Multisensor Fusion and Integration Meets Software 2.0

Dr. Ramesh Bharadwaj
4555 Overlook Avenue SW
Washington DC 20375
USA

Ramesh.Bharadwaj@nrl.navy.mil

ABSTRACT

The term "Software Crisis" was first mentioned in the proceedings of the first NATO Software Engineering Conference in 1968, and yet is still acknowledged as a crisis more than half a century later. In spite of recent advances in computing, software systems remain brittle, expensive, and delivered late with a number of latent flaws. The current state of affairs is particularly dangerous for Cyber-Physical Systems (CPSs), whose incorrect or deficient operation may lead to loss of lives, and cause irreparable harm to society. However, this "deploy and patch" attitude persists, leading to catastrophic accidents and spectacular failures. Recent advances in Machine Learning (ML), in particular Deep Learning (DL), have entered the arena like a breath of fresh air, leading many to speculate that these approaches will be game changers in the way CPSs are built and deployed. Some have gone so far as to label this approach Software 2.0 to highlight their Data-Centric, vs Human-Centric approach for software construction. Yet, Software 2.0 is no panacea. Systems constructed using this approach have unintended functions, and are sensitive to adversarial perturbations, which may lead to surprising failure modes in the field. In this paper we argue that, for CPSs built using ML/DL, fusion of additional sensor information, specifically in regions of uncertain operation, will be a game changer, and promote safe system deployments with predictable behaviours under fielded conditions. This approach will promote development and deployment of systems that are certifiably robust and free of unintended failures in their Operational Design Domain (ODD).

1.0 INTRODUCTION

The term "Software Crisis" was coined at the First NATO Software Engineering Conference held in 1968 in the small German town of Garmish-Partenkirchen [1]. It referred to the difficulty of writing useful and efficient computer programs in the required time, due to the rapid rise in complexity of the problems that could be tackled, given the exponential growth and availability of inexpensive computing hardware, leading to increased difficulty of software development and inadequacy of extant development methods [2]. Here we are, more than half a century later, having to cope with software systems that remain brittle, expensive, and delivered late, using a process disparagingly known as the "deploy and patch" approach, whose failures often have catastrophic societal consequences. However, recent advances in Machine Learning (ML) and Deep Learning (DL), have led many researchers to speculate that these may be game changers in the construction and deployment of software systems.

2.0 BACKGROUND

2.1 Software 2.0

Researchers such as Andrej Karpathy (Director of AI at Tesla) have gone so far as to label this new approach Software 2.0, to highlight their Data-Centric vs Human- Centric approach to software construction, thereby harbouring the hope that successful deployment of systems constructed using ML/DL may finally be the breakthrough to end the software crisis in its entirety [3]. In contrast to the "classical" approach to software development – dubbed Software 1.0 – which is carried out by the human-intensive process of providing explicit instructions to the computer using conventional programming languages, Software 2.0 involves a much more abstract and inscrutable process, such as choosing the weights of a neural network, with no human involvement in explicitly writing the code or choosing the weights. Instead, the human specifies a “cost function” as a measure of the desired behaviour of the system, with massive computation resources being used to search the solution space for a parameter set that minimizes the cost function, directed by a large (labelled or unlabelled) corpus of data for “training” the system under development [4].

2.2 Cyber-Physical Systems

The promise of Software 2.0 is particularly enticing for the design and construction of Cyber-Physical Systems (CPSs), i.e., systems which interact with the physical world (as opposed to Cyber Systems that are deployed solely in a virtual world), as their design, implementation, and certification are far more intricate and complex in comparison to conventional software systems, leading to escalating costs for their development, operation, and sustainment [5]. This has led to irrational exuberance among sponsors and developers of CPSs, such as autonomous combat aircraft and robotic vehicles, who see Software 2.0 as a panacea to bring down development costs and improve their reliability. Termed safety-critical systems, their unintended or incorrect operation when deployed in the physical world, however, have more serious consequences, such as loss of life and/or serious damage to infrastructure and property.

2.3 Pitfalls of Software 2.0 and their Mitigation

Sadly, such speculative developments are misguided as Software 2.0 is no panacea. Systems constructed using this approach have unintended functions [6], and are sensitive to adversarial perturbations [7], which may lead to surprising failure modes in the field. Further, these systems may fall prey to silently adopting biases in their training data [8], which is very hard for humans to analyse and diagnose. In this paper we argue that, for CPSs built using ML/DL, fusion of additional sensor information, specifically in regions of uncertain operation, will be a game changer, and promote safe system deployments with predictable behaviours under fielded conditions. This approach will promote development and deployment of systems that are certifiably robust and free of unintended failures in their Operational Design Domain.

3.0 RADAR SIGNAL PROCESSOR

In this section, we briefly describe the receiver signal processing subsystem of a UHF pulse-Doppler radar which was implemented using a conventional – Software 1.0 – approach, contrasted with a newer design which incorporates a signal processor based on deep learning – i.e., Software 2.0 – for automatically detecting and tracking objects of interest. We compare and contrast both approaches, and provide preliminary results of experiments incorporating multisensory data fusion of signal processors developed using the two approaches, and articulate a path forward for future work.

3.1 UHF Pulse-Doppler Radar

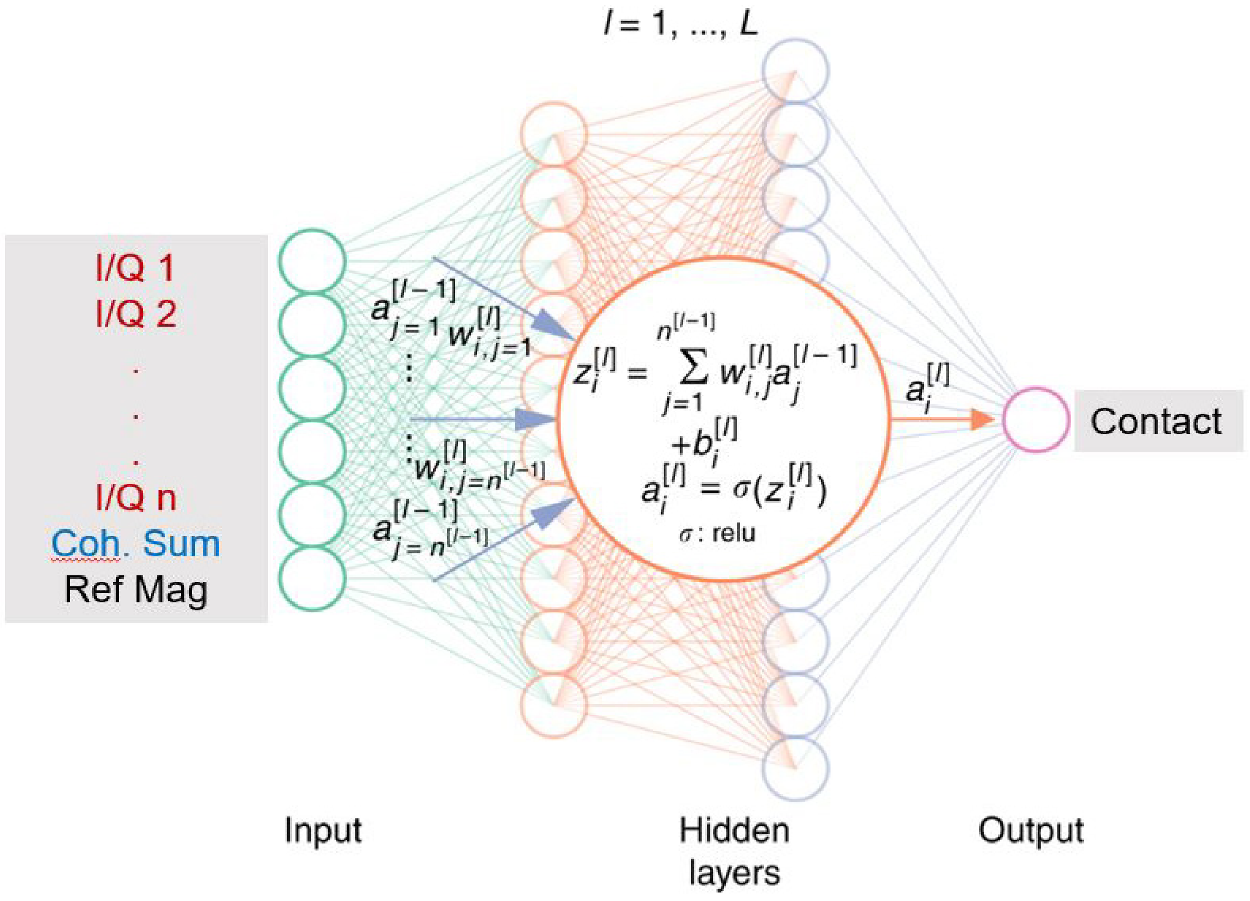
The system under consideration is a narrow beam UHF Naval pulse-Doppler radar used for very long-range air surveillance and tracking, similar to non-DoD pulse-Doppler radar systems whose information is public knowledge, as available on open source platforms [9].

3.2 Conventional Signal Processor

The radar provides sophisticated pulse-Doppler processing for automatic detection and reporting of targets within its surveillance volume. The receive signal processing includes accurate centroiding of target range, azimuth, amplitude, and radial velocity with an associated confidence factor to produce contact data for command and control systems. In addition, contact range and bearing information are provided for display on standard plan position indicator consoles. The radar has an automatic target detection capability with pulse-Doppler processing and clutter maps, ensuring reliable detection in normal and severe types of clutter.

3.3 ML/DL Based Signal Processing System

In this section we briefly describe the design of a pulse-Doppler receive processor subsystem based on Deep Neural Networks (DNNs). The design of the radar included the selection of a suitable neural network architecture (see Figure 1 courtesy Michael Colon). The DNN was then trained using deep learning (i.e., Software 2.0), on synthetically generated radar returns to automatically detect and track objects of interest.



$l = 1, \ldots, L$
I/Q 1
I/Q 2
I/Q n
Coh. Sum
Ref Mag
$a_{j=1}^{[l-1]} w_{i,j=1}^{[l]}$
$w_{i,j=n^{[l-1]}}^{[l]}$
$a_{j=n^{[l-1]}}^{[l-1]}$
$z_i^{[l]} = \sum_{j=1}^{n^{[l-1]}} w_{i,j}^{[l]} a_j^{[l-1]} + b_i^{[l]}$
$a_i^{[l]} = \sigma(z_i^{[l]})$
σ: relu
$a_i^{[l]}$
Contact
Input
Hidden layers
Output

Figure 1: Radar Processing Architecture.

3.4 Radar Signal Processor Architecture

Figure 1 illustrates the architecture of our signal processing chain [10]. The function of the signal processor was to detect targets for each range cell of a coherent processing interval (CPI) using In-phase and Quadrature (I/Q) data derived from the analogue front-end and the Analog to Digital (A/D) converter at the receiver front-end. We created a DNN for each range cell individually. In order to facilitate quick convergence, this data was conditioned based on our speculative understanding of how the radar signal processor functions. We accomplished this by providing the signal processor the following inputs:

- The I/Q data for each pulse.

- The magnitude of I/Q returns, i.e., the square root of the squared sum of the In-phase and Quadrature components.
- A coherent sum of all I/Q returns for all pulses within a Coherent Processing Interval (CPI).
- Magnitude of returns for reference target with 1 m^2 Radar Cross Section.
- The Fast Fourier Transform (frequency components) of I/Q returns of successive pulses.

The Neural Network architecture chosen was a fully connected network per range cell with ten inputs and one output (to indicate whether or not a target was present at that range cell), two hidden layers of 32 and 8 neurons respectively, with ReLU activation functions for each layer. The synthetic data generator comprised of 1,000 Coherent Processing Intervals, simulated with 50% probability of the presence or absence of a target within each range cell, together with Additive White Gaussian Noise independently generated using a seed and a pseudo-random number generator, for each range cell.

We trained on 900 CPIs using a batch size of 32 and validated against the remaining 100 CPIs, and achieved an overall accuracy of 90% for the test data set.

3.5 Synthetic Target Generator

In order to train the DNN on realistic target returns, we built a simulator which incorporates sophisticated electromagnetic system models, including RF propagation and realistic Radar Cross Section (RCS) computations. The training data was generated by suitable labelling of sensor inputs based on Ground Truth information from the synthetic target generator. The trained DNN was validated against recorded detections from the radar signal processor which was developed using conventional software development (Software 1.0) techniques. We achieved a detection accuracy of over 96% on data recordings derived from real targets.

4.0 FUTURE WORK

Although the results are preliminary, they are encouraging. However, we would like to experiment with Multisensor Data Fusion techniques using DNNs. Due to the Gaussian Process assumption of Kalman Filter based algorithms, we expect the performance of the DNN to far exceeded the conventional Kalman Filter based approach, both in terms of computational requirements and object tracking accuracy. Extant approaches seem to use a Neural Network for fusing tracks from independent Kalman Filters for each sensor, working asynchronously. However, it is our belief that using a Recurrent Neural Network for carrying out data filtering in addition to track creation will provide better track accuracy, thereby performing object identification at increased confidence levels.

5.0 CONCLUSIONS

In this paper we presented a novel approach – which is based on Software 2.0 – for the rapid implementation of signal processing algorithms and their deployment under controlled conditions. Use of Multisensor Data Fusion improves accuracy and mitigates the effects of domain shifts, enabling neural network-based systems to be deployed in the field. Other approaches to mitigate the effects of domain shift are needed for DNN based classifiers to be robust and reliable. However, their low cost of construction and deployment, together with the regularity of their run-time architectures, are compelling reasons why this approach will be embraced.

6.0 REFERENCES

[1] P. Naur and B. Randell (Eds.), Software Engineering Techniques: Report on a Conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968, NATO Scientific Affairs Division, Brussels, January 1969, (231 pages).

[2] Anonymous, Software Crisis, from Wikipedia, the free encyclopaedia, last accessed May 30, 2021.

[3] A. Karpathy, Software 2.0, at https://karpathy.medium.com/software- 2-0-a64152b37c35, last accessed May 30, 2021.

[4] D. Anderson and G. McNeill, Artificial Neural Networks Technology, ELIN: A011, Rome Laboratory, NY, August 1992.

[5] V. Insinna, Inside America's Dysfunctional Trillion-Dollar Fighter-Jet Program, The New York Times, August 21, 2019.

[6] SAE G-34/ EUROCAE WG-114, Artificial Intelligence in Aeronautical Systems: Statement of Concerns, SAE International, Document number AIR6988, issued Apr 30, 2021.

[7] I. Goodfellow and J. S. C. Szegedy, Explaining and Harnessing Adversarial Examples, in Proceedings International Conference on Learning Representations, 2015.

[8] H. Suresh and J. V. Guttag, A Framework for Understanding Unintended Consequences of Machine Learning, arXiv:1901.10002v3, Submitted 28 Jan 2019, last revised 17 Feb 2020 (v3).

[9] Wikipedia, Pulse-Doppler Radar, https://en.wikipedia.org/wiki/Pulse-Doppler radar, last accessed May 31, 2021.

[10] M. Colon and R. Bharadwaj, ATDT, NRL Internal Memo, April 2019, unpublished.