Challenges in Using Neural Networks in Safety-Critical Applications

Håkan Forsberg
School of Innovation, Design and Engineering
Division of Intelligent Future Technologies
Mälardalen University
721 23 Västerås, Sweden
Email: hakan.forsberg@mdh.se

Joakim Lindén
Gripen C/D, Saab Aeronautics, Järfälla, Sweden
Email: joakim.linden@saabgroup.com

Johan Hjorth
School of Innovation, Design and Engineering
Division of Intelligent Future Technologies
Mälardalen University
721 23 Västerås, Sweden
Email: johan.hjorth@mdh.se

Torbjörn Månefjord
Avionics Systems, Saab, Huskvarna, Sweden
Email: torbjorn.manefjord@saabgroup.com

Masoud Daneshtalab
School of Innovation, Design and Engineering
Division of Intelligent Future Technologies
Mälardalen University, 721 23 Västerås, Sweden
Email: masoud.daneshtalab@mdh.se

Abstract—**In this paper, we discuss challenges when using neural networks (NNs) in safety-critical applications. We address the challenges one by one, with aviation safety in mind. We then introduce a possible implementation to overcome the challenges. Only a small portion of the solution has been implemented physically and much work is considered as future work. Our current understanding is that a real implementation in a safety-critical system would be extremely difficult. Firstly, to design the intended function of the NN, and secondly, designing monitors needed to achieve a deterministic and fail-safe behavior of the system. We conclude that only the most valuable implementations of NNs should be considered as meaningful to implement in safety-critical systems.**

Keywords—avionics, safety-critical, machine learning, deep neural networks

I. Introduction

Machine learning (ML) in dependable systems are genuinely researched nowadays. ML aims at achieving artificial intelligence (AI) through learning from data. AI, in turn, is the theory and development of systems which are able to execute tasks normally requiring human intelligence [1]. Deep learning (DL) is a subset of ML that uses deep neural networks (DNNs) to bring the learning capability closer to the function of the human brain [2].

In autonomous ground-based vehicles, DNNs have been investigated for a long time to support object detection and classification. Mainly with focus on reliability and security. Reliability includes research on correct training and validation of input data, physical errors (mostly transient faults) in the deep neural network itself and data input distortion such as adverse weather conditions, i.e. snow, heavy rain or fog. Security includes adversarial attacks (human made attack on input data to fool the neural network as much as possible). Adversarial attacks are of specific interest for avionics in certain applications, e.g. autonomous landing. Both the reliability and security research concentrate on maximizing the correct classification of images (true positives) in presence of disturbances.

This work is partially supported by Vinnova within the project SafeDeep: Dependable Deep Learning for safety-Critical Airborne Embedded Systems and partially by the Swedish Knowledge Foundation within the project Dependable Platforms for Autonomous systems and Control.

For aviation, there has been a tremendous increase in interest of using AI in dependable systems the last years. In 2019, both SAE and EUROCAE, created standardization committees and working groups to prepare for the use of AI in aviation [3, 4]. The Aerospace Vehicles Systems Institute (AVSI) and their working group AFE-87 has been working on the topic of machine learning the last years. As a result from this research [5], they are currently setting up another working group, "Machine Learning Certification" [6]. In addition, the DEEL Project works with dependable, certifiable and explainable artificial intelligence for critical systems [7]. This project has participants from both academic and industry from several disciplines including aviation. The certification authorities also seriously focus on the issue of enabling AI in aviation. The European Aviation Safety Agency (EASA) released an *Artificial intelligence roadmap* [2] recently and their AI Task Force together with Daedalean AG, published a report in the field of concepts of design assurance for neural networks [1].

Reliability and security are both prerequisites for the use of deep neural networks in safety-critical systems. However, several other issues have to be dealt with for safety, including systematic faults, which include design faults in both hardware and software. Design faults are among other means dealt with using deterministic hardware and software and structured requirements driven development process, containing black box requirements written without knowledge of the internal structure, white box requirements (derived from the internal structure) and verification. Static neural networks which are not trained during operation, are indeed deterministic in the sense that the same input gives the same output. The problem is however that in practice the full input space cannot be used when training the neural network [8]. Also, signal distortions due to limited precision or weather disturbances (in case of image processing) will always be present. Therefore, only a statistical description of the neural network's function can be attained. In addition, a structured development process for developing a system containing neural networks must differ from that of the traditional case, since the white box requirements obviously need another approach.

When using DNNs for object classification in safety-critical applications it is important to correctly classify the objects. However, it is of equal importance to minimize incorrectly classified objects. For instance, it is of high significance not to mistake a highway for a runway when an aircraft performs a machine vision-guided approach. Once the aircraft is approaching the runway it is of little importance to correctly classify a car or a fire truck on the landing strip. It is more important to detect an obstacle (foreign object debris) even if it is incorrectly classified rather than not detecting it at all.

We believe diverse redundant systems are needed to cope with the scenarios above. These systems may or may not include the time domain (i.e. history of classified objects and moving targets) and may consist of deterministic or statistical monitors. Additional redundant architectures may be necessary for symmetric faults. For each of the challenges described, we discuss possible solutions including new or existing fault tolerant architectures. The main emphasis is on algorithm level, where deep neural networks are used in the decision loop and monitored by diverse redundant architectures, see Figure 1.

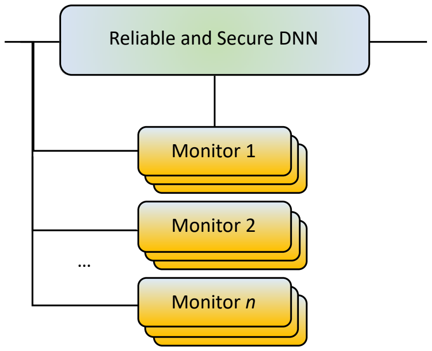

Figure 1. A reliable and secure DNN with diverse monitors, each specified for a task such as detecting transients in the DNN, untrained input data, reduced false negatives or false positives

In a broader scope, artificial intelligence may be used in the behavioral level (e.g. for planning and decisions), as add-on to monitor erroneous behavior in traditional deterministic systems, or even to detect anomalies in the development process for safety-critical products. These areas are subject to further research and not covered in this article.

The paper is organized as follows: In Section II, we present challenges in using neural networks in safety-critical systems. In Section III, we exemplify a possible implementation to cope with the challenges and in Section IV we discuss the challenges and the proposed implementation. Finally, Section V concludes the paper.

II. Challenges

In this section, we introduce challenges in using neural networks in safety-critical applications. Some of these challenges have arisen from previous research performed at Saab or MDH. In some cases, the challenges have emerged from other researchers. But rather than repeating their challenges, we expand or complement their ideas.

A. Relevant Operating Parameters

The representativeness of the dataset used for training a neural network (NN) for a specific function is perhaps the most essential objective of the learning process. It includes steps such as identifying which variables affect the data and the conditions and boundaries on those variables within which the system is intended to operate. In the case of visual perception avionics systems this could include weather conditions, geography, time of day, flight scenario, type of sensors used, and their properties related to perception.

1) Data Management

Data is usually split into three parts named according to their use – training, validation and test. The first two can be treated similarly from an assurance point of view and are used in the design phase of the NN. The test set, however, needs some special tending to; it needs to be prepared with care and independently from the other datasets, since its intended use is to verify the NN design.

The challenges with dataset creation are manifold, especially in the context of supervised learning, where data labels are required, and the annotation process is cumbersome and error prone. The process of acquiring the data itself often entails expensive flight tests. Therefore, the need for synthetically generated data is apparent. Synthetic data is also what enables the testing of specific corner cases identified which might not be reachable (in a safe manner) during normal or sub-normal operating conditions. Gaidon, Wang, Cabon and Vig [9] report that training on entirely synthetic data and testing on real data reduces accuracy compared to training exclusively on real data, which in turn performs worse than training on synthetic data first and then fine-tuning with real data training afterwards. Hence synthetic data is not only a necessity but also desired.

In creating the dataset, the traditional process of requirements-based engineering can be used. Before establishing the dataset, both real and synthetic, a holistic view of which properties the dataset shall have, in order for the data to be as complete as possible, must be specified. The use cases and the safety cases can be addressed in a dataset requirement specification. Special considerations for security and adversarial attacks, may also be added.

2) Weather Impact

Although the Use Case and ConOps perception system description in [1] is a commendable effort to lay the land for the introduction of NN in avionics systems, it is important to consider the impact adverse weather would have on such a system. Designing a perception system which should function in reduced visibility weather conditions will most probably require sensors of different modalities (e.g. Electro-Optical (EO), Short-Wave Infra-Red (SWIR), Long-Wave Infra-Red (LWIR) or other) since neither sensor modality alone will be able to penetrate weather like snow, fog and heavy rain.

3) Data Synthesis

Generating synthesized visual data in different scenic settings is a challenging task. Although simulator tools specializing in photorealistic rendering has emerged over the last couple of years, primarily to support the development of automotive perception systems, the tools mostly focus on emulating EO sensors. Hence there is a need for physically correct rendered sensor data for several types of weather for several sensor modalities. Some work in this direction [10] has been done by modeling transmission of electromagnetic radiation through atmospheric conditions using the MODTRAN (MODerate resolution atmospheric TRANsmission, [11]) software, although limited to IR wavelengths.

Using a framework such as Scenic scenario description language [12] coupled with a sufficiently realistic rendering backend could be one way of handling the generation of synthetic images, where parameters can programmatically be varied within the operating limits of the system.

B. *Uncertainties in classifying correct images*

A major problem with deep learning methods is the incapability of guaranteeing that the predicted outcome is correct. Both false and missed predictions may occur. Phan, Khan, Salay and Czarnecki [13] as well as Levin and Vidimlic [14] propose the use of Bayesian Neural Networks to mitigate the effects of such predictions.

Uncertainty may be categorized into two types, Aleatoric and Epistemic [1]. Aleatoric uncertainty is data dependent. Noise in the data is captured by the model which results in the ambiguity of training input. Epistemic uncertainty is model dependent and caused by the Neural Network's inability to interpret the input. It may be a result of incompleteness of training data [15].

Bayesian Neural Networks (BNN) implement probability theory and approximates the uncertainty of the network output. Thus, providing additional information to interpret objects in the operational environment correctly [14].

An example where a BNN would be beneficial to implement is when the network is presented with data that differs from training data. The network itself may be confident of the output, even though no possibility of correctly assessing the input is possible due to the lack of correct training data. By adding a new parameter, Network uncertainty, more knowledge may be gained before reaching a conclusion based on Network output.

The concept of uncertainty in predictions is also discussed by Cluzeau et al. [1] for design assurance of neural networks. They discuss two types of uncertainty estimations, multiple neural network implementations, where the networks should agree on the output, and Monte Carlo dropout. Monte Carlo dropout refers to a Bayesian model, where a probability distribution is acquired during the Network testing. The variance and mean may then be obtained from the probability distribution and used as metrics of confidence in the output.

Henne, Schwaiger, Roscher and Weiss [16] evaluate different methods for uncertainty estimations from a safety-critical perspective. In their paper, Monte Carlo dropout is compared to three other uncertainty estimators. Further, the Softmax output of the tested networks is used as a baseline, together with the uncertainty estimations. The Softmax function takes as input a vector y of R real numbers. Then it normalizes the vector y into a probability distribution consisting of R probabilities which are proportional to the exponentials of the input numbers [17].

The performance of the networks is quantified with network certainty and whether the prediction is correct, resulting in four possible outcomes. The most critical outcome in a safety-critical point of view is where the model is incorrect in its prediction, yet confident that it is correct. The model's performance is determined by comparing the number of certain and correct samples with the number of certain and incorrect samples. The benchmark evaluation by Henne et al. [16] show that all tested uncertainty estimations perform better when compared to Softmax predictions. When applying the developed benchmark, it is concluded that no one method can be said to have the best performance. Further, the benchmark shows that sampling free methods are more cautious, resulting in higher rejection rates of false predictions. However, the high rejection rates also result in predictions that are correct, becoming uncertain. Sampling-based methods also show promising results in the benchmark tests, but with certainty values higher when compared to sampling free methods.

C. *Hardware and software certification considerations*

When it comes to hardware and software certification of safety-critical systems using neural networks (NNs), traditional design assurance methods may only be used for some parts of the system. The software involved at algorithm level in the NN part must use another approach, mainly because of the design is dependent on the data set used for training the network. Also, the full input space can typically not be used in the training process due to the nature of the problems to solve, e.g. images in the size of 512 x 512 pixels, each with 2^24 possibilities of colors (assuming 8 bits per RGB color). The output results can also only be statistically confirmed. Cluzeau et al. [1] have addressed the concern with data dependent design by introducing the W-shaped development cycle for learning assurance as an extension to traditional design assurance. In their development cycle, data used for training, validation and test, become part of the development cycle and will be assured in the same way as the rest of the design, to make sure systematical faults have been reduced to an acceptable level. Cluzeau et al. [1] assume no new hardware to implement the neural networks and suggest therefore traditional hardware design assurance methods to be used for the underlying electronics. In addition, they assume a systems architecture which does not change dynamically under normal operation. Adaptive system architectures using machine learning are much more complex in its nature and is a challenge of its own for use in safety-critical systems. Yet, adaptive systems are common for similar systems in domains other than aviation.

Since deep neural networks (DNNs) are heavily compute intensive not only for training but for normal operation, it will be a challenge to use traditional hardware as the underlying electronics in the final system, including established commercial-off-the-shelf (COTS) components. In fact, even if these COTS components are well-established, they will be used differently when used in a DNN such that service experience assurance arguments may be of less significance. We believe new emerging COTS-based computing-platforms such as heterogeneous computing platforms exploiting massive parallelism, will be used for real implementations of DNN-based systems.

In previous research [18], we have suggested the use of Overarching Properties (Intent, Correctness and Acceptability) together with assurance cases to argument that assurance objectives can be met for future computing platforms based on new COTS technology. The informal meaning of the three overarching properties are [19]: 1) *What the product is supposed to do is properly captured*, 2) *the product does what it is supposed to do*, and 3) *the product does not cause harm*. The last one relies on that no development decisions do compromise the original safety assessment. Figure 2 shows a graphical example of an assurance case. For COTS we introduced sub-claims on two levels, at the isolated COTS component level and at the COTS integrated component level. The top-level claim is *COTS component operates demonstrably airworthy in its system context*. This claim is further divided into initial and continuous airworthiness [18]. Assurance cases may be used for both software and hardware design assurance [20].

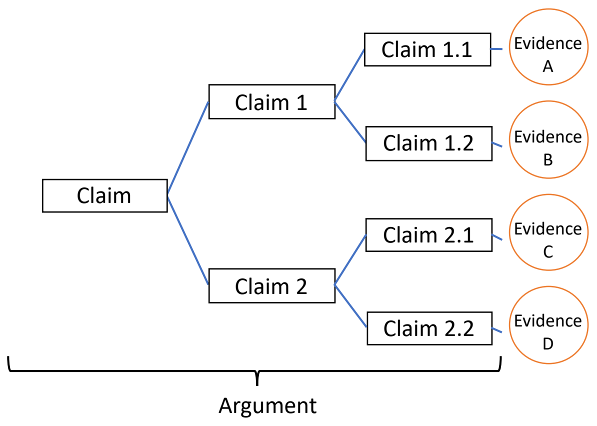

Figure 2. Graphical presentation of an assurance case. The top-level claim (leftmost) is decomposed until each sub-claim can be substantiated by evidence. The argument part consists of strategies used to decompose claims and sub-claims.

D. *Robustness and adversarial attacks*

To be able to use a DNN in a safety-critical system, it has to be robust. It must handle unintentional variations in the form of natural variations in inputs such as noise or degraded operations. It must also be able to handle intentional variations made by humans with the purpose to mislead the network to perform incorrectly [5]. A major problem with DNNs is that small changes in inputs (for humans seemingly invisible) can make the DNNs completely fail in detecting the correct output [21, 22]. Human made disturbances are also called adversarial attacks and can be applied during the training phase of the network or in normal operation. Neural networks that dynamically learn during normal operation are considered to be most vulnerable for these kinds of attacks [5] and should therefore be avoided in safety-critical systems. In addition, if proper learning assurance is applied, such as the W-model described in [1], the risk for adversarial attacks during training should be considered low.

The research field addressing intentional variations and some natural phenomena, e.g. sensor degradation and aging, is called Adversarial Machine Learning [5]. It has grown fast the last years. Promising research show significant improvements in making the networks robust, not only for intentional but also for unintentional alterations.

E. *Performance metrics*

Naturally, many neural networks are trying to mimic a perception function of some sort, but it is not universally defined what 'good perception' is, rather it depends on the context of the system in which the perception function is used. For the case of vision guided landing the task is to detect runway(s) on the ground and use the position of the detected runway to provide navigational support to other systems. In this context it is critical not to mistake a freeway or other road-like objects for a runway, i.e. the precision of the detector of the runway class needs to be high (minimizing false positives). Conversely when approaching ground and we have runway in sight, the perception task might now shift to detect foreign object debris (FOD) or other (misplaced) vehicles disqualifying the use of the runway. In this context it is critically important not to miss any such object as this could lead to catastrophic consequences, i.e. the recall of the detector is what matters (minimizing false negatives).

F. *Transient faults in deep neural networks*

While most of the discussions concern systematic faults when it comes to DNNs in safety-critical applications, transient and permanent faults should not be neglected. In fact, DNNs are in many cases vulnerable to soft-errors due to the internal hardware structure used (e.g. accelerators with a large amount of buffer memory) [23]. Traditional but expensive solutions may of course be used for this purpose, i.e. triple modular redundancy (TMR) with voting or rad-hardened hardware. In the case of triplicating DNNs, failed units have to be detected and restarted, if soft errors may occur often. Other solutions may use information redundancy, e.g. error correction codes or cyclic redundancy checks, to cope with soft errors in the DNN. In addition, there are other methods that can be used. Schorn, Guntoro and Ascheid [23] have developed a successful solution for dealing with soft errors in DNNs, an error detection and mitigation network (EDMN). They detect anomalies in the output of the DNN with another monitoring NN. A feed-forward neural network to detect critical errors in the large DNN. The monitor network still has to be correctly trained which increases the overall complexity of the system, but the results are impressing (at least for their application - detecting road signs). Since the EDMN is very small, this network may be triplicated without much overhead, to overcome soft errors in the monitor. To improve the error detection, they use time redundancy in the form of sequential similar images, which they feed their EDMN with.

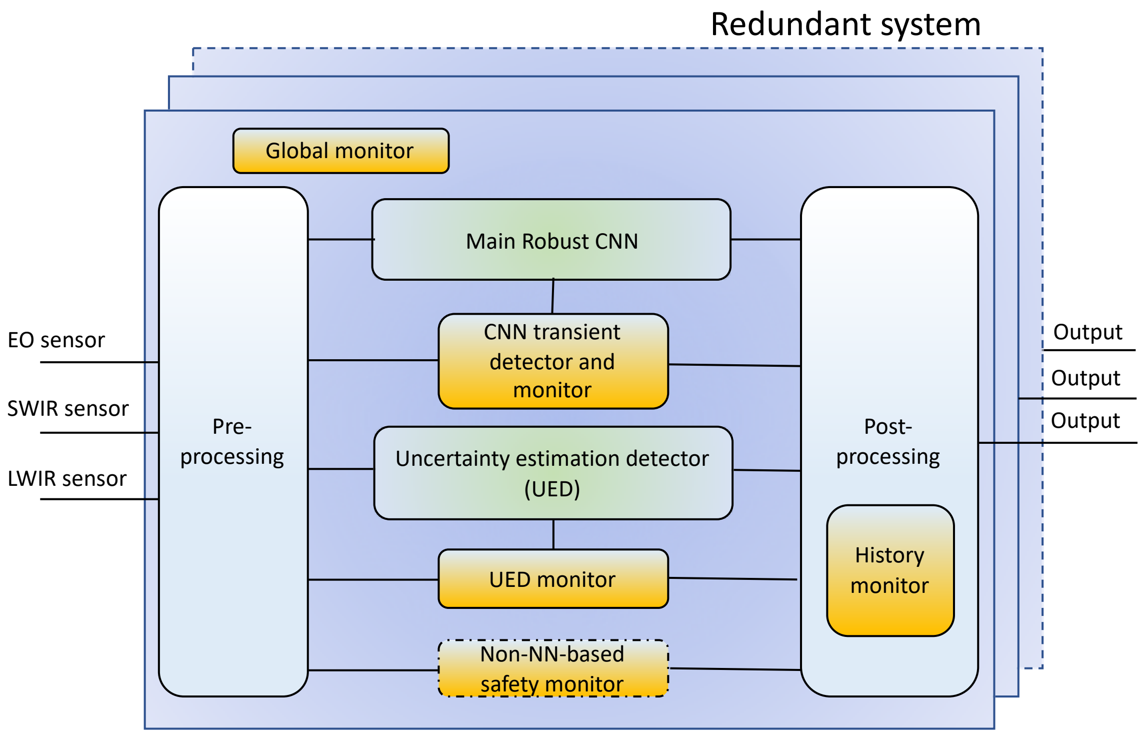

Figure 3. One possible implementation of a system for machine vision-guided approach using a deep neural network in the form of a convolutional neural network (CNN). The pre-processing merges the sensors and downscales the images. To detect transient errors in the robust CNN, a smaller neural network may be used as a monitor. To mitigate for false and missed predictions in the CNN, an uncertainty estimation detector is discussed to be used in parallel with the CNN. To mitigate for other incorrect CNN behavior, a history monitor is advised to be used in the post-processing part.

III. Possible Implementation

In this section we discuss one possible implementation of a machine vision-guided approach system using a convolutional neural network (CNN). A CNN is a is a class of DNNs which applies to analyzing visual imagery. To overcome the presented challenges, we believe a diverse redundant system is needed. Figure 3 shows our possible implementation and the 15 bullets below describe the solution in more detail. It should be mentioned that one inspiration source for the discussed implementation has been J.M. Cluzeau et al. [1] (Chapter 9.)

- Three different sensor types are used as inputs. These are Electro-Optical (EO), Short-Wave Infra-Red (SWIR) and Long-Wave Infra-Red (LWIR) sensors. By using diverse sensors, the effects of snow, fog and heavy rain seem more manageable.

- The preprocessing unit merges different sensor data and downscales the images.

- To reach specific corner cases when training the CNN, synthetic data must be created and used. Corner cases may not be reached in a safe manner through real flight data recording. Real flights are also very expensive. For autonomous landing, different approach angles and weather conditions should be synthesized. Software should be used to correctly handle realistic rendering of synthetic images, including programs to model weather conditions physically correct.

- The CNN should be trained with a combination of synthetic and real data. Synthetic data first and real data afterwards, see [9].

- The CNN should be robust enough to handle both intentional as well as unintentional variations, including untrained input data. At least to an acceptable level. The parameter describing the networks capability to correctly classify untrained input data is called generalizability [1].

- If a proper learning assurance such as the W-model [1] is used, the susceptibility to human made variations (adversarial attacks) during training is reduced.

- For the pre-processing and post-processing, traditional design assurance à la RTCA/DO-178C and RTCA/DO-254 should be used. For all neural networks where training is part of the design, a learning assurance [1] or similar process should be used in order to reduce systematic faults to an acceptable level. If any new COTS technology is used to implement the DNNs, assurance cases may be discussed with the certification authorities as an alternative method to argument that the assurance objectives can be met. For an example of how to use assurance cases for new emerging COTS technology, see Forsberg and Schwierz [18].

- For the training data, a data set requirement specification will address the properties of the data as well as the safety cases and security requirements. This will ensure the completeness of the training data and independence in defining the training data from test data.

- The neural networks used are static during operation, i.e. they are trained, frozen and baselined before use. Neural networks that dynamically learn during normal operation add significant complexity [1] and are considered more vulnerable to adversarial attacks [5].
- Different flight phases require different CNN characteristics. When detecting the runway, false positives should be minimized and when approaching the runway, the CNN should shift to detect FOD, i.e. minimizing false negatives. This could be performed by training the network for the first scenario followed by the second. At runtime, the CNN starts with the first configuration and then reloads for its second task when approaching the airstrip.
- The role of the uncertainty estimation detector in Figure 3 is extremely important. It mitigates (at least to some extent) the major problem with deep learning methods, to guarantee that the predicted outcome is correct. Bayesian neural networks (BNNs) seem very promising for this task, see e.g. [13] and [14].
- Depending on the underlying hardware, DNNs may be vulnerable to transient faults. Typically, it is too expensive to triplicate the DNNs, in terms of reliability, capacity and power. Instead a much simpler DNN detector and monitor may be used, see [23] for instance. Several researchers also try to prune the networks while maintaining robustness. Other methods to reduce the hardware cost (size and power) include the use of the residue number system [24].
- A typical monitor for real-time systems keep track of the history of previous outputs. For our CNN, the history of classified objects and moving targets may be used. We have added this monitor in the post-processing part in Figure 3. A similar approach is described by Cluzeau et al. [1]. A history monitor may be very useful in detecting incorrect CNN behavior.
- For our application, machine vision-guided approach, an additional non-neural network-based safety monitor that detects anomalies in the runway position by finding the runway outline perspective view lines, may be used.
- For symmetric faults, we assume redundant systems are used. See Figure 3.

IV. Discussions

From the discussed implementation above, it is easy to say that deep neural networks (DNNs) will be very hard to implement in safety-critical systems. The most advanced DNNs, the ones that can be trained during operation, or recurrent DNNs (not mentioned earlier in this article) are so complex to predict the results from such that they should be avoided in safety-critical systems, at least until the community knows more about how to handle those networks. But even for the "simpler" DNNs, it is not sufficient to use the DNN itself since it can incorrectly classify the objects to be identified or even miss to detect the objects, to a too high degree. It is also impossible to train or validate the complete input space. In addition, DNNs typically need high-performance hardware during normal operation, which often are vulnerable to soft errors. To protect from these errors and false predictions, many of the best solutions suggested from researchers often involve other simpler neural networks, simply due to their supremacy in detecting abnormal activations in large and complex networks. These simpler networks may also be vulnerable to soft errors and should be protected. Finally, the complexity in designing NNs including DNNs to work correctly for its intended operations need new assurance approaches to ensure systematic faults are limited to an acceptable level. Some of these approaches also seem to mitigate for intentional variations of the inputs, at least during the training phase.

There are several sayings used in the safety community, e.g. *keep it simple* or *complexity is your enemy*. So why on earth should DNNs be used in safety-critical systems? Simply because of their superiority in certain applications, e.g. object detection and classification. No other solutions seem to be close to the performance of the DNNs.

In our future research, we will study fault-tolerant architectures for safety-critical applications involving DNNs. Our goal is to keep the systems as simple as possible, yet sufficiently safe and efficient. We will also continue to research design assurance for new hardware technologies supporting DNNs.

V. Conclusions

In this paper we have introduced challenges in implementing DNNs in safety-critical applications and proposed a solution to cope with these challenges. The solution is mainly on a theoretical basis. It addresses the holistic view of implementing a working concept of DNNs in safety-critical airborne applications. The outcome indicates that a real application will be very complex to implement, which is in contrast to the simplicity the safety community always strives for. The complexity concerns both designing the intended function as well as the monitors to achieve a fail-safe behavior of the system.

References

[1] J.M. Cluzeau *et al.*, "Concepts of design assurance for neural networks (CoDANN)," Public Report Extract, EASA AI Task Force and Daedalean AG, Version 1.0, March 31, 2020.

[2] EASA, *Artificial intelligence roadmap – a human-centric approach to AI in aviation*. Version 1.0, February 2020.

[3] SAE, "Artificial intelligence in aviation," *SAE Standardization Committee G-34*. [Online]. Available: https://www.sae.org/works/committeeHome.do?comtID=TEAG34 [Accessed: July 15, 2020].

[4] EUROCAE, "Artificial Intelligence," *European Organisation for Civil Aviation Electronics*, Working Group WG-114.

[5] D. Redman, D. Ward and M. Carrico, "AFE 87 – Machine Learning," Aerospace Vehicles Systems Institute, Final Report, Issue 1.0, May 7, 2020.

[6] AVSI, "Machine Learning Certification," *Aerospace Vehicles Systems Institute*, Project AFE 89.

[7] The DEEL Project, "Dependable, Certifiable & Explainable Artificial Intelligence for Critical Systems." [Online]. Available: https://www.deel.ai/ [Accessed: July 15, 2020].

[8] R. Salay, R. Queiroz and K. Czarnecki. "An analysis of ISO 26262: Using machine learning safely in automotive software," (2017): n. pag. Print.

[9] A. Gaidon, Q. Wang, Y. Cabon and E. Vig. "Virtual worlds as proxy for multi-object tracking analysis," In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, Nevada, USA, 2016, pp. 4340–4349.

[10] J. Corné and U. Helander Sjöblom, "Investigation of IR transmittance in different weather conditions and simulation of passive IR imaging for flight scenarios," M.S. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2019.

[11] A. Berk, P. Conforti, R. Kennett, T. Perkins, F. Hawes and J. van den Bosch, "MODTRAN6: a major upgrade of the MODTRAN radiative transfer code," In Proc. SPIE 9088, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX, 90880H (June 13, 2014); doi:10.1117/12.2050433.

[12] D. J. Fremont *et al.* "Scenic: a language for scenario specification and scene generation," In Proc. of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI 2019 (2019).

[13] B. Phan, S. Khan, R. Salay and K. Czarnecki, "Bayesian uncertainty quantification with synthetic data," 10.1007/978-3-030-26250-1_31, 2019.

[14] A. Levin and N. Vidimlic, "Improving situational awareness in aviation: Robust vision-based detection of hazardous objects," M.S. thesis, IDT, MDH, Västerås, Sweden, 2020.

[15] S. Shafaei, S. Kugele, M.H. Osman and A. Knoll, "Uncertainty in machine learning: a safety perspective on autonomous driving," In: Gallina B., Skavhaug A., Schoitsch E., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2018. Lecture Notes in Computer Science, vol 11094. Springer, Cham, 2018.

[16] M. Henne, A. Schwaiger, K. Roscher and G. Weiss, "Benchmarking uncertainty estimation methods for deep learning with safety-related metrics" In *SafeAI@ AAAI* (pp. 83-90), 2020.

[17] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Ch. 6., ISBN 978-0-26203561-3, 2016.

[18] H. Forsberg and A. Schwierz, "Emerging COTS-based computing platforms in avionics need a new assurance concept," In IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), San Diego, CA, USA, 2019.

[19] M. C. Holloway, DOT/FAA/TC-xx/xx: *Understanding the overarching properties: first steps*, Limited release document, September 2018.

[20] J. Wlad, Verocel, "Certification initiatives ongoing for unmanned aircraft systems," in *Military Embedded Systems*, April 26th, 2018.

[21] A. Nguyen, J. Yosinski and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[22] E.R. Balda, A. Behboodi and R. Mathar, *Adversarial Examples in Deep Neural Networks: An Overview*. In: Pedrycz W., Chen SM. (eds) Deep Learning: Algorithms and Applications. Studies in Computational Intelligence, vol 865. Springer, Cham, 2020.

[23] C. Schorn, A. Guntoro, and G. Ascheid, "Efficient on-line error detection and mitigation for deep neural network accelerators," In International Conference on Computer Safety, Reliability, and Security, Springer, Cham, 2018, pp. 205-219.

[24] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev and N.I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, 2020.