Volume 4, Issue 2

Methods

Mouser's technology & solutions journal

Understanding A.I.

In this issue

Methods

Mouser's technology & solutions journal

Contributing Authors

Stephan Evanczuk
M. Tim Jones
Sally Eaves
Michelle **Nedashkovskaya**
Jim Romeo
Kyle Dent

Technical Contributors

Paul Golata
Joseph Downing
Christina Unarut

Design & Production

Robert Harper

Special Thanks

Kevin Hess
Sr. VP, Marketing

Russell Rasor
VP, Supplier Marketing

Jack Johnston, Director
Marketing Communication

Raymond Yin, Director
Technical Content

Creating Programs That Learn

By Stephan Evanczuk for Mouser Electronics

Artificial intelligence lies at the heart of dramatic advances in automotive, healthcare, industrial systems, and an expanding number of application areas. As interest continues to rise, the nature of AI has elicited some confusion and even fear about the growing role of AI in everyday life. The type of AI that enables an increasing number of smart products builds on straightforward but nontrivial engineering methods to deliver capabilities far removed from the civilization-ending AI of science fiction.

Definitions of AI range from its most advanced—and still conceptual—form, where machines are human-like in behavior, to a more familiar form where machines are trained to perform specific tasks. In its most advanced form, true artificial intelligences would operate without the explicit direction and control of humans to arrive independently at some conclusion or take some action just as a human might. At the more familiar engineering-oriented end of the AI spectrum, machine-learning (ML) methods typically provide the computational foundation for current AI applications. These methods generate responses to input data with impressive speed and accuracy without using code explicitly written to provide those responses. While software developers write code to process data in conventional systems, ML developers use data to teach ML algorithms such as artificial neural network models to generate desired responses to data.

How is a basic neural network model built?

Among the most familiar types of machine learning, neural network models pass data from their input layer through hidden layers to an output layer (**Figure 1**). As described, the hidden layers are trained to perform a series of transformations that extract the features needed to distinguish between different classes of input data. These transformations culminate in values loaded into the output layer, where each output unit provides a value representing the probability that the input data belongs in a particular class. With this approach, developers can classify data such as images or sensor measurements using an appropriate neural network architecture.

Neural network architectures take many forms, ranging from the simple type of feedforward neural network shown in **Figure 1** to deep neural networks (DNNs) built with several hidden layers and individual layers containing hundreds of thousands of neurons. Nevertheless, different architectures typically build on an artificial neuron unit with multiple inputs and a single output (**Figure 2**).

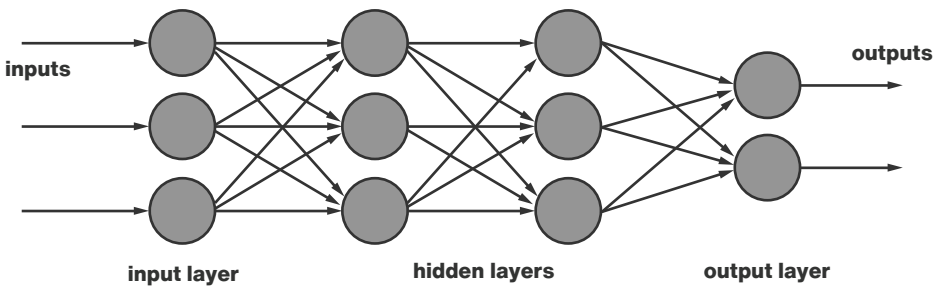


Figure 1: *Neural networks comprise layers of artificial neurons trained to distinguish between different input data classes. (Source: adapted from Wikipedia)*


AI

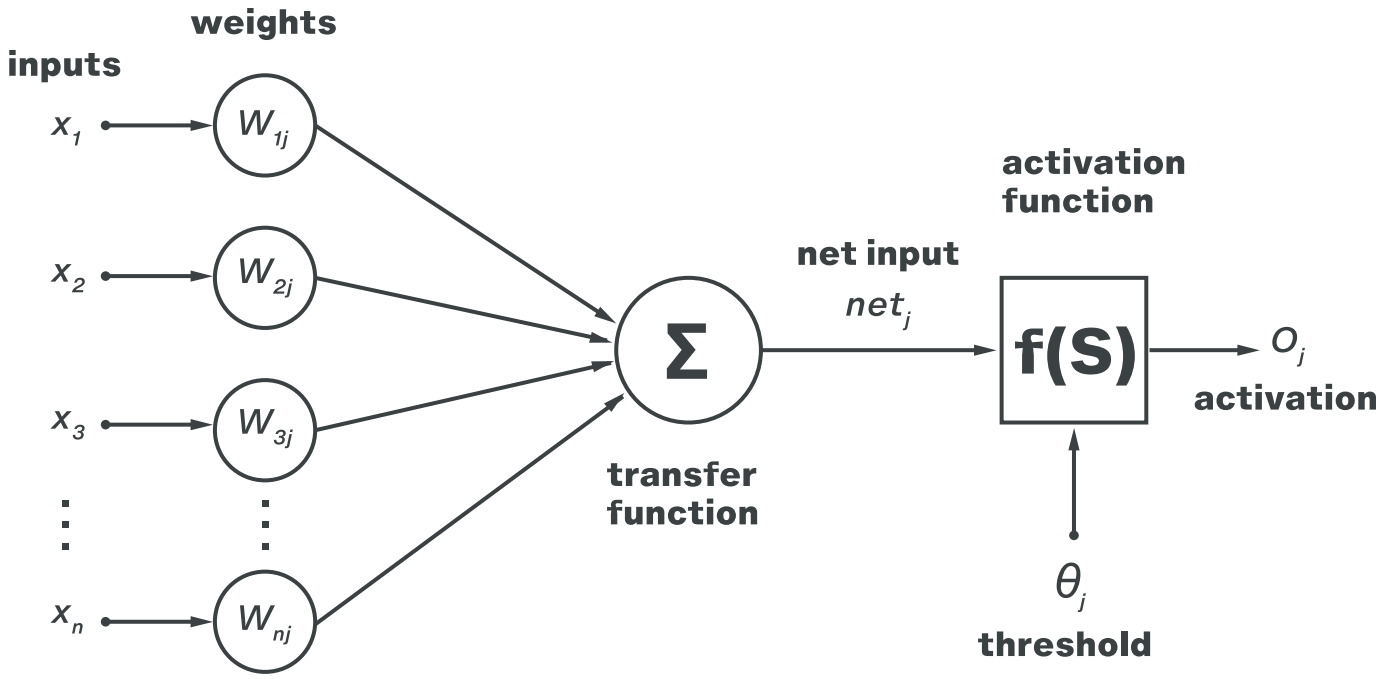


Figure 2: *An artificial neuron produces an output based on an activation function that operates on the sum of the neuron's weighted inputs. (Source: Wikipedia)*

In a feedforward neural network, a particular neuron n_{ij} in hidden layer j sums its i inputs, x_i, adjusted by an input-specific weight w_{ij}, and adds a layer-specific bias factor b_j (not shown in the figure) as follows:

$$S_j = \Sigma w_{ij} x_i + b_j$$

Finally, the summed value S_j is converted to a single value output by an activation function. Depending on requirements, these functions can take many forms, such as a simple step function, arc tangent, or non-linear mapping such as a rectified linear unit (ReLU), which outputs 0 for $S_j <= 0$ or S_j for $S_j > 0$.

Although they are all designed to extract the distinguishing features of data, different architectures might use significantly different transformations. For example, convolutional neural networks (CNNs) used in image-recognition applications use kernel convolutions. In this, functions, called kernels, perform convolutions on the input image to transform it into feature maps. Subsequent layers perform more convolutions or other functions, further extracting and transforming features until the CNN model generates a similar classification probability output as in simpler neural networks.

However, for developers, the underlying math for popular neural network architectures is largely transparent because of the availability of ML development tools (discussed elsewhere in this issue). Using those tools, developers can fairly easily implement a neural network model and begin training it using a set of data called the training set. This training data set includes a representative set of data observations and the correct classification for each observation—and represents one of the more challenging aspects of neural network model development.

How is a neural network model trained and deployed?

In the past, developers creating training sets had little option but to work through the many thousands of observations required in a typical set, manually labeling each observation with its correct name. For example, to create a training set for a road sign recognition application, they need to view images of road signs and label each image with the correct sign name. Public domain sets of prelabeled data let many machine-learning researchers avoid this task and focus on algorithm development. For production ML applications, however, the labeling task can present a significant challenge. Advanced ML developers often use pre-trained models in a process called transfer learning to help ease this problem.

Although emerging tools and services help facilitate data preparation, the training set characteristics nevertheless play a critical role in the effectiveness of the neural network model and overall application. The decisions made in choosing which observations to include and exclude have fundamental implications, including flexibility, specificity, and fairness that require careful consideration. As a result, the level of effort required to create an optimal training set can rival the effort required to implement the machine-learning program itself. After the training set is created and the neural network model is implemented, the model's training process iteratively runs the training data. At each iteration, the training process calculates a loss function that measures the difference between the desired result provided by the data labels and the calculated result generated by the model. Using a method called back propagation, that error information is used by the training process to adjust the weights and other model parameters for the next iteration. This process continues until the loss function falls within some threshold or fails to improve after some specified number of iterations.

When training completes, the model is converted to an inference model by performing several optimizations, including removing unneeded structures such as the back propagation mechanism, eliminating neurons that contribute little to the classification process, and even merging layers. Programs implement the inference model by loading a compact representation saved in various standard formats by ML tools and frameworks.

Neural networks not always the best solution

Although neural networks might be the more recognizable type of machine learning, they are by no means the only or even best choice for some applications. Neural networks fall into ML called supervised learning because they rely on labeled data sets to train the algorithm. In sensor-based applications such as the Internet of Things (IoT) or industrial systems, other supervised learning algorithms like support vector machines (SVMs) or decision trees provide an alternative that is simpler, more compact, and equally effective.

SVM methods classify data by finding where input data points lie within an n-dimensional space defined by the training data. Decision-tree methods use training data to construct a model that efficiently decomposes input data into a series of optimized decisions. As with the output layer of a neural network, the decision tree's final leaf nodes provide the probability that the data falls into a particular class. This approach is particularly efficient for classifying sensor data such as simultaneous accelerometer and gyroscope measurements to detect a complex movement. Support for decision trees is integrated into some inertial measurement units from STMicroelectronics.

These supervised learning methods are perhaps the most recognizable ML form. Still, other types of ML, including unsupervised learning, reinforcement learning, and many others, are already being applied to practical engineering problems. As the name suggests, unsupervised learning finds relationships within unlabeled data, using clustering techniques to identify similar characteristics. These techniques are particularly useful during training set development and feature engineering, where developers optimize the selection of data characteristics or features to be used for training and inference. Reinforcement learning finds applications in robotics systems programming, using utility measures to optimize training, not unlike loss functions in neural network training.

Conclusion

Machine-learning methods such as neural networks form the foundation of a growing array of smart products able to recognize and classify specific images or sensor measurements based on sophisticated mathematical concepts. For developers, implementing neural network models and other ML-based solutions in their applications follows a well-supported development process that is straightforward but by no means simple. M

Kria™ KV260 Vision AI Starter Kit

mouser.com/xilinx-kria-kv260-kit

XILINX®

Machine Learning Requires Multiple Steps

By M. Tim Jones for Mouser Electronics

Introduction

Deploying machine learning (ML) is a multi-step process. It involves selecting a model, training it for a specific task, validating it with test data, and then deploying and monitoring the model in production. Here, we'll discuss these steps and break them down to introduce you to ML.

ML refers to systems that, without explicit instruction, are capable of learning and improving. These systems learn from data to perform a particular task or function. In some cases, learning, or more specific training, occurs in a supervised manner where incorrect outputs result in adjusting the model to nudge it toward the correct output. In other cases, unsupervised learning occurs where the system organizes the data to reveal previously unknown patterns. Most ML models follow these two paradigms (supervised vs. unsupervised learning).

Let's now dig into what is meant by a model and then explore how data becomes the fuel for machine learning.

Machine-Learning Model

A model is an abstraction of a solution for machine learning. The model defines the architecture which, once trained, becomes an implementation. Therefore, we don't deploy models. We deploy implementations of models trained from data (more on this in the next section). So models plus data plus training equal instances of ML solutions (**Figure 1**).

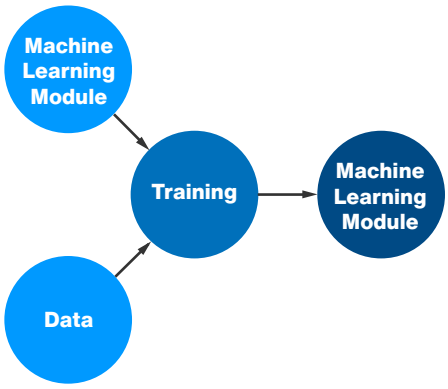


Figure 1: *From Machine Learning Model to Solution. (Source: Author)*

ML solutions represent a system. They accept inputs, perform the computation of different types within the network and then provide an output. The input and output represent numerical data which means that, in some cases, translation is required. For example, feeding text data into a deep-learning network requires encoding words into a numerical form that is commonly a high-dimensional vector given various words that could be used. Similarly, outputs might require translation from a numerical form back into a textual form.

ML models come in many types, including neural network models, Bayesian models, regression models, clustering models, and more. The model that you choose is based upon the problem at hand.

In the context of neural networks, models range from shallow multi-layer networks to deep neural networks that include many layers of specialized neurons (processing units). Deep neural networks also have a range of models available based upon your target application. For example:

- If your application is focused on identifying objects within images, then the Convolutional Neural Network (CNN) is an ideal model. CNNs have been applied to skin-cancer detection and outperform the average dermatologist.

ADuCM4050 Ultra Low Power Microcontroller

mouser.com/adi-aducm4050-ulp-mcu

ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

- If your application involves predicting or generating complex sequences (such as human language sentences), then Recurrent Neural Networks (RNN) or Long-Short-Term-Memory networks (LSTM) are ideal models. LSTMs have also been applied to machine translation of human languages.

- If your application involves describing the contents of an image in human language, then a combination of a CNN and an LSTM can be used (where the image is fed into the CNN and the output of the CNN represents the input to the LSTM, which emits the word sequences).

- If your application involves generating realistic images (such as landscapes or faces), then a Generative Adversarial Network (GAN) represents the current state-of-the-art model.

These models represent some of the more popular deep neural network architectures in use today. Deep neural networks are popular because they can accept unstructured data such as images, video, or audio information. The layers within the network construct a hierarchy of features that allow them to classify very complex information. Deep neural networks have demonstrated state-of-the-art performance over a wide number of problem domains. But like other ML models, their accuracy is dependent upon data. Let's explore this aspect next.

Data and training

Data is the fuel that drives machine learning, not just in operation but also constructing an ML solution through model training. In the context of training data for deep neural networks, it's important to explore the necessary data in the context of quantity and quality.

Deep neural networks require large amounts of data for training. One rule of thumb for image-based classification is 1,000 images per class. But the answer is dependent upon the complexity of the model and tolerance for error. Some examples from production ML solutions yield a spectrum of dataset sizes. A facial detection and recognition system required 450,000 images, and a question-and-answer chatbot was trained with 200,000 questions paired with 2 million answers. Smaller datasets can also suffice based upon the problem being solved. A sentiment analysis solution that determines the polarity of opinion from written text required only tens of thousands of samples.

Data quality is just as important as the quantity. Given the large datasets required for training, even small amounts of erroneous training data can lead to a poor solution. Depending upon the type of data necessary, your data might go through a cleansing process. This ensures that the dataset is consistent, lacks duplicate data, is accurate, and complete (lacks invalid or incomplete data). Tools exist to support this process. Validating data for bias is also important to ensure that data does not lead to a biased ML solution.

ML training operates on numerical data, so a pre-processing step can be required depending upon your solution. For example, if your data is human language, it must first be translated into a numerical form to process. Images can be pre-processed for consistency. For example, images fed into a deep neural network would be resized and smoothed to remove noise (among other operations).

One of the biggest problems in ML is acquiring a dataset to train your ML solution. This could be the largest endeavor depending upon your problem because it might not exist and require a separate effort to capture.

Finally, the dataset should be segmented between training data and test data. The training portion is used to train the model, and once trained, the test data is used to validate the accuracy of the solution (**Figure 2**).

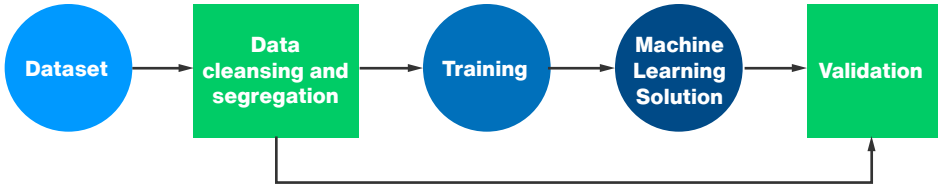


Figure 2: *Dataset Splitting for Training and Validation. (Source: Author)*

Tools exist to accomplish this process, and most frameworks include split functions to segregate training and test data. Let's now explore some of the frameworks that simplify the construction of machine-learning solutions.

Framework

It's no longer necessary to build your machine-learning model from the ground up. Instead, you can rely on a framework that includes these models and other tools to prepare data and validate your solution. This same framework also provides the environment through which you'll deploy your solution for production. Choosing a framework is typically done based upon familiarity, but if you're starting out, you can choose one that fits your application and the model you intend to use.

TensorFlow is the best of the deep-learning frameworks. It supports all popular models (CNN, RNN, LSTM, etc.) and allows you to develop in Python or C++. You can deploy TensorFlow solutions on high-end servers down to mobile devices. If you're just starting, TensorFlow is the place to start, if nothing else than for its tutorials and breadth of documentation.

CAFFE started as an academic project, but after being released into open source, has grown into a popular deep-learning framework. CAFFE is written in C++ but also supports Python for model development. Like TensorFlow, it supports a wide range of deep-learning models.

Facebook began work on a derivative of CAFFE called Caffe2 that included new models. But rather than bifurcate the CAFFE project, it was instead merged into another framework called PyTorch. PyTorch is based upon the wealth of information available, including hands-on tutorials to build different types of solutions.

The R language and environment is a popular tool for ML and data science. It's interactive, which allows you to prototype and build a solution incrementally while seeing the results in stages. Along with Keras (an open-source neural network library), you can build CNNs and RNNs with minimal development.

Model auditing

Once your model is trained and meeting your accuracy requirement, you deploy it in production. But once there, you'll need to audit your solution to ensure it meets your requirements. This is particularly important based upon the decisions made by your model and how they can impact people.

Some machine-learning models are transparent and can be understood (such as decision trees). But other models such as deep neural networks are considered black-box, and decisions are made by millions of calculations that the model itself cannot explain. Therefore, while periodic auditing was once acceptable, continuous auditing quickly became the norm in these black-box situations because mistakes are inevitable. Once a mistake is discovered, this information can be used as data to tweak the model.

The other consideration is the lifetime of the solution. Models decay and input data can evolve, resulting in changes in the model's performance. Therefore, accepting that a solution will be brittle over time, ML solutions must change along with the world around them

Conclusion

To deploy a machine-learning solution, we start with a problem and then consider possible models that solve it. Acquiring data is next, and once properly cleansed and segmented, the model can be trained and validated using an ML framework. Not all frameworks are the same and based upon your model and experience, one of many can be selected and applied. This framework is then used to deploy the ML solution, and with proper auditing, the solution operates in the real world with live data. M

Agency, Autonomy, and Protection in Artificial Intelligence

By Sally Eaves for Mouser Electronics

Artificial intelligence (AI) is a key component advancing intelligent smart environments, spanning home, work, health, education, supply chain, factory, city, and society. Although fairness and bias are currently human-designed aspects of AI, other aspects such as agency and autonomy present the potential for a duality between greater goods and harmful acts. For those who design, develop, and implement AI systems, challenges include reducing the risk of harm, balancing human control and machine autonomy, and solving problems through governance, standardization, and innovation. Here, we'll explore concepts of agency and autonomy, discuss the ever-present role of bias, and describe challenges in protecting humans from potential harm.

Agency

Definitions of agency vary significantly, but most focus on making something happen, including making something happen for something or someone else. A related concept is whether the agent can choose what to do, instead of being told what to do. Consider a decision that appears irrational to others. Out of all the different options available, the perceived irrational decision is made for reasons known only to the person making it. What characterizes the human agent is intelligence and our ability to select from a range of options.

The ability to choose what to do invites questions about enabling AI systems to make such decisions for themselves. What happens if AI makes a seemingly irrational decision to us as humans? Is it a malfunction, or is it reasoning that is not accessible to us? Drawing on science fiction, perhaps the classic case is HAL9000, the sentient computer in 2001: A Space Odyssey. So the question becomes: What autonomy should AI-empowered agents/devices possess?

Autonomy

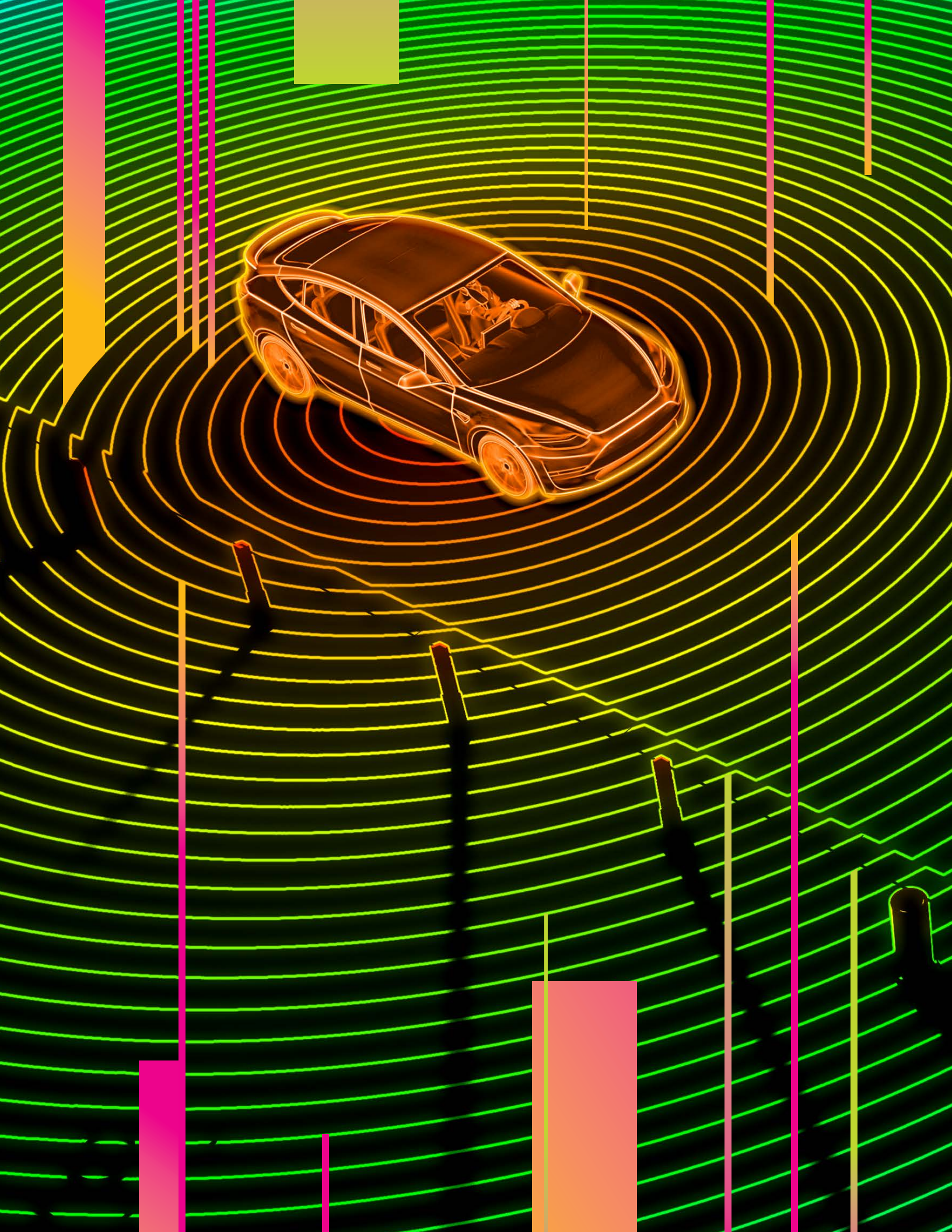
Autonomy invokes independence in decision-making and goal-setting, and it implies self-sustainment, with autonomy motivating agency. The nature of the independence differs from automated technologies: Automated technology is self-acting, as opposed to autonomous or self-regulating technology that can adapt to situations.

However, automation and independence are not black-and-white distinctions. They're a matter of degrees. Take, for example, the concept of AI-, Internet of Things-, and sensor-enabled driverless cars. Today, reaching the much-hyped fully autonomous vehicle is still some way off. Although we have seen innovation toward this vision and acceleration of other benefits—such as making human driving smarter and safer—significant moments have caused pause for reflection. One such moment was a March 2019 pedestrian fatality involving a self-driving car.

Such consequences raise the question: How much autonomy can be achieved, or is indeed desirable? The answer depends on degrees of autonomy and supervision:

Degrees of Autonomy

The U.S. National Highway Traffic Safety Administration (NHTSA) has defined six levels of autonomy in driverless cars, ranging from Level 0 to Level 5:



Xeon® Scalable Processors (3rd Gen)

mouser.com/intel-xeon-scalable-processors-3rd-gen

intel®

- **Level 0:** Humans do all the driving.
- **Level 1:** Some assistance is provided to the person driving, in steering, accelerating, or braking.
- **Level 2**: More advanced assistance that controls steering, accelerating, and braking but requires constant human monitoring.
- **Level 3**: In specific circumstances, the vehicle can carry out all parts of the driving but with the person ready to intervene when required by the system.
- **Level 4:** In specified circumstances, the vehicle can do all required to drive without a person paying attention.
- **Level 5:** Fully autonomous self-driving vehicle with the human as a passenger.

Despite outlining degrees of autonomy, ethics and legal questions remain. For example, how should a car decide between crashing and killing five school children at a bus stop or killing its two elderly passengers? Who is ultimately responsible when autonomous systems fail? Can and should self-learning robots be held responsible for their actions and be held liable if people are hurt or property damaged?

Degrees of Supervision

Part of the answer might lie in further defining degrees of autonomy according to degrees of supervision. Like autonomy, supervision can also be described on a spectrum, such as definitions identified by a company called re2 robotics:

- **Tele-Operation**: Intuitive human control of a robot.
- **Supervised Autonomy:** Autonomous operation under human supervision.
- **Fully Autonomous:** Complete robotic autonomy with no supervision required.

Putting these degrees into context, supervised autonomy would be allowing robots to perform duties that would be hazardous to humans but still giving robot operators full control over specific tasks.

Bias and its friends

Agency and autonomy matter, especially with artificial intelligence embedded across all aspects of life, increasingly involved in highly impactful decisions and accelerating with the wider rollout of 5G. One of the challenges within this smart connected and hybrid domain is the bias, conscious or otherwise, that can affect AI design, development, and application.

Agency and autonomy can have highly sensitive and serious long-term impacts in terms of unfair or discriminatory outcomes. One example is an algorithm to determine the likelihood of a criminal re-offending that has revealed a pattern of bias in its assessment. This is more than the undermining of trust; it is exposing people to discriminatory and undesirable consequences. Protection is vital.

Protections and their challenges

The very possibility of harm from artificial intelligence raises the question of how people can be protected. It provokes questions about who or what controls our individual and collective lives as a society, how much freedom and privacy we have, and what checks, balances, and accountability are in place to stop any of us from being harmed. Significant, complex, and interlinked questions suggest some level of protection is required.

One of the earliest attempts to propose rules for AI systems is Asimov's Law of Robot's published in the short science story "Runaround" in 1942. This introduces three laws:

1. Prevent harm to humans.

2. Require robots to be obedient to humans unless doing so clashes with the first law.

3. Require robots to protect themselves unless doing so clashes with the other two laws.

Are these adequate? Beyond any debate on the adequacy or necessity of these laws, this again draws attention to the distinction between automated and autonomous technologies.

Constraining autonomy

AI-enhanced robots with any degree of autonomy are part of a more general issue relating to how much autonomy should be given to AI and whether constraints should be imposed. For example,

is it desirable to have a weapon system that will act upon goals without human intervention but open to human intervention if necessary? Can we trust this? History is full of examples where technology has had negative consequences, intended or otherwise. Social media is, in principle, a good idea because it allows people to connect and share. However, its dark side has exposed many vulnerabilities, including cyberbullying, misinformation, and identity theft, among others.

As AI becomes more deeply embedded across our increasingly smart, intelligent, and connected world, the need for constraint becomes more complicated. Take, for example, facial recognition technologies that cross city centers and enter into residential areas. Although these might help make our streets safer, they also carry the potential to monitor people's movements and activities. Transparency concerning data usage is key to trust and acceptance, alongside protection around privacy and security, especially given the increased surface area for cybersecurity threats. Even smart toys can be hacked, so, really, how private can we be?

Governing AI

Although most people recognize that AI can bring transformative benefits to business and society, the potential dark side must be recognized, too, because that's how we can best mitigate and address it. In a recent article, Sundar Pichai, CEO of Alphabet and Google, has called for regulating AI and acknowledges challenges in regulating it. Regulation at a national level could lead to some nations having strong, enforceable governance mechanisms but leave others with weak regulations that attract malicious activity.

Further, legislation tends to lag behind technology developments, especially ones as rapidly developing as AI. This suggests the need for underlying principles and values to guide what is accepted or permitted. So while Google has its principles for the "ethical development and use of AI in our research and products," for example, reasons are few for others to adopt them.

To counter this, Pichai argues that the time is now for international alignment. This calls for a global effort, as exemplified in May 2019, when 42 countries signed up to adopt OECD Principles on Artificial Intelligence. These comprise "five values-based principles for the responsible deployment of trustworthy AI and five recommendations for public policy and international co-operation."

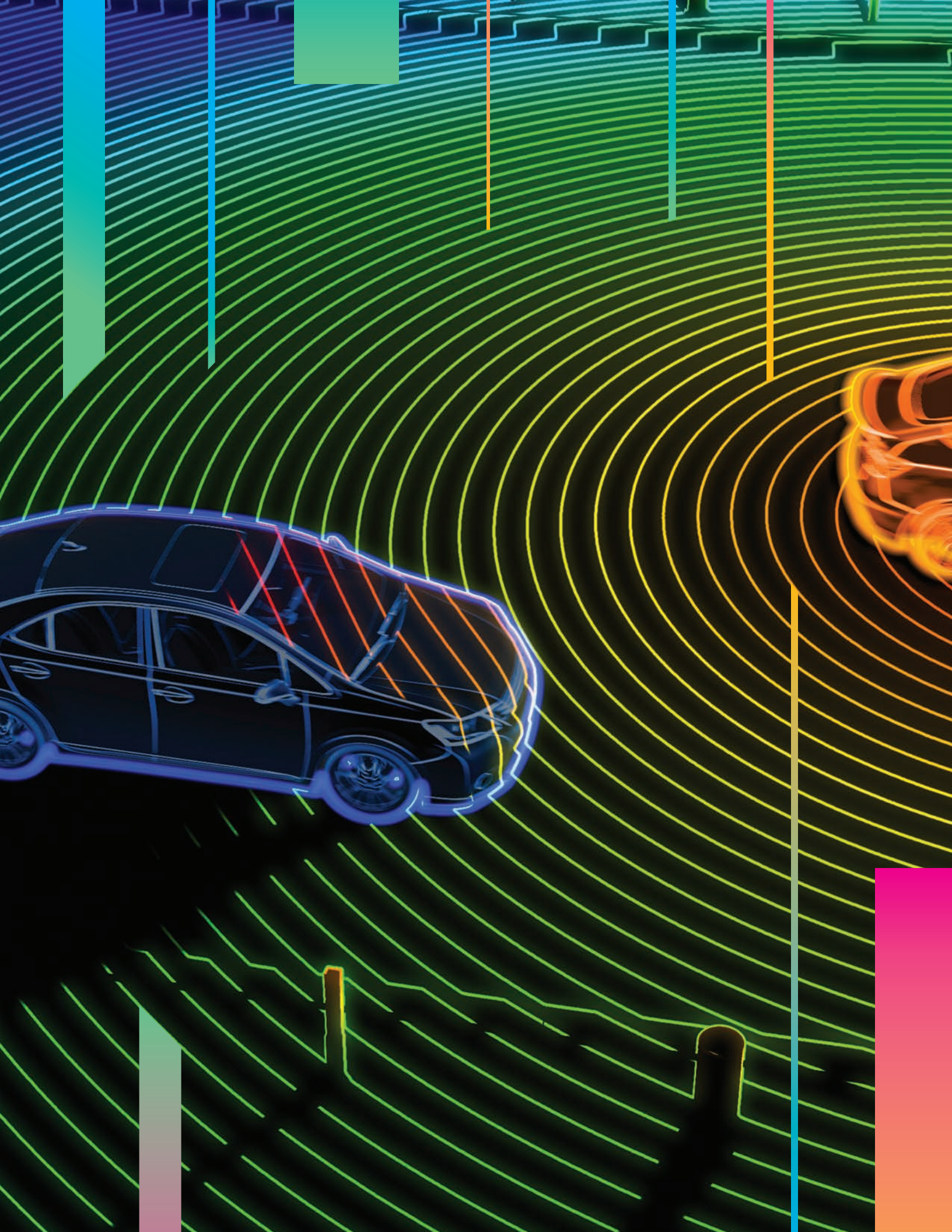
However, some argue that it will be near impossible to regulate AI. What if someone or some (political) organization decides to ignore these principles? For example, political agendas can argue for AI developments on the grounds of national security, particularly in the military sector. The malicious spread of misinformation and the distortion of reality with deep fakes are also concerns.

Assigning rights and responsibilities

So finally, protection must also be viewed from the perspective of the technology itself. First is the issue of identity. It might not be human, but a robot can have a unique presence, unlike inanimate objects. Taking this to a higher level recognizes that a robot can gain citizenship, as illustrated when Sophia was awarded full citizenship of Saudi Arabia back in 2017. During the same year, the European Union (EU) Parliament's Legal Affairs Committee published a controversial paper that suggested creating specific legal rights and responsibilities, rather than human rights, and encapsulating them under "electronic personalities."

Conclusion

Artificial intelligence is with us and here to stay, and like many aspects of life, it presents as a duality–with both the potential for helping the greater good and reaching new depths of nefarious uses. The challenge for those who design, develop, and implement AI systems is to reduce the risk of harm, achieve the delicate balance between human control and autonomy, and find an effective balance between governance structures, standardization, and innovation. Perhaps the onus actually lies best with us all to help steer the trajectory of AI toward the economic and ethical values we believe. In this way, we can move beyond a human versus technology narrative toward one of human-technology partnership that builds on our collective complementary strengths. M



Living With an Imperfectly Ethical AI

By Michelle Nedashkovskaya for Mouser Electronics

Balancing AI's potential for good against its ethical pitfalls

Ongoing developments in artificial intelligence (AI) hold immense promise for achieving social good. AI's potential to help us overcome some of the world's greatest challenges is already being explored across various sectors. From agriculture to astronomy, the breadth of AI applications seems limited only by our imagination.

Like any tool, however, AI could end up creating or perpetuating some problems while solving others, even when designed with the best intentions. The ethical risks associated with AI will broaden alongside the contexts in which it is applied. A perfect ethical framework for navigating this new landscape could prove elusive, but its importance matches the enormity of the challenge. New technological contexts will dictate the need for new norms.

To address these risks, technologists must:

1. Understand that bias exists in data and tools
2. Increase awareness of this bias and improve efforts to mitigate it
3. Institutionalize ethical thinking in engineering processes

Understanding bias in data & tools

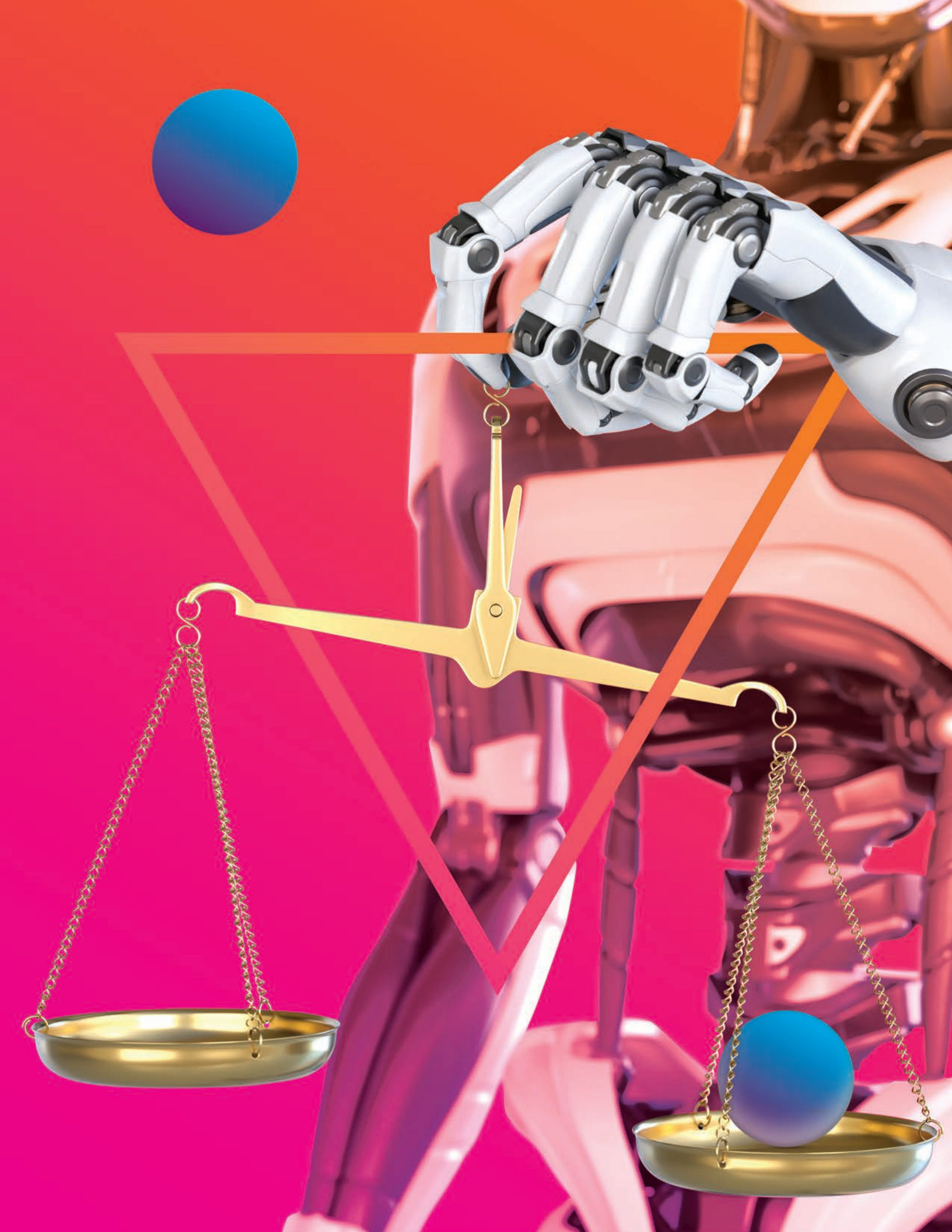
AI ethics have become a hot topic recently, in part because algorithmic decision-making and decision support systems are being integrated into public administration domains, including public health, law enforcement, and criminal justice. Such applications raise the ethical stakes of employing AI technology, amplifying the potential real-world ramifications of AI tools that might yield biased or unfair results.

Biased AI tools could have profound and long-lasting impacts on individuals' lives, affecting their criminal records, creditworthiness, and employment prospects. Bias can creep into an algorithm in many ways. Data bias can arise from flawed data collection or reflect a broader systematic bias at play. For example, if individuals from minority populations are arrested at higher rates than their white counterparts for the same crime, algorithms trained on the resulting data can perpetuate those inequities. Such risks related to prejudicial categorization have already played out in the real world. Pretrial risk assessment algorithms used in criminal justice proceedings, for example, have repeatedly been found to discriminate against racial minorities.

Bias can also arise in the way computer scientists frame problems and select the attributes an algorithm considers. For example, algorithms for job recruitment efforts rely on many assumptions: Which attributes should be associated with a worthwhile candidate? Could those attributes carry gendered or racial connotations? This sort of bias led Amazon to scrap an AI recruitment tool in 2018 after realizing that the model discriminated heavily against women: The model learned to associate strong candidates with maleness because the company employed more men than women.

Increasing awareness of bias and improving solutions

The implications of such cases of bias have prompted the development of a robust literature on AI's ethical considerations. Many philosophers, including



well-known experts, such as Oxford University philosophers Nick Bostrom and Luciano Floridi, have begun to devote themselves to developing frameworks around these issues. Such academic efforts have focused on "fairness, accountability, and transparency" in machine learning. Awareness of and engagement with this vast and growing literature is key to forging appropriate risk mitigation strategies.

Understanding and building awareness about AI ethics issues requires interdisciplinary cooperation. Several initiatives have already been developed with this goal in mind, from think tank programming like that of the Future of Life Institute to industry efforts such as Google's publication of Responsible AI Practices. The third annual Association for Computing Machinery Conference on Fairness, Accountability, and Transparency (ACM FAccT, formerly ACM FAT*) took place in January 2020 and brought together stakeholders from numerous fields to explore the ethics of computing systems. Such efforts are signs of progress, but it is incumbent on each individual to contribute to this interdisciplinary dialogue and learn from the work.

Equally important to initiatives dedicated to studying these issues is the widespread adoption of new ethical frameworks throughout the technology industry. Like the practice of building human-centered AI, we should consider developing new means of institutionalizing ethical and socially responsible thinking into the engineering process' every step. Ethics need not be perceived as an esoteric science reserved for ordained philosophers but rather a practice or skill that anyone can exercise and sharpen in every step of their work.

Institutionalizing ethical thinking in engineering processes

New modes of thinking about institutionalizing responsible AI can borrow heavily from disciplines outside engineering and computer science. For example, much of the thinking in international security focuses on worst-case scenarios. Just as military forces create contingency plans to prepare for unforeseen challenges, technologists can employ worst-case scenario thinking regarding AI ethics: What are all the ways a given product could go awry? How can we mitigate those risks? One thought experiment could mirror deliberations about weapon design and deployment: What could go wrong if bad actors acquire this new capability?

Other ways to institutionalize ethical thinking within technological development can hit closer to home. For example, the concept of iterative progress is perhaps best understood in the technology industry. Agile software development entails concurrent development and testing. It emphasizes incremental delivery, continual team collaboration, and learning. Innovations can be implemented iteratively to resolve technical kinks. Agile deployment and institutionalized feedback can help keep ethical considerations in perspective.

Conclusion

We must all continue to ask big-picture questions about technology limitations. For every product or approach that involves AI—especially in a public sector context—one might ask whether AI is best suited to perform that function. Human judgment is demonstrably imperfect and often falls victim to a confluence of cognitive biases, but we must remain vigilant to not replace one set of flaws with another. M

"AI could end up creating or perpetuating some problems while solving others, even when designed with the best intentions."

12+Gbps 0.8mm High-Speed Edge Card Connectors

mouser.com/amphenol-intercon-edge-card

Amphenol®

Machine-Learning Software Simplifies Development

By Stephan Evanczuk for Mouser Electronics

Machine learning (ML) has gained the most recognition among the different artificial intelligence (AI) types, thanks to a growing list of successful applications. As noted elsewhere in this issue, ML development flips the conventional model of software development. Rather than explicitly writing algorithms to process data, ML developers use data to train algorithms on how to process data. For production ML applications, developers might spend little time on the algorithms themselves and focus more on data engineering and writing code with proven algorithms. In contrast, ML researchers might spend most of their time writing code for new algorithms or optimizing existing ones, using standard data sets to compare improvements over earlier algorithms. In the following, key development resources needed to program ML applications are examined.

Both production and research efforts can take advantage of a wide array of development resources, ranging from low-level algebraic libraries used to implement new kinds of model algorithms to high-level automated ML environments that accept a set of data and return a trained model. In general, developers of production applications can complete their work with little need to involve themselves with low-level math libraries. Yet, when facing challenges such as developing ML models for Internet of Things (IoT) devices, production developers can still find themselves using some of the same sorts of tools and techniques employed by researchers.

Whether focused on research or production, ML projects require implementing an existing or novel ML algorithm using conventional coding methods. ML developers work with various conventional programming languages, including Python, C/C++, Java, Javascript, R, Go, and other more specialized languages. Among these, Python has emerged as the dominant language for ML development partly because developers can quickly become productive with this language; but largely because of the wide availability of add-on libraries or modules. Python provides different developers' methods to add external C/C++ functions or even create Python modules in C if no suitable module is available. In general, however, ML development with Python builds on a common set of tested, optimized modules.

Building on modules

Developers can quickly get started with ML development by importing a set of Python modules that provide fundamental capabilities required equally for developing production ML models or for creating new ML algorithms. Among these, some of the more commonly used modules include:

- NumPy, which provides array manipulation and algebraic functions commonly required in ML development;
- Scipy, which provides various scientific computing functions;
- Pandas, which supports high-level data structures and supports access to different file formats and databases;
- Matplotlib, which provides functions to visualize data and results.

In principle, a developer could use only these libraries, implementing an ML algorithm's underlying math operations using NumPy


mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
#selection at the end -add back the deselected mirror mo
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the ac

AURIX™ TC3xx Microcontrollers

mouser.com/infineon-aurix-tc3xx-mcus

Infineon

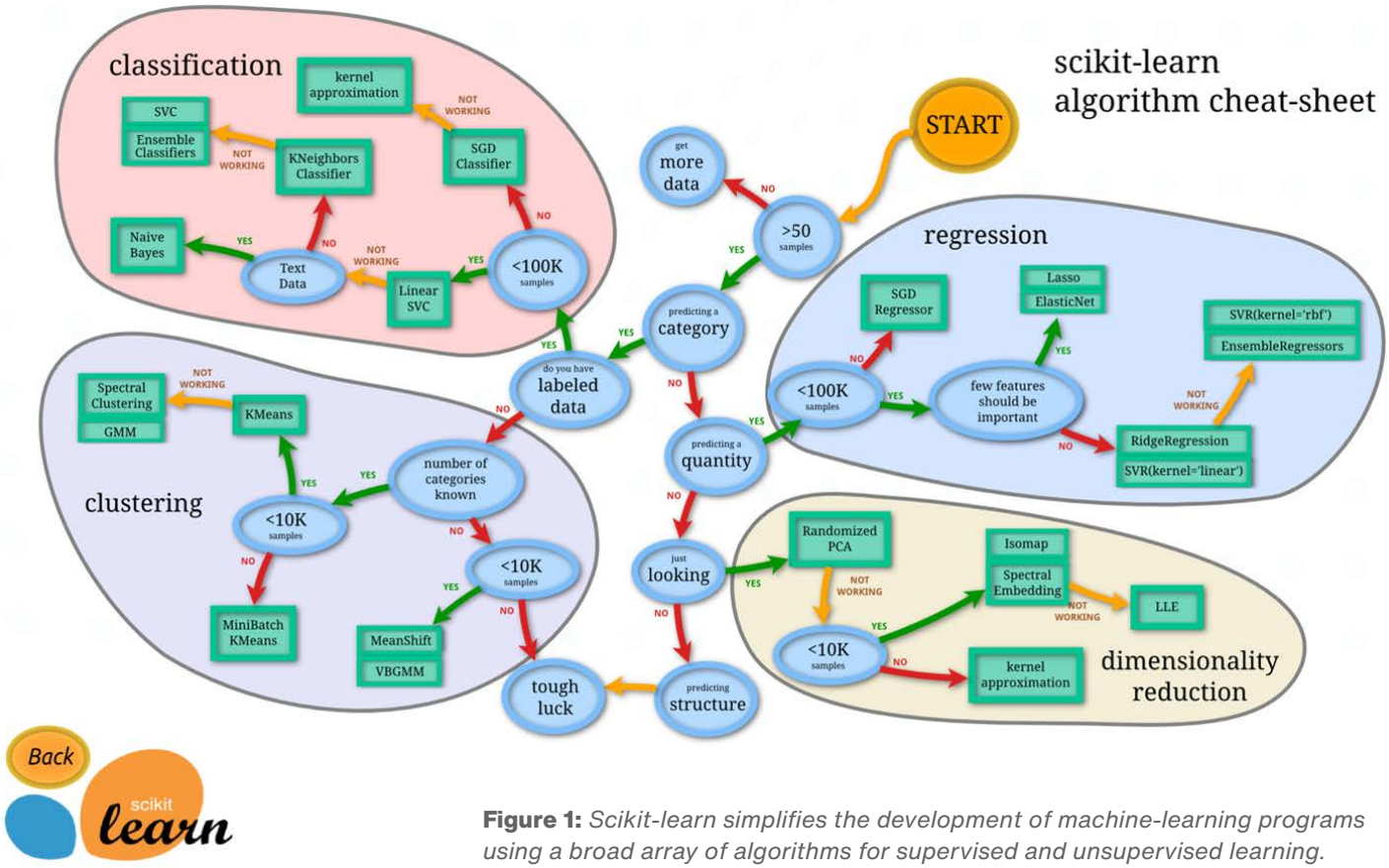


Figure 1: *Scikit-learn simplifies the development of machine-learning programs using a broad array of algorithms for supervised and unsupervised learning. (Source: scikit-learn.org)*

algebraic functions and visualizing results with Matplotlib. In practice, however, both researchers and production developers combine these libraries with several others. ML scientists exploring new algorithms might use the SymPy symbolic computing module to evaluate their equations or implement compute-intensive core functions in C using low-level routines from a basic linear algebra subprograms (BLAS) library such as OpenBLAS.

Production developers might find themselves turning to C/C++ libraries for performance reasons. However, in the early stages of development, they are more likely to use Python modules that support higher-level abstractions with intuitive functions specifically designed for implementing ML applications. Although this is perhaps the largest group of software resources for ML programming, some of the more commonly used ML packages include:

Scikit-learn, which natively supports perhaps the widest range of ML algorithms for supervised learning and unsupervised learning (**Figure 1**) with an accessible approach considered particularly effective for those new to ML development;

- Keras, which supports the efficient implementation of deep neural network (DNN) models, including convolutional neural networks (CNNs) through a comprehensive set of functions required to implement the various layers of a model;
- TensorFlow, which provides functions for model implementation as well as broader, end-to-end support for ML applications.
- PyTorch, which also provides both model implementation and end-to-end development capabilities.

Each of these libraries abstracts complex operations to a series of intuitive function calls. To build a DNN model, developers typically build up the model layer by layer using built-in functions that implement the layer's function. After the model is configured, other function calls invoke training with hyperparameters needed in the training process itself.

Some Python libraries, including TensorFlow and PyTorch, are supported by comprehensive ecosystems, so the core library is part of a more substantial framework for ML development. Although many such frameworks have emerged, TensorFlow and PyTorch have gained dominance among production developers and researchers. Researchers have generally preferred PyTorch because of its interactivity and flexibility. Industry developers have generally preferred TensorFlow for its performance efficiency. Still, each framework continues to evolve, addressing any shortcomings with capabilities that drive them closer to parity.

An even higher-level ML development resource class continues to emerge from commercial cloud-service providers such as Amazon Web Services (AWS), Google, IBM, and Microsoft and specialty cloud-platform providers. Intended to provide turnkey ML solutions, services such as AWS SageMaker, Google Fluid Annotation, IBM Cloud Annotations, and Microsoft Automated ML generate models from datasets for users with neither the time nor the expertise to create ML models on their own. Typically, users can pass the results to other tools in each provider's environment to create optimized inference models for deployment.

Optimization and deployment

Performance concerns are endemic to ML development projects. Although ML researchers continue to explore methods to speed lengthy training cycles, both researchers and production developers typically take advantage of the performance boost provided by graphics processing units (GPUs) and GPU-compatible libraries. For example, the GPU-enabled CuPy package can speed many core ML operations well over 100x than the compatible but non-GPU-enabled NumPy package.

For an overall gain in performance, developers can use the Numba compiler, which converts Python to machine code with optimizations, including GPU support. TensorFlow's XLA (Accelerated Linear Algebra) compiler can improve model speed and size with no source-code changes.

Alternatively, developers can use different versions of Python itself. Cython compiles Python-compatible Cython code, resulting in faster execution than possible with a standard Python's interpreter. The Intel® distribution for Python takes full advantage of performance enhancements available in the Intel® architectures.

MAX78000 Ultra-Low-Power Arm Cortex-M4 Processor

mouser.com/maxim-max78000-processor

maxim integrated™

For deployment on resource-constrained IoT devices, developers can take advantage of resource-optimized model architectures and processor-optimized libraries. For example, Google's MobileNet CNN architecture and its more recent EfficientNet CNN architecture achieve high accuracy with smaller, faster models. To speed execution of the model itself, developers can use libraries such as the Intel® oneAPI Deep Neural Network Library (oneDNN) or Arm® NN (neural network) Software Developer Kit (SDK) for Cortex®-A-based processors or its Cortex Microcontroller Software Interface Standard Neural Network (CMSIS-NN) library for Cortex®-M4-based processors.

Development environments

We've described only a bare-bones set of Python modules among the thousands in the Python Package Index repository just for ML. Of course, a typical development project will build on many module packages, each with their dependencies. Developers typically use virtual workspaces to isolate a project's development packages from different versions of common packages used in other projects or even their operating environment. The Anaconda platform provides an even simpler approach, combining package management with the simple deployment of virtual workspaces.

For both experienced ML developers and those just venturing into ML development, the combination of Anaconda and a popular AI development tool, JupyterLab, largely eliminates the setup and configuration tasks typically required to use any development environment. JupyterLab, such as its earlier version, Jupyter Notebook, lets users build notebooks that combine descriptive text with runnable code and results in a single package. Jupyter notebooks have emerged as a common medium of exchange of ideas, specific algorithms, and applications between developers, researchers, and even participants in ML competitions and courses on Kaggle and other sites.

Conclusion

Machine-learning development encompasses a wide set of activities focused on both preparing data and writing code to implement models with existing or new algorithms. To implement models, developers need only a few basic tools to get started, but generating optimized inference models might require them to reach deeper into the rich set of tools available for creating effective ML-based applications.

Open-Source Movement Affects AI Apps

By Jim Romeo for Mouser Electronics

Artificial intelligence (AI) is being used by many to accomplish great things beyond human intelligence. Open-source platforms, data, frameworks, and models are increasingly used in conjunction with AI development to improve and enhance AI projects.

The premise of open source is that everything is free and available to all. Source code, designs, and related intellectual property is shared and can be redistributed at large. It represents an open exchange where users participate and collaborate in a communal effort. Programmers use source code to program a software application. With source code, programmers and developers use it to perform and reach desired objectives. How does the open-source movement affect AI applications? Let's explore.

Interoperability and communal sharing for progress

In artificial intelligence applications development and the emergence of machine learning (ML), the open-source movement is more important than ever, Major software purveyors and software developers across various industries contribute their own source code and use others' codes.

Open-source code meets a common open-source standard. As an enterprise develops open-source code with open-source standards and without proprietary data formats, the resultant software is compatible with other software and applications. This enables interoperability.

A communal effort ensures interoperability. Interoperability is key because it means compatibility and an ability to integrate an application with other applications, allowing enterprise networks to grow and prosper from the software.

Open-source code and software are often loaded to a common platform, available to all, such as GitHub. Although it's a Microsoft site, GitHub is meant to be a working parking lot for open-source code and self-described as "a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers."

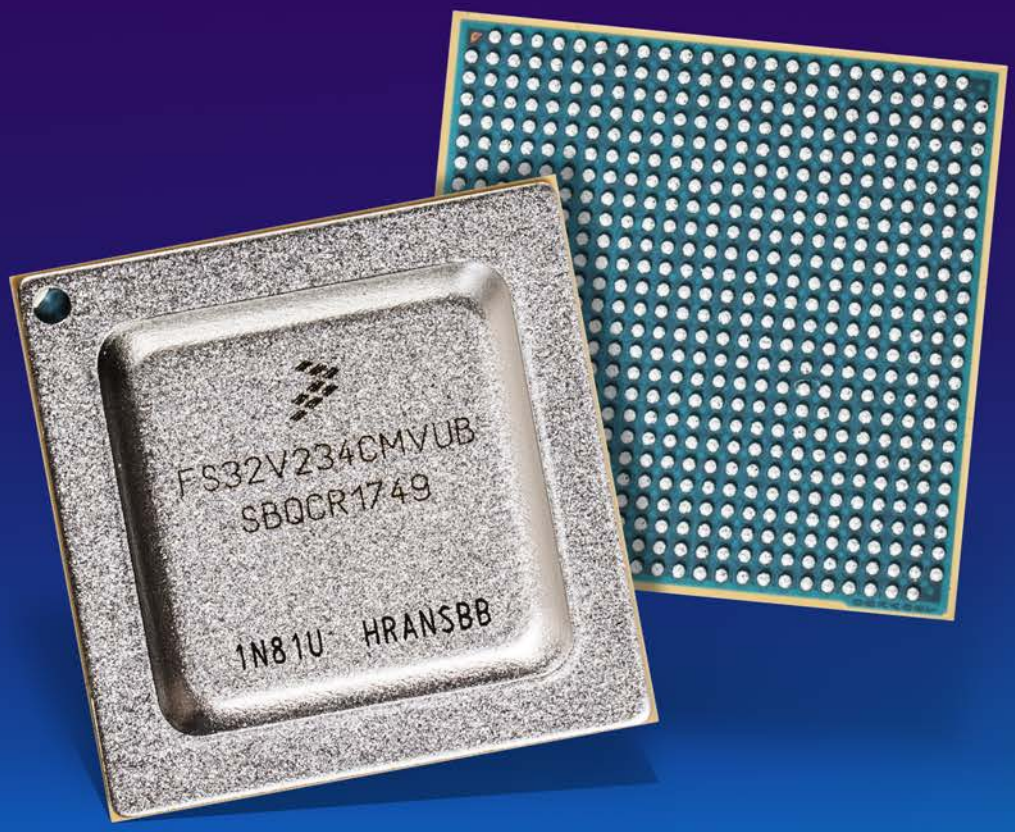
Another popular platform to share code is via the Apache Software Foundation (ASF). It provides a framework for source code where those committed to the open-source credo of sharing intellectual property can do so without worrying about infringement or liability.

The foundation touts "The Apache Way," as one in which "more than 730 individual Members and 7,000 Committers successfully collaborate to develop freely available enterprise-grade software, benefiting millions of users worldwide."

In AI, a data framework uses data to create predicted outputs. Such prediction is used to learn data and continuously train the model. This takes place via a feedback loop. Data is learned. Prediction is enabled. Data and predictions are continually fed back and improve the prediction's accuracy.

Ride-sharing company Lyft is an example of how open-source data tools are used in everyday applications. The company recently open-sourced a debugging tool for AI data it had been using for years, internal to its operations. The tool called Flyte

S32V234 Vision & Sensor Fusion Processor



mouser.com/nxp-s32v234-processor

NXP

has been used by Lyft in-house for the past three years and corrects and debugs data related to Lyft's pricing, locations, estimate time of arrivals, mapping, and self-driving developer teams. It is open source and can be downloaded and used by anyone.

Open-source frameworks for AI

Artificial intelligence has many open-source frameworks, available to all, used by many, resulting in creating applications that might be proprietary. But they are based on an open-source framework.

For example, Google's TensorFlow is a popular and useful open-source framework. This framework is a comprehensive ecosystem of tools and libraries that enables developers to apply ML to and serve their markets better. For example, Airbnb, eBay, Dropbox, and others use TensorFlow.

TensorFlow, unbeknownst to many, allows rental listings to follow an order suitable to a user as that user and their preferences are indicated per their profile and search. It might aid eBay to better present auction listings. Or it might establish a file hierarchy that follows a certain pattern for a Dropbox user.

Amazon SageMaker Neo is another open-source ML platform. Its project code helps AI developers build models that learn data and train an ML model via the cloud. It is compatible with sensors and other computing and connected devices used with Internet of Things (IoT) applications. It can be used by AI developers to make accurate predictions.

For example, SyntheticGestalt (an applied ML company) uses SageMaker Neo to train drug discovery models in the pharmaceutical and life-science industries. It processes and learns experiment data, evaluates it, and produces model results.

The Open Neural Network Exchange (ONNX) is an open-source AI ecosystem created specifically for AI to represent ML models. The system's operands are common and serve as basic building blocks for ML and deep-learning models. This aids AI developers in utilizing formats to allow models to work with different frameworks and tools for AI applications. Like other open-source platforms and formats, ONNX enables interoperability between other applications and solutions. By developing within a preferred framework, it curtails problems from incompatibility further downstream of the application.

Our open-source future

As artificial intelligence continues to gain traction to accomplish great things, the open-source mindset will continue to help accelerate it. It uses the power of a give-and-take community who use open-source data property, data, algorithms, platforms frameworks, and formats to perform amazing feats.

Open-source platforms will help technology use math, science, and other technology to improve our world in new and different ways via AI. In this way, we will move beyond what we ever thought possible.

AI's Evolution Demands Strong Ethics, Safety

By Kyle Dent for Mouser Electronics

Some inventions are more important than others. Certain innovations have an outsized effect on society, while others, even when they are pervasive, are mere conveniences in our lives. Consider the different impacts of microwave ovens and light bulbs. Although most people these days have a microwave oven, our lives would not be drastically different if some bizarre solar activity somehow zapped all microwaves tomorrow. Most of us would still probably have a cooktop, an oven, a toaster, and maybe even a grill, fire pit, crockpot, pressure cooker, or air fryer standing by and ready to use.

However, suddenly extinguishing all light bulbs would be a different story because convenient lighting has offered a huge boost to humanity's standard of living and economic well-being. Electric bulbs that illuminate vast areas at the flick of a switch replaced the messy lanterns and candles of eras past. They have made illuminated environments the standard rather than the exception for nearly 150 years. Newer advances, such as connected lighting systems and light fidelity (Li-Fi), further integrate light with other building and communication systems.

How important will artificial intelligence (AI) be? Will its impact be more akin to a microwave or to the light bulb? Probably both. Like many other inventions, AI is part of a larger spectrum of development that is solving problems. Potentially, AI could one day be so integrated that it becomes a modern necessity. It also brings strong needs for ethics and safety to ensure that the technology serves the greater good and values human life.

AI: Doing good today ... and beyond

If solar flares managed to evaporate all of artificial intelligence today, most of us would have other options for meeting needs, such as having a stovetop instead of a microwave or revert to processes and capabilities of our not-so-distant past. Business analytics, medical imaging, recommended products, and music playlists would be limited by human capabilities coupled with whatever technology is available. Here again, however, the absence of AI would be more akin to missing microwave ovens than missing light bulbs—at least for this fleeting moment.

At this juncture, several technologies, such as sensors, processing, and storage, have matured and converged to enable AI to solve tangible problems. Of the 160 cases the McKinsey Global Institute have tracked, only about a third have real-life uses, and many of those are still in the testing phase. That said, AI-driven solutions are emerging. For example, an organization called AI for Good Global Summit aims to connect "AI innovators with those seeking solutions to the world's greatest challenges to identify practical applications of AI that can accelerate progress towards the United Nations' Sustainable Development Goals (UN's SDGs)." Its work cites progress in agriculture, drug development, and computer vision for satellite imagery:

Farmers can now integrate massive amounts of data from various sources, including sensors in the field, weather data, markets, and satellite imagery. AI-based time series analysis provides recommendations for increasing crop yields and maximizing efficient land use (**Figure 1**).

Figure 1: *AI-based analysis provides recommendations for increasing crop yields and maximizing efficient land use.*

- Pharmaceutical companies also realize benefits from AI modeling. Researchers can generate and search over huge, even exponential numbers of possible treatments using digital models of molecules and their interactions. As human genome sequencing also progresses, better treatments can be customized for individual cases.
- Organizations use satellite imagery to detect wildfires (a harder problem than you might think) and carbon emissions and even to locate areas of extreme poverty in the world.

AI's potential to do good is quickly transitioning from solving tangible, more immediate needs to becoming increasingly integrated into larger, more abstract solutions. The UN SDGs go beyond solving local and short-term problems to challenging the world to devise solutions to poverty, inequality, climate change, environmental degradation, and peace and justice. Could AI have a role in achieving these goals? The McKinsey Global Institute indicates many of these roles might be related to the UN's SDGs in particular. Some of the nearer-term applications are expected in education and health care. However, applying AI solutions significantly to difficult problems will require action across a spectrum of groups, from governments and organizations to private industry (**Figure 2**).

Ethics in doing good

As artificial intelligence advances and solutions become more far-reaching, ethics will become increasingly more important for ensuring that technologies are used for good and emphasize the value of human life. Most of the published guidelines cite the need for ethical AI as a way to extract the greatest benefit from it. One of the leading groups discussing the ethics of AI is AI4People, which tries to influence governments and organizations. Its goal is to shape the social impact of new applications of AI and lay out the foundational principles, policies, and practices for building a "Good AI Society" framework. The group has made 20 specific recommendations for achieving that end. One of its major concerns is that AI could be underutilized if the public does not trust AI solutions and rejects them. If this happens, the world will not derive the great benefit that could otherwise come from AI.

The European Commission released a white paper in February 2020 promoting the development and deployment of AI to ensure benefits that conform to European values. The white paper emphasizes the ethical implications while it calls for scientific breakthroughs that improve lives and respect human rights. The white paper also warns of the downsides if AI takes on a larger, more intrusive role in human lives. Regulations that already exist can cover some aspects of AI, but existing rules might have to be adapted or clarified concerning AI products. Transparency of AI models will make enforcement of regulations more difficult. Clear regulations are needed to protect citizens and give businesses legal clarity.

The European Commission also emphasizes the importance of a unified effort to reach the scale needed to solve big problems. A fragmented approach from different member countries with different directions and unnecessary duplication of effort risks creating AI solutions that do not scale to the necessary level. The commission plans to involve multiple stakeholders and provide incentives to industry to prioritize Europe's current strengths in technology, including manufacturing automation and quantum computing. It also expects to coordinate among academic centers of excellence. Finally, the white paper mentions using AI to achieve the SDGs, viewing the technology as especially relevant to climate and environmental goals.

The United States has similar ambitions. Individual agencies within the federal government are making plans for AI and publishing white papers describing their expectations. Recently, the Secretary of Energy Advisory Board created a working group to examine and report on AI and the U.S. Department of Energy's (DOE's) role in supporting the development and promotion of AI technologies. The board recently released its report. The DOE emphasizes the urgency of developing AI, seeing it as a new space race with China, which is making large investments in AI.

Prioritizing AI for human benefit could help solve some of the most pressing societal challenges, such as climate change, environmental degradation, and even the protection of democracies. To achieve these ambitions, a strategy

Figure 2: *Prioritizing artificial intelligence for human benefit could help solve some of society's most pressing challenges, such as climate change, environmental degradation, and even the protection of democracies.*

QSFP-DD Connectors, Cages & Cable Assemblies

mouser.com/te-qsfp-dd-connectors-assemblies

TE connectivity

AUTHORIZED DISTRIBUTOR

is needed to coordinate the many stakeholders. Many governments, intergovernmental agencies, organizations, and companies are publishing similar statements and AI recommendation papers.

Safety in doing good

Risks exist in these and future AI uses, a potentially powerful tool that could also be misused and comes with the high likelihood of unintended consequences. In traditional engineering disciplines such as structural engineering or aviation, designs include safeguards wherever possible. New products and systems are extensively tested, and solutions are subjected to stresses to better understand the limits of their capabilities. Armed with knowledge, industry has had success deploying engineered solutions that provide benefit with minimal risk.

The engineering culture and mindset are so far absent from AI, even as we deploy it for real-life situations, fully expecting its significant impact on people's lives. Much of the best practices in traditional engineering have been codified in regulations. As useful as microwave ovens are in our lives, if they spewed radiation throughout our homes, we would not be able to use them. The European Union (EU) white paper suggests that regulations are likely required, but other similar documents from other entities are less clear about how to mitigate potential harms. The U.S. statement on AI specifically calls for minimizing regulation.

In the absence of leadership and guidance from government (except the EU), many other entities have stepped in to fill the gap. A whole body of ethical guidelines has been developed, mostly directed at developers and researchers. At this point, no real mechanism for enforcement exists, but the principles and recommendations identified can help elucidate the key aspects of the conversation when lawmakers and regulators get involved.

Conclusion

Considerable work remains to advance artificial intelligence to the point where it can help solve some of the world's greatest challenges. As we expand AI's potential, we must be cautious of its dark side. The core task of AI is to automate what would otherwise be human decision-making. Doing that requires vast amounts of data but garnering that data risks intruding into our personal lives to understand us better. Balancing these risks against AI's tremendous potential is tricky, but it is also likely to be the difference between the impact of a microwave and a light bulb. M

What's your Tech Quotient?

Put your engineering smarts to the test. Play now!

Download on the App Store

GET IT ON Google Play

MOUSER ELECTRONICS