



The Mind in the Machine

Machine learning is a **scientific revolution**
that is changing how science gets done.



WE'RE NOT TALKING ABOUT ROBOTS (though it would be cool if we were). Sentient humanoid automatons, whether benevolent or malevolent, walking and talking amongst us, are still science fiction. But some things that used to be impossible are now science fact—like computers that can tell if two disparate images are actually showing the same thing or that can predict if and when a supercomputer will crash. Los Alamos has always excelled at data science, and the data-science techniques known collectively as machine learning are now taking data analysis to the next level. Through machine learning, or ML, scientists are exploring new ways of answering old questions, and, in some instances for the first time, they are actually getting some answers.

Machine learning is a natural product of increased computational power. The questions aren't necessarily new, and the math isn't necessarily new. But the machines are, and what scientists are doing with them certainly is. Enabled by major advances in computer hardware and software, and by the massive amounts of data newly available, tech entities from social media companies to national laboratories are using and developing ML.

But while social media and computer companies are mostly working on problems like targeted marketing, virtual assistants, and self-driving cars, Los Alamos scientists are working on mission-critical science problems like nuclear nonproliferation, global security, and ensuring the safety, efficacy, and reliability of the country's nuclear arsenal. The level of performance required for the Laboratory is more stringent owing to the

knowledge. The broad body of physics expertise that exists at the Laboratory, when married to ML, makes new approaches to national security possible.

Scientists at the Laboratory are using and developing ML in a plethora of ways. Some are going after answers to age-old questions, some are asking brand new questions, and some are pioneering new ways of doing and thinking about ML itself. By no means comprehensive, this article provides several examples of machine learning being done at Los Alamos.



What it is and what it isn't

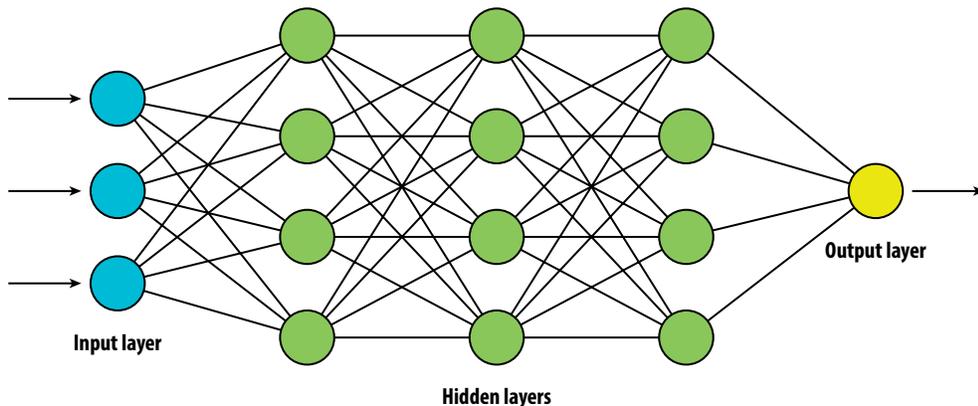
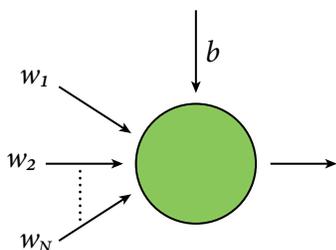
ML is not synonymous with artificial intelligence (AI). General AI refers to learning and reasoning by machines without the intervention of humans, and most scientists agree that we aren't there yet. ML is a specialized subset of AI, wherein a human still writes the code, but the output of the code depends on data—usually vast amounts of it—that is also chosen and fed to the computer by a human. And the computer usually needs to be told, by a human, whether or not it is doing the right thing with the data.

Machine learning is a natural product of increased computational power.

high-stakes nature of these challenges. What sets the Laboratory apart from other entities pushing ML is the intersection of problems and solutions found here—Los Alamos offers leading-edge scientific solutions rooted in rich institutional

The U.S. Defense Advanced Research Projects Agency describes the evolution of AI thus far as having occurred in three distinct waves: The first wave, from the 1970s through the 1990s, was characterized by computers with the ability to reason—but not learn or generalize—as illustrated by IBM's Deep Blue, the chess-playing computer that repeatedly beat the reigning human world champion. The second wave, from the 2000s through the present, is characterized by computers with the ability to learn and perceive—but not to generalize—as illustrated by virtual assistants like Apple's Siri and Microsoft's Cortana (and others). The third wave, mostly still in research labs and not yet producing products ready for mass consumption, will likely last for 10–20 years and be characterized by computers that can reason, learn, perceive,

In a neural network the nodes are called neurons, and each neuron has learnable parameters: multipliers (w) are called weights, and adds (b) are called biases. The exact values of the weights and biases for each neuron are arbitrary at first and become progressively more finely tuned through the iterative training process.



A neural network is a common machine learning method designed and named after biological neuronal systems. The mathematical computation occurs in the hidden layers, which may consist of any number of neurons from one to hundreds. Similarly, the network itself may have anywhere from one to hundreds of hidden layers between the input and output layers.

and generalize. A third-wave computer would, for example, be able to converse in natural language and explain to a human its decision-making process.

So ML is not quite AI, but nor is it simply good programming. For example, mobile navigation apps that find the fastest travel routes aren't using ML; they've just been intelligently programmed. For it to be ML, the machine has to learn an algorithm without being explicitly told how. It's a bit like human toddlers learning to walk. They practice a little and learn what works, or even hold a parent's hand, but the parent isn't saying, "transfer your weight to your right leg; bend your left hip, knee, and ankle simultaneously to lift your left foot off the ground while keeping your right leg straight; move your left leg forward several inches before placing your foot, heel first, back on the ground." And even if the parent were saying all that, those instructions would not compute in the toddler's brain. The parent says, "this is walking, now you walk," and the child figures it out. The steps, or algorithm, are internally generated—they aren't specified by the external programmer (parent), and when the desired outcome is achieved, the machine (toddler) has learned.

With traditional computer programming, data and rules go in, and answers come out. With ML, however, there are two phases: during the training phase, data and answers go in and rules come out; during the inference phase, data goes in and predictions come out.

One popular ML method, out of many, is the use of a neural network. Named after the way neurons organize in brains, an ML neural network is a mathematical model that is organized according to an extreme simplification of living neural systems. Neural networks can be simple, consisting of an input layer, output layer, and single computing layer in between, or they can be complex, or "deep," with many computing layers in between the input and output layers. The computing layers are called "hidden layers" because the user doesn't interact with them, as with the input and output layers. As a calculation progresses through the layers, the resolution becomes finer. Each layer of a DNN can learn a different concept, so part of the challenge for scientists is figuring out the best combination and order of layers.

Training a deep neural network (DNN) is just minimizing a cost function using a collection of known inputs and outputs called a training dataset. Each input goes in, gets worked on by the succession of hidden layers, then emerges as some output, which is at first not very close to the expected known output. The difference between expected output and actual output is the cost, and the way to reduce it is to go back to each neuron, or node within a hidden layer, and adjust what it does, mathematically, to the input it receives. Every neuron has "weights," by which inputs get

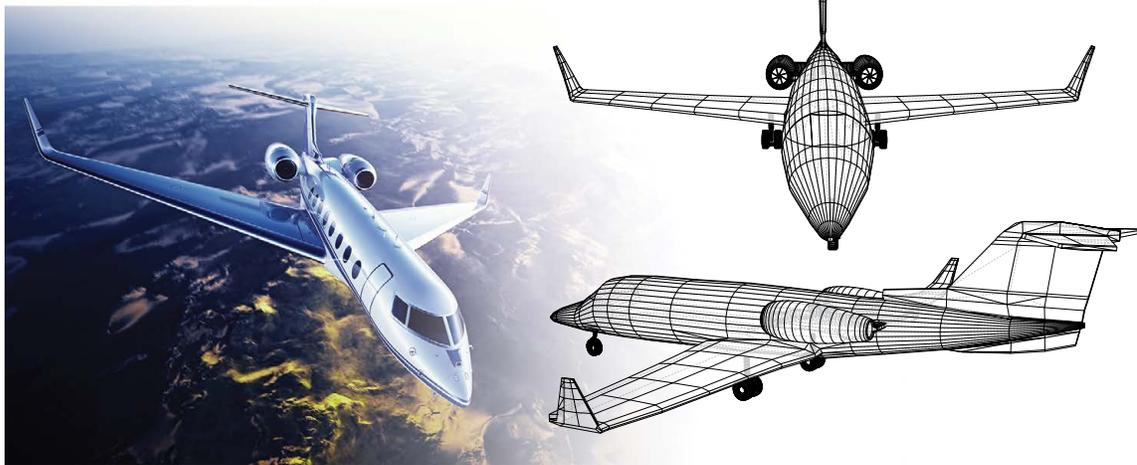
multiplied, and a "bias" that gets added to the output. The weights and biases are the learnable parameters, so their values are arbitrary at first and get repeatedly modified through the iterative training process. Once the cost function is minimized, meaning all the weights and biases are dialed in so that the actual output is as close to the expected output as it can get, the model is trained.



Graphics and schematics

One thing that brains have historically done better than computers is vision. But brains are helping the computers get better. Online reverse-image search tools, for example, when provided with a photo, can find other instances of the same photo or other visually similar photos. And when it comes to classification—is this a photo of a dog or a cat?—ML models are now performing better than humans. But what if the image of interest is not a photograph, but a technical drawing, like a wiring schematic, blueprint, or data graph? In those instances, brains still reign supreme.

For an ML model to assign a photograph to a predefined class (e.g., dog vs. cat), the model compares numerical values for each pixel to the values of its eight neighboring pixels. But unlike photographs, which contain information about color and intensity in nearly every pixel, resulting in texture and



Most image-classification ML models work well on images like this one of a jet, but they fail miserably on technical drawings like these schematics for a similar aircraft. To a typical adult human brain, it is almost immediately apparent that the two technical drawings show the same item from two different angles. Diane Oyen is developing a computer model that will similarly be able to determine that the two drawings show the same thing, despite the fact that they have very few lines in common.

shading throughout the image, technical drawings are line drawings that have very little per-pixel information, so standard image-classification ML won't work. Los Alamos ML expert Diane Oyen is working on ML models to automate the analysis of technical drawings for the Laboratory's nuclear counter-proliferation mission.

"Just like ML models for classifying photographs won't work for technical drawings, our ML approach for technical drawings wouldn't work for photos," says Oyen. "Diagrams have a lot of white space, so our models don't have to go pixel by pixel."

ML photo classifiers fail for technical drawings because these models don't retain spatial relationships—a circle inside a square is the same as a circle next to a square. But Oyen and colleague Liping Yang use a graph-based approach that preserves these relationships. A graph, in this context, is a logical structure defined as an ordered pair of a set of vertices and a set of edges. How images get represented graphically is to first define points, edges, and the relationships between them, then to organize and store this information in a data matrix. Oyen and Yang's method uses non-ML classical computer-vision techniques to delineate the edges and corners of a technical drawing, then a graph-based approach to extract the topology—

Transfer learning is particularly useful for the kind of datasets found at the Laboratory. Many ML models get trained with internet-scale data, consisting of millions of labeled examples (e.g., photos of dogs and cats). But Los Alamos datasets often consist of a small number of highly specialized examples, and labeling them requires a human expert and a lot of time. Being able to transfer what a model has learned on a large, less-specialized dataset to a small, highly specialized dataset is invaluable.

So far, Oyen's model is good at matching replicated images, which, Oyen says, has uses beyond national security applications, such as detecting plagiarism. In scientific research, careers are built on the novelty and ingenuity of data and designs, so plagiarism is a high-stakes affair. If, say, a scientist publishes a data chart in a research paper, then a screenshot of that chart is used in a presentation slide by another scientist, and someone photographs the slide during the presentation and posts that photo to a social media page, a tool like Oyen's could connect the social media photo to the original research publication and the image's rightful owner.

Tech entities from social media companies to national laboratories are using and developing machine learning.

meaningful spatial relationships within the image. In the graph-based approach, different model layers do different tasks for organizing topological information: some layers combine lines and surfaces into shapes, such as circles and squares, and other layers describe the spatial relationships among those shapes. Then the extracted topological features are used to train a standard DNN to image-match the drawing against a collection of known drawings.

Because there is so much ML in development, a developer doesn't always have to train a new model—he or she can use pre-trained models. Taking pre-trained models and stringing them together in new ways, or modifying them for a new purpose is called transfer learning and is one of Oyen's specialties. The main benefit of transfer learning is that it shaves off a lot of the data and computation needed to train the model. DNNs can have millions of weights and biases, but those contained in the early layers of the model (just after input) are very general and affect the rough approximation. Therefore only the later layers (just before output), where more complex features get resolved, need to be changed to adapt the model for a new use.

The way to secure ownership over a new idea or design is to obtain a patent. Here too, Oyen's image-classification tool will be of use. If two different scientists have invented highly similar items, it's unlikely that their individual technical drawings will be identical. Differences in perspective alone would be enough to confound most image-matching computer programs, so the images all have to be evaluated by human brains. Pairwise comparison of hundreds of thousands of images quickly becomes mind-numbingly tedious. Oyen is working to bring semantic information into her technical-drawing-classification tool, so that it will be able to confirm that two images, which contain the same shapes in the same relationship to one another despite differences in scale, rotation, occlusion, or perspective, are indeed showing the same thing.

Through transfer learning, the model can be adapted for other purposes. It can be customized so that subject-matter experts can use it without needing an ML expert on hand. This is called interactive learning, which is a subfield within transfer learning. The model is created from a preexisting model (transfer), but it's built in such a way that the end user can modify it (interactive) according to his or her own needs.

"There are two ways to capture domain knowledge, or subject-matter expertise, in an ML model," explains Reid Porter, Los Alamos interactive learning expert and a colleague of Oyen. "You can either go in and talk to a domain expert, then build an ML tool specific to that person's needs, or you can put the ML directly into the domain expert's hands and let him or her customize it by using it."

Interactive learning is useful for highly specialized applications because it enables general-purpose tools to be fine-tuned on the data at hand, which is often limited in quantity. Microscope

images, satellite images, and time-series data are examples of limited, specialized datasets that would be candidates for analysis by an interactive learning approach. For a real-world, nuclear nonproliferation example, consider the scenario of federal agents encountering some sort of radioactive material being transported across a border. Samples must be sent to domain experts at Los Alamos for determination of the material's provenance. Every sample is unique, so the data are limited, yet confidence is crucial. With an interactive ML tool, experts could automate some parts of, say, microstructural image analysis, in order to spend more of their time on non-automatable conclusions and validation.



Faster output

From its inception over 70 years ago, the Laboratory has been a world leader in computer simulations of atomistic and molecular systems. ("Atomistic" refers to the tracking of each atom in a collection of atoms, as in a material, in contrast with "atomic," which generally refers to single atoms and their substructure.) Predicting how groups of atoms or molecules will interact with one another is and has always been central to Los Alamos's mission.

Laboratory physicist Nicholas Lubbers develops physics-informed ML methods to help materials scientists, chemists, and molecular biologists model the chemical and physical properties of the atoms and molecules they study. They want to know, for example, how a shock wave will propagate through a certain kind of metal, or how an enzyme will interact with DNA inside a human cell. For these domains, a model has to obey the laws of physics, but mainstream DNNs don't typically include even basic physics principles. Lubbers and his colleagues have been working to marry the flexibility of ML techniques with the constraints of

physics. They build DNN architectures that encode exact physics properties, such as translation and rotation invariance, as well as approximate properties that are found in atomistic systems, such as locality. The result is physically valid ML models that are much more robust and accurate for large, complex systems.

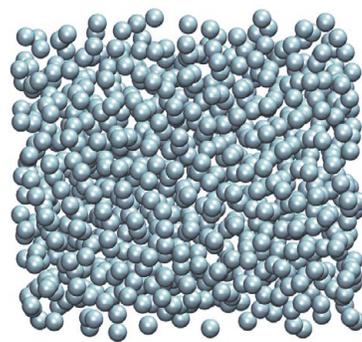
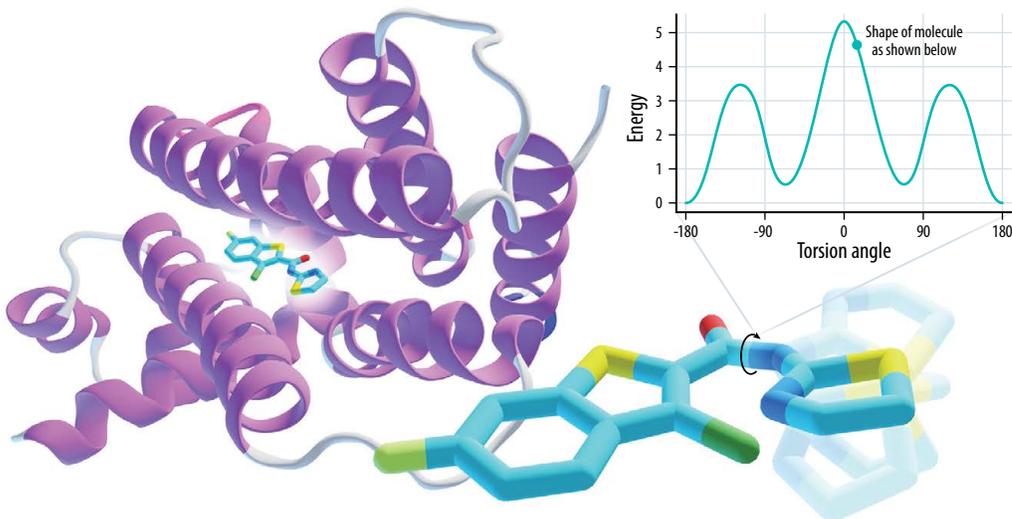
When it comes to methods for computationally modeling the energies of individual atoms, bonds between atoms, clusters of atoms, individual molecules, and molecular interactions, there tends to be a tradeoff between affordability and accuracy. Methods using the equations of classical physics are very affordable but are of limited accuracy and lack transferability to other domains, while methods using the equations of quantum physics are highly accurate and transferable but scale poorly—quickly becoming cost restrictive as the number of atoms in a simulation grows.

The scientists have been using transfer-learning techniques to develop best-of-both-worlds solutions. Basically, they train a DNN on vast quantities of classically computed approximate data to get the model's basic structure, then retrain the last few layers on higher-quality but lower-quantity data generated by the best quantum-physics calculations to perfect and polish the model. In this way, the model operates with classical cost and quantum accuracy, allowing scientists to simulate larger systems over longer time scales. If necessary, the ML model can occasionally be compared to quantum-mechanical calculations as a sort of spot check to ensure accuracy.

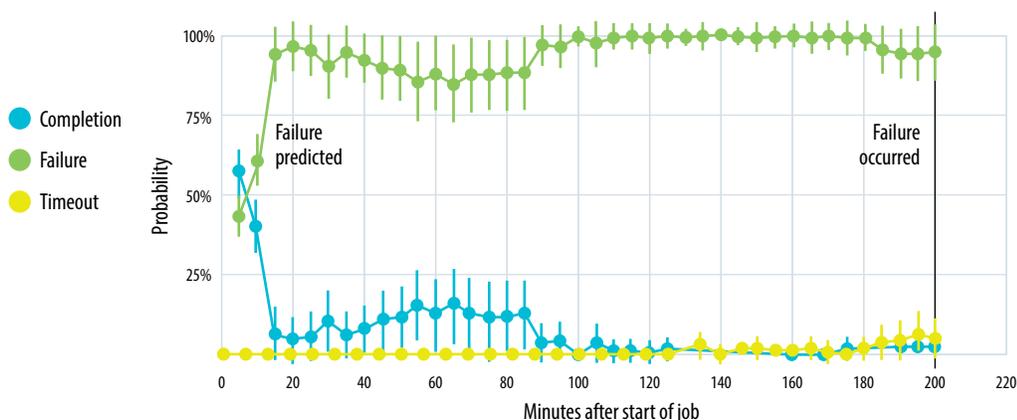
When training an ML model, it is possible to overtrain, which undermines transferability. For example, if you were

ML is not quite AI but it is much more than good programming.

Nicholas Lubbers and colleagues use ML to accelerate the simulation of various complex systems. This simulation shows an experimental drug molecule interacting with a protein from *Mycobacterium tuberculosis*, the pathogen that causes tuberculosis. Some bonds between atoms are flexible, allowing the drug molecule to take different shapes, each of which might interact differently with the pathogen protein. Lubbers' model predicts how much energy it takes for each possible shape to form, which tells the drug designers how likely it is that the drug will take that shape when it interacts with the pathogen protein. For example, as the right-hand portion of the drug molecule rotates about the indicated carbon-nitrogen bond (torsion angles from -180 through $+180$), thereby changing the molecule's shape, the energy of that bond also changes. Every atom of the pathogen protein and the drug molecule is included in the simulation, which helps drug developers understand how the molecules might be expected to interact, thereby helping them assess how effective the drug might be. CREDIT: Justin Smith/LANL



ML models can be repurposed through retraining. Here, the same method that was used for the pathogen-drug interaction simulation at left has been applied to study the properties of liquid aluminum. ML is better than classical computing methods at simulating disordered arrangements like the atoms or molecules of a liquid at a high temperature. CREDIT: Justin Smith/LANL



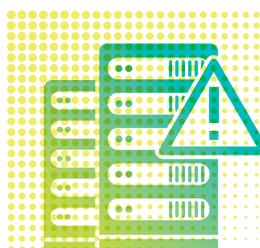
Lissa Moore is developing ML methods to improve high-performance computing by predicting job timeouts and system failures before they occur. In this proof-of-principle experiment, the ML tool was given a job that was known to have failed after about 200 minutes. Initially the model predicted equivalent likelihoods of completion or failure. By about 15 minutes in, nearly three hours before the job actually failed, the model began predicting a much higher probability of failure than success. This work has implications for improving the efficiency and reliability of supercomputing simulations.

trying to learn the names of a group of people, and each person wore the same color shirt many days in a row, you may inadvertently learn that “Bob” equals “red shirt” and “Barbara” equals “yellow shirt.” But what happens when, one day, Bob and Barbara are both wearing green? This kind of false cue, called overfitting, can thwart ML as well, resulting in a model that has memorized rather than learned, and so is overspecialized and not transferable. The model has to incorporate known physics to answer known questions, but it also has to be transferable in order to get at unknown questions.

Whereas ML offers accelerated computation for atomistic systems, Lubbers’ broader goal is to create models and methods that are transferable, so that they might be applied across many domains. Using transferable models can help scientists learn things they didn’t know before and predict things they have never seen. It can help them define the very questions they need to ask.

The questions one has to ask are tied to the nature of knowledge.

“The questions one has to ask in scientific machine learning are tied to the nature of knowledge,” Lubbers says. “If the algorithm learns, but the human doesn’t, what has really been achieved? As we explore what’s possible with machine learning, we are also learning how to approach a problem so that we will gain scientific knowledge as well.”



Better outcome

High-performance computing (HPC) simulations are one of the main ways that high-quality, high-stakes science gets done. And ML, whether physics-informed or not, can improve the reliability and performance of mission-central HPC simulations themselves.

“If the coin is the intersection of high-quality science with national security challenges,” explains Laboratory ML expert and former Navy research scientist Ben Migliori, “then the two sides of that coin, the two ways we can use ML to that end, are, yes, embedding the physics in the model when we know the physics of the system, but also, when we don’t actually know the physics and therefore can’t embed it, understanding how and when it will fail.”

Los Alamos ML expert Lissa Moore is developing methods to do just that. Not to be confused with *output*, which is the answer to the question being calculated, the *outcome* of a simulation is whether the simulation itself will run successfully, or if instead the system will crash, time out, or otherwise stall before completion. These kinds of interruptions increase the time and cost of HPC simulation, but preventing them requires knowing about the health of the HPC cluster and anticipating abnormal behavior.

An HPC cluster, or supercomputer, is basically 2000 computers networked together to do one thing. There are different ways to set it up, but no matter how it’s set up, it needs continual monitoring.

Both hardware—processors, networks, memory modules, etc.—and software—operating systems, user codes, job schedulers, etc.—get monitored, which generates terabytes of system-health data every day. Moore uses ML to take in all that data and make sense of it so that

decisions can be made about if, when, and how a person should intervene to optimize the outcome.

Scientists queue for 12-hour time slots on the Laboratory’s HPC clusters, so their codes have to run to completion in that time. If the operators who supervise the HPC clusters knew in advance that the system was going to crash, they could kill a simulation early and restart it, perhaps with enough time to complete a re-run. Or, if they knew that the simulation was going to time out, the operators could save the data midway, so that

when the user re-runs the simulation, it can pick up where it left off, rather than starting over and timing out again.

“This is important,” says Moore, “because until now we’ve had only rudimentary tools looking for known problems. ML can look for new problems, ones we haven’t anticipated or seen before. The machine learns what normal and healthy looks like, then tries to spot any deviation from the norm.”

At present, the learning task is just recognizing potential problems. If the model predicts something unusual is going to happen, it raises an alert to a human who makes the call as to whether or not it is a real problem, and if so, what to do about it. The next step will be teaching the model to make that determination itself, and after that will come teaching it which actions to take in certain circumstances. Humans will still be in charge, but the ML tool will help them keep HPC simulation as efficient as possible.

Independently of her supercomputer-health monitoring work, but not unrelated, Moore also works on explainable machine learning. Firmly in third-wave AI territory, explainable ML is when the model not only makes a decision by itself, but can tell the user why it decided what it did.

A brain out in the world will learn things without explicitly trying.

What’s going on under the hood of a trained ML model is typically not intuitive. Because the hidden layers are hidden and the algorithm is not defined by the programmer, the details of the mathematics in those layers are a bit of a black box. The programmer can learn about the trained model by sort of poking at it phenomenologically: vary the input and see what effect it has on the output. This is how one might discover that a model has learned Bob and Barbara’s shirt colors, rather than their faces. But with explainable ML, the model could actually report back, “I know this is Bob because of the red shirt he’s wearing,” or, “I know this is Bob because I recognize his face.” Being third wave, the field of explainable ML is still quite new and still rapidly advancing, but the potential is tremendous, offering a new level of performance and reliability for HPC simulations and other ML applications alike.



More like life

The Laboratory’s Trinity super-computer runs on 20 megawatts of electricity while the human brain runs on about one millionth of that, yet it’s unclear which is more powerful. They

can do roughly the same number of operations per second but they have very different skill sets.

“Here’s the interesting thing about brains from an ML perspective,” explains Los Alamos ML expert Andrew Sornborger, “Their main characteristic is that they learn. You send a brain out into the world, and it will interact with its environment and learn things without explicitly trying.”

“But,” adds Garrett Kenyon, a Laboratory physicist-turned-neuroscientist who collaborates with both Sornborger and

Migliori, “no one really knows how brains do it. How we train machines is not how we ourselves learn.”

Neuronal communication works through activity spikes. In a brain these spikes are ion-mediated charge fluctuations called action potentials, which send neurotransmitters across synapses to neighboring neurons. Spiking neurons offer more computing power for less electrical power, but most ML uses conventional, non-spiking communication schemes. Kenyon, Sornborger, and Migliori want to build ML models that operate more like brains, and they are all-in on spiking schemes. But spiking schemes don’t map well to traditional computer architectures. So the scientists are using Loihi (low-EE-hee), a neuromorphic chip made by Intel, that directly emulates neural processing and synaptic plasticity in the brain by computing with biophysically-inspired spiking neurons.

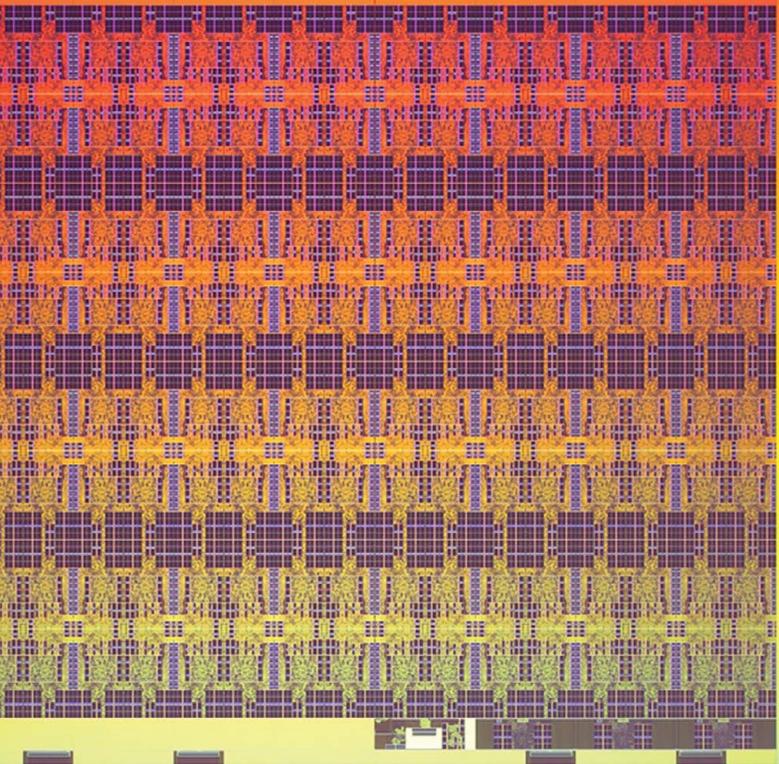
Nearly all ML is supervised learning, meaning that a human tells the machine what its task is and when the task has been achieved. Brains also do supervised learning, but when they are brand new, they learn in an unsupervised way. In unsupervised learning, there is no specific task, other than taking in whatever data there is to take in and organizing it somehow. Just like

toddlers learning to walk, human babies learning to see are not given stepwise instructions. As they are carried through the world with their eyes open and their brains on, they rapidly learn that things are different from other things and automatically create cognitive categories for all the things. The neurons of the baby’s brain self-organize during unsupervised learning into a cortex that can process and represent all the incoming sensory data. This natural learning (unsupervised) is a prerequisite for the brain to be able to make use of the more structured training (supervised) the child will receive later, for example, when he or she goes to school.

Kenyon and Migliori are using Loihi as an unsupervised learning module that learns to represent visual input through self-organization. Just like a newborn, Loihi will be carried through the world seeing whatever there is to see (through a special camera), until it self-assembles all of its neurons into a kind of primary visual cortex.

Kenyon likens it to a digital crow. “Crows are very intelligent,” he says. “A Loihi brain with a silicon retina, mounted on a drone, could be lightweight, low-power, solar-rechargeable, and fully autonomous. That’s what I want to build.”

But to assign a specific task to the digital crow, like “find blue cars and squawk when you see one,” requires going back to supervised learning. By having done unsupervised learning first, the amount of training data needed for subsequent supervised training is drastically reduced. But, even though the amount of training data is reduced, the ML model still has to be “rewarded” and “punished” mathematically as it learns how to minimize the cost function. This can’t be done on Loihi in the same way it is done on other processors, and that is where Sornborger comes in.



The Loihi chip, a neuromorphic processor introduced by Intel in 2017, is enabling the development of new machine learning models that emulate learning by human brains. The chip includes 130,000 neurons and 128 cores arranged in an architecture that is optimized for spiking neural network algorithms that mimic the electrical spiking behavior of neurons in human brains.

CREDIT: Intel

On a DNN, doling out rewards and punishments is the feedback-motivated adjustment of weights and biases across the network, which is also called backpropagation of error, or simply backpropagation. But, until recently, backpropagation algorithms had not yet been implemented on Loihi, because of its unique, neurally based architecture. Brains have all the necessary information for changing their connections right there where the change occurs, at the synapse. Similarly, Loihi, as a brain emulator, implements local learning rules, where error is addressed right at the connection between neurons, rather than across the network. Sornborger and

colleagues have taken a fundamentally nonlocal learning rule, backpropagation, and figured out how to implement it on Loihi, a system that only uses local rules.

“This is what I’m most excited about,” says Sornborger, “we’ve had backpropagation algorithms for years, but we didn’t have the computers. Then we got the neuromorphic processor but couldn’t do ML on it. Now that we’ve figured out backpropagation on Loihi, very big gains stand to be made.”

These gains could be as pragmatic as drastically reducing the energy consumption of big server farms, as futuristic as the digital crow, or as lofty as a fundamental understanding of how our own human brains operate.



Making Connections

From solving old problems in new ways to finding new problems to solve, ML is a leading-edge technology that is only going to increase in popularity. This omnipresence is a boon to creative problem solving, but as different scientists across the Laboratory pursue ML to various ends, there is a risk of reinventing the wheel. To address this risk, a small team of researchers from three divisions across the Laboratory have been working for the past two years to coalesce what they’ve dubbed a “community of interest” in ML at Los Alamos.

“The community of interest consists of people who are developing ML as well as people who are interested in learning to use it,” says Juston Moore, a Laboratory ML and cybersecurity researcher who first proposed the community-building effort. “It’s difficult to implement ML in a safe and reliable manner, suitable for the Laboratory’s critical national security mission. It should be a capability that is distributed across the Lab and a tool that is accessible to anyone who needs it. Domain experts should be able to vet their ML algorithms using a network of connected ML experts.”

“We want to foster institutional knowledge,” elaborates Migliori, who is part of the team, “just like the institutional physics knowledge we’ve amassed. Ideally we would have ML evangelists embedded in each group to help found ML projects.”

Whereas the Laboratory’s physics expertise has had 70 years to amass, ML is a relative newcomer, so Moore, Migliori, and their team want to speed things along.

There are hundreds of people across the Laboratory who work with or on ML, and the community-of-interest team has so far built two main mechanisms to help bring all those people together. First, they have launched a seminar series on adversarial machine learning—an emerging sub-field that uses competing attacker-defender models to strengthen performance—to get people into the same room and begin breeding familiarity. Second, they have established an internal topical chat service so people can converse in real time with ML colleagues, without clogging one another’s email inboxes. Both efforts have been well received, word is getting around, and the network of ML people is solidifying.

There are many more projects that use ML at the Laboratory than the ones covered here. From improving cancer diagnosis to predicting earthquakes, from power-grid optimization to turbulence modeling, machine learning is revolutionizing how national security science gets done. **LDRD**

—Eleanor Hutterer

How we train machines is not how we ourselves learn.