COMPUTING edge

- Machine Learning
- Quantum Computing
- Smart Manufacturing
- History

DECEMBER 2024

www.computer.org

IEEE COMPUTER SOCIETY

IEEE

COMPUTING edge

IEEE COMPUTER SOCIETY

IEEE

IEEE COMPUTER SOCIETY computer.org

Printed with inks containing soy and/or vegetable oils

STAFF

Circulation: *ComputingEdge* (ISSN 2469-7087) is published monthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

Postmaster: Send address changes to *ComputingEdge*-IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Printed in USA.

IEEE Computer Society Magazine Editors in Chief

DECEMBER 2024 • VOLUME 10 • NUMBER 12

COMPUTING edge

Machine Learning

Quantum Computing

Smart Manufacturing

History

Departments

Magazine Roundup

The IEEE Computer Society's lineup of 12 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to visualization and microchip design. Here are highlights from recent issues.

Computer

Certifiability Analysis of Machine Learning Systems for Low-Risk Automotive Applications

In this article, featured in the September 2024 issue of *Computer*, the authors analyze the existing safety standard, ISO 26262, for automotive applications, determining the certifiability of machine learning (ML) approaches used in low-risk automotive applications. This can help with assuring the security and safety of ML-based autonomous driving systems, gaining the trust of regulators, certification agencies, and stakeholders.

Computing in Science & Engineering

Providing a Flexible and Comprehensive Software Stack Via Spack, an Extreme-Scale Scientific Software Stack, and Software Development Kits

To manage the complex demands of modern high-performance computing (HPC), software applications increasingly depend on software developed by other teams, often at other institutions. An HPC software ecosystem approach is required to support dependencies on third-party scientific software. This January–March 2024 *Computing in Science & Engineering* article describes the U.S. Exascale Computing Project (ECP) contributions to HPC software ecosystem challenges.

IEEE Annals of the History of Computing

Making Innovation in the Mexican Silicon Valley: The Early Years of El Centro de Tecnología de Semiconductores (1981–2001)

This article, featured in the April–June 2024 issue of *IEEE Annals of the History of Computing*, tells the early story of El Centro de Tecnología de Semiconductores (CTS) as a site of innovation. It argues that, along with economic and scientific development goals, CTS furthered political and geopolitical change agendas for IBM and Mexico. These included reorganizing labor around global supply chains and maintaining specific power dynamics between the Global North and South.

IEEE Computer Graphics and Applications

Integrating GPT as an Assistant for Low-Cost Virtual Reality Escape-Room Games

In this article, featured in the July/August 2024 issue of *IEEE Computer Graphics and Applications*, the authors explore the integration of generative pre-trained transformer (GPT), an AI language model developed by OpenAI, as an assistant in low-cost virtual escape games. Their study focuses on the synergy between virtual reality (VR) and GPT, aiming to evaluate its performance in helping solve logical challenges within a specific context in the virtual environment while acting as a personalized assistant through voice interaction.

IEEE Intelligent Systems

Effective Adversarial Examples Identification of Credit Card Transactions

Credit cards are a prevalent method of transactions but are

 2469-7087/24 © 2024 IEEE

susceptible to forgery, leading to numerous cases of fraud. As a result, much research has been focused on employing artificial intelligence (AI) to achieve high detection performance. However, the accuracy of these AI-based methods may be challenged by attack techniques using adversarial examples. To address this issue, this July/August 2024 *IEEE Intelligent Systems* article utilizes neuron activation status distribution and deep neural networks as detection tools. Furthermore, the experiments employ three methods to generate adversarial examples, showcasing the effectiveness of the proposed detection approach.

IEEE Internet Computing

Digital-Twin-Driven Deception Platform: Vision and Way Forward

Digital twin (DT) technology provides new opportunities to enhance the robustness and resilience of critical infrastructure. In this July/August 2024 *IEEE Internet Computing* article, the authors discuss the potential of DTs as a proactive security enabler and present a vision for using DTs as a deception platform to thwart cyberattacks on critical national infrastructure (CNI). They propose a generic DT-driven deception-based solution called securing cyberphysical systems through DT-driven deception (INCEPTION), which can serve as a research-based DT-driven deception platform for CNI.

IEEE micro

Mosaic Pages: Big TLB Reach With Small Pages

In this article, featured in the July/August 2024 issue of *IEEE Micro*, the authors introduce mosaic pages, which increase translation lookaside buffer (TLB) reach by compressing multiple, discrete translations into one TLB entry. Their results show that Mosaic's constraints on memory mappings do not harm performance, and there are no conflicts before memory is 98% full—at which point a traditional design would also likely swap.

IEEE MultiMedia

Adaptive Detachable Partition-Based Reference Frame Recompression for Video Coding

In this April–June 2024 *IEEE MultiMedia* article, the authors present an adaptive detachable partition-based reference frame recompression (RFRC) scheme which is capable of compressing a variable-size to a fixed bit of access unit. Compared with the conventional schemes, which compress fixed-size partition to variable bits, this work can increase the compression ratio by greatly reducing the redundancy of the partitions' boundaries.

IEEE pervasive COMPUTING

Energy Communities: Pervasive Technologies and Collective Futures

Energy communities are emergent sociotechnical constellations, where local actors collectively organize green energy initiatives. In this article, featured in the April–June 2024 issue of *IEEE Pervasive Computing*, the authors draw on emerging research and five situated design cases to illustrate the many nuances involved in envisioning and developing technology for energy communities. Reflections from these cases suggest a need for design approaches that better account for the diverse social lifeworlds into which these technologies are meant to foster local engagement toward sustainable futures.

IEEE SECURITY & PRIVACY

A Viewpoint: Safer Heaps with Practical Architectural Security Primitives

In this article, featured in the July/August 2024 issue of *IEEE Security & Privacy*, the authors argue that architectural security primitives are a promising basis for fast and secure program heaps. They discuss MPKAlloc, a recent research effort demonstrating the concrete benefits of this approach using Intel MPK to harden a production allocator.

IEEE Software

Integrating Static Quality Assurance in CI Chatbot Development Workflows

To fill a gap in proposals to integrate automated quality assurance mechanisms into the chatbot development workflow, the authors of this article from the September/October 2024 issue of *IEEE Software* present a continuous integration workflow for chatbot development, implemented as GitHub actions, and show its usefulness by its application to open source chatbots.

IT Professional

ChatGPT for Software Development: Opportunities and Challenges

Rapid natural language processing advances, such as OpenAI's ChatGPT, promise profound transformations across multiple domains, including software development. In this May/June 2024 *IT Professional* article, the authors discuss ChatGPT's role in software engineering, including an investigation of implications and applications highlighting ChatGPT's code-assistance capabilities. Through a series of analyses, they discuss the real impact of ChatGPT on open source software development.

Editor's Note

Keeping Up With the Rise of Machine Learning

The applications of machine learning (ML) have skyrocketed in recent years. ML is now being used in a vast range of industries including transportation, healthcare, marketing, cybersecurity, customer service, data analysis, social media, and even space exploration. The possibilities of ML seem endless. And yet—are software developers keeping up? This issue of *ComputingEdge* examines the applications of ML, where it needs improvement, and how to improve it, as well as the intersection of ML and quantum computing. The articles go on to discuss cyber threats in quantum computing as well as history lessons about software engineering and computer networking.

Developers are coming up with new ways to apply ML with improved specifications. In "Orchestrating Networked Machine Learning Applications Using Autosteer," from *IEEE Internet Computing*, the authors present AUTOSTEER, a software platform for deploying ML applications across hardware, cloud, and edge devices. The authors of *IEEE Software* article, "Feature Interactions on Steroids: On the Composition of ML Models," propose rethinking ML model composition to create stronger and more accurate specifications.

Despite the exciting potentials of quantum computing, there are also risks and obstacles to consider. The authors of "Distributed Quantum Machine Learning: Federated and Model-Parallel Approaches," from *IEEE Internet Computing*, explore the challenges of implementing two types of quantum ML methodologies and suggest potential solutions. "The Quantum Cybersecurity Threat May Arrive Sooner Than You Think," from *Computer*, reveals how quantum computers will eventually have the ability to break current encryption.

Automation is being employed for agricultural purposes—to improve farming and water regulation. *Computer* article "'Sensoring' the Farm" analyzes how the Internet of Things (IoT) can be used to increase farming efficiency and addresses the challenges of agricultural computing. The authors of the *Computer* article "AI for Water" show how AI can improve water access, treatment, and management.

Present-day engineers can look back on their predecessors' contributions to draw valuable lessons about computing that are still applicable today while avoiding the same mistakes. The article "Computer Networking Initiatives in One of the World's Remote Cities," from *IEEE Annals of the History of Computing*, tells the story of the risky—but rewarding—purchase of the first time-sharing computer in Australia. In "Lessons From the Father of Software Engineering," from *Computer*, the author reflects on the life of Barry Boehm and how he impacted the design and management of software.

EDITOR: Ewa Deelman, deelman@isi.edu

This article originally appeared in IEEE Internet Computing vol. 26, no. 6, 2022

DEPARTMENT: VIEW FROM THE CLOUD

Orchestrating Networked Machine Learning Applications Using Autosteer

Zhenyu Wen and Haozhen Hu, *Zhejiang University of Technology, Hangzhou, 310023, China*
Renyu Yang, *University of Leeds, LS2 9JT, Leeds, U.K.*
Bin Qian, Ringo W. H. Sham, and Rui Sun, *Newcastle University, NE1 7RU, Newcastle, U.K.*
Jie Xu, *University of Leeds, Leeds, LS2 9JT, U.K.*
Pankesh Patel, *University of South Carolina, Columbia, SC, 29208, USA*
Omer Rana, *Cardiff University, CF24 3AA, Cardiff, U.K.*
Schahram Dustdar, *TU Wien, 1040, Vienna, Austria*
Rajiv Ranjan, *Newcastle University, NE1 7RU, Newcastle, U.K.*

A platform for orchestrating networked machine learning (ML) applications over distributed environments is described. ML applications are transformed into automated pipelines that manage the whole application lifecycle and production-grade implementations are automatically constructed. We present AUTOSTEER, a software platform that can deploy ML applications on various hardware resources—interconnected using heterogeneous network resources—across cloud and edge devices. Device placement optimization and model adaptation are used as control actions to support application requirements and maximize the performance of ML model execution over heterogeneous computing resources. The performance of deployed applications is continually monitored at runtime to overcome performance degradation due to incorrect application parameter settings or model decay. Three real-world applications are used to demonstrate how AUTOSTEER can support application deployment and runtime performance guarantees.

Machine learning (ML) systems and applications are intrinsically nondeterministic and need to operate in an environment that is constantly evolving, and contains ever-changing data. Typically, a networked ML application consists of a variety of components for data collection, device control, model inference (e.g., speech recognition, object detection), which are deployed and managed at different locations, i.e., either on locally managed servers or remotely in cloud data centers or edge environments.

1089-7801 © 2022 IEEE
Digital Object Identifier 10.1109/MIC.2022.3180907
Date of current version 23 December 2022.

ML applications executing over a networked platform are arguably complex systems, which have to be continuously updated and maintained. ML applications need to be transformed into automated pipelines that manage the whole application lifecycle and build production-grade ML implementations. A pipeline workflow, typically in the form of a graph representing the component interconnections in an ML application, can comprise: data management, model learning (model selection, training, and hyperparameter selection), model testing, and validation and model deployment. Thereafter, runtime management is responsible for ensuring performance guarantee, i.e., end-to-end model performance optimization and model update,[1] so that the deployed ML applications can be dynamically modified to runtime environment.

Doing so manually is generally unrealistic and not scalable, particularly when thousands of ML applications

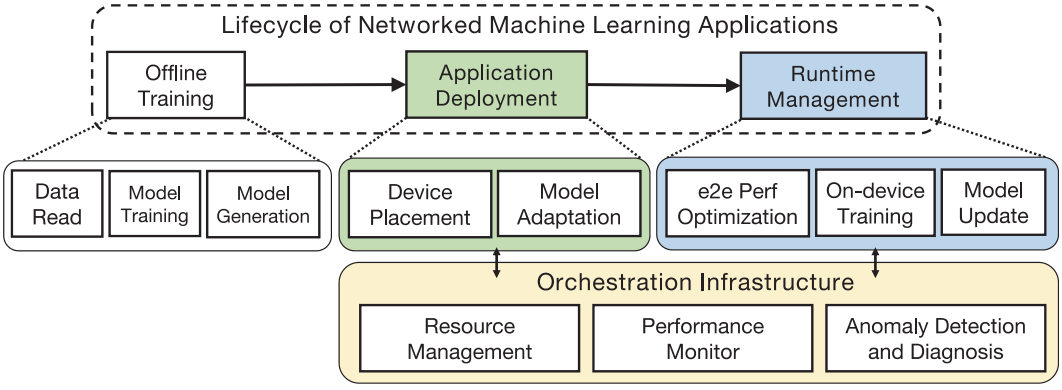


FIGURE 1. Conceptual workflow.

are submitted and maintained in edge and cloud platform that may be composed of hundreds of devices with heterogeneous hardware and software specifications. Continuous and automatic orchestration plays a pivotal role in deploying, managing, and synchronizing models of the ML applications across multiple tiers in a distributed computing environment. For instance, the trained models will be published and delivered to specific cloud servers or edge devices to run inference. Some specific applications, e.g., federated learning tasks require on-device training, indicating more complex device placement and model synchronization. Moreover, model decay arising from changes in data, would inevitably diminish model accuracy over time. Hence, an orchestrator calls for observation of the performance deviation and redeployment of the updated models.

Deploying such networked ML systems, particularly in an IoT and edge environment can be challenging due to the difficulty in managing the complexity of heterogeneous network and hardware resources. A variety of devices are used for data exchange, model training, and data analysis encompassing edge devices (such as IoT gateways and base stations) and servers (such as GPU, CPU, and TPU-based devices). Existing ML model development can be computationally expensive and resource intensive, which impede the effective deployment of applications, particularly those with strict latency requirements to resource-constrained devices.

In this article, we propose a platform solution to deployment and runtime management for the pipelines of networked ML applications. We devise *AUTOSTEER*, a management system that can automatically deploy networked ML applications over heterogeneous network and hardware resources while ensuring their performance through deployment plan optimization and model adaptation. At runtime, *AUTOSTEER* continually monitors the performance of deployed applications and automatically performs model update to mitigate performance degradation caused by obsolete application parameters setting or model decay. Finally, we use three real-world applications that are executed upon *AUTOSTEER* to showcase how the mechanisms are engaged in the application deployment and runtime maintenance.

MOTIVATION

Motivating Examples

We primarily categorize the networked ML applications into a) *centralized off-site* ML applications that can be trained offline or offsite, and b) *distributed on-site/federated* ML applications that must build their models using local dataset on individual device and, in some cases, share and aggregate models with other peer devices.

Centralized off-site learning applications: A smart home application allows users to observe the occupancy of their house, remotely control the smart devices (e.g., LEDs, air conditioner) via smartphone and even automatically control the smart devices. For example, a smart home application can automatically adjust the temperature of air conditioners based on the occupancy, weather, and so on.

Distributed on-site/federated learning applications: A high-quality brain tumor detection application relies on a huge amount of magnetic resonance imaging data that is only locally available and managed within a specific institution domain due to GDPR and other privacy regulations. A shared model is typically distributed to different data owners and trained locally. Locally trained models will be combined into a consensus model.

Research Scope and Overview

In general, the pipeline for such an application can be depicted as the workflow in Figure 1. The pipeline starts with and augments an initial model that has been trained offline along with a reference to metadata and the associated data sources on which the model has

been trained. Thereafter, the workflow management platform typically addresses two fundamental problems: planning for device placement and model adaptation in the *deployment* phase and model execution performance guarantee in the *runtime* phase.

Determining the placement of ML components on available resources remains a key challenge—especially due to heterogeneity of resources. In addition, models have to be converted, for example through model pruning,[2] posttraining quantization,[3] and identifying a "focus" for the associated model through *distillation* techniques. This enables the generated models to best fit the target device, balancing the model size with accuracy of prediction. Significant recent efforts in this area include TinyML and EdgeML.

Once the plan of deployment comes into effect, runtime management ensures that the model performance can be monitored and overcomes model staleness. In the automated and continuous pipeline, triggers can be used to update application parameters or retrain the stale model with fresh data when performance observably degrades due to dynamic environment changes, such as network speed drop, workload bursting, model drift, or lack of generalization. For applications of federated learning and distributed training, the platform runtime also needs to enforce efficient on-device training.

A key focus of this work is to devise an orchestration system for supporting multiple ML model development and performance optimization. In addition, the system needs to scale to support both application size and resource heterogeneity. To underpin precise performance monitoring and anomaly detection while measuring platform health and resource utilization, we also need to track and inspect (distributed) system *fingerprints*—consisting of various performance indicators and application metrics, such as drift and prediction scores.

CHALLENGES

We elaborate on these specific challenges facing the ML workflow platform in the following notable aspects.

Complexity of device placement and model adaptation: Planning for a pipeline of a given ML application indicates a mapping procedure between awaiting models and available computing resources on the devices. To accommodate the specific demands of diverse distributed or federated learning applications, infrastructure resources have become increasingly heterogeneous, making the planning a far more intricate task.

1) *Device placement:* Successfully deploying sizeable components of the ML applications served in the platform requires stringent capacity check and optimization solution under numerous constraints. The manifestation of heterogeneity intrinsically stems from the static attributes of the hardware, such as CPU, GPU, memory, SSD, and network bandwidth, and of the software including operating system version, clock speed, and particularly software libraries. The compatibility of a given hardware or library version even becomes a hard constraint, for any violations of such requirements would completely fail the deployment. For example, some components are compiled for ARM Mali cannot be executed on Nvidia GPU. The network constraints, such as bandwidth sharing among colocated components or network latency specified by each individual component, will further exacerbate the planning complexity.

2) *Model adaptation:* The advancement of deep models, such as recurrent neural network and convolutional neural network leads to the substantially increased parameter number and the resultant computational cost, which hinders the real-world model deployment into embedded and edge devices. Hence, model pruning and compression can be used to reduce model size, remove redundant weights, such that pretrained models can better adapt to portable devices with limited resources (e.g., memory, CPU, power, and bandwidth) and be applied into real-time applications.

3) *Enabling dependent components within a pipeline:* Each individual ML model has its own specification and format of input and output data. Dependencies are referred to as the interactions, such as the data flows and remote callings, among interconnected components. This would be problematic and challenging particularly when components deployed on various devices are interconnected via different network types and protocols. Hence, it is imperative to design an effective data messaging system to orchestrate the data flow and manage the network traffic across different models while considering the particular specification and data format.

Optimized runtime management: Improper application parameter setting or model decay could result in poor performance of an ML application and even failures. The first task of runtime management is to perform end-to-end and intraapplication optimization. Application parameters (e.g., model accuracy, task offloading rate) need to be adjusted at runtime to ensure the allocated resource can guarantee the expected performance level. To do so, the orchestration system should be capable of automatically detecting any performance degradation of the deployed applications and then dynamically work out the optimal configuration to rescue the abnormal performance. Second, in the face of any model failures, the orchestration system should automatically perform local on-device training while

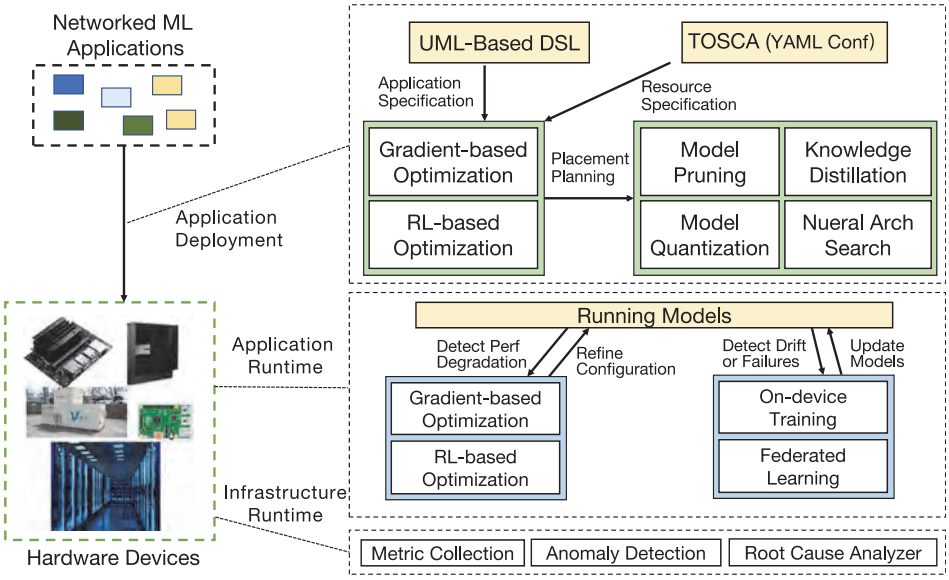


FIGURE 2. Architecture of *AUTOSTEER*.

synchronize and aggregate the up-to-date global models on the fly.

Low-cost platform monitoring and troubleshooting: Monitoring is one of the primary issues in maintaining ML applications and systems; outline or anomaly detection is important to find out unexpected model prediction or any system-wide issues in the early stage. However, anomaly detection and trouble-shooting could be challenging as high-quality labeled data are sparse and difficult to obtain and hence only semisupervised or unsupervised approaches could be applied. The overhead is another non-negligible consideration when designing application instrumentation and metric collection. This usually indicates a tradeoff between the accuracy and granularity of the measured data. Hence, the platform solution of infrastructure monitor should have an overall co-design of metric sampling, storage and real-time analysis.

SYSTEM DESIGN

In response to the aforementioned challenges, we develop *AUTOSTEER*, an orchestration platform for application deployment and runtime management. In this section we mainly highlight a set of key techniques used for implementing the orchestration mechanism. Figure 2 describes the architecture of *AUTOSTEER*.

Automatic Application Deployment

Application and resource specification: The user submits an ML application with execution logic, pretrained models and specifies the pertaining requirements, such as model accuracy and end-to-end latency. To achieve an automatic deployment, we need to translate this knowledge to machine-understandable language. We use a UML-based visual domain specific language[4] that can easily represent the component dependencies within an application and specify the format and source of input and output of each individual component. As a result, the interactions between components, such as data flows and service calls, are loosely coupled through interfaces and agnostic about any model updates. Apart from the application specification, standardized resource specification is the key to automatic and efficient deployment. We exploit[5] for specifying the available underlying computing resources and the hardware and software requirements of each application.

Planning optimization for device placement: To navigate the algorithmic complexity, the orchestrator in *AUTOSTEER* adopts two optimization techniques: gradient-based optimization[6] and reinforcement learning (RL).[7] Gradient-based approaches work upon a realistic model to formalize an optimization problem and usually have relatively low time complexity without the need of *a priori* knowledge or experience, which are therefore suitable for new applications. In contrast, RL-based methods can learn the optimal planning from the experiences and can better support the uncertainties compared the gradient-based solutions.

We also construct an efficient data messaging subsystem where two types of dependencies are defined—*data flow* and *service call*. Since the orchestration system needs to deliver a large volume of data

in distributed environments, high system throughput becomes a critical system objective. We employ the publish/subscribe paradigm implemented in Apache Kafka to underpin the data flows. The service call, on the other hand, is implemented through RESTful APIs, as the precise command delivery is the primary goal. Both the *AUTOSTEER* publish/subscribe and RESTful paradigms can be implemented upon a vast majority of network types and protocols, hence capable of supporting most networked ML applications.

Model Adaptation

Computation optimization aims to improve the execution efficiency of different computation units associated with the model (e.g., vector–vector, vector–matrix, and matrix–matrix operations) on various hardware. Optimizing the execution pipeline of the computation graph of a neural network can further improve model performance. We use TensorRT along with the adjustment of weights and numerical precision associated with the activation function (e.g., INT8 and FP16). Model architecture optimization improves the efficiency of on-device computation through well-designed models, such as MobileNetV2, ShuffleNet etc.,—part of the TensorFlow-Lite toolkit). We use *YOLOv3*[8] to strike a balance between computation efficiency and model accuracy.

In addition, more advanced and customizable approaches, such as neural architecture search (NAS)[9] and model compression can be implemented in *AUTOSTEER* further. NAS automates the search of an optimal network structure with the aid of RL or genetic algorithm-based approaches. However, it is computation-intensive and tends to be problematic given the portable devices with limited resources. Model compression is thus extensively studied in three notable aspects: *model pruning* that removes the redundant parameters within the networks; *quantization* that reduces the weights precision, and *knowledge distillation*[10] that trains a new small model based on a larger model. Quantization is the most straightforward approach at the risk of precision degradation and model pruning is the most well-established approach but requires extra calibration process. Integrating mixed techniques in the platform is already underway for building more adaptive and robust models.

End-to-End Application Optimization

In a networked ML system, computational and network resources are dynamically available at different levels. Application parameters, such as input rate and the targeted accuracy need to be adjusted, in response to the ever-changing traffic congestion, to assure the end-to-end latency or system throughput.

We specify model parameters based on extensive benchmarking experiments and transform the problem of finding the "best" setting of parameters into an optimization problem using techniques, such as convex optimization, evolution- and gradient-based methods. RL is an alternative approach that uses statistical or deep learning model where the application parameters are the actions of the agent, and the available computing resources represent the environment. The system performance is represented by the reaction of the environment to the actions. As opposed to the optimization-based approaches that have better interpretability but need extra hand-crafted modeling process, the RL-based approaches have better representation capabilities and can learn to set optimal application parameters from experience.

Model Update

Coping with the drift: During the lifecycle of an ML application, the relationship between the input variables and the performance of the targeting prediction inevitably experiences constant change and drift over time. The model drift usually originates from the following aspects. 1) *Invalid measurement indicator:* the replacement of data collection devices may give rise to different value spaces and a broken device could always deliver nil reading. 2) *Concept drift:* data distribution or statistical characteristics, which is uncertain and frequently varying over time, may lead to concept drift. 3) *Data drift:* the model effectiveness is also prone to inherent changes, such as the seasonal temperature rise and fall. Drifts can be roughly categorized into several classes: sudden drift (sudden change of the data pattern), gradual/incremental drift (new pattern that replaces the old ones within a period of time), and reoccurring drift (old patterns repop up later).

It is imperative to detect such drifts, understand the degree of drift and intervene the model for adapting to changing environments. There are three representative classes of drift detection. 1) Error rate-based approaches focus on the online detection of errors or sudden changes for triggering the model update. 2) Data distribution-based approaches mainly measure the statistical similarities between the original data and the new data and check if the difference is sufficient for model update. 3) Hypothesis test-based approaches, built upon the previous two methods, apply various hypothesis tests to quantify further the severity of model drift. Based on these approaches, our solution can determine *when to intervene* according to the starting and ending points of the drift, *where to intervene*, i.e., localizing the concept/data drift in the feature space, and *how to intervene*, in

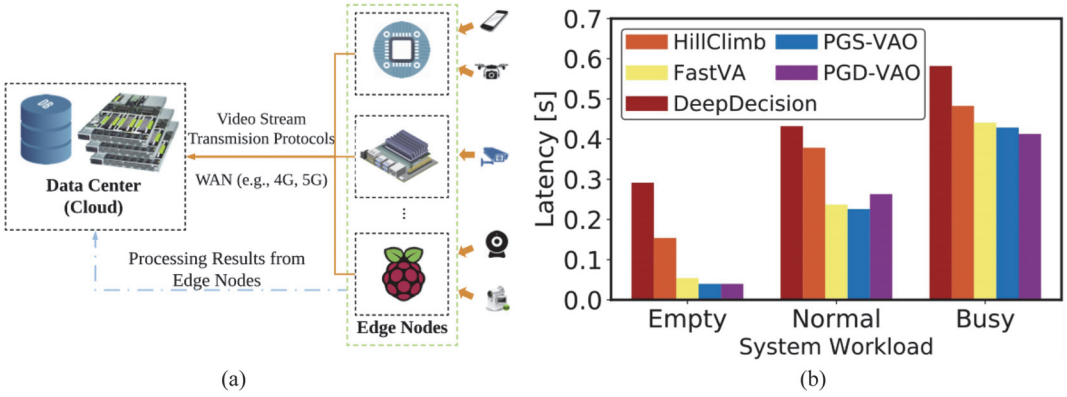


FIGURE 3. Edge-cloud video analysis application and an early performance comparison. (a) Illustration of an edge-cloud video analysis system. (b) Performance of workload optimizer in different system working conditions.

the light of the type and degree of the drift, by adaptively choose model update strategies. The most straightforward approach is the model retraining and updating. For concept drift, we ensemble several base classifiers or utilize knowledge transfer learning for the emerging new target variables.

System implementation: The amount of data engaged in the model update has an impact on the training effectiveness and the system overhead: less data can reduce computation and storage cost but only reflect the latest data distribution; more data are beneficial for reshaping models with higher precision, along with increased overhead. We employ an adaptive window-based solution to select the optimal data amount used for on-device training and/or global model synchronization via ADWIN[11] algorithm: instead of using a fixed time window, the algorithm calculates the drift rate from all possible windows and selects the best cut that reveals the optimal drift level. We modularize and implement the drift detection and alarming system in *AUTOSTEER*. The detection module is responsible for data retrieval and extraction of data statistical properties, and we then leverage hypothesis tests to evaluate the drift degree. Once the alarming system confirms the existence of the model drift, we employ techniques in *Section Model Adaptation* for efficient on-device training. For federated learning applications, once local model has been updated, we also trigger gradient aggregation to keep the global model up-to-date.

Infrastructure Monitor and Maintenance

To learn how the applications perform, we either collect general-purpose telemetry metrics in a black-box manner or instrument, as an integral part of the models, subsystems or system services, in a white-box manner. The metric tracking and tracing system of our orchestration infrastructure collects system logs, model metrics (task execution status, prediction statistics, and evaluation metrics as baselines), system metrics (request latency, error rates, network status, etc.), and resource metrics (CPU utilization, memory utilization, GPU usage, etc.) in real time, and ships them to a centralized analytic platform. We adopt the random sampling mechanism on each agent that is deployed on each physical node, for reducing the overhead of data collection. More advanced technologies, such as sketch[12] can be further added. *Anomaly Detector* comprises real-time event-based processing units, used for identifying per-application performance degradation while *Root-cause Analyzer* is implemented to troubleshoot the causes of performance degradation based on the collected performance indicators.

CASE STUDY: EDGE-BASED REAL-TIME VIDEO ANALYTICS

In this section, we showcase a real-world application backed up by the deployment and runtime management mechanisms in *AUTOSTEER*.

As shown in Figure 3(a), we develop a video analytical application following the edge-cloud paradigm. A set of video generating devices (e.g., traffic surveillance cameras, drones, mobile phones) produce live video streams, which are then processed either on low-power edge devices (e.g., Raspberry pi, Jetson Nano, computing chips), or GPU cluster in cloud datacenters. We prototype the video analytic application via object detection models yolo3 and the wide area network communication between edge devices and the data center is implemented by using the real-time video stream transmission protocol.

The heterogeneity of edge nodes and the interplay among the edge and cloud introduce uncertainties regarding network latency, hardware slowdown, or failures. As discussed in the "End-to-End Application Optimization," section the collected fingerprints and system status are mathematically modeled with a hierarchy queuing model that reveals the relationships between the workload offloading rate (between the edge and cloud) and the system latency and throughput. We then formulate a min-latency optimization problem bounded by a minimal throughput threshold. For model optimization, we implement two gradient-based optimization algorithms (i.e., PGD-VAO and PGS-VAO) to ascertain a solution to minimizing the overall latency. All components are containerized and deployed at both the edge and the cloud side via *AUTOSTEER*.

Figure 3(b) shows the performance of our proposed algorithms under *empty*, *normal,* and *busy* system workloads. Specifically, we insert video chunks into system buffering queues to simulate different workloads. Then, we test our algorithms against the other state-of-the-art task-offloading approaches, i.e., DeepDecision and FastVA. We can see that with the increase of the workload, the system latency is increasing as well. It is also clear that our modeling-based algorithms (e.g., PGS-VAO, PGD-VAO, FastVA) perform better than nonmodeling-based algorithms.

CONCLUSION

Most prior work related to ML applications focuses on algorithm design and optimization for better training ML models.

Although such work is essential for specific applications, there are few studies on the holistic orchestration solution to maintaining the lifecycle of networked ML applications. In this article, we first highlight several key challenges facing the orchestration systems. We then present a set of techniques to deploy ML applications onto resources across cloud and edge devices and assure their runtime performance, making models being served free from model decay and performance degradation due to inappropriate parameter setting. These assist in finding effective pathways to automating the management of networked ML applications at production level, although, admittedly, it still calls for significant effort in large-scale engineering practices and integration with wider domain-specific scenarios.

ACKNOWLEDGMENTS

This work was supported in part by U.K. EPSRC under Grant EP/T01461X/1, in part by U.K. Alan Turing Institute Post-Doctoral Enrichment Award Programme, and in part by U.K. Alan Turing Pilot Project.

REFERENCES

1. B. Qian *et al.*, "Orchestrating the development lifecycle of machine learning-based IoT applications: A taxonomy and survey," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–47, 2020.
2. M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017, *arXiv:1710.01878*.
3. B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.
4. T. Eterovic, E. Kaljic, D. Donko, A. Salihbegovic, and S. Ribic, "An Internet of Things visual domain specific modeling language based on UML," in *Proc. 25th Int. Conf. Inf., Commun. Automat. Technol.*, 2015, pp. 1–5.
5. T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "TOSCA: Portable automated deployment and management of cloud applications," in *Advanced Web Services*. New York, NY, USA: Springer, 2014, pp. 527–549.
6. D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2113–2122.
7. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.
8. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, *arXiv:1804.02767*.
9. T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.
10. J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4793–4801.
11. A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. 2007 SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.
12. T. Yang, Y. Zhou, H. Jin, S. Chen, and X. Li, "Pyramid sketch: A sketch framework for frequency estimation of data streams," in *Proc. VLDB Endowment*, 2017, pp. 1442–1453.

ZHENYU WEN is a professor with the Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou, 310023, China. Contact him at zhenyuwen@zjut.ed.cn.

HAOZHEN HU is currently working toward the master's degree with the College of Information, Zhejiang University of Technology, Hangzhou, 310023, China. Contact him at 2112003108@zjut.edu.cn.

RENYU YANG is a research fellow with the University of Leeds, LS2 9JT, Leeds, U.K. Contact him at r.yang1@leeds.ac.uk.

BIN QIAN is a postgraduate research student with the school of computing, Newcastle University, NE1 7RU, Newcastle, U.K. Contact him at b.qian3@ncl.ac.uk.

RINGO W. H. SHAM is a research technician with the school of computing, Newcastle University, NE1 7RU, Newcastle, U.K. Contact him at ringo.sham@newcastle.ac.uk

RUI SUN is a postgraduate research student in the school of computing, Newcastle University, NE1 7RU, Newcastle, U.K. Contact him at r.sun5@newcastle.ac.uk.

JIE XU is a chair professor with the School of Computing, the University of Leeds, LS2 9JT, Leeds, U.K., and chief scientist of BDBC, Beihang University, Beijing, China. Contact him at j.xu@leeds.ac.uk.

PANKESH PATEL is a researcher with AI Institute, University of South Carolina, Columbia, SC, 29208, USA. Contact him at dr.pankesh.patel@gmail.com.

OMER RANA is a full professor with the School of Computer Science and Informatics, Cardiff University, CF24 3AA, Cardiff, U.K. Contact him at ranaof@cardiff.ac.uk.

SCHAHRAM DUSTDAR is a full professor of computer science with TU Wien, 1040, Vienna, Austria. Contact him at dustdar@dsg.tuwien.ac.at.

RAJIV RANJAN is a chair and professor with Newcastle University, NE1 7RU, Newcastle, U.K., and with the China University of Geosciences, Wuhan, China. Contact him at raj.ranjan@ncl.ac.uk.

EDITOR: Tim Menzies, North Carolina State University, tim@menzies.us

This article originally appeared in
IEEE Software
vol. 39, no. 3, 2022

DEPARTMENT: SE FOR AI

Feature Interactions on Steroids: On the Composition of ML Models

Sven Apel, Christian Kästner, and Eunsuk Kang

FROM THE EDITOR

Is machine learning (ML) different? What lessons can we learn from software engineering (SE) that would help ML applications? Here, it is argued that "feature interaction" (a well-studied problem in ML) is a natural framework within which to discuss developing ML apps.

This column publishes commentaries on the growing field of SE for AI. Submissions are welcomed and encouraged (1,000–2,400 words, each figure and table counts as 250 words, try to use fewer than 12 references, and keep the discussion practitioner focused). Please submit your ideas to me at timm@ieee.org.—*Tim Menzies*

One of the key differences between traditional software engineering and machine learning (ML) is the lack of specifications for ML models. Traditionally, specifications provide a cornerstone for compositional reasoning and for the divide-and-conquer strategy of how we build large and complex systems from components, but these are hard to come by for machine learned components. While the lack of specification seems like a fundamental new problem at first sight, in fact, software engineers routinely deal with iffy specifications in practice. We face weak specifications, wrong specifications, and unanticipated interactions among specifications. ML may push us further, but the problems are not fundamentally new. Rethinking ML model composition from the perspective of the feature-interaction problem highlights the importance of software design.

Digital Object Identifier 10.1109/MS.2021.3134386
Date of current version: 18 April 2022

CHALLENGES IN COMPOSING ML MODELS

Many systems do not use just one ML model but compose multiple models to solve complex problems. To automatically generate captions for images, one could try to learn a model that directly takes an image and produces a caption, but a state-of-the-art solution decomposes the problem into three steps with different models.[1] First, a *visual detector* (a convolutional neural network) predicts which of 1,000 common objects is visible in the image, then a *language model* (a maximum-entropy model) takes these objects and generates 500 plausible sentences with them, and finally a *caption ranker* (a deep multimodal similarity model) takes both the original image and the generated sentences and scores the combination to pick the best sentence as the caption.

At a first glance, this looks like a great divide-and-conquer story. We break down the problems into steps. Each model can be developed and tested independently, using different modeling and

 Published by the IEEE Computer Society

implementation techniques, possibly reusing models or training data from other domains.

At a second glance though, it seems we cannot really reason modularly about problems. For example, as shown by Nushi et al.,[2] all three models somehow contribute to the poor caption "A blender sitting on top of a cake" for the image in Figure 1. The visual detector detects a blender where there is none, the language exhibits low common-sense awareness with "a blender sitting," and the ranking model picks that sentence, even though it includes a word with a low object-detection score. So, no component is behaving perfectly or compensating for problems in others. There are no clear responsibilities or boundaries between the components that could be used to assign blame. So, if we already have a problem with composing three models, how do we expect to build reliable systems with, say, the 18 models in Baidu's self-driving car system?[3]

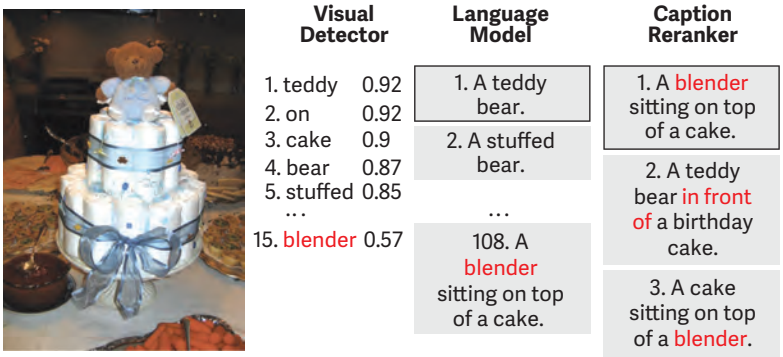


FIGURE 1. Problems in each component leading to the poor caption. (Copyright © 2017, Association for the Advancement of Artificial Intelligence. All rights reserved.)

THE ROOT OF THE PROBLEM: A LACK OF SPECIFICATIONS

The core of the problem is that we do not have clear specifications for what each model is supposed to do. For traditional software components, a specification tells us whether a component's output is either correct or wrong for a given input—not "pretty good" or "95% accurate." We also would not accept correct answers for 98% of all inputs, but would instead consider the component to have a bug if it produced a wrong result. In systems with multiple components, we can assign blame by checking which component produced a wrong output according to its specification (even if our specifications in practice are often weak and textual).

Specifications enable modular reasoning based on logic: we can understand the consequences from combining two modules in terms of the composed specifications. This is a cornerstone for a sound divide-and-conquer strategy and modularity, and it enables the vast reuse of libraries in building modern and complex software systems.

For ML components, we do not have any meaningful specifications in the traditional sense. For many problems, we already have a hard time capturing what it means for a single prediction of a model to be "correct" (humans may not agree). Even if we had a strict binary notion of correctness for any single prediction, we would not expect a model to make correct predictions for every possible input. In line with the aphorism "all models are wrong, but some are useful," we do not evaluate the correctness of a model but whether the model fits our problem, usually approximated through accuracy measures. These do not compose nicely.

WHILE THE LACK OF SPECIFICATION SEEMS LIKE A FUNDAMENTAL NEW PROBLEM AT FIRST SIGHT, IN FACT, SOFTWARE ENGINEERS ROUTINELY DEAL WITH IFFY SPECIFICATIONS IN PRACTICE.

THE LIMITS OF DECOMPOSITION: FEATURE INTERACTIONS

While it may seem that modular decomposition in traditional software systems is obvious and clean, unfortunately, that is not always the case either. It is not even just a question of whether we write formal or informal, strong or weak specifications, but a question of whether this would be possible in the first place. The study of feature interactions has made this amply clear.

There are many examples of feature interactions, and they typically follow a common pattern. We decompose the system into components, then design, develop, and test these components in separation, but finally observe unexpected behavior when composing them. The canonical example is a call forwarding feature in a phone system that competes with an independently developed call waiting feature on how to respond to the same call on a busy line. We can often blame interaction problems on weak specifications that did not anticipate the interaction, but in general the problem is much bigger.

LIKELY THE MOST POWERFUL STRATEGY IN ADDRESSING THE FEATURE-INTERACTION PROBLEM IS TO PREPARE FOR THE POSSIBILITY OF INTERACTIONS AS PART OF THE SYSTEM DESIGN.

In complex systems, it is often not possible to cleanly separate behavior and divide a problem into subproblems. Behavior is often antimodular. This is particularly apparent when the real world is involved. Consider a home-automation system with a heating component, a ceiling fan, and a component to open windows. We could specify the behavior of each controller and reason about how each component interacts with the environment, but the actions of components may influence each other through physical processes in the environment (such as heated air moving through the ceiling fan to the open window). Components may also compete for the same resources, such as electricity or human attention. To truly understand how the components behave in concert, we would need to fully understand the environment in addition to the actual components. Even if we could model the environment (such as the room layouts and thermodynamics), we could not reason about components individually but only about the system as a whole. As feature-interaction pioneer Michael Jackson framed it: "the physical world has no compositionality."

A key insight from the study on feature interactions is that, even when systems are complex, we decompose them anyway and, necessarily, make simplifying assumptions that may not actually hold. Humans simply cannot deal with complexity beyond a certain scale—we do not have the cognitive capacity. We need to abstract and decompose. We may make simplifications and create specifications that are weak or even wrong for specific cases. We may do this intentionally for good reasons, hoping that we can resolve issues at composition time.

The good news is that, most of the time, a divide-and-conquer approach pays off and imperfect decompositions work. For example, most home-automation components work well together as a rule, and control mechanisms can often dynamically adjust for unanticipated interactions. The problems that reach the surface are the remaining unanticipated interactions that surprise us.

COPING WITH FEATURE INTERACTIONS

Feature interactions have been actively studied since at least the 1990s. The community has learned how to build systems that work reasonably well despite weak or even partially wrong component specifications. Dedicated analysis, design, and control techniques can anticipate and compensate for some modularity violations.

While not a silver bullet, it is likely that there are insights from a long tradition of coping with feature interactions that may help us to better understand and build systems composed of both multiple ML and traditional components. In both worlds, we explicitly deal with modularity and composition problems stemming from weak or missing specifications. Let's look at three strategies to manage feature interactions.

Detection Through Testing

It is widely recognized that unit testing or component verification is clearly not sufficient, even in traditional software systems; integration testing and system testing are important too. This observation also holds for systems with ML components, which is just another reminder to evaluate the entire system (often in production) and not just the prediction accuracy of individual models.

Detection Through Better Specifications

A long history of research on feature interactions has shown that better requirements engineering can help to anticipate interactions, for example through systematic inspection of potential interaction points or through model checking of combined specifications. Even weak specifications can be useful to detect problems through inspection, such as goal models and resource models, for example, to analyze whether multiple home-automation components might compete for electricity or human attention. Nhlabatsi et al. provide a concise overview of common kinds of conflicts and different kinds of interactions that can help to guide an inspection.[4] In an ML setting, we may be able to reason to some degree about goals, resources, and maybe even weak specifications to detect certain kinds of interactions, especially if models interact through the environment and shared resources. However, given how hard it is to provide even weak specifications of ML components, leveraging system design might be a more promising solution.

Design for Interactions

Likely the most powerful strategy in addressing the feature-interaction problem is to prepare for the possibility of interactions as part of the system design, designing the system to 1) prevent certain interactions by isolating components and 2) prepare for resolving interactions when they eventually occur.

Isolation is a common design strategy to shield components from each other; an example is Android apps that cannot access each other's internal state or files. However, full isolation is rarely desired as components should often work together to achieve a goal. Designs will therefore typically allow specific kinds of interactions but often require the use of permitted and possibly controlled communication channels; the system can intercept, modify, or block messages at runtime if it serves the system specification (for example, not making phone calls without the corresponding app permissions).

Once an interaction has been detected, it can be resolved with additional coordination logic, such as one component overwriting the behavior of another. Anticipating the need for resolution as part of the system design will make it easier for developers or end users to resolve interactions when found. If we anticipate that interactions may happen, clever system designs can automatically select default resolutions, that is, design the system to handle unknown interactions gracefully.

In several domains, automatic domain-specific default-resolution mechanisms have been successful. For example, in self-driving cars, multiple components (such as cruise control, emergency braking, and maps) may provide possibly conflicting suggestions for the target speed, and, anticipating such conflicts, the system can be designed to resolve them by always picking the lowest (safest) suggested speed.[5] In Android, the system is designed to ask the user which of multiple apps should open a link if a conflict is detected at runtime. It is important to note that these strategies usually need to be designed for a specific problem, which is their strength and weakness at the same time: the system can resort to resolutions that leverage domain knowledge, but it is difficult to transfer this type of solution to other domains.

Systems with multiple ML components usually already naturally isolate the models and communicate through messages where resolution can be focused. Data fusion can be considered as a resolution strategy that is already common in many system architectures. The fusion strategy can either be defined manually (like picking the lowest speed) or learned with another ML model. In a sense, the ranking component of our initial image-captioning scenario can be seen as a domain-specific fusion mechanism that combines outputs of object detection and language models, trained on task-specific data. The ranking component is carrying out coordination logic to resolve interactions that has been learned itself! We suspect that there are many opportunities to think very deliberately about communication channels and formats to restrict the kind of data exposed from models and, more importantly, data fusion steps to define or even learn default resolutions for interactions.

Even though the lack of proper specifications may make ML models appear special compared to traditional software systems, there are many parallels around how to design a system to anticipate interactions with a healthy dose of system thinking. That

is, we need to focus on system design, not just the design of ML model architectures.

The key point is to realize that decomposition without perfect modularity is okay. Decomposition is a best-effort approach, but we need to anticipate interactions, prepare for them as part of the system design and development process, and make feature interactions a first-class concern to reason about them when they occur. We need to embrace design methods of managing interactions, and ML itself may provide a powerful tool for learning feature-interaction-resolution strategies.

At the same time, it is worth exploring what kind of specifications, however partial, we can provide for ML models. Describing goals of models or assumptions made in training data selection or modeling can help us to reason at least partially, about compositions. The recent adoption of more structured documentation, such as model cards[6] and data sheets,[7] can provide inspiration for providing structured, and possibly even machine-readable, information about models.

An extended version of this article can be found in Kästner et al. 2021.[8]

ACKNOWLEDGMENT

Sven Apel's work has been funded by the German Research Foundation, grant 389792660, as part of the Transregional Collaborative Research Center 248–Center for Perspicuous Computing; see https://perspicuous-computing.science/.

REFERENCES

1. H. Fang *et al.*, "From captions to visual concepts and back," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 1473–1482, doi: 10.1109/CVPR.2015.7298754.
2. B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, "On human intellect and machine failures: Troubleshooting integrative machine learning systems," in *Proc. 31st AAAI Conf. Artif. Intell.*, vol. 31, Feb. 2017, pp. 1017–1025.
3. Z. Peng, J. Yang, T.-H. (Peter) Chen, and L. Ma, "A first look at the integration of machine learning models in complex autonomous driving systems: A case study on Apollo," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2020, pp. 1240–1250, doi: 10.1145/3368089.3417063.
4. A. Nhlabatsi, R. Laney, and B. Nuseibeh, "Feature interaction: The security threat from within software systems," *Prog. Informat.*, no. 5, pp. 75–89, 2008, doi: 10.2201/NiiPi.2008.5.8. https://www.nii.ac.jp/pi/n5/5_75.html
5. C. Bocovich and J. M. Atlee, "Variable-specific resolutions for feature interactions," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Hong Kong, China, Nov. 2014, pp. 553–563, doi: 10.1145/2635868.2635927.
6. M. Mitchell *et al.*, "Model cards for model reporting," in *Proc. Conf. Fairness, Accountability, Transparency*, Atlanta, GA, USA, Jan. 2019, pp. 220–229.
7. T. Gebru *et al.*, "Datasheets for datasets," Mar. 23, 2018, arXiv:1803.09010v3.
8. C. Kästner, E. Kang, and S. Apel, "Feature interactions on steroids: On the composition of ML models," May 13, 2021, arXiv:210506449K.

SVEN APEL is the chair of software engineering at Saarland University and Saarland Informatics Campus, Saarbrücken, 66123, Germany. Contact him at apel@cs.uni-saarland.de.

CHRISTIAN KÄSTNER is an associate professor in the School of Computer Science at Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA. Contact him at kaestner@cs.cmu.edu.

EUNSUK KANG is an assistant professor in the School of Computer Science at Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA. Contact him at eunsukk@andrew.cmu.edu.

IEEE COMPUTER SOCIETY

www.computer.org

PURPOSE: Engaging professionals from all areas of computing, the IEEE Computer Society sets the standard for education and engagement that fuels global technological advancement. Through conferences, publications, and programs, IEEE CS empowers, guides, and shapes the future of its members, and the greater industry, enabling new opportunities to better serve our world.

OMBUDSMAN: Contact ombudsman@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

PUBLICATIONS AND ACTIVITIES

***Computer*:** The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals: The society publishes 12 magazines, 18 journals

Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.

Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.

Technical Communities: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The society holds more than 215 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society offers three software developer credentials.

AVAILABLE INFORMATION

To check membership status, report an address change, or obtain information, contact help@computer.org.

IEEE COMPUTER SOCIETY OFFICES

WASHINGTON, D.C.:
2001 L St., Ste. 700,
Washington, D.C. 20036-4928
Phone: +1 202 371 0101
Fax: +1 202 728 9614
Email: help@computer.org

LOS ALAMITOS:
10662 Los Vaqueros Cir.,
Los Alamitos, CA 90720
Phone: +1 714 821 8380
Email: help@computer.org

IEEE CS EXECUTIVE STAFF

Executive Director: Melissa Russell
Director, Governance & Associate Executive Director: Anne Marie Kelly
Director, Conference Operations: Silvia Ceballos
Director, Information Technology & Services: Sumit Kacker
Director, Marketing & Sales: Michelle Tubb
Director, Membership Development: Eric Berkowitz
Director, Periodicals & Special Projects: Robin Baldwin

IEEE CS EXECUTIVE COMMITTEE

President: Jyotika Athavale
President-Elect: Hironori Washizaki
Past President: Nita Patel
First VP: Grace A. Lewis
Second VP: Nils Aschenbruck
Secretary: Mrinal Karvir
Treasurer: Darren Galpin
VP, Member & Geographic Activities: Kwabena Boateng
VP, Professional & Educational Activities: Cyril Onwubiko
VP, Publications: Jaideep Vaidya
VP, Standards Activities: Edward Au
VP, Technical & Conference Activities: Terry Benzel
2023–2024 IEEE Division VIII Director: Leila De Floriani
2024–2025 IEEE Division V Director: Christina M. Schober
2024 IEEE Division V Director-Elect: Thomas M. Conte

IEEE CS BOARD OF GOVERNORS

Term Expiring 2024:
Saurabh Bagchi, Charles (Chuck) Hansen, Carlos E. Jimenez-Gomez, Daniel S. Katz, Shixia Liu, Cyril Onwubiko

Term Expiring 2025:
İlkay Altintaş, Mike Hinchey, Joaquim Jorge, Rick Kazman, Carolyn McGregor, Andrew Seely

Term Expiring 2026:
Megha Ben, Terry Benzel, Mrinal Karvir, Andreas Reinhardt, Deborah Silver, Yoshiko Yasuda

IEEE EXECUTIVE STAFF

Executive Director and COO: Sophia Muirhead
Interim General Counsel and Chief Compliance Officer: Ahsaki Benion
Chief Human Resources Officer: Cheri N. Collins Wideman
Managing Director, IEEE-USA: Russell Harrison
Chief Marketing Officer: Karen L. Hawkins
Managing Director, Publications: Steven Heffner
Staff Executive, Corporate Activities: Donna Hourican
Managing Director, Member and Geographic Activities: Cecelia Jankowski
Chief of Staff to the Executive Director: Kelly Lorne
Managing Director, Educational Activities: Jamie Moesch
IEEE Standards Association Managing Director: Alpesh Shah
Chief Financial Officer: Thomas Siegert
Chief Information Digital Officer: Jeff Strohschein
Managing Director, Conferences, Events, and Experiences: Marie Hunter
Managing Director, Technical Activities: Mojdeh Bahar

IEEE OFFICERS

President & CEO: Thomas M. Coughlin
President-Elect: Kathleen Kramer
Past President: Saifur Rahman
Director & Secretary: Forrest D. Wright
Director & Treasurer: Gerardo Barbosa
Director & VP, Publication Services & Products: Sergio Benedetto
Director & VP, Educational Activities: Rabab Kreidieh Ward
Director & VP, Membership and Geographic Activities: Deepak Mathur
Director & President, Standards Association: James E. Matthews III
Director & VP, Technical Activities: Manfred J. Schindler
Director & President, IEEE-USA: Keith A. Moore

IEEE

revised 29 October 2024

EDITOR: Qun Li, liqun@cs.wm.edu

DEPARTMENT:
EMERGING INTERNET TECHNOLOGIES

This article originally appeared in
IEEE Internet Computing
vol. 28, no. 2, 2024

Distributed Quantum Machine Learning: Federated and Model-Parallel Approaches

Jindi Wu, Tianjie Hu, and Qun Li, *William & Mary, Williamsburg, VA, 23187, USA*

In this article, we explore two types of distributed quantum machine learning (DQML) methodologies: quantum federated learning and quantum model-parallel learning. We discuss the challenges encountered in DQML, propose potential solutions, and highlight future research directions in this rapidly evolving field. Additionally, we implement two solutions tailored to the two types of DQML, aiming to enhance the reliability of the computing process. Our results show the potential of DQML in the current Noisy Intermediate-Scale Quantum era.

Distributed quantum machine learning (DQML) is an emerging field that combines quantum machine learning (QML) with distributed computing. QML utilizes distinctive quantum mechanics properties, like superposition and entanglement, to potentially enhance traditional machine learning algorithms. However, the current stage of quantum computing technology, often referred to as the *Noisy Intermediate-Scale Quantum* (*NISQ*) *era*, imposes limitations on the size and complexity of QML models implemented with variational quantum circuits (VQCs). These constraints can restrict the performance and applicability of QML methods. To address these challenges, integrating QML with distributed computing has emerged as a strategic and forward-looking approach. This hybrid approach aims to overcome the individual limitations of quantum devices by harnessing distributed quantum computing's power to manage and process complex tasks across multiple quantum computing nodes, thus amplifying the capabilities of QML models.

DQML holds promise for a diverse array of applications, especially those demanding complex computations that can utilize the distinct advantages of quantum computing. For instance, DQML can significantly enhance molecular simulation and drug discovery by enabling more efficient modeling of molecular interactions. Furthermore, DQML can be highly beneficial in fields such as financial modeling and medical image processing, which require the efficient handling of large datasets while preserving local privacy. The current cost of training even a modest QML model is quite high. For instance, for a QML circuit of a quantum convolutional neural network (CNN) that utilizes eight qubits and includes approximately 150 trainable parameters on a training set of 500 training instances, it may take approximately $20,000 to train a model on current quantum computers. QML and DQML are generally not practical at present, but in the future, we believe that they have the potential to perform better than classical computers.

The DQML approaches, while promising, confront unique and significant challenges distinct from those in classical distributed machine learning. In this article, we focus on two specific areas within the realm of DQML: quantum federated learning (QFL) and quantum model-parallel learning.

QFL is a case of distributed learning with data parallelism.[1] In this approach, multiple quantum computing nodes, each with its own local dataset, collaborate to train a shared QML model. Each node processes its own data independently, ensuring privacy and security by not transferring raw data between nodes. Instead, only model updates are communicated across the network. The updates generated by each computing node are collectively aggregated, effectively synthesizing the insights learned from distinct local datasets. These consolidated updates are then redistributed to each node. This approach effectively combines computational power and data from diverse sources, enhancing the learning process while maintaining data confidentiality, a key aspect in scenarios where data privacy is crucial.

1089-7801 © 2024 IEEE
Digital Object Identifier 10.1109/MIC.2024.3361288
Date of current version 16 April 2024.

Quantum model-parallel learning is a method of distributed learning characterized by the paradigm of model parallelism. The model parallelism in quantum model-parallel learning is particularly advantageous for handling large-scale QML models that exceed the computational and memory capacities of individual quantum nodes.[2] In this framework, the QML model is partitioned into submodels, distributed across multiple quantum computing nodes, with each node handling a distinct submodel of the entire model. The computing nodes process their assigned partition of data and compute intermediate results in parallel, which are then communicated to other nodes or a central coordinator for generating the final outcome. The unique aspect of quantum model-parallel learning is its ability to leverage the individual computational strength of each node while jointly contributing to the construction of a comprehensive model. Through the distribution of computational workload and the facilitation of parallel processing, quantum model-parallel learning opens up new possibilities for addressing more intricate QML tasks.

QFL and quantum model-parallel learning each present unique benefits for specific QML tasks, yet they also face several challenges. In our study, we conduct a thorough examination of these challenges and propose potential solutions to mitigate these issues. These challenges include quantum errors, scalability, communication, and hardware diversity. Moreover, we implement two specific solutions to enhance the reliability of the two types of DQML approaches. In particular, a key challenge in QFL arises from the global model being susceptible to a wide array of local errors. Because the local models are trained under device-specific errors, the aggregation of these error-impacted local models can significantly compromise the efficacy of the overall QML model. To address this issue, we minimize the impact of errors on local models as much as possible. For quantum model-parallel learning, devising a strategy to partition a large-scale QML model without compromising its functionality and reliability presents a considerable challenge. We address these issues by carefully designing the submodels of the QML model, taking into account both the architecture and the reliability of the available quantum computing nodes. Our results demonstrate the significant potential of DQML in the current NISQ era.

BACKGROUND

Quantum Computing

Quantum computing is a revolutionary paradigm of computation that leverages the principles of quantum mechanics to perform complex calculations.

Qubit

A qubit, which carries quantum information, is the fundamental building block of quantum computing. Due to its unique property of superposition, it can exist in multiple states simultaneously. The state of a single qubit can be represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$. Here, $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit as $|0\rangle$ and $|1\rangle$, respectively, i.e., $|\alpha|^2 + |\beta|^2 = 1$. Superposition empowers quantum computers with increased computational power, allowing them to explore and process numerous potential solutions to a problem in parallel. Furthermore, entanglement allows two or more qubits to establish correlations in which changes in the state of one entangled qubit instantaneously impact the state of the other, irrespective of the spatial separation between them.

Quantum Gates

Quantum computation involves manipulating qubit states with quantum gates, represented by unitary matrices denoted as U (satisfying the conditions $U^\dagger U = UU^\dagger = I$). The fundamental quantum gates include the Pauli-X, Pauli-Y, Pauli-Z, Hadamard (H), and controlled-X (CNOT) gates. These gates serve as foundational building blocks for assembling more intricate quantum algorithms. Among these gates, the CNOT gate is a two-qubit gate that establishes correlations between qubits. In addition, quantum rotation gates, such as RX(θ), RY(θ), and RZ(θ), provide precise manipulation of quantum states through an angle θ.

Quantum Errors

Quantum computing is error-prone due to the inherent instability of the quantum system and the immature manufacturing of quantum computers. A quantum circuit (program) comprises a set of gates that manipulate quantum data, and the processed data are acquired through quantum measurement operations. Each operation in this process can introduce errors into the quantum system, potentially resulting in inaccuracies in the quantum circuit's outcomes. In particular, errors in quantum computing arise from various sources, including quantum state preparation, quantum gates, measurement, and crosstalk. Moreover, the state of qubits can be influenced by errors due to decoherence and dephasing.

QML

VQCs are a widely adopted approach for constructing QML models. These models are tailored for executing particular tasks, including optimization and classification, by harnessing the capabilities of quantum circuits and integrating them with classical optimization methods.

Ansatz

A VQC, implementing a QML model, is composed of three parts: 1) the *encoding unit* is responsible for converting classical data into quantum data, which can be processed by the quantum circuit. Popular encoding techniques include angle encoding and amplitude encoding. 2) The *variational block* constitutes the core of the VQC, featuring a sequence of parameterized quantum rotation gates. The rotation gates equipped with trainable parameters function analogously to neurons in classical neural networks. The entire variational block is organized into layers, and a block consisting of L layers can be represented as

$$U(\theta) = U_L(\theta_L)U_{L-1}(\theta_{L-1})\ldots U_1(\theta_1) \quad (1)$$

(3) The *measurement unit* involves the measurement of one or more qubits to obtain the outcome corresponding to the input data with observable O.

Optimization

QML employs a hybrid quantum–classical approach to train the model. Once a model on a quantum computer processes specific input data, the classical computer takes over to optimize the model's parameters, guided by a predefined cost function

$$C = \langle 0|U^{\dagger}(\theta)OU(\theta)|0\rangle. \quad (2)$$

This optimization can be executed through various methods, including gradient-based approaches like the stochastic gradient descent algorithm or non-gradient-based techniques, such as the parameter-shift algorithm. The entire training procedure in QML involves repeating the optimization step until the parameters of the model converge.

QFL

QFL is a sophisticated process that blends the principles of quantum computing with federated learning's distributed model training approach. The QFL system consists of a central server and multiple quantum computing nodes, each allocated to different clients. In this configuration, clients maintain their data locally on their assigned quantum computing nodes. The objective is to collaboratively train a QML model that benefits from the aggregated data across all nodes, while ensuring that each client's private information remains unshared and secure.[3] Formally, the QFL procedure for a VQC-based QML model can be represented as follows:

1) *Initialization*: Each client initiates a local model $U(\theta)$ with the same ansatz and parameters on their respective local devices.
2) *Local training*: On the i-th quantum computing node, the client trains its individual model $U(\theta_i)$ using the private dataset with several update steps.
3) *Local model submission*: Each participating client submits their updated local parameters θ to the central server.
4) *Model aggregation*: The central server aggregates the received local parameters from clients and integrates them into a global model represented as $\theta' = \sum_{i=1}^{N} n_i\theta_i$, where N is the number of clients involved in this procedure, and n_i denotes the corresponding weight of the i-th client.
5) *Global model distribution*: The updated global parameters θ' are distributed to clients for the subsequent round of training.
6) *Iteration*: Repeat steps 2–5 for multiple rounds until convergence or the desired model performance is achieved.

Quantum Model-Parallel Learning

Quantum model parallelism is a distributed approach in QML, especially for large-scale QML models. In this method, a complex quantum model is partitioned into submodels, which are then distributed across multiple quantum computing nodes.

In such a large-scale QML model, the overall unitary operation $U(\theta)$ is partitioned into K submodels $\{U_1(\theta_1), U_2(\theta_2), \ldots, U_k(\theta_k)\}$. Each submodel, denoted as $U_i(\theta_i)$, is allocated to a distinct quantum computing node, with i ranging from 1 to K. Similarly, the input data D are divided into subsets $\{D_1, D_2, \ldots, D_n\}$, corresponding to the submodels $\{U_1(\theta_1), U_2(\theta_2), \ldots, U_n(\theta_n)\}$ within the input layer of the QML model, where $n \leq K$. Starting with the submodels in the input layer, each submodel within the same layer operates independently. The output generated by a submodel is then transmitted to the submodel in the subsequent layer, using either classical or quantum communication channels. The submodel in the final layer of the QML model generates the ultimate output that corresponds to the specific input data. This process guarantees the seamless flow of information and computation across the distributed quantum system.

RELATED WORK

The design of the QFL model can be varied to align with the specifications of the available quantum devices and data. For instance, a hybrid quantum–classical transfer learning model is developed and deployed on

each computing node for federated learning.[4] This model utilizes a pretrained classical model to compress classical input data into a small size, followed by leveraging a VQC-based QML model for decision making. In addition, a QFL approach based on quantum data is proposed.[5] Furthermore, beyond a VQC, the QFL model can be constructed using various methods. For example, the model may consist of several layers that incorporate a differing number of qubits.[6,7]

A scalable QML is an instance of quantum model-parallel learning.[2] The approach divides a large-scale QML model into two distinct layers. The first layer comprises individual subcircuits, each designed to learn from segments of a training instance. The second layer then aggregates these intermediate results, enabling further exploration of the correlations between data segments. In addition, the approach can be implemented with a single quantum computing node due to the independence between the submodels of the large QML model. Similarly, a quanvolutional neural network achieves scalability by constructing quantum convolutional kernels that emulate the functionality of the classical convolutional kernel used in classical CNNs.[8] This quantum kernel slides over the input data to extract abstract features, mirroring the process in classical CNNs.

CHALLENGES

Although QFL and quantum model-parallel learning offer benefits by integrating distributed computing with QML, such as enhanced privacy protection and improved model scalability, there are still several challenges that need to be addressed.

Quantum Errors

Quantum errors present a significant challenge in distributed quantum computing systems as the errors accumulate through noisy operations and vary over time in an unpredictable manner.

During the training process of a QML model, parameter updates serve a dual purpose: they learn from the training dataset and simultaneously capture the error pattern to mitigate the impact of quantum errors. Nevertheless, QML models cannot completely eliminate the influence of these errors. The error pattern captured by a QML model is influenced by both the ansatz of the model and the specific quantum device. When the level of error is sufficiently high, the reliability of the model can be significantly compromised. Furthermore, error patterns vary across different quantum devices. Consequently, in QFL approaches, the aggregated model suffers from varying errors that originate from multiple computing nodes, which can lead to suboptimal performance. Additionally, quantum model-parallel learning produces final outcomes based on noisy intermediate results. As the complexity of the model increases, errors accumulate, consequently diminishing the fidelity of the outcomes.

Error-mitigation strategies such as circuit optimization and result postprocessing can be utilized to effectively improve the reliability of each computing node. For instance, the solution presented in the subsequent section aims to alleviate the influence of various errors in the overall model. It achieves this by specifically reducing the error impact on each computing node through the application of circuit-optimization techniques. However, a comprehensive solution for error mitigation has not yet been fully realized. In addition, quantum errors continuously change in unpredictable ways. Therefore, a model trained to mitigate errors during the training phase may not be effective against errors encountered during testing. A potential solution to address fluctuating quantum errors involves continuously updating the model to accommodate current quantum error conditions, but this method may lead to considerable overhead. Alternatively, the model can be trained while accounting for shifted errors.[9]

Scalability

Although distributed approaches can enhance the scalability of QML tasks to a certain extent, scalability remains a significant challenge in the field. For QFL, the scale of the QML model deployed on an individual quantum computer is constrained by the limited quantum computing resources. The circuit width, for instance, is confined by the number of qubits available on the quantum hardware. These qubits, serving as the register for data processing, limit the data's dimensionality that the model can process. To mitigate this, a common strategy is to compress data to fit the available qubit capacity. Additionally, various encoding methods have been proposed to represent data within a limited number of qubits. However, these methods can potentially diminish the utility of data for specific tasks or introduce significant overhead and errors, posing a tradeoff between data representation efficiency and the fidelity of the information processed. Moreover, the depth of the circuit is constrained due to the accumulation of errors and the instability of the quantum system. Therefore, for QFL tasks, the scale and complexity of local models remain limited. This limitation can subsequently constrain the performance and effectiveness of these local models.

Within quantum model-parallel learning, the scale of the QML model is dictated by the specifications of

available quantum devices and the requirements of circuit partitioning. The capacity of each individual quantum computing node plays a critical role in determining how the QML model should be partitioned. Excessive partitioning of the QML model into numerous small segments can compromise its integrity. Hence, achieving an optimal balance in the partitioning process is crucial for the effective functioning of a large-scale QML model.

Furthermore, as the scale of the QML model increases, the challenge of the barren plateau emerges.[10] This phenomenon, encountered in the training of large-scale quantum neural networks, is characterized by the gradient of the cost function becoming extremely small. As a result, it becomes inefficient to train the QML model. To tackle the barren plateau problem, recommended strategies include careful parameter initialization and the adoption of problem-specific ansatzes. Nevertheless, there remains a need for more advanced solutions to effectively address this issue.

Communication

The communication channels, both classical and quantum, between quantum computing nodes in a distributed QML system present challenges. In quantum model-parallel learning, the outputs of subcircuits, obtained through measurements, are transmitted to the central node via classical communication channels. However, these measurement results may not entirely represent all the information of the intermediate states as measurements are typically made on a single basis, which can result in the loss of significant information. One potential solution to this issue is the classical shadow technique, which involves performing a series of measurements on a quantum state and using the results to create a "shadow" or a classical approximation of the state. However, the overhead will rise exponentially as the number of qubits increases. An effective strategy for reducing measurement overhead is to reconstruct the complete state of a single qubit and fully utilize the information of this single qubit to scale up the size of the problem that the model can solve.[11] Alternatively, the processed quantum states at the nodes could be transmitted through quantum communication channels. Yet, the development of stable and reliable quantum communication channels, essential for a functional quantum network, is still in its nascent stages. Overcoming these communication hurdles is crucial for the efficient operation of distributed QML systems.

Hardware Diversity

Hardware diversity, particularly the various techniques used to implement qubits, poses a significant challenge in distributed QML. Different quantum systems use different physical implementations for qubits, such as trapped ions, superconducting circuits, or topological qubits, each with unique characteristics and limitations. This diversity poses a challenge when attempting to create a standardized QML model capable of running on various quantum hardware platforms. Furthermore, the interoperability between diverse quantum systems becomes complex due to differences in their control and measurement protocols. This necessitates the development of adaptable QML algorithms that can be efficiently transpiled and optimized for various hardware architectures. In essence, hardware diversity in distributed QML demands a robust and flexible approach to algorithm design and system optimization to ensure effective and reliable performance across heterogeneous quantum computing platforms.

APPROACHES

In this section, we introduce two strategies designed to enhance the reliability of QFL and quantum model-parallel learning by applying an error-mitigation technique. These techniques primarily involve minimizing gate errors by optimizing quantum circuit design. In particular, we design the QML models to minimize the necessity for SWAP gates according to the qubit topology on the machines. These gates are inserted into the circuit to enable the application of multiqubit gates to nonadjacent qubits. Moreover, we prioritize choosing qubits and gates with lower error rates to further reduce the error rate of the QML model.

Based on the distinctive characteristics of QFL and quantum model-parallel learning, various strategies can be employed to design the QML model. In the context of QFL, all participating nodes train the QML model using the same logical ansatz. Nevertheless, the transpiled circuits of the model vary and capture distinct error patterns, adversely affecting the performance of the aggregated model. In contrast, in the quantum model-parallel learning approach, a large-scale QML model is partitioned into several submodels, each potentially with a different ansatz. The overall performance of the entire model depends on the reliability of each submodel. Therefore, our approach to QFL centers around designing the QML model while taking into account the qubit topology and quality of all the participating computing nodes. On the contrary, for quantum model-parallel learning, we design submodels of the entire QML model by separately considering the qubit topology and quality of each node involved.

QFL

In QFL, our strategy begins with a careful selection process, where we carefully select a subset of qubits from each participating node. These chosen subsets are characterized by similar qubit connectivity and relatively lower error rates. Subsequently, we construct the QML model to adapt the qubit connectivity of these selected subsets. This method guarantees that the transpiled QML models require the minimum number of SWAP gate insertions across all nodes. By doing so, we aim to minimize the impact of errors on each local model and, consequently, reduce the disturbance in the aggregated model caused by differing local error patterns. This method provides a more unified and error-resilient approach to QFL, optimizing both individual and aggregated model performance.

For concreteness, given a QML task focused on a binary classification task that categorizes images composed of four pixels [as shown in Figure 1(a)], we construct a QML model that incorporates four qubits. There are several open source tools available for implementing QML. For instance, Qiskit by IBM, PennyLane by Xanadu, and TensorFlow Quantum by Google. These tools provide support for QML, but currently, there are no tools specifically designed for DQML. In our research, we created a simulated federated learning environment utilizing Qiskit on the IBM Quantum platform. For this setup, three quantum computers, *ibm_lagos*, *ibm_perth*, and *ibm_nairobi*, were selected as individual quantum computing nodes to train local QML models. These three computers have identical qubit topologies, as depicted in Figure 1(b). From each computer, we select a subset of qubits within which three qubits are adjacent to a central qubit, and these are chosen for their lower error rates, as illustrated in the green box in Figure 1(b). Then we construct the QML model as depicted in Figure 1(c). This specific ansatz ensures that no noisy SWAP gates are required during the circuit transpilation, thereby preserving the reliability of the circuit. This QML model is then deployed on nodes, selecting the highest-quality qubit set that meets this topology requirement. This strategy not only enhances the fidelity of the outcomes but also minimizes the impact of errors on the trained model.

We assess the effectiveness of our proposed approach by conducting the training phase on a simulator that incorporates the noise models of specific quantum devices. Subsequently, we test the trained model on actual quantum devices. The trained model attained accuracies of 93%, 96%, and 96% on the three respective nodes involved, as illustrated in Figure 1(d). For a baseline comparison, the accuracies of a model trained without accounting for the specific architecture of the quantum devices across nodes are 90%, 86%, and 88%. The observations indicate that the model with circuit optimization using our approach outperforms the baseline model without circuit optimization in terms of accuracy across all computing nodes, highlighting the efficacy of our method. Moreover, the lower accuracy achieved by the baseline indicates that although the trained QML model can offset errors to a certain degree, its effectiveness is still notably diminished by these errors. This is mainly because the transpiled circuit of the model incurs a large volume of errors due to the extensive insertion of SWAP gates, which are intricately eliminated in our proposed approach.

Quantum Model-Parallel Learning

Typically, the width of a QML model is determined by the size of the data to be processed. For instance, to encode a data instance consisting of eight components, a QML model would require eight qubits using the angle-encoding method. In general, a large-scale

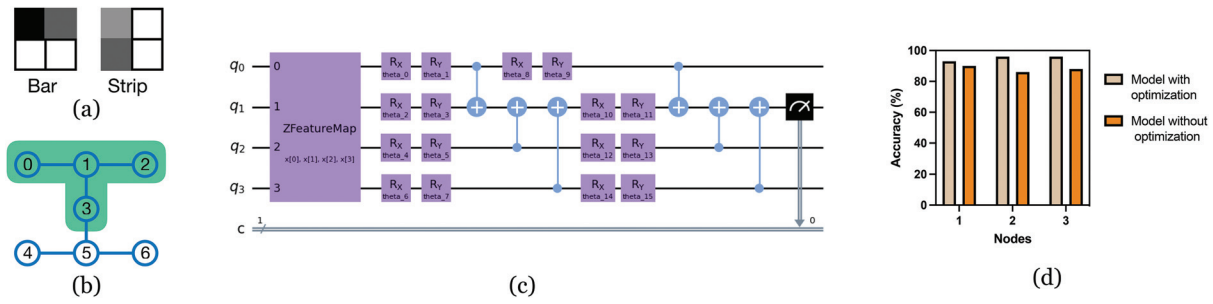


FIGURE 1. QFL with error mitigation. (a) The dataset comprises two classes, with each image containing four pixels. (b) The qubit topology of quantum computing nodes involved in the QFL environment. (c) The specifically designed QML ansatz, tailored with the qubit topology to eliminate the need for inserting SWAP gates. (d) The test accuracy of the model on actual quantum devices.

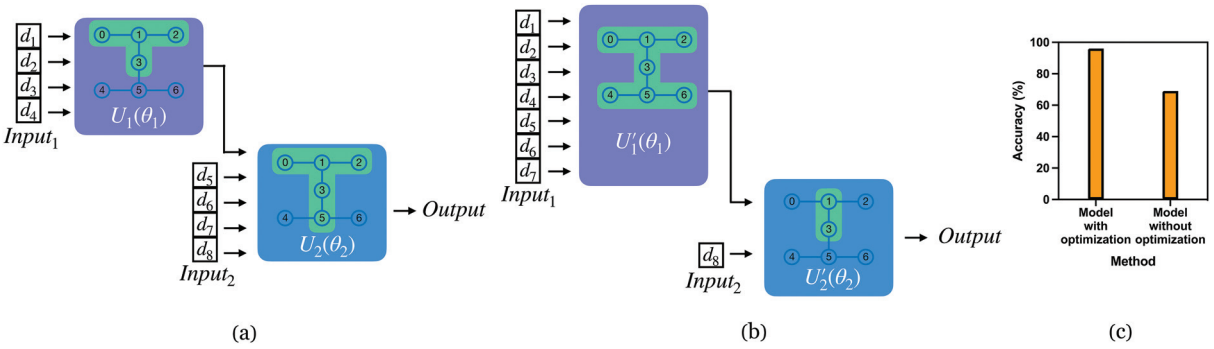


FIGURE 2. Quantum model-parallel learning with error mitigation. (a) A QML model is designed with two subcircuits, each constructed with consideration of the architecture and reliability of the involved nodes. (b) A QML model is constructed and then partitioned based on the number of qubits available on the nodes. (c) The test accuracy of the models implemented with different approaches.

QML model is first built and then divided into submodels according to the capacities of the available quantum computing nodes.

This approach frequently necessitates inserting additional gates and can compromise the correlation within data instances, leading to a significant reduction in the model's reliability. Conversely, our approach entails designing the submodels while considering both the qubit topology and the quality of resources available on the quantum computing nodes. These submodels are then integrated into a complete QML model. This strategy aims to preserve the fidelity of the output of each submodel, thereby enhancing the overall performance of the model.

For example, we consider a QML-based classification task focused on handwritten digits 0 and 1, where images from the Modified National Institute of Standards and Technology dataset are downscaled to eight components using the principal component analysis method. The quantum system in this scenario comprises two quantum computing nodes: *ibm_lagos* and *ibm_perth*, with their qubit topology illustrated in Figure 2. Based on the characteristics of these two quantum computing nodes, we design a QML model consisting of two submodels: the first subcircuit, consisting of four qubits, processes the first half of the image, while the second subcircuit, encompassing five qubits, handles the second half of the image and integrates the processed results from the first half. The complete QML model is depicted in Figure 2(a). As a baseline for comparison, another method implemented for this task involves partitioning the QML model based solely on the resource capacity of the computing nodes. For processing data instances with eight values, the first subcircuit is constructed using all the qubits (seven qubits) of one node to process seven components of the data instance. Meanwhile, the second subcircuit utilizes two qubits to process the remaining component and to integrate the intermediate results, as shown in Figure 2(b).

In Figure 2(c), we depict the accuracy of two QML models implemented using distinct methods: one with circuit optimization achieving 96% accuracy and another without optimization achieving 69% accuracy. It's clear that the optimized model exhibits significantly superior accuracy compared to the baseline model without circuit optimization. There are three main reasons for the observed performance difference. First, our proposed method for designing subcircuits using circuit-optimization techniques eliminates the need for noisy SWAP gates, thereby reducing the overall error rate. Second, data instances are partitioned evenly in our approach. This method preserves coherence within each data segment and ensures that the final result is evenly influenced by both parts. In contrast, the baseline method partitions the data unevenly, leading to a biased final result. Third, although the baseline method can also be optimized to minimize additional SWAP gates, the first submodel in this approach still includes many gates necessary for the model's functioning. Given that the fidelity of a quantum circuit exponentially decreases with the increasing number of gates, the larger size of the subcircuit in the baseline method likely results in significantly lower fidelity. Therefore, carefully partitioning the model across available computing nodes is advantageous for maintaining higher fidelity.

CONCLUSION

In this article, we explore two methodologies within DQML: QFL and quantum model-parallel learning.

A significant challenge in distributed QML is managing quantum errors. To address this, we introduce an error-resilient approach to QFL and quantum model-parallel learning, employing an ansatz construction based on qubit topology. Empirical evaluations highlight the effectiveness of our solutions, underscoring the potential of distributed QML methodologies.

REFERENCES

1. C. Ren et al., "Towards quantum federated learning," 2023, *arXiv:2306.09912*.
2. J. Wu, Z. Tao, and Q. Li, "wpScalable quantum neural networks for classification," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 38–48, doi: 10.1109/QCE53715.2022.00022.
3. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.
4. S. Y.-C. Chen and S. Yoo, "Federated quantum machine learning," *Entropy*, vol. 23, no. 4, pp. 460–473, 2021, doi: 10.3390/e23040460.
5. M. Chehimi and W. Saad, "Quantum federated learning with quantum data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 8617–8621, doi: 10.1109/ICASSP43922.2022.9746622.
6. Q. Xia and Q. Li, "QuantumFed: A federated learning framework for collaborative quantum training," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–6, doi: 10.1109/GLOBECOM46510.2021.9685012.
7. Q. Xia, Z. Tao, and Q. Li, "Defending against byzantine attacks in quantum federated learning," in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 145–152, doi: 10.1109/MSN53354.2021.00035.
8. M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," *Quantum Mach. Intell.*, vol. 2, no. 1, p. 2, 2020, doi: 10.1007/s42484-020-00012-y.
9. Z. He, B. Peng, Y. Alexeev, and Z. Zhang, "Distributionally robust variational quantum algorithms with shifted noise," 2023, *arXiv:2308.14935*.
10. J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 4812, doi: 10.1038/s41467-018-07090-4.
11. J. Wu, T. Hu, and Q. Li, "MORE: Measurement and correlation based variational quantum circuit for multi-classification," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 208–218, doi: 10.1109/QCE57702.2023.00031.

JINDI WU is a Ph.D. student focusing on quantum error mitigation and quantum machine learning in the Department of Computer Science, William & Mary, Williamsburg, VA, 23187, USA. Contact her at jwu21@wm.edu.

TIANJIE HU is a Ph.D. student focusing on quantum error correction in the Department of Computer Science, William & Mary, Williamsburg, VA, 23187, USA. Contact him at thu04@wm.edu.

QUN LI is a professor with the Department of Computer Science, William & Mary, Williamsburg, VA, 23187, USA. Contact him at liqun@cs.wm.edu.

EDITOR: Joseph Williams, Infosys Consulting, joseph.williams@infosys.com

DEPARTMENT: COMPUTING ARCHITECTURES

This article originally appeared in
Computer
vol. 56, no. 2, 2023

The Quantum Cybersecurity Threat May Arrive Sooner Than You Think

Pete Ford, *QuSecure*

Recent articles predict that the quantum computing market will expand 500% by 2028. Billions of dollars are currently pouring into the quantum computer industry to build the first fault-tolerant quantum computers.

Fault-tolerant quantum computers (FTQCs) will be a gold mine for humanity and business. Recent articles predict that the quantum computing market will expand 500% by 2028, so we know there's a quantum race building where quantum computers could solve some of our most pressing and important problems. As a result, billions of dollars are currently pouring into the quantum computer industry to build the first FTQC. However, there is a dark side to this wonderful invention as quantum computers have been mathematically proven to have the ability to break the cybersecurity that most of the world uses. In this article, I discuss a few ways quantum computers may be used for theft or disruption sooner than we might think.

CRYPTOGRAPHICALLY RELEVANT QUANTUM COMPUTERS

When a quantum computer is powerful enough to break our current encryption, most people have no idea how the balance of power in the world could shift. Think about what could happen if an adversarial nation state had access to national secrets, financial systems, and data, and treasure troves of personal data. A quantum computer with this power is sometimes called a *cryptographically relevant quantum computer* (*CRQC*), and in an ever-increasing digital world, we can all understand the gravity of such an event.

Digital Object Identifier 10.1109/MC.2022.3227657
Date of current version: 8 February 2023

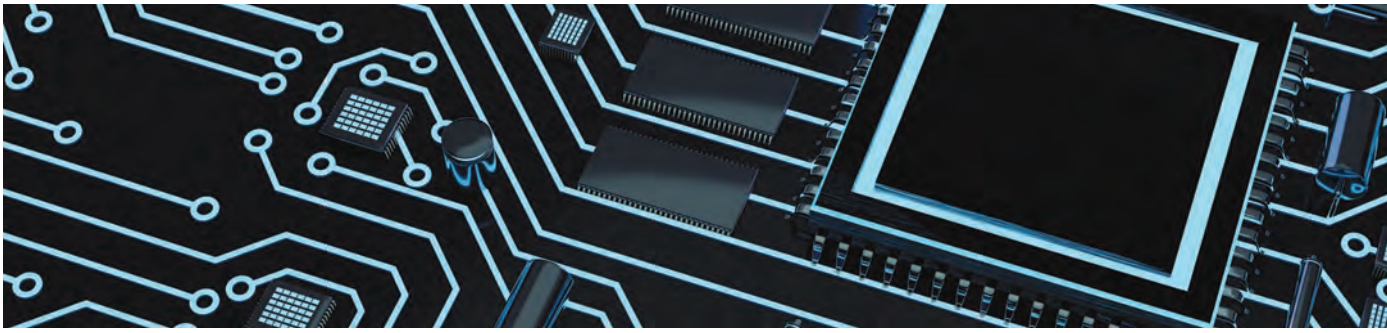
Recall Y2K, when we had to upgrade many of the computing systems in the world to prevent potential computer errors related to the formatting and storage of calendar data for dates in and after the year 2000. Now we have Q-Day or Y2Q, where the entire world needs to change the encryption so we can safely use the Internet.

Many ask, "When will a CRQC be available?" Like searching for gold in a mine, the answer[1] is more opaque than many admit, with views clouded by technical jargon and very complex descriptions.

If you have not heard of Q-Day, you are not alone; it is the time when FTQCs will be available and can break current asymmetric-key encryption (the encryption the entire world uses) and become CRQCs. Y2K and Q-Day have several things in common. They are/were both global computing efforts, cost a lot of money, and have a time constraint. Although we knew about Y2K seven years beforehand, our awareness of Q-Day's timing is opaque, yet we do know that billions are being spent to build an FTQC.[2]

The pending upgrade to postquantum-resilient cybersecurity will be the largest upgrade in information technology history. Current World Economic Forum predictions[3] indicate that more than 20 billion devices will need to be upgraded or replaced to operate safely and successfully in a postquantum world to keep data safe. Q-Day will arrive, but we don't know the exact timing like we did for Y2K. The first strategic crisis of the 21st century is already underway, and thankfully, we are aware of it and can act now.

 Published by the IEEE Computer Society



The first well-known encryption-breaking proof came from Peter Shor in 1994. His algorithm[4] showed that an FTQC with 4,099 coherent qubits could break the public-key encryption standards we use today. This algorithm and coherent qubit counting have been a focal point of Q-Day timing. Deep technical debates on FTQC gains, daily physics breakthroughs, and technology funding are riddled with scientific details. Certainly, an FTQC with 4,099 qubits will be one of the big treasures buried in the quantum mines. National governments are the main sources paying for this quantum breakthrough as national security and strategic surprise are at stake. India, Russia, Japan, the European Union, and Australia have substantial programs. However, the two main countries investing in this quantum race[5] are China and the United States. China is leading by far, spending an estimated US$13 billion since 2015. U.S. spending during the same period is estimated at US$2.1 billion.

NOISY INTERMEDIATE SCALE QUANTUM COMPUTERS

However, there are other ways to get to a CRQC without needing 4,099 qubits. Noisy intermediate scale quantum computers (NISQs) and hybrid classic-quantum computing[6] are quietly making strong strides toward solving nonpolynomial hard math in the near term. As these quantum techniques advance, we may find quantum-approximate optimization algorithms[7] or new hybrid classic quantum solutions that can break current encryption before a large FTQC or CRQC is online. As an example, Zapata Computing used a heuristic algorithm called *variational quantum factoring*[8] to show that a quantum hybrid solution has promise. This technique of blending classical and quantum computing allowed better tradeoffs between available quantum resources (primarily quantum circuit depth and solution accuracy). These smart tradeoffs leveraging readily available quantum computing resources could be a workaround of the NISQ problem. Although the time until Q-Day is still unknown, these advances should alert us to be ready for a strategic surprise.

Other quantum scientists[9] are exploring tradeoffs between circuit depth-efficient and qubit-efficient models as a mixed approach to encoding schemes that are easier to deploy on a NISQ machine. Other scientists are reducing errors in NISQ machines by making them more visible[10] and easier to work around. These new techniques and algorithms offer faster paths to solve the nonpolynomial hard math problems protecting our encryption. This is another threat to breaking asymmetric encryption, even before an FTQC is ready. Studying these developments is important as the race to Q-Day speeds up, and the threat could become a reality sooner than we realize.

THE PENDING UPGRADE TO POSTQUANTUM-RESILIENT CYBERSECURITY WILL BE THE LARGEST UPGRADE IN INFORMATION TECHNOLOGY HISTORY.

These new options using NISQ's and hybrid compilers to break encrypted data shed light on other possible nation-state plans. As competitive nation states spend many billions of dollars on quantum computing, not all the funding in these nation states will be spent on developing FTQC's and quantum hardware. Obviously, some funds will be spent to rapidly decrypt data that have already been stolen. When a well-funded group[11] of smart scientists and programmers pull in the same direction, they can accomplish surprising results. We should be aware of others developing these or similar techniques to accomplish devastating results. It is critical for us to prepare now before a dramatic strategic surprise makes it too late.

Commercial and government experts are constantly reviewing the traction gained toward a CRQC using current scientific methods. For reluctant bystanders, the future will be troublesome as it is volatile, uncertain, complex, and ambiguous if we rely on rigid certainty for encryption standards.

There are solutions to the CRQC problem, which can now be implemented. Postquantum cybersecurity (PQC) refers to a new form of cybersecurity that is resistant to quantum computing attacks from CRQC and the nearer-term approaches. Unlike existing encryption that will be broken by powerful quantum computers, PQC uses specific quantum-resilient algorithms[12] in combination with other features to ensure that enterprise and government organizations will have protections in place to withstand these attacks, even before CRQCs arrive. For PQC to be deployed quickly and effectively, building in backward-compatible PQC software solutions that protect our encryption on existing hardware allows us to react quickly as quantum computing advances happen.

Designing future systems with this resilience, and not just ruggedness for success, should be our standard. This will help in the opaque nature of the race upon us. Time is never on our side as defenders. FTQCs or nearer-term alternative solutions may come online well before we expect. With a resilient offensive design and mindset, we should embrace crypto agility as our postquantum information goals are not only achievable but are a race we can win if we put the right measures in place.

REFERENCES

1. "QRP3–Quantum race: Sprint, marathon, false start?" Project Q Sydney. Accessed: Oct. 5, 2022. [Online]. Available: https://projectqsydney.com/qrp3-quantum-race-sprint-marathon-false-start/
2. "Global quantum computing market." SkyQuestt. Accessed: Sep. 28, 2022. [Online]. Available: https://skyquestt.com/sample-request/quantum-computing-market
3. "Transitioning to a quantum-secure economy," World Economic Forum, Geneva, Switzerland, 2022. [Online]. Available: https://www3.weforum.org/docs/WEF_Transitioning20%to_a_Quantum_Secure_Economy_2022.pdf
4. P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134, doi: 10.1109/SFCS.1994.365700.
5. S. Rollo, "The quantum tech arms race is on," *Asia Times*, Mar. 2022. Accessed: Oct. 3, 2022. [Online]. Available: https://asiatimes.com/2022/03/the-quantum-tech-arms-race-is-on/
6. M. Swayne, "Quantum AI may need only minimal data—Proof takes step toward quantum advantage," *Quantum Insider*, pp. 2–3, Aug. 2022. [Online]. Available: https://thequantuminsider.com/2022/08/24/quantum-ai-may-need-only-minimal-data-proof-takes-step-toward-quantum-advantage
7. A. H. Karamlou, W. A. Simon, A. Katabarwa, T. L. Scholten, B. Peropadre, and Y. Cao, "Analyzing the performance of variational quantum factoring on a superconducting quantum processor," 2020, *arXiv:2012.07825v1*.
8. A. Katabarwa and Y. Cao. "Analyzing the performance of variational quantum factoring on a superconducting quantum processor." Zapata. Accessed: Oct. 8, 2022. [Online]. Available: https://www.zapatacomputing.com/publications/analyzing-the-performance-of-variational-quantum-factoring-on-a-superconducting-quantum-processor/
9. A. Glos, A. Krawiec, and Z. Zimborás, "Space-efficient binary optimization for variational quantum computing," *npj Quantum Inf.*, vol. 8, Apr. 2022, Art. no. 39, doi: 10.1038/s41534-022-00546-y.
10. "Researchers propose new quantum error correction technique." HPCwire. Accessed: Oct. 7, 2022. [Online]. Available: https://www.hpcwire.com/off-the-wire/researchers-propose-new-quantum-error-correction-technique/
11. M. Masiowski, N. Mohr, H. Soller, and M. Zesko, "Quantum computing funding remains strong, but talent gap raises concern," *McKinsey*, pp. 4–6, Jun. 15, 2022. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/quantum-computing-funding-remains-strong-but-talent-gap-raises-concern
12. "eSecurity planet reports: Best encryption software for 2022," QuSecure, San Mateo, CA, USA, 2022. [Online]. Available: https://www.qusecure.com/esecurity-planet-reports-best-encryption-software-for-2022/

PETE FORD is senior vice president of federal operations at QuSecure, San Mateo, CA 94403 USA. Contact him at pete@qusecure.com.

Unlock Your Potential

WORLD-CLASS CONFERENCES — Stay ahead of the curve by attending one of our 195+ globally recognized conferences.

DIGITAL LIBRARY — Easily access over 900k articles covering world-class peer-reviewed content in the IEEE Computer Society Digital Library.

CALLS FOR PAPERS — Discover opportunities to write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog and its range of offerings.

ADVANCE YOUR CAREER — Search the new positions posted in the IEEE Computer Society Jobs Board.

NETWORK — Make connections that count by participating in local Region, Section, and Chapter activities.

Explore membership today at the IEEE Computer Society

www.computer.org

IEEE COMPUTER SOCIETY

IEEE

EDITOR: Joanna F. DeFranco, The Pennsylvania State University; jfd104@psu.edu

DEPARTMENT: INTERNET OF THINGS

This article originally appeared in
Computer
vol. 56, no. 10, 2023

"Sensoring" the Farm

Joanna F. DeFranco, *The Pennsylvania State University*
Nir Kshetri, *University of North Carolina at Greensboro*
Jeffrey Voas, *IEEE Fellow*

Smart farming is gaining attention. But research needs to continue to address the challenges that come with farming efficiency.

Seventy-five years ago, *The Farmer's Almanac* was a key predictor for when to plant and reap. Today, it is still used for longer-range planning and forecasting, however, today's newest weapon for "when to do what" on a farm during a planting season is not a book, it's sensors embedded into the land and air that provide real-time data such that plans for how to make a crop more successful emerge before and after planting.

Agricultural tools have evolved for thousands of years. The first plows were made out of forked sticks to pull through the soil. That was the predecessor to plows integrating machinery to reduce soil resistance, and now, modern-day plows utilize GPS systems, allowing for precise operations. In addition to innovative machinery, farmers today have access to many computing developments that assist with increased efficiency and improvement of agricultural productivity, precision, and predictability.

In the mid-19th century, the agriculture industry began educating farmers as the United States needed to develop reliable food supplies for the betterment of society. Accordingly, baccalaureate degrees in agriculture were launched. The first American school to offer baccalaureate degrees in agriculture was The Farmer's High School of Pennsylvania, now known as *The Pennsylvania State University*. Its first graduating class in 1861 had 13 students. In 2020, 37,721 agricultural degrees were awarded across the United States.[1]

Digital Object Identifier 10.1109/MC.2023.3296762
Date of current version: 20 September 2023

Agriculture is now a US$1.2 trillion dollar industry.[2] In the mid-20th century during the arrival of computers, farming applications began to emerge. Today, the industry continues to advance in this area, where the Internet of Things (IoT) is the main technology that facilitates smart and precision farming. Given the labor intensiveness of this industry, it makes sense that farmers use computing power to increase precision and efficiency.[3]

AGRICULTURAL COMPUTING

Agricultural computing innovations are assisting the industry with improving efficiency, increasing crop yield, and preserving natural resources. According to the statistics portal Statista, the IoT was the most influential agricultural technology (AgTech) innovation of 2022 and had a higher impact on the agricultural market than any other AgTechs.[4] The primary computing technologies related to the IoT are under the umbrella of smart farming, which includes precision farming technologies. Recently, agricultural researchers have created systems and frameworks to help farmers efficiently connect the numerous sensor nodes and devices that monitor soil, water, and environmental data as well as control the flow of water and fertilizer to crops.[5,6] Other researchers have created

DISCLAIMER

The authors are completely responsible for the content in this message. The opinions expressed here are their own.

 Published by the IEEE Computer Society

TABLE 1. Agricultural IoT technologies.

Technology	Examples
Livestock monitoring	Livestock feeding is optimized using sensors to track behavior, health, and activity.
Soil monitoring	Moisture, nutrient, and temperature data are used to determine fertilization and watering needs.
Irrigation management and automation	Water usage is optimized.
Agricultural robots for crop harvesting	Picking and harvesting tasks are automated; aerial crop assessments are provided.
Crop monitoring (yield and health)	Using sensors, images, and artificial intelligence, these systems can detect signs of disease, pests, and harvest to optimize and predict yield. Sensors are used to collect in real time and use historical data to predict the quality of soil and crop health.
Equipment monitoring and maintenance	Sensors are used to monitor machine health, performance, fuel, and maintenance.
Security	Intrusion and theft monitoring/alert services are provided.

systems to optimize the use of natural resources such as water and solar power.[7] Some of the key examples of the IoT's uses in agriculture are listed in Table 1.

LIVESTOCK MONITORING

South African multinational mobile telecommunications company MTN, which operates in many African and Asian countries, has partnered with Aotoso Technology to provide IoT-based connected collars for cattle in the Sudanese market. Farmers use Subscriber Identity Module cards on the collar and on their cell phones to get vital information about the cattle. The information helps them develop feeding and breeding strategies. The device also monitors reproduction and lactation, which can help farmers increase their income.[8]

SOIL MONITORING

The IoT can help maintain appropriate soil moisture conditions and nutrient availability, which can help increase the effect of fertilizers. As an example, Kenya's IoT-based smart irrigation system, Illuminum Greenhouses, are powered by solar panels and sensors, which work together to create an optimal environment for crops.[9] Farmers can use their mobile phones to control key indicators such as temperature, humidity, and soil moisture. An automated watering system supplies a precise amount of water if the sensors detect that the soil is dry. The solutions are cost efficient and thus accessible for smallholder farmers.[10]

THE PRIMARY COMPUTING TECHNOLOGIES RELATED TO THE IOT ARE UNDER THE UMBRELLA OF SMART FARMING, WHICH INCLUDES PRECISION FARMING TECHNOLOGIES.

IRRIGATION MANAGEMENT AND AUTOMATION

IoT-based irrigation systems are helping improve the availability of water for farmers. For example, Nairobi, Kenya-based solar irrigation company SunCulture provides smallholder farmers irrigation and solar pumping solutions.[11] Using SunCulture's off-grid technology, farmers can extract up to 3,000 L of water per hour from wells up to 70-m deep.[12] The system also has built-in algorithms, which study the weather to optimize performance. Based on the predicted weather patterns, SunCulture's smart machines send phone and text alerts to farmers about the appropriate irrigation timing.[13] If too much water is pumped out of the ground, pressure in the aquifer gradually decreases. To ensure that the water supply does not

dry out, other IoT solutions can be used to evaluate the optimal amount of water to be pumped. As noted previously, farmers, for instance, can use sensors to detect soil moisture and pump water only when needed. IoT-based irrigation systems can thus minimize water consumption.

AGRICULTURAL ROBOTS

To assist with helping with labor shortages and increasing population, robotics has been integrated into the agricultural domain in many areas to reduce some of the heavy lifting, such as picking and gathering fruit, harvesting vegetables, and dealing with weeds. In addition, drones, considered flying robots, provide aerial images that assist farmers with quick assessments of crop health. Other uses of robots in farming are greenhouses (mentioned earlier), which can provide more vegetables to urban areas.[14]

AUTONOMOUS ROBOTS HAVE A 98% ACCURACY WHEN PICKING FRUIT BUT DO REQUIRE THE SUPERVISION OF ONE INDIVIDUAL.

Tortuga AgTech is a farming robot system. In addition to the labor-intensive tasks, it provides a way to address challenges such as damage caused by human hands during the harvesting process. Autonomous robots have a 98% accuracy when picking fruit but do require the supervision of one individual.[15] iRobot, another farming system, has also entered the harvest-automation arena. In addition to picking and harvesting, this technology can also continue running in extremely hot weather.[16]

CROP MONITORING

Vodacom's MyFarmWeb has tools that collect and analyze data from multiple IoT sensors across a farm, which can help to improve productivity and optimize farming practices. For instance, its ITESTLeaf feature can be used to view and compare leaf tissue based on historical data. Its precision pest-monitoring feature provides a historical analysis of pests and trends. Likewise, the MyYield feature can help analyze areas with various levels of yields and identify the underlying causes of areas with low yields. As of early 2022, MyFarmWeb was being used by 7,200 farmers in the United States, South Africa, Australia, and New Zealand.[17]

EQUIPMENT MONITORING AND MAINTENANCE

In March 2023, Indian multinational conglomerate Mahindra's AgTech arm Krish-e launched IoT-based Krish-e Smart Kit (KSK), which provides equipment owners with detailed insights about their farm equipment, such as tractors harvesters and rice transplanters. The KSK uses GPS to track and remotely monitor various parameters using a smartphone or desktop.[18] The kit can be installed on any brand of equipment, which can help improve fleet performance, reduce equipment downtime, prevent unauthorized usage, and reduce maintenance costs.[19]

SECURITY

IoT-based solutions can also help prevent theft of livestock and agricultural produce. The U.K.-based IoT solutions provider Smarter Technologies Group's cattle collars send out a signal to a livestock management dashboard every 15 min. This enables farmers to gain remote visibility of their livestock. Cattle owners can also set up specific geofences and alerts, which notify them when a livestock moves outside a designated area. The company's smart fence gate alarms can inform cattle owners when there are unauthorized cattle movements.[20] Likewise, MTN's IoT-based connected collars, discussed previously, can also help prevent cattle theft.

AGRICULTURE TECHNOLOGY AS A SERVICE

The agriculture technology-as-a-service (ATaaS) market, valued at more than US$3.4 billion, is driven by small-scale farmers and their growing demand for precision agriculture.[21] Smaller-scale farmers can use ATaaS to enhance productivity and efficiency. The industry is predicted to reach US$4.93 billion by 2028.[22] The IoT's potential to transform farming practices has been one of the key drivers of the ATaaS market.[23]

ATaaS can

- streamline operations and increase efficiency using crop monitoring, livestock monitoring, soil analysis, irrigation, crop yield, and autonomous farming machinery

- provide real-time information about crops to make informed decisions about planting, fertilizing, and pest control.

To achieve these goals, ATaaS providers integrate advanced sensors and other technologies such as remote sensing and drones in their solutions. Farmers have access to a wide range of indicators that the sensors collect and measure, which can be used for accurate crop monitoring, and take proactive actions that can help them generate the highest revenue. For instance, farmers can detect potential problems such as disease, pest infestations, and soil degradation, which can help with timely interventions.[24]

AGRICULTURE COMPUTING CHALLENGES

In general, smart farming is environmentally friendly given the improvements discussed earlier. But society needs to consider the challenges that come along with changes. For example, some research states that agricultural efficiency reduced expenses of water and pesticides, but some claim that the positive outcomes of more efficient use of machinery and irrigation systems may result in higher use of pesticides and fertilizers, which can also be a result of climate change.[25] It is a fact that climate change has increased pests that destroy crops. The point is, the entire system needs to be considered, even when improvements are made.

To grasp this statement, one can use the analysis approach of systems thinking. Specifically, one of the systems thinking laws by Peter Senge is, "Today's problems come from yesterday's solution." For example, Brazil is the largest coffee producer but also the largest pesticide consumer.[26] The pesticides used are effective for coffee production, but not good for the ecosystem and health of the system in which it resides (this includes people). The pesticides used have contaminated some water supplies and been linked to respiratory problems, high blood pressure, cancer, and cardiovascular disease. Sustainable alternatives could be considered. Agroforestry, the practice of integrating trees, shrubs, and livestock, has been known to create benefits.[27]

Like the victory gardens encouraged in the United States after World War I and World War II to meet population demands, urban planners are looking to the public to source some of their food. For example, farmers in India are also using IoT systems to increase food production in urban areas.[28] Greenhouse technology is used as well as innovative solutions such as vertical farms (growing crops on top of each other), hydroponics (growing plants without the use of soil), aeroponics (growing plants with only water and nutrients), and aquaponics (growing plants and fish together in the same environment), are being utilized along with sensors, data collection, and a device to monitor.

As the world's population increases, so too does the need for food. However, tillable land is being turned into data centers and wind farms, while other tillable land is becoming untillable due to climate change and political conflicts. AgTech is an exciting and ever-expanding science with the potential to address these challenges. Computing plays a large role, not only from sensors and the IoT, but from computing's role in creating new fertilizers, pesticides, and seeds.

Many of the systems described in this article are used by farmers to make informed decisions to optimize the use of things like fertilizer and pesticide use and natural resources (water/fuel) to minimize environmental impact. For example, water delivery can be adjusted based on real-time weather sensor data. Controlling water usage not only preserves a natural resource but also reduces costs and improves crop yields.

Data collection (from different sensors) also can assist in predicting crop yield, reducing equipment downtime (watching equipment health), and, overall, improves efficiency in many aspects of farming.

Therefore, smarter farming is just smarter, and this topic will be gaining more attention in the immediate years.

REFERENCES

1. "Agriculture." DATAUSA. Accessed: Jul. 14, 2023. [Online]. Available: https://datausa.io/profile/cip/agriculture
2. "Ag and food sectors and the economy," U.S. Dept. Agriculture, Washington, DC, USA, Jan. 26, 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://www.ers.usda.gov/data-products/ag-and-food-statistics-charting-the-essentials/ag-and-food-sectors-and-the-economy/#:~:text=Agriculture%2C%20food%2C%20and%20related%20industries%20contributed%20roughly%20%241.264,of%20this%20sum%E2%80%94about%200.7%20percent%20of%20U.S.%20GDP

3. "Precision farming vs. digital vs. smart farming: What's the difference?" DTN, Burnsville, MN, USA, Mar. 2021. Accessed: Jul. 14, 2023. [Online]. Available: https://www.dtn.com/precision-farming-vs-digital-farming-vs-smart-farming-whats-the-difference/
4. "Smart agriculture 2022." Statista. Accessed: Jul. 14, 2023. [Online]. Available: https://baba-blog.com/smart-agriculture/
5. A. K. Nalendra, D. Wahvudi, M. Mujiono, M. N. Fu'ad, and N. Kholila, "IoT-Agri: IoT-based environment control and monitoring system for agriculture," in *Proc. 7th Int. Conf. Informat. Comput. (ICIC)*, Denpasar, Indonesia, 2022, pp. 1–6, doi: 10.1109/ICIC56845.2022.10006964.
6. S. K. Sah Tyagi, A. Mukherjee, S. R. Pokhrel, and K. K. Hiran, "An intelligent and optimal resource allocation approach in sensor networks for smart Agri-IoT," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17,439–17,446, Aug. 2021, doi: 10.1109/JSEN.2020.3020889.
7. V. V. Reddy S, B. Jaison, A. Balaji, D. Indumathy, S. Vanaja, and J. J. Jeya Sheela, "Agri-IoT: A farm monitoring and automation system using Internet of Things," in *Proc. 2nd Int. Conf. Electron. Renewable Syst. (ICEARS)*, Tuticorin, India, 2023, pp. 639–642, doi: 10.1109/ICEARS56392.2023.10085235.
8. "Leading digital solutions for Africa's progress," MTN Group Ltd., Fairland, South Africa, Dec. 2020. Accessed: Jul. 14, 2023. [Online]. Available: https://www.mtn.com/wp-content/uploads/2023/03/2020-UN-Global-Compact-Report.pdf
9. M. Mutiga, "Kenya's smart greenhouse texts when your tomatoes need watering," *The Guardian*, Jan. 2016. Accessed: Jul. 14, 2023. [Online]. Available: https://www.theguardian.com/global-development/2016/jan/05/kenya-smart-greenhouse-tomatoes-watering-farming
10. J. Stewart, "Challenges surrounding IoT deployment in Africa," *Compare the Cloud*, 2019. Accessed: Jul. 14, 2023. [Online]. Available: https://www.comparethecloud.net/articles/challenges-surrounding-iot-deployment-in-africa/
11. "SunCulture wins project of the year." EEP Africa. Accessed: Jul. 14, 2023. [Online]. Available: https://eepafrica.org/sunculture-wins-project-of-the-year/#:~:text=SunCulture%20is%20based%20in%20Nairobi
12. "The 10 most innovative European, Middle Eastern, and African companies of 2021," *Fast Company*, Mar. 2021. Accessed: Jul. 14, 2023. [Online]. Available: https://www.fastcompany.com/90600369/europe-middle-east-africa-most-innovative-companies-2021
13. E. Mungai, "AI is a game changer for small companies," *Africa Sustainability Matters*, pp. 1–8, Feb. 2021. Accessed: Jul. 14, 2023. [Online]. Available: https://africasustainabilitymatters.com/ai-is-a-game-changer-for-small-companies
14. S. Gossett, "16 agricultural robots and farm robots you should know," *Builtin*, Mar. 7, 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://builtin.com/robotics/farming-agricultural-robots
15. "Tortuga AgTech." Builtin. Accessed: Jul. 14, 2023. [Online]. Available: https://builtin.com/company/tortuga-agtech
16. "Harvest automation." Builtin. Accessed: Jul. 14, 2023. [Online]. Available: https://builtin.com/company/harvest-automation
17. S. Brooks, "Vodafone brings digital transformation to farming in Europe," *Enterprise Times*, Feb. 2022. Accessed: Jul. 14, 2023. [Online]. Available: https://www.enterprisetimes.co.uk/2022/02/25/vodafone-brings-digital-transformation-to-farming-in-europe/
18. "Krish-e launches IoT-based Smart Kit for farm equipment monitoring," *Agriculture Post*, Mar. 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://agriculturepost.com/agritech/krish-e-launches-iot-based-smart-kit-for-farm-equipment-monitoring/
19. "Krish-e launches IoT based Smart Kit for farm equipment," *Mahindra,* Mar. 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://www.mahindra.com/news-room/press-release/en/krishe-launches-iot-based-smart-kit-for-farm-equipment
20. B. Wingrave, "The smart way to combat livestock theft." Smarter Technologies. Accessed: Jul. 14, 2023. [Online]. Available: https://smartertechnologies.com/blog/livestock-theft/
21. "Agriculture technology-as-a-service global market report 2023: Decreasing agriculture workforce drives demand," *GlobeNewswire*, Mar. 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://www.globenewswire.com/en/news-release/2023/03/28/2635448/28124/en/Agriculture-Technology-as-a-Service-Global-Market-Report-2023-Decreasing-Agriculture-Workforce-Drives-Demand.html
22. Emergen Research, "Agriculture technology as a service market to witness explosive growth by 2028 | IBM Corporation, Accenture plc, Trimble," *News Wires*, Apr. 2022. Accessed: Jul. 14, 2023. [Online]. Available: https://www.einnews.com/pr_news/566974142/agriculture-technology-as-a-service-market-to

-witness-explosive-growth-by-2028-ibm-corporation-accenture-plc-trimble
23. "Global agriculture technology-as-a-service market." Spherical Insights. Accessed: Jul. 14, 2023. [Online]. Available: https://www.sphericalinsights.com/reports/agriculture-technology-as-a-service-market
24. "Agriculture technology-as-a-service market - A global and regional analysis: Focus on product, application, and country analysis - Analysis and forecast, 2022-2027," *BIS Res.*, 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://bisresearch.com/industry-report/agriculture-technology-service-market-report.html
25. "What is smart farming, and what are its advantages and disadvantages?" Hurley Farms. Accessed: Jul. 14, 2023. [Online]. Available: https://www.visithurleyfarms.com/what-is-smart-farming-and-what-are-its-advantages-and-disadvantages/
26. University of Copenhagen, "Unsustainable coffee production is making more and more people sick, says study," *Phys.Org*, Jun. 2023. Accessed: Jul. 14, 2023. [Online]. Available: https://phys.org/news/2023-06-unsustainable-coffee-production-people-sick.html
27. M. Vaughan, N. Adamson, and K. MacFarland, "Using agroforestry practices to reduce pesticide risks to pollinators and other agriculturally beneficial insects," U.S. Dept. Agriculture, Washington, DC, USA, Jun. 2017. Accessed: Jul. 14, 2023. [Online]. Available: https://www.fs.usda.gov/nac/assets/documents/agroforestrynotes/an35g09.pdf
28. B. Anuradha, R. Pradeep, E. Ahino, A. Dhanabal, R. J. Gokul, and S. Lingeshwaran, "Vertical farming algorithm using hydroponics for smart agriculture," in *Proc. Int. Conf. Intell. Syst. Commun., IoT Secur. (ICISCoIS)*, Coimbatore, India, 2023, pp. 432–437, doi: 10.1109/ICISCoIS56541.2023.10100527.

JOANNA F. DeFRANCO is an associate professor of software engineering, associate director of the D.Eng. in Engineering program at The Pennsylvania State University, Malvern, PA 19355 USA, and an associate editor in chief of *Computer*. Contact her at jfd104@psu.edu.

NIR KSHETRI is a professor of management in the Bryan School of Business and Economics, University of North Carolina at Greensboro, Greensboro, NC 27412 USA, and the "Computing's Economics" column editor for *Computer*. Contact him at nbkshetr@uncg.edu

JEFFREY VOAS, Gaithersburg, MD 20899 USA, is the editor in chief of Computer. He is a Fellow of IEEE. Contact him at j.voas@ieee.org.

EDITOR: Hsiao-Ying Lin, IEEE Member, hsiaoying.lin@gmail.com

DEPARTMENT: ARTIFICIAL INTELLIGENCE/ MACHINE LEARNING

This article originally appeared in
Computer
vol. 56, no. 3, 2023

AI for Water

Feras A. Batarseh, *Virginia Tech*
Ajay Kulkarni, *Commonwealth Cyber Initiative, Virginia Tech*

Water availability is a prerequisite for flourishing economies, protecting public health, and national prosperity. An overview of the potentials of artificial intelligence for water is presented.

In the United States, there are about 153,000 public drinking water systems[1] and more than 16,000 publicly owned wastewater treatment plants.[2] While systems control and data acquisition (SCADA) has become a standard in large water treatment and distribution plants,[3] a very small (unknown) percentage of those plants have cyber defenses in place.[4]

MOTIVATION

SCADA infrastructures, as evidence shows,[5] are very prone and vulnerable to cyberthreats. Additionally, there has been a rising number of attacks, especially on water plants, in the United States and around the world. The traditional cybersecurity approach to utilizing firewalls and double authentication is beneficial and can mitigate multiple forms of cyberattacks; however, more sophisticated attacks, such as data poisoning, data manipulation, minimum perturbations, concealed attacks,[6] info stealer, botnets, and ransomware, require algorithms that can detect unusual activities (that is, outlier events)[7]; classify the source of adversarial actions; and perform attack mitigation activities. The current state of the art provides evidence that artificial intelligence (AI) is the leading approach to such defenses[8] due to its ability to adequately identify unwarranted pattern shifts in networks and datasets, a feature that is not achievable using traditional approaches.

Digital Object Identifier 10.1109/MC.2022.3231142
Date of current version: 8 March 2023

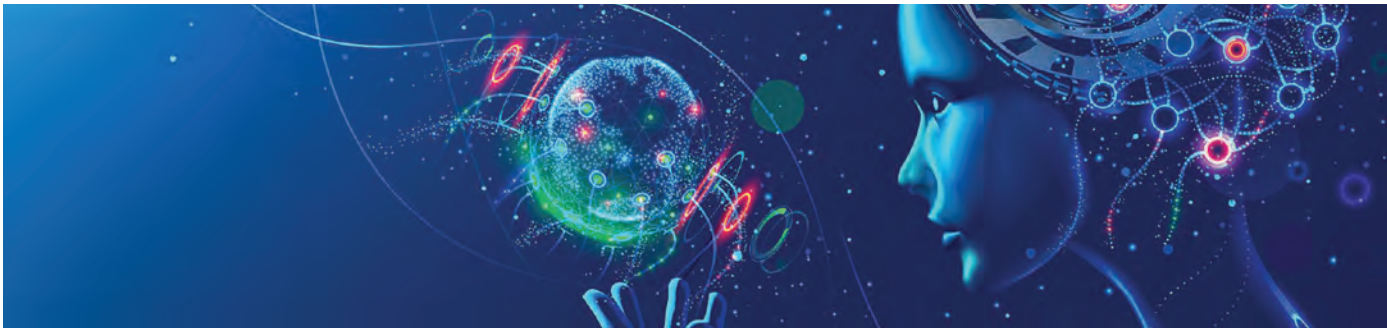
STORIES OF INTEREST

Critical infrastructures, such as smart grids, nuclear plants, medical monitoring systems, smart farms, and intelligent water systems (IWSs), are deemed obsolete (and dangerous to human life) without comprehensive measures to protect them and secure their outcomes.[9] The workings of these systems are usually governed by laws and policies as well as domain-specific best practices.[10] However, the challenge of securing those cyberphysical systems is exacerbated by the dependency of their cyber, human, biological, and physical components on each other. For instance, a programmable logic controller controlling the pH levels of water at a treatment plant is an example of a cyber and bio interdependency; smart sensors reading water flow in municipal water pipes is an example of physical and cyber dependencies; and so on.[11] Wastewater treatment plants dump treated water (effluent) into rivers all over the country, while U.S. Environmental Protection Agency policies dictate acceptable amounts of phosphorous and nitrogen. For instance, an unwarranted modification of that value as being measured by sensors at a wastewater treatment facility can cause severe environmental damage to rivers and lakes due to excess amounts of such chemicals—that is not hypothetical; recent motivation examples of applying AI for water are presented.

Data poisoning and water poisoning

In the last two decades, U.S. water systems have been exposed to different cyberthreats.[12] In 2015, the

 Published by the IEEE Computer Society



U.S. Department of Homeland Security responded to 25 cyber incidents related to the water sector, representing a 78.6% increase in the number of reported cases since 2014[2]—one of the highest rate increases among all sectors. Cyberattacks have continued to grow exponentially relative to the attention given to the sector by governments, operators, practitioners, and academics—deeming the sector unsafe.[13] In February 2021, a plant operator in Oldsmar, FL, saw his cursor being moved around on a computer screen, starting various software functions that control the water being treated. While the operator assumed that it was another employee accessing the system remotely, the intruder boosted (that is, data poisoning) the value of one data point—the sodium hydroxide (lye) amount—by 100 times. Sodium hydroxide, the main ingredient in cleaning liquids, is added to treat the acidity of water and remove metals from water for purposes of human consumption (that is, drinking). Increasing its amount to higher levels can cause poisoning, burns, and multiple other health risks.[4]

Water, farmers, and irrigation

The Brookings Institution reports an average of 30 farmers committing suicide per day in India due to water-related events.[14] Albeit this is a fluctuating rate, the cause of farmers' uncertainty is increasingly due to water access, extreme weather events, rainfall, the lack of water, inaccurate forecasts of water levels in rivers and lakes,[15] and unwarranted chemicals in water used for agricultural irrigation. Such events affect agricultural yield, the quality of crops, crop disease, and even the prices of agricultural commodities around the world.[16] As is known, nimble agriculture and food security is vital to human survival. In the past three years, global agriculture has been negatively affected by many water-related shocks. Unprecedented uncertainties (such as floods and droughts) have affected the range of decisions starting at the farm and reaching the household consumption of certain goods.

Water Security

We suggest that the status quo definition of *water security* ought to expand from merely covering water "availability" to including cyber-, physical, and biosecurity aspects.[17] Conventional definitions don't satisfy the rising need to cover all areas of how water can be secure. The United Nations provides a working definition for water security as follows[11]: "Water security is defined here as the capacity of a population to safeguard sustainable access to adequate quantities of acceptable quality water for sustaining livelihoods." Based on the three-pillared challenge involved in the sector (cyber, bio, and physical), we propose the following new definition for water security encompassing emerging trends and conventional challenges related to water availability and quality:

> *"Water security is the capacity of nations to safeguard the quality and availability of water for all desired purposes of society, which includes ensuring measures related to securing access; valid treatment processes; cyber hygiene; and mitigating risks associated with environmental factors, data collection, biological threats, and emerging technologies."*

While we understand that this definition is debatable, the goal of presenting such a definition is to urge scientists and practitioners in the water sector to consider the novel aspects.

AREAS OF APPLICATION

Water has an obvious effect on economies besides farming and drinking water. Societies flourish next to water bodies[18]; sanitation and health[19] are not

manageable or possible without access to water; and manufacturing is heavily dependent on reliable water sources. It is rather difficult to capture all potential AI applications in the water sector; here, a brief review of three examples is elaborated.

Water treatment and management

As AI becomes more developed and deployed further across critical infrastructure, operators will be blindsided if they rely only on their past experience or expertise when making decisions (such as deciding on the number of pumps to operate during a storm, assessing the performance of the adsorption process, and evaluating water quality). Future leaders need to possess a fundamental knowledge of AI to better lead and protect their institutions.[20] A modern water treatment plant, referred to as an *IWS*, has hundreds of sensors and actuators—for pipes, tanks, reservoirs, and pumps. Such equipment has inter- and intradependencies that increase the complexity of detecting a breach.[21] Accordingly, AI can be used at wastewater treatment plants for multiple use cases in decision making.[22] Decision-making support involves optimization techniques to allocate an optimal combination of factors that maximize/minimize a numerical objective function (that is, the factor affected by the decision). From the onset of hydraulic modeling, optimization techniques have been critical to water distribution networks.[23] Areas of application that can benefit from AI optimization algorithms include: optimizing energy consumption, the number of pumps used at a certain point in time, the optimal design of monitoring and control networks, and the management of tunnels and pipes during extreme weather events.[24]

> *"WATER SECURITY IS THE CAPACITY OF NATIONS TO SAFEGUARD THE QUALITY AND AVAILABILITY OF WATER FOR ALL DESIRED PURPOSES OF SOCIETY, WHICH INCLUDES ENSURING MEASURES RELATED TO SECURING ACCESS; VALID TREATMENT PROCESSES; CYBER HYGIENE; AND MITIGATING RISKS ASSOCIATED WITH ENVIRONMENTAL FACTORS, DATA COLLECTION, BIOLOGICAL THREATS, AND EMERGING TECHNOLOGIES."*

Optimization through AI is performed via multiple approaches, but the most common ones include genetic algorithms (GAs), deep learning (DL), and reinforcement learning (RL). For instance, if a water treatment plant aims to minimize nitrogen in the effluent, then a DL or GA optimization approach could be utilized. RL algorithms can, for instance, reinforce practices that lead to better water quality or increase water volume (gallons) processing per day.

Agricultural irrigation and farming

Essential crops (such as wheat, corn, and soybeans), livestock, fruit, and all agricultural commodities require water to survive. However, access to water is not always guaranteed; for instance, in drought-prone areas, smart irrigation is a critical strategy due to the scarcity of water.[25] Urban and vertical farming are similar; in those scenarios, precision irrigation is important to maintain farm finances and create profit for the farmers.[16] Such AI-driven decision-making processes (for example, crop yield prediction, livestock price forecasting, and optimizing the right mix of biodegradable pesticides) are heavily dependent on big data. Data, however, are prone to poisoning, validation issues, and incorrectness. Poisoned data can change farming recommendations; manipulate smart-irrigation systems' outcomes; and compromise water meters, humidity sensors, water pumps, and other agricultural control devices.

Water economics and policy

Water and the environment are difficult spaces to regulate, mainly due to the shared nature of their resources (that is, the tragedy of the commons). Policymakers need to refer to experts to understand the domain and create reasonable policies that can govern the space fairly. Park et al.,[26] for instance, utilized AI, Shapley Additive exPlanations (SHAP), and partial dependence plots for identifying important variables that affect algal bloom in rivers: generally, a challenging (and potentially subjective) aspect to measure. One of the theories in computational quantification is referred to as *value loading*.[27] It presents matters with a subjective concept, such as policy evaluation—of

the Clean Water Act 33 U.S.C. §1251 et seq. (1972), for instance—that could be defined mathematically and measured in a more empirical manner.

However, we argue the following for the application of AI for public policy. Science used as a foundation for statutes is ever changing, and in many cases, different contexts could lead to different results. Accordingly, data-driven lawmaking has to be one of the major ways of constructing and evaluating the success of statutes—a direction that is becoming progressively inevitable and is also increasingly further backed up by the public.[28] One of the ongoing debates in water and environmental law is the Water of the United States issue—that is, which authority (state versus federal) prevails and who has control over water bodies of the United States.

Multiple administrations have tackled this issue, but what is very interesting is the involvement of different scientific assumptions as a basis for criteria that define answers to the debate. For instance, the Obama administration used the significant nexus test, while the Trump administration used the narrower definition derived from the Supreme Court's Rapanos versus USA case, 547 U.S. 715 (2006). Both are deemed reasonable, but one would ask: Which one should be followed? And without sufficient data, how does one measure the outcome? The answer lies in empirical evaluations and scientific experimentation (not expert opinions or political partisanship)—multiple research institutes produce research that generates results and directions that ought to support a certain direction, although contradicting in some cases, but referring to such scientific debates usually leads to a consensus that is backed up by different dimensions.

In some cases, U.S. law cannot be defined in isolation; therefore, the national well-being also has to be in the balance when it comes to ratifying statutes that contradict or confirm international treaties. For such difficult goals to be balanced in the same legal realm, AI can be one of the main referees in determining the best version of a statute as it is being crafted or amended.

AI has been helping cure disease, create art, drive cars, perform surgery, and identify crime; the water sector is no different. In this article, the use cases, AI methods, and applications aim to encourage the water industry to investigate and further apply AI for decision making while considering assurances (such as security) for deriving actionable intelligence.

Besides cybersecurity, explainability, and correctness, other challenges involved with AI's deployability to the water sector include adoption by operators; the black box nature of AI models; and data privacy issues—all of which are applicable to most sectors. Ultimately, however, these issues ought to be addressed, especially because water has no substitute.

REFERENCES

1. R. Clark, S. Panguluri, T. Nelson, and R. Wyman, "Protecting drinking water utilities from cyberthreats," *J. Amer. Water Works Assoc.*, vol. 109, no. 2, pp. 50–58, Feb. 2017, doi: 10.5942/jawwa.2017.109.0021.
2. "Water and wastewater systems sector-specific plan," U.S. Dept. Homeland Secur., Washington, DC, USA, 2015. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/nipp-ssp-water-2015-508.pdf
3. J. Hubert, Y. Wang, E. Alonso, and R. Minguez, *Using Artificial Intelligence for Smart Water Management Systems*. Mandaluyong, Philippines: Asian Development Bank, 2020.
4. A. Hassanzadeh et al., "A review of cybersecurity incidents in the water sector," *J. Environmental Eng.*, vol. 146, no. 5, May 2020, Art. no. 03120003, doi: 10.1061/(ASCE)EE.1943-7870.0001686.
5. M. Hameed et al., "Application of artificial intelligence (AI) techniques in water quality index prediction: A case study in tropical region, Malaysia," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 893–905, Dec. 2017, doi: 10.1007/s00521-016-2404-7.
6. J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-Pois: An attack-agnostic defense against data poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3412–3425, May 2021, doi: 10.1109/TIFS.2021.3080522.
7. J. Zheng, P. Chan, H. Chi, and Z. He, "A concealed poisoning attack to reduce deep neural networks robustness against adversarial samples," *Inf. Sci.*, vol. 615, pp. 758–773, Nov. 2022, doi: 10.1016/j.ins.2022.09.060.
8. H. Mehmood, D. Liao, and K. Mahadeo, "A review of artificial intelligence applications to achieve water-related sustainable development goals," in *Proc. IEEE/ITU Int. Conf. Artif. Intell. Good (AI4G)*, 2020, pp. 135–141, doi: 10.1109/AI4G50087.2020.9311018.

9. L. Corominas, M. Garrido-Baserba, K. Villez, G. Olsson, U. Cortés, and M. Poch, "Transforming data into knowledge for improved wastewater treatment operation: A critical review of techniques," *Environmental Modelling Softw.*, vol. 106, pp. 89–103, Aug. 2018, doi: 10.1016/j.envsoft.2017.11.023.
10. J. Keck and J. Lee, "Embracing analytics in the water industry," *J. Water Resour. Planning Manage.*, vol. 147, no. 5, May 2021, Art. no. 02521002, doi: 10.1061/(ASCE)WR.1943-5452.0001375.
11. "Water security and the global water agenda: A UN-Water Analytical Brief," UN Water, Hamilton, ON, Canada, 2013. [Online]. Available: https://collections.unu.edueserv/UNU:2651/Water-Security-and-the-Global-Water-Agenda.pdf
12. N. K. Velayudhan, P. Pradeep, S. N. Rao, A. R. Devidas, and M. V. Ramesh, "IoT-enabled water distribution systemsA comparative technological review," *IEEE Access*, vol. 10, pp. 101,042–101,070, Sep. 2022, doi: 10.1109/ACCESS.2022.3208142.
13. A. Harsha Vardhan, B. Subramanyam, M. Lakshmi Reddy, G. Gowtham Reddy, and A. Ramesh, "Anomaly detection in water distribution systems," in *Proc. 4th Smart Cities Symp. (SCS)*, 2021, pp. 219–222, doi: 10.1049/icp.2022.0344.
14. R. Shamika, *A Reality Check on Suicides in India* (Brookings India IMPACT Series). New Delhi, India: Brookings Institution India Center, 2015.
15. M. Lunani, "Artificial intelligence for water and wastewater: Friend or foe?" *Opflow*, vol. 44, no. 6, pp. 6–7, Jun. 2018, doi: 10.1002/opfl.1017.
16. F. Batarseh, M. Gopinath, A. Monken, and Z. Gu, "Public policymaking for international agricultural trade using association rules and ensemble machine learning," *Mach. Learn. Applicat.*, vol. 5, Sep. 2021, Art. no. 100046, doi: 10.1016/j.mlwa.2021.100046.
17. A. Chastain-Howley. "How big is big data among water utilities?" Water Online. Accessed: Oct. 29, 2022. [Online]. Available: https://www.wateronline.com/doc/how-big-is-big-data-among-water-utilities-0001
18. G. Abramowitz, "Towards a benchmark for land surface models," *Geophys. Res. Lett.*, vol. 32, no. 22, Nov. 2005, Art. no. L22702, doi: 10.1029/2005GL024419.
19. N. Hellen and G. Marvin, "Explainable AI for safe water evaluation for public health in urban settings," in *Proc. Int. Conf. Innov. Sci., Eng. Technol. (ICISET)*, 2022, pp. 1–6, doi: 10.1109/ICISET54810.2022.9775912.
20. E. E. Kim et al., "Water economy with smart water system in the City of Carouge," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*, 2022, pp. 1–6, doi: 10.1109/COINS54846.2022.9854990.
21. J. Ktari, T. Frikha, M. Hamdi, H. Elmannai, and H. Hmam, "Lightweight AI framework for industry 4.0 case study: Water meter recognition," *Big Data Cogn. Comput.*, vol. 6, no. 3, p. 72, Jul. 2022, doi: 10.3390/bdcc6030072.
22. L. Hao, J. Gao, and J.-a Wang, "Simulating the effects of water reuse on alleviating water shortage," in *Proc. 4th Int. Conf. Bioinf. Biomed. Eng.*, 2010, pp. 1–4, doi: 10.1109/ICBBE.2010.5515327.
23. G. Sebestyen, A. Hangan, and Z. Czako, "Anomaly detection in water supply infrastructure systems," in *Proc. 23rd Int. Conf. Control Syst. Comput. Sci. (CSCS)*, 2021, pp. 349–355, doi: 10.1109/CSCS52396.2021.00064.
24. W. Y. Mao et al., "Trustworthy AI solutions for cyber-biosecurity challenges in water supply systems," *Int. FLAIRS Conf. Proc.*, vol. 35, May 2022, Art. no. 130664, doi: 10.32473/flairs.v35i.130664.
25. V. Radhakrishnan and W. Wu, "IoT technology for smart water system," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun.; IEEE 16th Int. Conf. Smart City; IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, 2018, pp. 1491–1496, doi: 10.1109/HPCC/SmartCity/DSS.2018.00246.
26. J. Park, W. H. Lee, K. T. Kim, C. Y. Park, S. Lee, and T.-Y. Heo, "Interpretation of ensemble learning to predict water quality using explainable artificial intelligence," *Sci. Total Environ.*, vol. 832, Aug. 2022, Art. no. 155070, doi: 10.1016/j.scitotenv.2022.155070.
27. N. Bostrom, *Superintelligence: Paths, Dangers, Strategies*. Oxford, U.K.: Oxford Univ. Press, 2015.
28. F. Batarseh and R. Yang, *Federal Data Science: Transforming Government and Agricultural Policy Using Artificial Intelligence*. London, U.K.: Elsevier, 2017.

FERAS A. BATARSEH is an associate professor with the Department of Biological Systems Engineering, Virginia Tech, Arlington, VA 22203 USA. He is a Senior Member of IEEE. Contact him at batarseh@vt.edu.

AJAY KULKARNI is a postdoctoral associate with the Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA 22203 USA. Contact him at ajaysk@vt.edu.c

IEEE COMPUTER SOCIETY

Publications Seek 2026 Editors in Chief

Application Deadline: 1 March 2025

IEEE Computer Society seeks applicants for editor in chief for the following publications:

- *Computer* magazine
- *IEEE Computer Graphics and Applications*
- *IEEE Security & Privacy*
- *IEEE Transactions on Emerging Technologies in Computing*
- *IEEE Transactions on Mobile Computing*
- *IEEE Transactions on Services Computing*
- *IEEE Transactions on Software Engineering*
- *IEEE Transactions on Visualization and Computer Graphics*

Our publications are the cornerstone of professional activities for our members and the community we serve. We seek candidates who are IEEE members in good standing, have strong familiarity with our publications, and possess an excellent understanding of the field as it relates to academic, industry, and governmental areas. Applicants must have successful experience developing a diverse team of individuals to serve key editorial board roles. Demonstrated managerial skills are also required to ensure content and issue development, and timely processing of submissions. Terms begin 1 January 2026.

For complete information on how to apply, please go to
www.computer.org/press-room/seeking-2026-editors-in-chief

Apply Today!

IEEE COMPUTER SOCIETY

IEEE

EDITOR: Alexander B. Magoun, a.b.magoun@ieee.org

DEPARTMENT: ANECDOTES

This article originally appeared in
IEEE Annals of the History of Computing
vol. 45, no. 4, 2023

Computer Networking Initiatives in One of the World's Remote Cities

T. Alex Reid, *The University of Western Australia, Crawley, WA, 6009, Australia*

This article describes some computing initiatives made by members of the University of Western Australia, located in arguably the most isolated capital city in the world. These initiatives center around online and networking capabilities, predominantly arising from the installation, in 1965, of the first time-sharing computer in Australia. This far-sighted, if risky, purchase set the university on a course that led to many more initiatives, encompassing significant computer resource sharing, a ground-breaking online library system, early online education programs, and an early multihost packet-switched network. Most research concerning isolation and innovation suggests that isolation operates as a break on innovation, but the Western Australian experience belies that conclusion.

WESTERN AUSTRALIA MADE A LATE START IN COMPUTING

Perth, the capital city of the State of Western Australia, is often considered to be the most isolated capital city in the world. Regarded by many at the time as something of a "quiet backwater," so it was not surprising that it was not until 1962 that Perth (and hence the whole of the 2.6 million square kilometres of Western Australia) took delivery of its first computer, a Bendix G15, bought by the Main Roads Department to help in the design of freeway extensions and interchanges [28]. It was followed that same year by an IBM 1620 bought by the University of Western Australia (UWA) [19], [28]. This was well after most other Australian capitals (and universities) had acquired computers [17], [18], [19].

Given this slow start, it is perhaps rather surprising that Perth did not long remain quiet in computing. By 1964, the capacity of the university's IBM 1620 was overwhelmed by demand from university and external users, and a search began for a much larger replacement. IBM were convinced that the university would just buy it from their stable, especially since they had just announced their System/360 range, being the first line of computers incorporating the same instruction set and designed for all users large and small [7], [14].

TIME-SHARING COMES TO AUSTRALIA

However, the university's Computing Centre Director, Dennis Moore, was familiar with MIT's multiuser, time-shared computing Project MAC, and believed that this was the way of the future [8], [11], [14]. There were only two companies worldwide at the time offering to sell such computers, though none had yet been sold. These were the Canadian division of the British firm Ferranti Limited, and the Digital Equipment Corporation (DEC) in Massachusetts, USA [1]. IBM did not support time-sharing on its System/360 range until 1970 [7].

The funding of public universities in Australia at the time was split equally between the Federal and State governments, with triennial grants made available for capital items such as computers. The university obtained such a capital grant and issued tenders. There were four contenders—-IBM, ICT (later ICL), DEC, and Ferranti. Each made presentations to the university's Computer Users Group [13]. The Ferranti-Packard salesman was very persuasive about the benefits of time-sharing and "almost had the Computer Users Group cheering" for its 6000 model, recounted Moore [14]. He was in fact selling the DEC computer [11], [13]. Designed originally to share computing with equipment operating in laboratories, the PDP-6 was a

1058-6180 © 2023 IEEE
Digital Object Identifier 10.1109/MAHC.2023.3325497
Date of current version 23 November 2023.

full time-sharing machine incorporating the hardware to do this efficiently [1].

Despite efforts by IBM to reverse the decision, the University resolved to buy a PDP-6. This was an extraordinary decision. Digital Equipment was a small, almost unknown, company; time-sharing was a great laboratory experiment, but there were no systems commercially available and in use in the field; and Perth was the furthest place in the world, the almost exact antipodes, from Digital's factory in Maynard, Massachusetts. When the computer was delivered in May 1965, it became the first time-shared computer to be delivered to Australia [2], and among the first in the world. The foresight and bravery of Moore and Birkett Clews and the university was quite remarkable.

Along with the computer came the DEC systems engineer responsible for developing the time-shared operating system, who proceeded to spend much time refining the software—he sometimes used a camp stretcher so he could work on the computer overnight [14]. Thus, time-sharing was introduced to Australia, incidentally bolstering Digital Equipment's market position there.

SHARED COMPUTING

Numerous terminals were connected to the PDP-6 over telephone lines. These were originally teletypes, adapted from telex machines, operating at 110 or 300 bits per second (approximately 10 and 30 characters per second), but the PDP-6 promised more than just connecting terminals; it was acquired in part to enable a range of laboratory experiments to connect and be directly controlled by the computer. These included a diffractometer and a spectrometer in the Physics Department, an analogue computer and field recorder in engineering, a flying spot scanner in the Crystallography Laboratory, a rat race and a perception laboratory operated by Psychology, and a flying spot scanner in the Pathology Department [14]. In the early days, these devices were interfaced by an electronics engineer, Ian Nicholls, employed by the Centre, who undertook some ground-breaking work [14].

Other agencies around Perth, including CSIRO and the technical divisions of various government departments, had drawn on the capabilities of UWA's IBM 1620. With the advent of the much more powerful PDP-6 in 1965, along with its time-sharing and remote connectivity capabilities, this usage burgeoned, encouraged by the university who saw this sharing almost as an obligation given the remoteness of Perth. This laid the groundwork for the high degree of collaboration and sharing that characterized public sector computing in Western Australia for many years to come, paving the way for the creation of the Western Australian Regional Computing Centre in 1972 (see below).

This acquisition of a revolutionary and architecturally pioneering computer system allowed a wide range of novel applications to be developed, in particular running online experiments in laboratories around the university. This was exemplified by the Department of Psychology, one of whose fields of research was in visual perception. Being able to use the computer in real time to control images that human subjects were allowed to see, and how they perceived them, enabled significant advances in understanding the human perception system [30]. This led subsequently to the invention of the Betagraph, a visual display system that relied on the human eye and brain's ability to fill in missing information, as when a moving car is perceived through a picket fence [30]. Controlling such experiments by computer is widespread today, but it was a new experience for researchers in the 1960s. Similar advances were made in the fields of crystallography, physics, and physiology, among others.

LATIN INSCRIPTIONS

Not all uses of the PDP-6 were based on its online capability, though that was often exploited at different stages of the project. An example was Professor E. John Jory's creation in the early 1970s of an index to all the inscriptions found on ancient monuments around Rome. These had been accumulating since 1862 into Volume VI of the *Corpus Inscriptionum Latinarum* (*CIL*). The solution was to create a key word in context (KWIC) index, but much of the data entry and program development was undertaken online. The data entry took well over two years, the main problem being that the inscriptions were two to three thousand years old, and every inscription entered produced something new to be programmed for, because there was no standard format. Resolving these issues with Jory able to access the data online from his office shortened the project considerably.

When the index's 7315 pages were published in 1975 by the Academy of Sciences of the German Democratic Republic, Jory received international recognition. Since the appearance of Jory's KWIC index, computer applications and databases have had a major influence on epigraphic studies [8], [10].

UNIVERSITY OF WA FORTRAN TRANSLATOR, UNIWAFT

Another project which did not altogether rely on the online nature of the PDP-6 was the development of a

Fortran compiler. The PDP-6's standard Fortran compiler produced correct, fast, and reliable code, but it was unsuitable for teaching purposes: it was slow to compile, produced arcane error messages, and stopped analyzing ("parsing") the program at the first error encountered. Of course, the online nature of the PDP-6 meant that many users could develop and test their Fortran programs online. However, the volume of students wishing to do so outstripped the computer's capacity to support them simultaneously. The goal was therefore to develop a fast Fortran compiler that would have rich error messages and which undertook as much parsing as possible at each attempt. The result, released for the start of the 1971 academic year, was a system called Uniwaft—University of Western Australia Fortran Translator. It was itself written in Fortran, which made coding the system that much faster—and eased its transport to subsequent computers [22], [24].

The system was extended in 1972 to WA State high schools, called Miniwaft, using specially prepared preprinted punched cards. Chads on these were pushed out using a paper clip, based on a similar arrangement developed at Monash University [31].

ONLINE EDUCATION

The potential for online teaching presented by the PDP-6 was not missed, even though systems like Uniwaft and Miniwaft were batch processing systems. Before 1977, there was no department of computer science at UWA, but Centre staff were active in the Australian Computer Society, with the WA Branch having been founded by the then Director, Dennis Moore [19]. Staff was keen to promote this new computer architecture and its implications for the future of computing. A series of professional development seminars was launched, with some using the PDP-6 [23], [26]. In particular, a simple database package, Data Management Package (DAMP), that implemented the CODASYL-linked database architecture, was used to teach database architecture. In one of the first examples of online education in Australia, students (in this case, Computer Society members) created an online database and interacted online with their data [26].

THE FOUNDATION FOR THE WA REGIONAL COMPUTING CENTRE, WARCC

One of the consequences of having a time-shared computer, featuring online connections on campus and off, was that the technical divisions of many government departments were able to connect to it remotely. UWA encouraged this sharing and established charges for computer use in order to ensure equitable access. Many of these external organizations in any case had close ties with the engineering or scientific disciplines within the university, who, like the Computer Society, ran regular professional development seminars for their alumni. Accordingly, considerable use was made of the PDP-6 by organizations like CSIRO, and the technical divisions of the Main Roads Department, the WA Water Authority, and the State Electricity Commission. In addition, a number of private engineering consultancy firms also became customers.

Inevitably, it was not long before the capacity of the PDP-6 was sorely tested. Initially, various upgrades were made (e.g., additional memory, magnetic tapes, and disk drives for storage and program "swapping"). But by 1970, it became clear that a major upgrade or replacement would be required [8], [14]. As indicated above, major purchases of this kind were funded on a triennial basis by the Australian Universities Commission (AUC), an agency of the Australian federal government. For the triennium 1970–1972, the AUC had ruled that it would only support grants to universities which were prepared to share computing resources, in order to take advantage of the economies of scale available at that time for large computers. Discussions ensued between the universities and colleges in all the Australian capitals to attempt to reach agreement. However, only in the case of Western Australia was agreement reached: A heritage of the sharing that had been happening there for several years. No university in any other city received a computer grant in that triennium [19].

In order to set this arrangement for sharing on a more formal basis, it was agreed to set up the Western Australian Regional Computing Centre (WARCC), housed and managed by the University of Western Australia, under the direction of a Board of Management made up of representatives of the major users (see [8, Section V]). The primary users would consist of the UWA, other tertiary education institutions, state government departments, and statutory authorities, and the CSIRO, serving in particular their scientific and engineering computing needs. The computer would be expected to support local batch processing, remote batch processing, and remote interactive processing (the role of direct computer-controlled laboratory experiments by that time had largely been taken over by minicomputers). The Centre would operate along commercial lines, charging all users (including the university) according to the computer time

FIGURE 1. Reconstructed IBM 1620 console [photo credit Alex Reid].

used, and all expenses (including staff salaries) would be met from this revenue. The university would provide accounting and some administrative services, along with power, cleaning, space, etc., and would bill the Centre accordingly. Computer charges were to be set at a level that would cover all expenses, and also to allow the accumulation of a small reserve for purchasing additional equipment.

WARCC officially came into existence on 1 January 1972, after over two years of discussion and negotiation [8], [13], [14]. With $A470,000 available from the 1970–1972 Triennium, and the prospect of similar funding available from the 1973–1975 Triennium, proposals were received from Digital Equipment Corporation for a PDP-10 (a later version of the PDP-6) and Control Data Corporation for a CDC 6400. In the end, and after some time, the decision was made to purchase a Cyber 72 (a later model of the 6400) from CDC, which was delivered to the Centre, by then housed in a new extension to the university's Physics Building, in August 1972 (see Figure 3).

FIGURE 2. The PDP-6 after delivery to UWA, L to R: DEC Australian sales manager Ron Smart, UWA deputy vice-chancellor professor John Birkett Clews, UWA Computing Centre director Dennis Moore [photo credit UWA].

FIGURE 3. Front entrance to WARCC, UWA [photo credit UWA].

The Cyber 72 was the most powerful computer yet installed in WA, and was among the most powerful in the country (see [8, Section V-E]). It had 60-bit words (compared with the PDP-6's and PDP-10's 36-bit words), and all peripheral input/output activity (e.g., disk drives, tape drives, card reader, printer, remote communications) was handled by multiple peripheral processors, so that the main central processing unit was dedicated to calculating, at which it excelled. The 60-bit words gave it very high precision in arithmetic calculations, which greatly pleased the scientists and engineers among its users. This Cyber 72 was later upgraded to a Cyber 73, and subsequently to a Cyber 720. Curiously, and presumably for marketing purposes, the Cyber 72 was in fact identical to the Cyber 73, but incorporated a circuit designed to slow it down; upgrading it to a Cyber 73 involved simply removing that circuit (and the payment of additional funds). The computer ran the SCOPE (later NOS/BE) operating system and provided a wide range of scientific and other software. The Uniwaft Fortran compiler was transported to the Cyber with little difficulty.

Soon, many terminals and a good number of remote job entry systems were connected to the Cyber from across Perth. The PDP-6 had been transferred to WARCC in 1972, and its time-sharing feature was still very much favored for certain applications. In due course, therefore, it was decided to replace it with a compatible, but much faster PDP-10 with its KA10 CPU. Use of these systems continued to grow, and

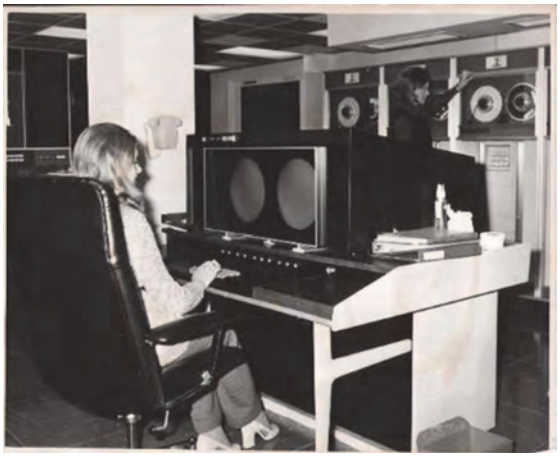


FIGURE 4. The main console of the cyber72 [photo credit UWA].

WARCC with it, with regular upgrades of the main computing equipment funded by the growth in usage and thence revenue (the PDP-10 was replaced by a DECsystem-10 with a KL10 CPU in March 1980). At the same time, the WARCC diversified its services into applications software programming, networking, minicomputer support, and subsequently microcomputer sales and support, and facilities management (e.g., housing and operating the State Health Department's computers).

By 1990, WARCC had a staff of over 100, an annual turnover of $A10 million, and was highly regarded throughout the State (see Figures 5 and 6 [27]). Indeed, the Public Accounts and Expenditure Review Committee of the WA State Legislative Assembly strongly recommended the WARCC model of computing for the rest of the WA public sector [29]. Ultimately, however, mainframe computers became uneconomical in this environment, and by 1992 those at WARCC were decommissioned, with the Centre's diversified

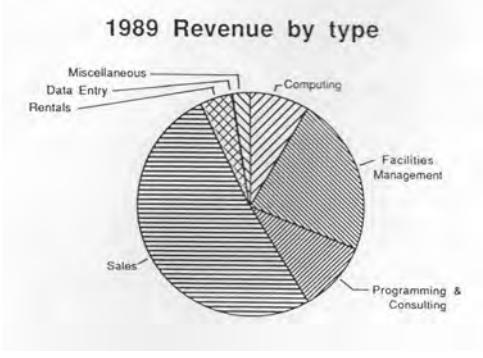


FIGURE 5. WARCC sources of revenue 1989 ([27, p. 1]).

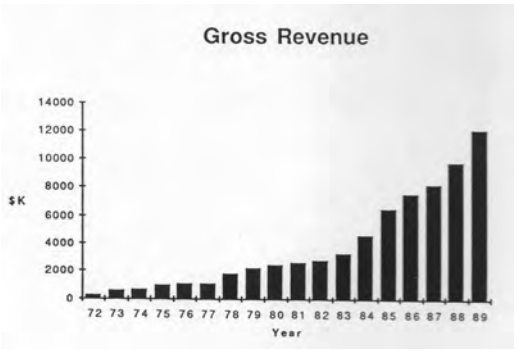


FIGURE 6. WARCC revenue growth 1972–1989 ([27, p. 1]).

activities ensuring its continued economical operation [32]. By then, WARCC had been operating successfully for 20 years, with gross revenue growing at an annual compound rate of 25% (see Figure 6), a testament to the networking among public-sector computer users in WA, seeded by what the PDP-6 had initiated.

LIBRARY CIRCULATION SYSTEM, LOANLY

Inspired by the power of online systems, David Noel, the systems librarian in the University Library, had a vision for a loan system where library users recorded their loans themselves using online terminals [15]. Some public libraries in the U.K. had embarked on this course, but no university or college libraries worldwide could be found to have attempted this. The volumes in the UWA library had already each been equipped with an 80-column punched card, which provided the title, author, and basic identification data; students and staff of the university had identification cards that resembled abbreviated, plastic, punched cards containing their unique staff or student number.

FIGURE 7. LOANLY users at work [photo credit UWA].

Consequently the library commissioned WARCC to develop this system, with custom-built terminals for reading book and borrower cards from a U.K. company. A DEC PDP-11/40 was selected, and programming began at the start of 1973. Despite some setbacks, a working system was implemented by mid-1975 [15], [25]. This integrated the records of books that had been borrowed with records of all the books in the library's collection. This enabled remote catalog enquiries: if sought items were on loan, they could be recalled by the enquirers themselves online. Thus was implemented a radical, self-charging, university circulation system and described at the time as "trend-setting" [4], [16]. Its name, LOANLY, was a homophonic nod to the character in the very popular, if bleak, TV spy series *Callan*. A caricature of Lonely was used in promoting the system.

Some valuable lessons about the nature of online systems were learnt through this project and conveyed to the Australian computing community through a lecture tour conducted by Alex Reid, together with a lead article in the *Australian Computer Bulletin* [20].

Of course, circulation control is just one of many applications of automation in libraries. Since it is perhaps where library staff time could save the most time, it has generally been the first function to be automated. Other applications followed, including acquisitions, catalog maintenance, serials management. Initially developed independently, commercial providers overtook in-house applications in the mid-1980s with comprehensive software systems like URICA and LIBERTAS [4], [9].

PACKET-SWITCHED NETWORKING

Based on its early start in data communications, connecting ON- and OFF-campus remote terminals to a central computer over telephone lines, WARCC soon realized that data communications networks represented significant potential. Initially, the focus had been upon hardware developments, to facilitate connecting terminals. These included the design and construction of modems (to enable reliable transmission over long distances), and multiplexors—-to enable sharing of communications links by several terminals [13]. In the late 1960s and early 1970s, the cost of telephone lines in Australia was relatively prohibitive—in part due to its long distances, but also due to the monopoly enjoyed by the Postmaster-General's Department (later Telecom), which was not only the sole provider but was also the regulator. For example, the Computing Centre staff had developed modems for use ON-campus, but was unable to get them approved for use off-campus [8], [14].

The problems encountered in licensing data communications equipment was one reason why hardware initiatives gave way to the use of software to enhance data communications capabilities. The main reason was however the flexibility that software afforded, and it was clearly going to be a much more fruitful approach in the long run. Several networking projects were initiated, starting with the clustering of remote terminals. Some computer manufacturers already had "terminal cluster" systems, but they were very expensive and not particularly flexible. It was decided instead to buy small minicomputers (e.g., the DEC PDP-11/10) and program them to emulate these terminal clusters. This relatively simple adaptation increased staff competence in communications software that opened up many more possibilities [21].

At that time (1973) WARCC operated a Cyber 73; it also still ran the PDP-6 then the PDP-10. It had become obvious quite early on that these two architectures complemented each other rather well—one being a powerful batch processor, with the other ideal for remote multitasking. Neither firm's machines could undertake both modes well together.

A PDP-11/10-based terminal cluster system was installed at the Western Australian Institute of Technology (later, Curtin University) to connect the institute to the WARCC PDP-10. It was also found beneficial to replace the terminal handling system on the PDP-10 with a similar minicomputer, emulating the DEC-supplied interface. Beyond the significant savings, this move more importantly opened up the opportunity to adapt both sets of emulators to accommodate traffic destined for the Cyber as well as the PDP-10. The Cyber relied heavily on remote job entry (RJE) systems, which provided a card reader and printer, and a communications link to the Cyber: This enabled remote users to submit their jobs to the Cyber using equipment on their own premises. These RJE systems were also expensive, and replacing them by much cheaper emulators had the added benefit that these could be programmed to perform other functions. One of these was to turn them into conversational RJEs (CRJEs), whereby remote users could develop their programs interactively, submit them to the Cyber for processing, and interact online with the results [13], [21].

Another challenge which proved to be simple given the background of the WARCC Network team, was interfacing the Cyber and DECsystem-10 directly with each other, so that data files could be moved between the two systems. It had become clear that the ideal

way to manage data traffic between computers and end-user devices was to employ "packet switching." As early as 1974 the Centre's staff began researching the technique, but trod very cautiously in view of others' failure in this field. By 1977 packet switching had become fairly well-established overseas, e.g., with ARPANET (precursor to the Internet [6], [12]) connecting some universities and research laboratories in the USA, but it was not extended to Australia. The monopoly data communications provider in Australia, Telecom, had plans for a common packet-switching service, but was slow to implement them [3], [19].

Meanwhile, talk of a Western Australian Regional Computer Network started to gain momentum, again with some caution, lest some agencies felt that this was a political move to grab control of public sector computing [8], [13]. But no one could deny the logic that, in a state as large as WA, and with such a scattered population, it made sense for public sector entities to share communication lines to remote centers. While the political investigations continued, with the State government, for instance, collecting data on traffic volumes and projections, WARCC continued to investigate the technology.

Overall, the goal was to devise a networking environment, which would rationalize the diversity of communications links required, while providing a vehicle for both terminal and file traffic between multiple hosts and other networks.

Computer suppliers were canvassed, but few rose to the challenge. Ultimately, only DEC's offering, DECNET, seemed promising, and WARCC launched a project to experiment with it. It was found to be surprisingly straightforward to connect the DEC-10 at WARCC with one that by then had been installed at WAIT. By the spring of 1977, a packet-switched multihost network had been installed between WARCC and WAIT that employed a 4800-bps synchronous line rented from Telecom, using PDP-11/40-based DN87 interfaces in front of each DEC-10 [21]. It performed remarkably well. The team then interfaced WARCC's Cyber to this network through emulation—making the Cyber look like a DEC-10 to the network and the network look like a standard terminal cluster to the Cyber. No changes were made to the operating systems of either the DEC-10 or the Cyber—an important requirement to minimize future support.

By the beginning of 1978 the Cyber had joined this fledgling, packet-switched, multihost network. The next phase of expansion was to connect PDP-11 computers at various agencies across Perth. This proved surprisingly difficult, as it was discovered that there were two incompatible versions of the DECNET

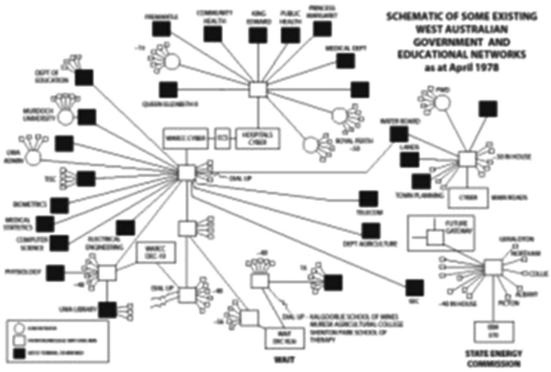


FIGURE 8. WA regional network in 1978 (from [13]).

software and protocols. In the end, the same approach was taken as with the Cyber, that is making the PDP-11s look like DEC-10s to the network; the first such link was installed early in 1978 on the PDP-11/34 at the Nedlands College of Advanced Education over a 2400-bps line. The network expanded rapidly, with a CRJE being connected, as well as the Cyber 172, at Main Roads Department, along with various other minicomputers and terminal clusters. Toward the end of 1980, when CSIRONET became operational, a two-way gateway based on a PDP-11/40 was installed. Work continued on connecting other public-sector computers, such as the Interdata 8/32 at Murdoch University, and various IBM systems that were installed across the public service, as well as gateways to other national and international networks, such as ARPANET [21].

Thus, what appears to be the first packet-switched multihost network in Australia came into operation by early 1978 (see Figure 8). It was recognized that there were several overheads in employing packet-switching (as opposed to use of devices like PABX's), but it was clear that this was the path of the future in view of its flexibility and versatility, and this has indeed been borne out by history. Interestingly, as with ARPANET, one of the chief benefits of this regional network was the widespread uptake in the use of email [21].

EPILOGUE

Research findings from a variety of sources (e.g., [5]) suggest strongly that isolation impedes innovation. However, the evidence from the projects described in this article, originating in arguably the world's most isolated city, would seem to belie that formulation. The cited research is based on patents and start-ups, but those are only one measure of innovation. No patents were sought from any of UWA's initiatives, and there were no notable start-ups, with one significant

exception—Monte Sala, whose work is described elsewhere in this issue.

The simplest explanation of why so many notable initiatives arose at UWA is the early start, which it gained in online systems and networking through its bold purchase of Australia's first time-shared computer. All the initiatives described have at their core the online paradigm and demonstrate a progressive realization of computer networking. The networking went beyond physical data communications to foster the close-knit networking of computer users throughout the city of Perth. At the time of these developments, Perth was a city of half a million people, but has since quadrupled in population. It has lost the characteristic which led Dennis Moore to observe that it was small enough for people to be able to talk to each other, but large enough to support technical initiatives [13], [14]. Now that further developments of networking, in particular the Internet, have shrunk distances worldwide, Perth can no longer be considered a "backwater"—if indeed it deserved that epithet 60 years ago.

ACKNOWLEDGMENTS

The author would like to gratefully acknowledge the very helpful advice given by the reviewers and editors. Any remaining errors are entirely the author's responsibility.

BIBLIOGRAPHY

[1] G. Bell, C. Mudge, and J. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*. Bedford, MA, USA: Digital Press, 1978.

[2] P. Budne, "DEC PDP-6 serial numbers," Apr. 2022. Accessed: Sep. 12, 2023. [Online]. Available: https://www.ultimate.com/phil/pdp10/pdp6-serials.html

[3] R. Clarke, "Origins and nature of the internet in Australia," 2004. Accessed: Sep. 12, 2023. [Online]. Available: https://www.rogerclarke.com/II/OzI04.html

[4] L. Clyde and M. Middleton, "Library automation in Western Australia," *Library Autom. Syst. Inf. Exchange*, vol. 21, no. 1, pp. 5–15, Jul./Aug. 1990.

[5] A. Contigiani and M. Testoni, "Geographic isolation, trade secrecy, and innovation," *Res. Policy*, vol. 52, no. 8, Oct. 2023, Art. no. 104825, doi: 10.1016/j.respol.2023.104825.

[6] S. D. Crocker, "Arpanet and its evolution–A report card," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 118–124, Dec. 2021.

[7] B. O. Evans, "System/360: A retrospective view," *IEEE Ann. Hist. Comput.*, vol. 8, no. 2, pp. 155–179, Apr.–Jun. 1986, doi: 10.1109/85.150016.

[8] K. Falloon, "Cyberhistory," M.S. thesis, Univ. Western Australia, Crawley, WA, Australia, 2001. Accessed: Sep. 12, 2023. [Online]. Available: https://research-repository.uwa.edu.au/en/publications/cyberhistory

[9] H. Groenewegen, "Four decades of library automation: Recollections and reflections," *Australian Library J.*, vol. 53, no. 1, pp. 39–53, Feb. 2004.

[10] E. J. Jory, "Problems and prospects for the production of computer compiled indices to epigraphic works," *Antiquities Africaines*, vol. 9, no. 1, pp. 15–22, 1975.

[11] J. A. N. Lee, R. M. Fano, A. L. Scherr, F. J. Corbato, and V. A. Vyssotsky, "Project MAC (time-sharing computing project)," *IEEE Ann. Hist. Comput.*, vol. 14, no. 2, pp. 9–13, 1992.

[12] B. Leiner et al., "Brief history of the internet," *Internet Soc.*, 1997. [Online]. Available: https://www.internetsociety.org/internet/history-internet/brief-history-internet/

[13] D. Moore, "Computers, communications and cooperation," Annual WA Comput. Soc. Conf. Bunbury, 1978. Accessed: Sep. 12, 2023. [Online]. Available: https://alex-reid.com/History/Moore-Bunbury-ACS-1978.pdf

[14] D. Moore, interview with Penny Collings for the National Library's, "History of ICT in Australia oral history project," recorded Feb. 7, 2015. Accessed: Sep. 12, 2023. [Online]. Available: http://catalogue.nla.gov.au/Record/6807543?lookfor=ORAL%20TRC%20moore%20%23[format:Audio]&offset=4&max=128, https://alex-reid.com/History/Dennis-Moore-7Feb15-complete-transcript.htm

[15] D. G. Noel and T. A. Reid, "System and file design in the LOANLY real-time circulation system," in *Proc. 17th Biennial Conf.*, 1973, pp. 479–490.

[16] D. G. Peake, "Library automation in Australia: The state of the art," *Prog., Electron. Library Inf. Syst.*, vol. 15, no. 1, pp. 11–23, 1981.

[17] Pearcey Foundation, "CSIRAC among the first electronic stored program computers," 2021. Accessed: Sep. 12, 2023. [Online]. Available: https://www.pearcey.org.au/initiatives/csirac/csirac-among-the-first-electronic-stored-program-computers/

[18] T. Pearcey, *A History of Australian Computing*. Dandenong, Australia: Chisholm Inst. Technol., 1988.

[19] G. Philipson, "A vision splendid: The history of Australian computing," Australian Comput. Soc., 2017. Accessed: Sep. 12, 2023. [Online]. Available: https://www.acs.org.au/content/dam/acs/acs-publications/ACS-ebook-2017_A-Vision-Splendid_The-History-of-Australian-Computing.pdf

[20] A. Reid, "The trials and tribulations of an on-line computer project," *Australian Comput. Bull.*, vol. 2, no. 1, pp. 6–14, Feb. 1978.

[21] A. Reid, "Network developments at WARCC 1965-1980," *Australian Comput. J.*, 1981. Accessed: Sep. 12, 2023. [Online]. Available: https://alex-reid.com/History/Network-Developments-WARCC-1981.pdf
[22] T. A. Reid, "A guide to Fortran programming and UNIWAFT," UWA Computing Centre, Perth, Western Australia, 1971.
[23] T. A. Reid, "Programming for on-line data management," Australian Computer Society Professional Development Seminar, 1971.
[24] T. A. Reid, "UNIWAFT— The University of WA Fortran translator," in *Proc. 5th Australian Comput. Conf.*, 1972.
[25] T. A. Reid, D. G. Noel, and C. P. R. Greaves, "LOANLY— A real-time library circulation system," in *Proc. 6th Australian Comput. Conf.*, 1974, p. 45.
[26] T. A. Reid and D. G. Moore, "Computer programming and data management for on-line systems," Univ. Western Australia Extension Service, 25-week course, 1971.
[27] T. A. Reid, "Director's annual report," WA Regional Computing Centre, Perth, Western Australia, 1990.
[28] T. A. Reid, "Computing," in *Historical Encyclopedia of Western Australia*, J. Gregory and J. Gothard, Eds. Perth, Western Australia: UWA Press, 2009, pp. 223–225.
[29] E. S. Ripper and G. I. Gallop, "Report on computing in government," Public Accounts and Expenditure Review Committee, WA Legislative Assembly, Western Australia, Australia, 1990.
[30] J. Ross, "A new type of display relying on vision's sensitivity to motion," *J. Physiol.*, vol. 271, no. 2, pp. 2P–3P, Oct. 1977.
[31] L. G. Whitehouse, "MINITRAN Monash University student fortran," Monash Univ. Computer Centre, Melbourne, Australia, 1969.
[32] B. Whitford, "Mainframes and micros: Western Australian regional computing centre," in *Computer Excellence: Computer Companies in Australia are Seeking and Attaining Excellence*, B. Whitford, Ed. Perth, Australia: Beaumont Publishing House, 1991, pp. 76–81.

T. ALEX REID is an honorary professorial fellow at the University of Western Australia, Crawley, WA 6009, Australia. Contact him at alex.reid@uwa.edu.au.

IEEE COMPUTER SOCIETY D&I FUND

Drive Diversity & Inclusion in Computing

Supporting projects and programs that positively impact diversity, equity, and inclusion throughout the computing community.

DONATE TODAY!

IEEE COMPUTER SOCIETY

IEEE Foundation

EDITOR: Phil Laplante, The Pennsylvania State University, plaplante@psu.edu

DEPARTMENT: SOFTWARE ENGINEERING

This article originally appeared in
Computer
vol. 56, no. 1, 2023

Lessons From the Father of Software Engineering

Ricardo Valerdi, *University of Arizona*

The recent passing of Barry Boehm motivated the opportunity to synthesize lessons from his contributions in software engineering. This article summarizes six lessons that impacted how software is designed, built, and managed.

Former Brooklyn Dodgers baseball player Jackie Robinson once said, "A life is not important except in the impact it has on other lives." Barry Boehm, known as the "father of software engineering," was a perfect example of this. His contributions literally changed how software is builtbuilt—from Beijing to London and everywhere in between. More importantly, he influenced academics and practitioners in the field to think about how to embed more scientific rigor and discipline into software engineering.

It would be impossible to summarize all of Barry's contributions in one article. Instead, it would be more beneficial to identify some of the lessons learned that every software developer and software manager should know. In fact, I strongly believe that Barry's contributions apply beyond the context of software development. Their relevance to product development and technology management is driven by the broad applicability of his work. Here are six lessons I learned from having Barry as my mentor during my doctoral studies and subsequently as a collaborator for the past 20 years. (See also "Barry on the Tennis Court.")

LESSON 1: STRIVE FOR WIN-WIN

Barry's objective function was to make everyone a winner. He referred to this as win-win, or *Theory W*. The idea was that software project managers will be fully successful if and only if they make winners of all of the other participants in the software process: superiors, subordinates, customers, users, maintainers, and so on.[1] If any success critical stakeholder is left out, then the project is at risk of failing. This expanded focus captured the many views of how we define success for software developments. Rather than emphasizing just performance, Barry valued happy developers, satisfied superiors, and pleased users equally.

The characterization of a manager as a negotiator was innovative at the time because it shifted the emphasis from an organizer of tasks to a "packager of solutions." It also placed a focus on the developer and the user as equal participants. Barry understood that people are motivated by good will and self-interest. Accordingly, it was important to let their value preferences surface to understand what features were most critical. This idea applies to any project that has multiple collaborators. Whether the objective is to build a sports arena or to write a new corporate policy, Theory W reminds us that the most important thing is to ensure which success critical stakeholders need to be satisfied.

LESSON 2: RESOLVE MODEL CLASHES

Sometimes it is impossible to make everyone happy. Barry developed a simple way to think about such "sticky" situations through the *model-based systems architecting and software engineering approach*.[2] Disagreements between parties were reframed into "clashes" between them and represented as different types of models: product models, process models, property models, and success models. Through this reframing of preferences, it became possible for success-critical stakeholders to visualize or reason about the prospective system and its likely effects to better deal with them. These models captured product attributes; the process for developing the system; properties such as cost, performance, or dependability; what it means for the system to be successful; and the means

Digital Object Identifier 10.1109/MC.2022.3219527
Date of current version: 9 January 2023

 Published by the IEEE Computer Society

to resolve the many conflicts that occur as the product is architected, designed, developed, tested, and fielded.

An example of a model clash is when a product model aims for high reliability and high performance, while a property model aims for low cost and a short development schedule. Barry's approach gave stakeholders a way to surface their preferences, perform tradeoffs, and come to the negotiating table with specific needs. This facilitated negotiations about things that mattered the most by requiring collaborators to prioritize and reach consensus on what was important. The simplicity of this approach facilitated its use throughout the software community. But model clashes occur in areas outside of software too. As a result, the approach was put to work across systems, hardware, and software communities to resolve conflicts and reach decisions as to what was best for all of those concerned.

LESSON 3: ITERATE TO GAIN KNOWLEDGE AND UNDERSTAND STAKEHOLDER PREFERENCES

Another of Barry's fundamental contributions was the idea that software development should be an iterative process, rather than a specification-driven process. This approach, appropriately called the *spiral model*, was created to overcome the limitations of the waterfall model, which recommended that software be developed in successive stages.[3] Such top-down structured approaches had other difficulties, such as assuming uniform progression of the system's evolution and the inability to accommodate software reuse, among other things.

The innovation of the spiral model is that it reframed software development as an incremental process with multiple milestones along the way. Ultimately, the goal was to identify risks early in the development process and resolve them as early as possible. The spiral model eventually became the universal software development process. Barry once said that he knew he had made it when he read about the spiral model in a Dilbert cartoon. Such an iterative approach can be useful in situations where the end result is not well defined and would benefit from incremental steps that provide information and feedback along the way. Each iteration, or spiral, allows stakeholders to identify risks, experiment with solutions, identify corrective actions, and reevaluate the results, which would ultimately lead to better products.

BARRY UNDERSTOOD THAT PEOPLE ARE MOTIVATED BY GOOD WILL AND SELF-INTEREST.

LESSON 4: A SOLUTION THAT IS TOO EXPENSIVE IS NOT A SOLUTION

Economic analysis techniques were relatively new to software engineering in the 1980s. Limited computing power and memory increased the importance of cost-benefit analyses of software product features. Barry knew that "it is often worth paying for information because it helps us make better decisions."[4] To make this possible, he developed the *constructive cost model* (*COCOMO*). Besides helping people understand the cost consequences of their decisions, the model allowed its users to conduct tradeoffs and come up with a plan that helped them realize the goals set by collaborators. Barry elevated the importance of cost as a critical decision criterion for projects and democratized the ability for people to generate their own cost estimates based on a set of parameters that were known to influence the cost of software.

COCOMO became the standard measuring stick for industry and government to better understand how to keep software development affordable. It also allowed decision makers to quantify the effects of different parameters on cost, such as software reliability, programmer capability, and project schedule. The broad adoption of COCOMO gave rise to the development of algorithmic models that could be used to forecast the cost and schedule of projects in their early stages.

BARRY ON THE TENNIS COURT

Barry Boehm was known to many as a humble intellectual—a gentle giant in our field. But if you ever faced him on the tennis court, you would witness an entirely different side of him. With a racquet in his hand, the man was a beast. His style of play was aggressive, and his stamina on the court was on par with a Wimbledon champion because he ran three miles every morning.

When I first met Barry in the early 2000s as a new graduate student in his lab at USC, he was wearing a Wimbledon sweater. I didn't think much of it. I thought he was simply a tennis fan. Barry walked slowly, spoke quietly, and took *veeeeeery* long pauses between thoughts. I asked him if he played tennis, and he confidently said, "I do!" and invited me to play. I thought we would meet at the courts on campus or maybe at a park in Santa Monica. Instead, he invited me to his house because he had his own tennis court in his backyard. That's when I realized my professor was serious about tennis.

Going to someone's house to play tennis is not an everyday occurrence, especially in urban Los Angeles. Nevertheless, I was confident in my tennis game because I had played competitive tennis since I was young. When I showed up at his house, Barry was dressed in old, beat-up tennis attire from the 1980s. His wardrobe featured a partially ripped shirt, shoes with holes in them, and glasses that fit crookedly on his face. I soon learned that this was simply a facade. He was about to rain down aces on me and approach the net on every single point. Every point. Who does that? Barry did. Because Barry learned to play tennis in the 1950s when the racquets were wooden and the only way to hit a forehand was with a long backswing and a long follow-through. One-handed backhands were the norm, and (as Barry demonstrated) the sooner you came up to the net to hit a volley, the more likely you were to win the point.

Barry's game wasn't his only secret weapon. He had home court advantage. This came from knowing where all of the cracks were on the tennis court. You see, Barry's house was near the Santa Monica fault line, which meant that the ground moved over time. This led to sporadic cracks on one side of the tennis court. They were small cracks, but if a tennis ball bounced on one, there was no way of predicting where it would go. Barry seemed to always aim for the cracks and would occasionally hit them. When he did, it was impossible to return the ball. It was an unfair match. An opponent in great physical condition with an aggressive approach and the home court advantage was a difficult one to beat.

During the five years of being Barry's grad student, I did not beat him once. As soon as I defended my doctoral dissertation, I decided it was time for his streak to end. (By this time I had learned where all the cracks were.) But, more importantly, I had learned to never hit it to his forehand because it was his aggressive side. I loved playing tennis with Barry because he loved to compete. Initially, I was embarrassed to lose to someone 40 years older than me, but he was the best player of his generation I had ever played against. When I am his age, I hope to be half as talented, half as humble, and half as competitive. I will dearly miss my professor and tennis partner, but I recognize that the ball is in my court to carry on his legacy.

IF TWO DESIGNS HAVE THE SAME FUNCTIONALITY BUT DIFFER IN COSTS, A MODEL LIKE COCOMO CAN HELP BREAK THE TIE.

This capability is also helpful when comparing possible design alternatives and conducting a wide range of tradeoffs. If two designs have the same functionality but differ in costs, a model like COCOMO can help break the tie. Alternatively, if a project exceeds a budget, then either its forecasted cost or schedule should be adjusted or it should be declared infeasible.

LESSON 5: BE PROBLEM DRIVEN INSTEAD OF METHOD DRIVEN

As a researcher, Barry was not limited to a fixed set of tools or methods. Instead, he was motivated by real-world problems that required robust solutions and an open mindset. The best example is the COCOMO suite of models, which was motivated by questions that the basic COCOMO model could not answer. The COCOMO model answered two questions: How many software developers do I need to build this system, and how long will they take to complete the project? As the sophistication of software products evolved over time, new development paradigms emerged, the

effects of which could not be covered by COCOMO. One of these paradigms was the increased use of commercial off-the-shelf (COTS) software to accelerate development schedules and reduce labor costs. Another was the deployment of agile methods. To address these new challenges, Barry developed new calibrations or other models, like COCOTS.[5]

This important lesson from Barry's work also highlights his versatility as a researcher. By being problem driven, Barry focused on identifying a problem to be solved and then worked diligently to solve it. This resulted in an impactful body of work that has left a lasting legacy in software engineering. Such an approach can apply anywhere there are challenging problems to solve. Being method driven might still be helpful in some situations, but one could become trapped by "having a hammer and only being interested in nails." Barry has demonstrated that being problem driven can be much more gratifying in the long run.

LESSON 6: EMBRACE UNCERTAINTY

Barry's approach to being risk driven in a world with model clashes was centered on the idea that life is full of uncertainties. To illustrate this notion, he popularized the *cone of uncertainty*, which is the idea that uncertainty decreases at an increasing rate as a project progresses.[4] Cost estimates are subject to a high degree of uncertainty at the beginning of a project when information is limited. As more information is obtained from prototypes, user feedback, and iterative development, the uncertainty tends to decrease.

The applications of the cone of uncertainty extend to scenarios where uncertainty must be accounted for in a forecast. This notion is used in hurricane forecasting, where the projected path of a hurricane is bounded by an error cone. Instead of uncertainty decreasing as in software development, uncertainty increases to reflect the error associated with the possible paths of a storm. In both cases, uncertainty is incorporated into a forecast, which leads us to embrace uncertainty by modeling it rather than being intimidated by it.

The underlying premise of the cone of uncertainty is that risk needs to be quantified and tradeoffs must be made to perform this quantification. Such tradeoffs allow "value" to be determined as different parameters are evaluated and tradeoffs are made (cost versus schedule, cost versus reliability, schedule versus personnel availability, and so on). Such tradeoffs are important in light of current development frameworks because they allow decision makers to look at the big picture rather than just at the economics of the situation.

THE UNDERLYING PREMISE OF THE CONE OF UNCERTAINTY IS THAT RISK NEEDS TO BE QUANTIFIED AND TRADEOFFS MUST BE MADE TO PERFORM THIS QUANTIFICATION.

Just as Babe Ruth was known as the greatest baseball player of all time, Barry is considered to be the best software engineer of all time. Barry was a Fellow of multiple professional societies, including IEEE, and he had the power to convene thought leaders from around the globe to move the field forward. Barry was an intellectual giant: every one of his contributions was a home run. Most important of all, Barry was the Most Valuable Person.

REFERENCES

1. B. W. Boehm and R. Ross, "Theory-W software project management principles and examples," *IEEE Trans. Softw. Eng.*, vol. 15, no. 7, pp. 902–916, Jul. 1989, doi: 10.1109/32.29489.
2. B. W. Boehm, "Escaping the software tar pit: Model clashes and how to avoid them," *SIGSOFT Softw. Eng. Notes*, vol. 24, no. 1, pp. 36–48, Jan. 1999, doi: 10.1145/308769.308775.
3. B. W. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Softw. Eng. Notes*, vol. 11, no. 4, pp. 14–24, Aug. 1986, doi: 10.1145/12944.12948.
4. B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.
5. B. W. Boehm et al., *Software Cost Estimation With COCOMO II*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2000.

RICARDO VALERDI is a professor and head of the Department of Systems & Industrial Engineering at the University of Arizona, Tucson, AZ 85721 USA. He completed his Ph.D. at the University of Southern California under Barry Boehm's mentorship and played many tennis matches against him. Contact him at rvalerdi@arizona.edu.

Conference Calendar

IEEE Computer Society conferences are valuable forums for learning on broad and dynamically shifting topics from within the computing profession. With over 200 conferences featuring leading experts and thought leaders, we have an event that is right for you. Questions? Contact conferences@computer.org.

JANUARY

4 January

- VLSID (Int'l Conf. on VLSI Design and Int'l Conf. on Embedded Systems), Bangalore, India

14 January

- ICOIN (Int'l Conf. on Information Networking), Chiang Mai, Thailand

27 January

- AIxVR (IEEE Int'l Conf. on Artificial Intelligence and eXtended and Virtual Reality), Lisbon, Portugal

FEBRUARY

9 February

- BigComp (IEEE Int'l Conf. on Big Data and Smart Computing), Kota Kinabalu, Malaysia

17 February

- ICNC (Int'l Conf. on Computing, Networking and Communications), Honolulu, Hawaii, USA

26 February

- VISIGRAPP (Int'l Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications), Porto, Portugal
- WACV (IEEE/CVF Winter Conf. on Applications of Computer Vision), Tucson, USA

MARCH

1 March

- HPCA (IEEE Int'l Symposium on High Performance Computer Architecture), Las Vegas, USA

4 March

- SANER (IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering), Montreal, Canada

8 March

- VR (IEEE Conf. Virtual Reality and 3D User Interfaces), Saint Malo, France

17 March

- PerCom (IEEE Int'l Conf. on Pervasive Computing and Communications), Washington, DC, USA

31 March

- DATE (Design, Automation & Test in Europe Conf.), Lyon, France
- ICSA (IEEE Int'l Conf. on Software Architecture), Odense, Denmark
- ICST (IEEE Conf. on Software Testing, Verification and Validation), Napoli, Italy

APRIL

9 April

- SaTML (IEEE Conf. on Secure and Trustworthy Machine Learning), Copenhagen, Denmark

16 April

- COOL CHIPS (IEEE Symposium on Low-Power and High-Speed Chips and Systems), Tokyo, Japan

22 April

- PacificVis (IEEE Pacific Visualization Conf.), Taipei City, Taiwan

26 April

- ICSE (IEEE/ACM Int'l Conf. on Software Eng.), Ottawa, Canada

28 April

- VTS (IEEE VLSI Test Symposium), Tempe, USA

MAY

4 May

- ARITH (IEEE Symposium on Computer Arithmetic), El Paso, USA
- FCCM (IEEE Annual Int'l Symposium on Field-Programmable Custom Computing Machines), Fayetteville, USA
- MOST (IEEE Int'l Conf. on Mobility, Operations, Services and Technologies), Newark, USA

2469-7087/24 © 2024 IEEE

5 May
- HOST (IEEE Int'l Symposium on Hardware Oriented Security and Trust), San Jose, USA

11 May
- ISPASS (IEEE Int'l Symposium on Performance Analysis of Systems and Software), Ghent, Belgium

12 May
- SP (IEEE Symposium on Security and Privacy), San Francisco, USA

19 May
- CCGrid (IEEE Int'l Symposium on Cluster, Cloud and Internet Computing), Tromsø, Norway
- ICDE (IEEE Int'l Conf. on Data Eng.), Hong Kong

26 May
- FG (IEEE Int'l Conf. on Automatic Face and Gesture Recognition), Tampa/Clearwater, USA

JUNE

2 June
- MDM (IEEE Int'l Conf. on Mobile Data Management), Irvine, USA

3 June
- IPDPS (IEEE Int'l Parallel and Distributed Processing Symposium), Milano, Italy

4 June
- ISMVL (IEEE Int'l Symposium on Multiple-Valued Logic), Montreal, Canada

10 June
- CVPR (IEEE/CVF Conf. on Computer Vision and Pattern Recognition), Nashville, USA

18 June
- CBMS (IEEE Int'l Symposium on Computer-Based Medical Systems), Madrid, Spain
- ICHI (IEEE Int'l Conf. on Healthcare Informatics), Rende, Italy

21 June
- ISCA (ACM/IEEE Annual Int'l Symposium on Computer Architecture), Tokyo, Japan

23 June
- CSF (IEEE Computer Security Foundations Symposium), Santa Cruz, USA
- DSN (Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks), Naples, Italy
- SVCC (Silicon Valley Cybersecurity Conf.), San Francisco, USA

26 June
- IEEE Cloud Summit, Washington, DC, USA

30 June
- EuroS&P (IEEE European Symposium on Security and Privacy), Venice, Italy
- ICME (IEEE Int'l Conf. on Multimedia and Expo), Nantes, France

JULY

6 July
- ISVLSI (IEEE Computer Society Annual Symposium on VLSI), Kalamata, Greece

7 July
- IOLTS (IEEE Int'l Symposium on On-Line Testing and Robust System Design), Ischia, Italy
- SERVICES (IEEE World Congress on Services), Helsinki, Finland

8 July
- COMPSAC (IEEE Annual Computers, Software, and Applications Conf.), Toronto, Canada

15 July
- ICALT (IEEE Int'l Conf. on Advanced Learning Technologies), Changhua, Taiwan

21 July
- ICCP (IEEE Int'l Conf. on Computational Photography), Toronto, Canada
- ICDCS (IEEE Int'l Conf. on Distributed Computing Systems), Glasgow, United Kingdom

IEEE COMPUTER SOCIETY CAREER CENTER

Career Accelerating Opportunities

Explore new options—upload your resume today

careers.computer.org

Changes in the marketplace shift demands for vital skills and talent. The **IEEE Computer Society Career Center** is a valuable resource tool to keep job seekers up to date on the dynamic career opportunities offered by employers.

Take advantage of these special resources for job seekers:

- JOB ALERTS
- TEMPLATES
- WEBINARS
- CAREER ADVICE
- RESUMES VIEWED BY TOP EMPLOYERS

No matter what your career level, the IEEE Computer Society Career Center keeps you connected to workplace trends and exciting career prospects.

IEEE COMPUTER SOCIETY

IEEE