.1023/A:1010933404324.

[7] Leo Breiman, Adele Cutler, Andy Liaw, and Matthew Wiener. 2022. Breiman and Cutlers Random Forests for Classification and Regression. en. (Jan. 2022). Retrieved Dec. 13, 2024 from https://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

[8] Jochen L. Cremer, Ioannis Konstantelos, and Goran Strbac. 2019. From Optimization-Based Machine Learning to Interpretable Security Rules for Operation. *IEEE Transactions on Power Systems*, 34, 5, (Sept. 2019), 3826–3836. doi:10.1109/TPWRS.2019.2911598.

[9] Airtec GmbH. 2023. CYPRES Saves List. (Sept. 2023). Retrieved Apr. 25, 2025 from https://www.cypres.aero/documents/cypres-save-list/.

[10] John C. Knight. 2002. Safety critical systems: challenges and directions. In *Proceedings of the 24th International Conference on Software Engineering* (ICSE '02). Association for Computing Machinery, New York, NY, USA, (May 2002), 547–550. ISBN: 978-1-58113-472-8. doi:10.1145/581339.581406.

[11] Fabian Küppers, Jonas Albers, and Anselm Haselhoff. 2019. Random Forest on an Embedded Device for Real-time Machine State Classification. In *2019 27th European Signal Processing Conference (EUSIPCO)*. ISSN: 2076-1465. (Sept. 2019), 1–5. doi:10.23919/EUSIPCO.2019.8902993.

[12] Nancy G. Leveson. 2017. The Therac-25: 30 Years Later. *Computer*, 50, 11, (Nov. 2017), 8–11. doi:10.1109/MC.2017.4041349.

[13] Microchip. 2021. SAM D21 Family Data Sheet. en. (2021). Retrieved Dec. 13, 2024 from https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21DA1-Family-Data-Sheet-DS40001882G.pdf.

[14] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. 2016. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. (Mar. 2016), 1310–1315. Retrieved Apr. 30, 2025 from https://ieeexplore.ieee.org/abstract/document/7724478.

[15] Kexiong Zeng, Sreeraksha Kondaji Ramesh, and Yaling Yang. 2014. Location spoofing attack and its countermeasures in database-driven cognitive radio networks. In *2014 IEEE Conference on Communications and Network Security*. (Oct. 2014), 202–210. doi:10.1109/CNS.2014.6997487.

Received 30 April 2025; revised dd month yyyy; accepted dd month yyyy