DARPA's Role in Machine Learning

Joshua Alspector, Thomas G. Dietterich

■ *Machine learning methods provide a way for artificial intelligence systems to learn from experience. This article describes four threads of machine learning research supported and guided by the Defense Advanced Research Projects Agency — probabilistic modeling for speech recognition, probabilistic relational models, the integration of multiple machine learning approaches into a task-specific system, and neural network technology. These threads illustrate the Defense Advanced Research Projects Agency way of creating timely advances in a field.*

Machine learning (ML) is a subset of artificial intelligence (AI) methodologies in which the AI system learns from experience. Experience can include observational data, labeled training data, interactions of the AI system with its user and environment, and teacher demonstrations and instruction. Today, ML approaches are ubiquitous in real-world applications including online assistants and computer vision. ML systems can formulate natural-sounding language responses and recognize faces to unlock phones. There are also lesser-known ML application areas such as manufacturing, health and medicine, finance, law, and agriculture. The Defense Advanced Research Projects Agency (DARPA) has invested heavily in developing ML technologies since before computer hardware was powerful enough to realize their potential.

This article describes four threads of ML research supported and guided by DARPA. The first thread concerns probabilistic modeling, which was first developed by DARPA

 ISSN 0738-4602

in the context of speech recognition. Through a long series of steps, this thread led to the development of probabilistic graphical models (also known as Bayesian networks) that enabled AI systems to reason efficiently with uncertainty. The second thread — probabilistic relational models (PRMs) — was launched by the Evidence Extraction and Link Discovery (EELD) program; research in this thread developed ways of combining logical and probabilistic reasoning, enabling, for example, AI systems to reason about the various tuples and relations in relational databases. Using such models, researchers were able to develop systems for populating knowledge bases by reading news stories, web pages, and Wikipedia articles. The third thread concerns methods for combining multiple ML components into an integrated AI system that can be trained end-to-end. The final thread involves the progressive development of neural network (NN) technologies from the earliest days of single-unit networks to the multilayer perceptrons of the 1990s, and finally to the deep learning (DL) techniques that revolutionized computer vision, speech recognition, and natural language processing in the 2010s.

Each of the four threads illustrates the power of the DARPA Way in which a program manager, working with the research community, articulates a technical challenge and a performance target that guides performers to rapidly and significantly advance the field.

ML Overview

Although often there is little distinction between AI and ML as colloquial terms, on a technical level ML is a clear subset of AI. ML itself has several subdisciplines, including NNs and statistical methods such as support vector machines (SVMs). Table 1 summarizes the three main ML tasks and eight major ML technologies.

Different from AI in general, ML technologies use one of the three methods to learn from experience: supervised learning, reinforcement learning, or unsupervised learning.

Supervised learning, or learning to perform a task through imitation, can be considered learning from a teacher whereby the system is shown the correct output for each set of inputs. After some number of repetitions (dependent on the exact technique), the system can reproduce what the teacher does. This is used for image and sound classification and is the main method for DL in computer vision and speech recognition.

Reinforcement learning, or learning to perform a task through trial and error, can be considered learning from a critic that does not show the correct answer but indicates whether the system's answer should be rewarded with some numerical score or an up or down vote.

Unsupervised learning has no signal for learning other than the structure of the data itself, or by learning structural relationships in the world that can be used in subsequent problem solving. If many of the patterns are similar, they form clusters that serve to organize the data into groups of similar patterns.

The earliest supervised learning methods borrowed techniques from statistics and engineering (for example, linear discriminant analysis, quadratic discriminant analysis, linear threshold perceptrons). Samuel's Checkers Player (Samuel 1959) was the first reinforcement learning system. Clustering and principal component analysis were among the earliest unsupervised learning methods. The same three learning methods are still used today.

Suitable AI Tasks for ML

To apply ML to a problem, we must first represent the solution in a learnable representation, described below with an example. Then we must collect or present data (for supervised and unsupervised learning) or provide a way for the learning system to collect data itself by interacting with the environment (for reinforcement learning). Finally, we must define a measure of accuracy or error; this is known variously as the loss function, the cost function, or the reward function. The goal of learning is to find a specific representation in the space of learnable representations that optimizes the loss function when measured on new data.

For example, suppose we wish to read handwritten letters and numerals. Following the lead of a research group at AT&T in the 1990s, we will choose a five-layer convolutional NN similar to the one shown in figure 1. The computational behavior of this network is determined by a set of numerical weights in each layer. We collect training data in the form of labeled training examples in which the input is a black-and-white image that contains one letter or numeral and the known output is the corresponding label that specifies the letter or numeral. Learning consists of adjusting the weights in the network such that the loss function (which measures the difference between the correct answer and the answer computed by the network) is minimized.

We can use the character recognition example above to explain a learnable representation. A learnable representation must have an algorithm that can find an instantiation of the representation that minimizes the loss on the data. The easiest tasks to approach with ML are one-step prediction tasks such as character recognition, as described above. In such tasks, the goal is to learn a function. There is no memory or inference in such tasks. The input is transformed through a series of one or more steps into the output.

Slightly more complex are sequence-to-sequence mapping problems. For example, suppose we have an image containing a handwritten word and we want to transform this into the sequence of letters in the word. We could map each one separately, but if we learn a model for typical letter sequences (for

Tasks and Techniques	Probabilistic Graphical Models	Nonparametric Probabilistic Models	Decision Trees	SVMs	Nearest Neighbor Methods	Linear Models	1- and 2-Layer NNs	Deep NNs
Supervised Learning	X	X	X	X	X	X	X	X
Reinforcement Learning					X	X	X	X
Unsupervised Learning	X	X	X	X		X	X	X

X indicates where there is a well-established approach to solving the selected task using the indicated technique.

Table 1. Summary of ML Tasks and Techniques.

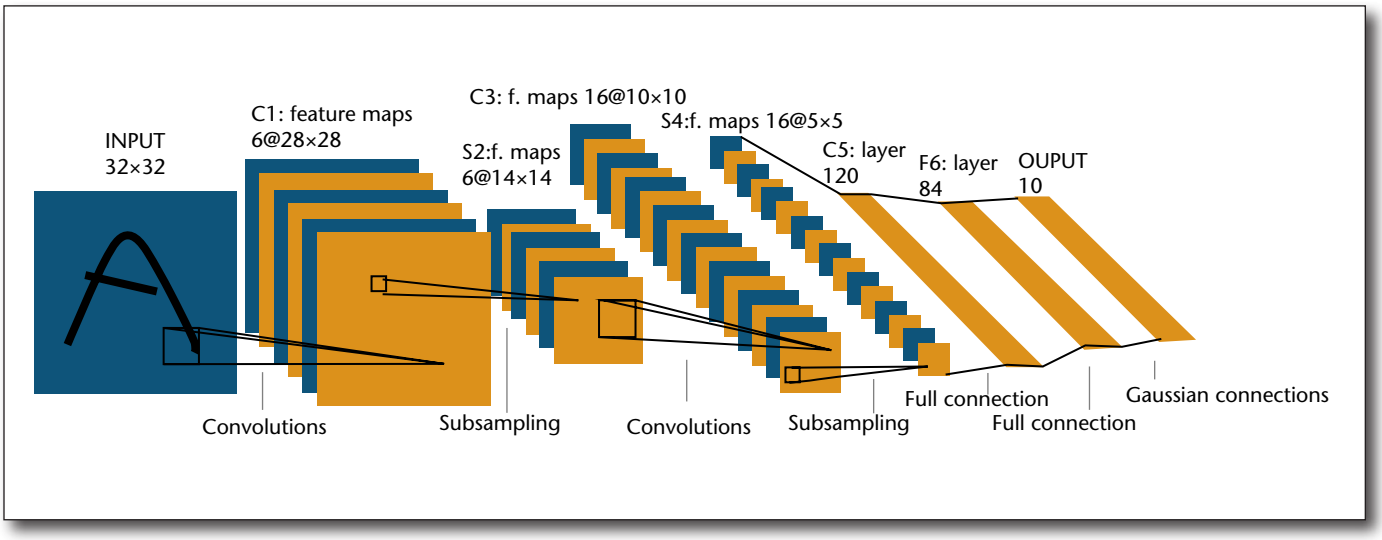

Figure 1. A Convolutional Neural Net, LeNet-5.

From Lecun et al. (1998), ©1998, IEEE, reprinted with permission.

example, that *g* is often followed by *h* in English), then we can do a more accurate job. In other words, if there are correlations among the component steps (i.e., recognizing the individual characters), then we can learn and exploit those correlations to improve performance. A simple way to do this is to replace the function with a finite state transducer. Probabilistic finite state transducers are known as hidden Markov models (HMMs). NN transducers include recurrent NNs, which process one input at a time while maintaining an internal state, and transformer networks (Vaswani et al. 2017), which learn to pay attention to the parts of the input sequence that are relevant to producing each output element.

Even more complex than sequence-to-sequence problems are tasks that include some form of optimization. For example, suppose we want to design music playlists. Music recommendation systems can recommend individual songs based on the songs a person has liked in the past. But to create a good playlist, we need to select a set of songs that constitutes a good playlist, meaning that the sequencing of the songs should make an overall playlist good for a particular user. This can be accomplished by first choosing an optimization algorithm that can take a set of candidate songs as an input and arrange them into a good sequence of the desired length. Then a learning algorithm is given two tasks: first to learn the user's definition of a good sequence and second to learn to produce lists of songs that, when given to the optimizer, will result in a good playlist. This is known as learning through an optimizer.

Similar challenges arise in playing games. One way to create a game-playing program is to learn an

Pushing DARPA in New Directions

I was a very young junior faculty member at Stanford when I first received funding from DARPA. At that time, the reigning approaches for reasoning and even learning were largely based on symbolic logic. I had come from the world of probabilistic modeling and ML, and was hoping to use those methods for reasoning about complex, richly structured domains. Dave Gunning, the program manager for the High Performance Knowledge Base (HPKB) program, was willing to give me some funding, even though the approach I was advocating was a major departure from the other methods used in this program. Through a series of conversations, I convinced Dave Gunning that probabilistic methods could be useful for analyzing relational data such as the links between people, places, organizations, and so on. These methods turned out to be better able to deal with the high degree of uncertainty in these networks and consequently produce more robust and reliable results.

While my work was an outlier in the original program, the success of these methods induced Dave and others at DARPA to incorporate them into the Broad Agency Announcement for the EELD program, from which I also subsequently received funding. My initial experience with HPKB and EELD taught me that DARPA is open to considering new methodologies, even if they are a departure from the original trajectory of the agency. The key is to have deep technical discussions with program managers, and to be able to demonstrate that these alternative techniques solve real problems that are relevant to DARPA's mission.

These programs were central in providing support for the work of my students Avi Pfeffer, Lise Getoor, and Ben Taskar in the formalization of the framework of PRMs and the development of efficient algorithms for learning and reasoning with these models. These provided the impetus for the creation of a research community that studies Statistical Relational Learning and continues to hold regular workshops. This line of work, and the associated funding by EELD and the Transfer Learning program, was also the inception of the seminal paper by my students Ben Taskar and Carlos Guestrin on Max-Margin Markov Networks.

DARPA's willingness to let us explore an alternative and untrodden path in the early days of HPKB provided both a catalyst and the support for a new and important paradigm shift in ML — the move away from simple single-outcome prediction problems, and toward the broader paradigm of structured prediction, where the goal is to simultaneously label sequences, grids, graphs, or even general networks of interrelated objects. While the ML approaches have evolved considerably, this perspective of discriminative models for complex domains remains, and is now routinely applied across multiple real-world applications in natural language understanding, computer vision, and even computational biology, where I hope that it will allow us to provide a new approach for improving human health.

– Daphne Koller

evaluation function that assigns a goodness score to each game state such that a limited-horizon search algorithm (such as Monte Carlo Tree Search) will find a sequence of moves that leads to a win. Learning a good evaluation function is another example of learning through an optimizer — in this case through a search procedure.

Many interesting AI tasks have been formulated and solved using ML. In natural language processing, tasks such as language modeling, part-of-speech tagging, named-entity recognition, semantic role labeling, dependency parsing, and many forms of information extraction have all been attempted. In computer vision, tasks such as object recognition, object detection, object segmentation, object counting, object tracking, gait recognition, and activity recognition have been studied. Many problems in science, engineering, and medicine have been addressed via ML including ranking precancerous cells in medical images, disease diagnosis, and treatment planning.

Although there have been many successes, it is also constructive to consider tasks that have resisted ML. Extended conversations in natural language, for example, require a theory of mind in which Participant A must reason about what Participant B thinks Participant A believes. Attempts to learn this from observed conversations have largely failed, presumably because we lack an effective way to represent the required knowledge. Self-driving cars are another application with challenges that ML has not solved. For example, a driver who looks in the mirror at a car following too closely may infer that the following car's driver wants to pass and will move over, but self-driving cars have been unable to incorporate this theory of mind. Other examples include those with limited training data such as training a robot by demonstration to perform novel manipulation tasks. People can generally learn from one demonstration, but existing methods for generalizable imitation learning

or reinforcement learning typically require hundreds or thousands of trials. Finally, existing ML methods only work in closed worlds where the situations encountered at run time are similar to the training data. Extending ML methods to deal robustly with surprises in open worlds is an open and important research challenge.

DL and NN History

ML has been revolutionized by the advent of a particular NN method known as DL or deep NNs, the focus of the fourth thread of this paper. Deep neural nets are NN models that are vastly larger and more complex than all previous ML models. Consequently, they are able to solve much more complex tasks at the expense of more training data and computation. The DL revolution was sparked by a combination of three factors: much larger data sets; algorithmic innovations that could train these complex networks; and implementation of those algorithms on graphical processing units (GPUs) that could deliver the required computation.

One large data set is ImageNet LSVRC-2010, which contains 1.2 million images labeled with 1,000 categories of objects. In 2012, a DL model developed by Krizhevsky et al. (2012) won the annual ImageNet competition by a huge margin. Since that moment, DL has come to dominate the landscape for recognizing images and videos, understanding speech, and translating language. It has become commercially valuable, especially in large technology companies such as Google, Amazon, Microsoft, and Baidu. For example, Facebook employs DL methods with many layers of neuron-like processing to recognize faces. DL is being applied to perception and decision-making for self-driving cars. Financial firms are trading securities using DL prediction on the data from stocks and bonds.

The history of NN approaches to ML dates from 1943 when neurophysiologist Warren McCulloch and mathematician Walter Pitts modeled a simple binary state threshold neuron with weighted synaptic inputs. Networks of such devices with binary inputs could implement ANDs and ORs, and therefore, any Boolean function. In 1949, Donald Hebb proposed that weights increase when connecting neurons fire simultaneously. Sophisticated variants of such simple neurons and Hebb's simple synaptic weight adjustment rule, or learning algorithm, along with architectures of neural connections, comprise NN learning today.

The perceptron learning algorithm for simple binary NN was invented in 1957 at Cornell by Frank Rosenblatt. In 1959, Bernard Widrow and Marcian Hoff of Stanford developed similar models for analog values called ADALINE and Multiple Adaptive Linear Elements (MADALINE). Still, in 1970 the Marvin Minsky and Seymour Papert book *Perceptrons* suggested that the single-layer NN approach, which was the only approach with a known learning algorithm, was hopelessly limited. This was partially responsible for the decline of NN funding, known as the *neural net winter*; traditional von Neumann architecture dominated AI research and funding in the 15 years following.

The 1980s saw the return of biologically inspired AI starting with a brain-like NN model of vision by Kunihiko Fukushima called the neocognitron based on earlier neuroscience results. This model has similarities to the convolutional NNs in use by DL vision systems today. In 1982, physicist John Hopfield of Caltech presented a content-addressable memory with bidirectional connections between McCulloch-Pitts neurons and, as a result, many people became interested in NNs again. The Boltzmann Machine (Ackley, Hinton, and Sejnowski 1985) extended this model with a learning rule that could learn synaptic weights in two or more layers. In 1986, several groups of researchers came up with similar ideas to extend analog ADALINEs to multiple layers. This method is called back-propagation of errors (Rumelhart, Hinton, and Williams 1986) and is more computationally efficient than the Boltzmann Machine. This started a revolution in which many applications for NNs were demonstrated. The hardware, software, and training data available allowed for learning two layers of weights which, in theory, is all that is necessary for learning any function.

In the 1990s, statistical methods such as decision trees, SVMs, and Bayesian techniques were shown to have properties similar to NNs but with better theoretical foundations. Consequently, they dominated the ML landscape until the DL techniques of the past decade showed their superiority and brought NN methods back into vogue.

Many layers of interconnected neurons were shown to be learnable in the 2010s (the deep convolutional character recognition model of Lecun, Bouttou, Bengio, and Haffner [1998] previewed this revolution), allowing sophisticated feature hierarchies to be learned rather than hand-coded, which led to spectacular improvements in performance. The needed learnability for deep networks eluded earlier researchers largely because the scale of computation necessary was not available. Current approaches rely on the availability of GPUs to parallelize and accelerate the necessary computations for learning on the massive datasets assembled by using internet data and other sources.

This history is sketched in figure 2 with funding milestones by DARPA. The four colors represent the four threads to be discussed in the next section. Uncolored areas represent the dates of other DARPA programs and significant events.

Four Important Threads in DARPA-Sponsored ML

Over the years, DARPA has sponsored research in many aspects of AI. Early programs did not fund ML directly, but rather included it as part of larger

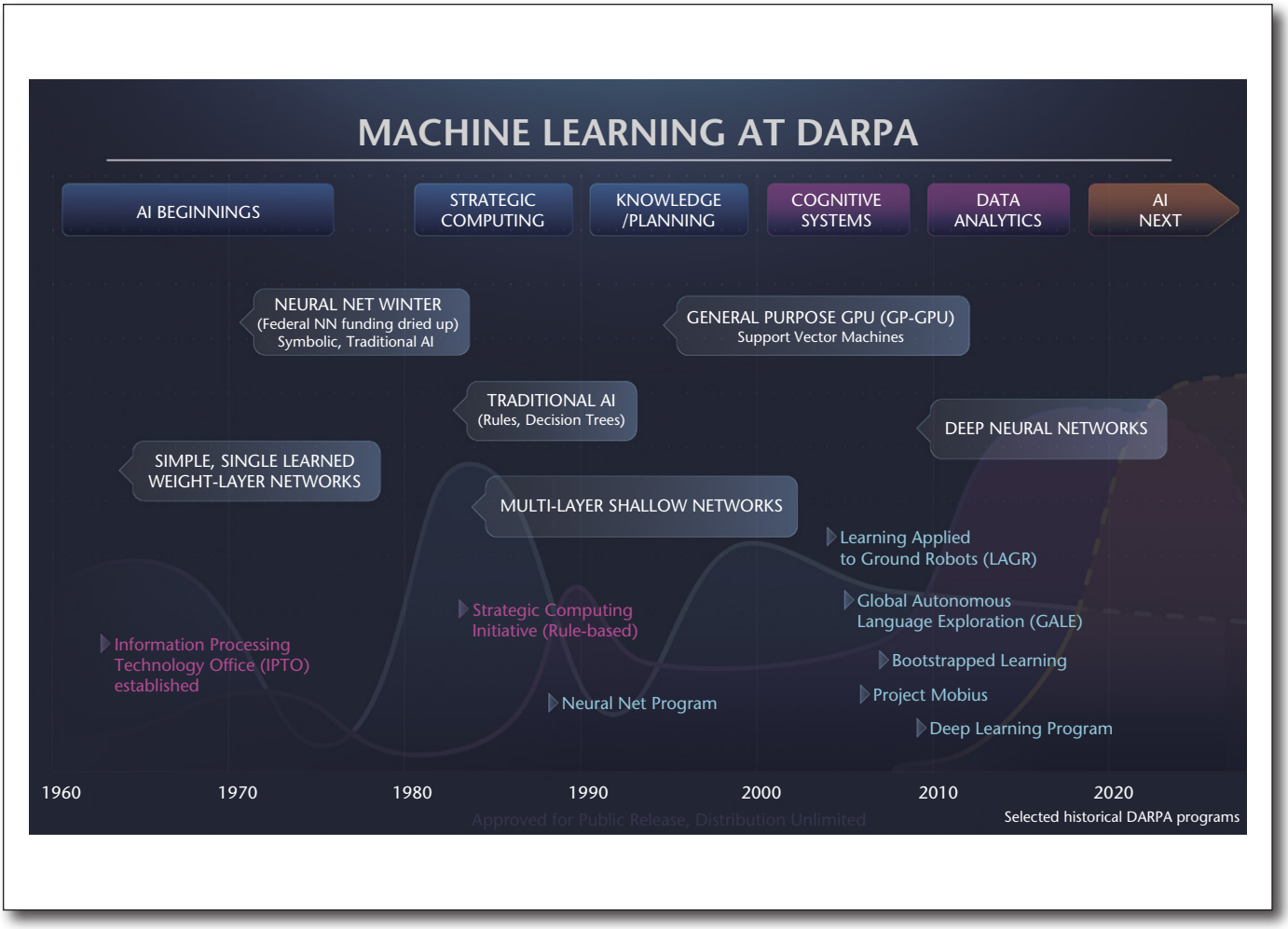

Figure 2. ML at DARPA.

Selected historical DARPA ML programs and technologies. *Figure courtesy of DARPA.*

efforts. This is typical of the DARPA style, which is to pose a challenge to the research community without specifying exactly how that challenge should be achieved. Some teams developed ML approaches to meet those challenges. A good example of this is the role of ML in speech recognition.

ML Approaches to Speech Recognition

In 1971, DARPA launched a program on speech understanding systems; the first thread of ML.

> Instead of setting vague objectives, DARPA and the research community proposed a set of specific performance goals (Newell et al. 1971). The system was required to accept connected speech from many speakers based on a 1000-word vocabulary task-oriented grammar, within a constrained task. The system was expected to perform with less than 10% semantic errors, using about 300 million instructions per second of speech (MIPSS) and to be operational within a five-year period. (Carnegie Mellon University Computer Science Speech Group 1977, p. 1.)

At Carnegie Mellon University, two teams attacked the challenge. One team, which involved many people including Lee Erman, Frederick Hayes-Roth, Victor Lesser, and Raj Reddy, developed the Hearsay-I and Hearsay-II systems based on what came to be known as the Blackboard Architecture (Erman et al. 1980). The second team, consisting of James Baker and Raj Reddy, developed the DRAGON system (Baker, 1975) based on building and training a HMM. Both systems were very influential in subsequent years. The Blackboard architecture was developed into a reusable platform for creating complex problem-solving systems. Meanwhile, the application of HMMs (and closely-related models) continues to the present.

An HMM is an example of a probabilistic generative model. The model is designed by imagining that the person speaking chooses a meaning to express as speech. To express the meaning, the speaker generates a sentence (as a sequence of words). To generate

the speech, the sequence of words is expanded into a sequence of phones, which are then expanded into an acoustic waveform. We can summarize the model as a conditional probability distribution $P(AS|M)$, where AS is the acoustic signal and M is the meaning. To recognize a spoken sentence, this generative model can be inverted by applying the Bayes' theorem to compute

$$\arg\max_M P(M|AS) = \arg\max_M P(AS|M)P(M).$$

In the case of HMMs, this computation can be performed relatively efficiently using the Viterbi dynamic programming algorithm. The probabilities in the model were computed from a set of training data.

The Harpy system, a successor that adopted the DRAGON approach, incorporated several ideas from HEARSAY-I and was able to surpass the program goals within the five-year period. By today's standards, the system was very limited. The training and test data came from a fixed set of five speakers and the set of possible utterances was quite narrow. Despite these limitations, the approach of applying ML (in the form of a complex probabilistic model) was general. It was a precursor to the development of probabilistic graphical models (also known as Bayesian Networks) that burst onto the AI scene in 1988 with the publication of Pearl's book *Probabilistic Reasoning in Intelligent Systems* (Pearl 1988).

The Speech Understanding Systems program is an excellent example of the way in which DARPA designed a program with ambitious performance targets in consultation with the research community. While the goals were precisely specified, the means of achieving them was left open. This resulted in remarkable innovation with two highly-influential AI system architectures developed.

EELD: Probabilistic Relational Models

The EELD program was created in 2002 in the wake of the 9/11 attacks with the goal of uncovering terrorist networks through the analysis of multiple diverse information sources (DARPA 2003). From a technical standpoint, the project involved identifying entities and relationships that make up a graph and then predicting additional relationships, or links, missing from the graph. EELD marked the start of the second ML thread.

Among the many research directions that were explored, the one with the greatest impact was an approach combining probabilistic modeling, relational logic, and ML to create PRMs (Getoor, Friedman, Koller, and Pfeffer 2001). Prior to EELD, probabilistic models in AI and ML were essentially the probabilistic versions of propositional logic. Such models cannot represent the concepts of objects and the relations between them. For this, we need to use relational structures provided by first-order logic. A long-standing goal in knowledge representation had been to find an approach to incorporating probabilities into first-order logic. PRMs took an important step in this direction by handling a subset of first-order logic sufficient to model relational databases.

Given a relational database and its associated entity-relationship model, the structure and probability distributions of a corresponding probabilistic relational model can be learned. Then, as new entities are added to the database, the learned model can predict the probability that specified relationships will hold. This is precisely the ability to predict links missing from a graph. For example, although EELD predated the creation of modern social networks such as Facebook, the tools that it developed have facilitated the creation of the field of computational social science that seeks to understand social structures based on evidence such as social network information.

PRMs and the EELD program have also had a huge impact in the development of more mature tools for extracting structured knowledge from the internet. For example, DARPA has funded research that aims to automatically populate a knowledge base by extracting information from news stories and other public documents on the web. Tom Mitchell's group at Carnegie Mellon University produced the Never Ending Language Learner, which builds a large knowledge base of entities and relationships this way (Mitchell et al. 2015). The National Institute of Standards and Technology regularly holds the Text Analysis Conference-Knowledge Base Population challenge for evaluating systems of this type. All of these rely on methods first developed under EELD.

Prior to EELD, ML and probabilistic modeling could only be applied to problems involving propositional logic (or feature vectors, in ML terminology). DARPA set an ambitious goal with the EELD program — motivated by a national security need — that produced dramatic innovation and technological improvements.

Integrated AI Systems with Multiple Learned Components

The third ML thread launched in 2003 with DARPA's Personal Assistant that Learns (PAL) program — the goal was to create an integrated AI system for the desktop knowledge worker. Two large teams were funded, one led by SRI that produced the CALO system (Cognitive Agent that Learns and Organizes) and the other led by Carnegie Mellon University (and later also SRI) that produced the RADAR system (Reflective Agents with Distributed Adaptive Reasoning). The PAL program addressed a wide range of capabilities including organizing email and documents; planning, scheduling, and dynamically rescheduling events; assisting the user in preparing documents (especially PowerPoint presentations); and assisting teams during meetings (for example, by transcribing the conversation, recording the action items, deadlines, and assigned parties, and so on.)

ML pervaded all aspects of CALO and RADAR. Supervised learning and clustering methods were developed for organizing email and documents and

for helping prepare PowerPoint presentations. To teach CALO to execute procedures, three different methods were developed that involved a mix of demonstrations and verbal explanations. A general scheduling algorithm learned the preferences of each user in a flexible way and was able to coordinate multiple meetings involving overlapping groups of people. The meeting understanding component included several forms of supervised learning.

One of the core challenges in building large systems such as CALO and RADAR is the integration of multiple, separately-learned components. As a simple example, CALO contained both an email classifier that predicted to which project an email message belongs and a document classifier that predicted to which project a document belongs. These classifiers needed to be different because email has different properties than documents. In particular, the email message contains the list of recipients, which is very useful for identifying the relevant project, while such information is often less obvious in a document; yet, we would like the two classifiers to learn from each other because the two tasks are very similar. This was achieved in CALO through a process of relational co-training that took advantage of the fact that many email messages have documents attached to them. It is highly likely that if an email message M1 belongs to Project K then its attached files F1 and F2 also belong to the same project. Hence, when the user labels an email message M1 with "Project K," this label can also be passed to the document classifier to provide training examples for F1 and F2. The process can then iterate. If the document classifier changes its prediction about some other document F3 that was attached to email message M2, the updated prediction can be passed to the email classifier to provide a (probabilistic) label for M2. This technique is a step toward achieving end-to-end learning for AI systems composed of multiple, separately-engineered ML components (Dietterich and Bao 2008).

The task learning components of CALO had the biggest impact. A reusable integrated task learning system was built that combined three of the task learning components into a single system. It was then integrated into the Army's Command Post of the Future (CPOF) to make it easy for CPOF users to teach CALO-CPOF routine procedures (e.g., for the morning daily briefing or for configuring the CPOF workspace to prepare standard reports).

The most visible spinout from CALO is Siri, the Apple voice assistant. Siri was a startup company (spun out from SRI) that included some of the CALO researchers as cofounders. The original vision was to allow vendors and end-users to teach Siri new procedures, but of course, the deployed product delivers a more limited, but more robust, capability. At the time of writing, Apple, Google, Microsoft, and Amazon all offer speech-based assistants.

The Personal Assistant that Learns program shows how DARPA can push forward applied research by sponsoring the development of integrated systems. As AI research advances, the systems integration challenges will become increasingly critical. There is significant merit in setting an AI system integration challenge every 10 years to test integration methods and identify AI capability gaps. DARPA is the only funding agency in the US Government that can fund such large efforts. Each such effort uncovers and highlights fundamental research questions as well as practical engineering issues. In CALO, the problem of end-to-end learning was one such issue. A second issue was the problem of requiring all components in a large system to work within a single shared ontology, a problem discussed in detail in the Knowledge Representation and Reasoning article in this issue.

NN Methods and DL

Although neural net research at DARPA has revived recently with the advent of DL successes, it actually has a long history at DARPA. This fourth ML thread began in 1987, soon after the promising NN learning algorithms of the Boltzmann Machine (Ackley, Hinton, and Sejnowski 1985) and back-propagation (Rumelhart, Hinton, and Williams 1986) were published. At that point, DARPA funded a study through the Massachusetts Institute of Technology Lincoln Labs on NNs. In an atmosphere of excitement about the promise of this technology, two years of funding was allocated for the Artificial Neural Network Technology exploratory program that started in 1988. Successes in this initial phase led to a full-scale program that continued for a total of eight years. Many of the methods that we take for granted in ML today began during this period. Evaluations of the technology for speech recognition, automatic target recognition, and sonar signal detection showed clear superiority. Very large-scale integration chips of both analog and digital hardware systems for learning and classification were developed (Alspector et al. 1991; Graf and Henderson 1990; Griffin et al. 1991). Although Intel developed their own analog Electronically Trainable Analog Neural Network chip (Intel 1990), the relentless progress in digital technology paced by Moore's Law superseded such specialized chips. Theoretical results led to SVMs as an alternative ML technology (Cortes and Vapnik 1995), which, unlike neural nets, had a clear mathematical foundation for performance.

After a bit of a downturn in neural net funding, the DL program commenced in 2009. At the time, the best example of learning in a deep system (more than two layers of learnable-weighted connections between neurons) was a deep belief net with a discriminative, bottom-up path from inputs to outputs and a generative, top-down path from output to input (Hinton, Osindero, and Teh 2006). The learning algorithm, similar to that of the Boltzmann Machine, was local to the neurons involved except for a global value signal, and used a form of Hebbian learning called contrastive divergence. The history of DL since then, mirroring its counterpart in the

Hardware for NL — DARPA

Because of the massive amount of computation that is required, there have been many attempts over the years to create specialized hardware for NN emulation and training. Neural models are highly parallel and relatively regular and consist primarily of matrix operations.

In the late 1980s and early 1990s, several commercial NN chips were developed. These include the CNAPS chip designed by my company, Adaptive Solutions, and two chips, an analog chip, Electronically Trainable Analog Neural Network, and a low precision digital chip, Ni1000, designed by Intel. In both of these commercial efforts as well as of a number of experimental technologies, DARPA played a key role investing and guiding the efforts.

We have now entered a new era in NN technology. Although algorithm improvements and the development of large data sets have been important, we should not underestimate the role Moore's Law has played. Processors today are thousands of times faster than those of the early 90s, and primary memory is thousands of times larger and faster.

Moore's Law has also led to the development of more specialized architectures such as the GPUs with almost 20 billion transistors and computing at 10 TeraFlops.

More recently, DARPA programs such as The Unconventional Processing of Signals for Intelligent Data Exploitation (probabilistic analog computing) and the Cortical Processor (biologically inspired ML) have been focused on the synergy of algorithms and hardware for AI.

A number of companies are now developing neural-net chips. Google has its Tensor Processor Unit and Intel has numerous chip efforts. Likewise, there are around 40 start-ups developing various kinds of digital and analog NN chips.

Moore's Law is now ending. NN algorithms will continue to tackle ever more complex applications. But without Moore's Law how can we maintain our current momentum? In some ways Moore's Law has inhibited architecture innovation: Why design a new architecture when I will get a 2× performance/price-increase by just moving to a new process? Many, including DARPA, believe that the end of Moore's Law may unleash even more creative chip design!

DARPA has responded with their Electronics Resurgence Initiative, one of whose objectives is to make it easier to build large, complex chips. Although NNs are not the specific target of the Electronics Resurgence Initiative, the initiative will create tools and other capabilities that will allow rapid exploration of the neural/ML architecture space. We are in for an exciting time!

– Dan Hammerstrom

1980s, has been mostly of purely discriminative (as opposed to generative) training using the nonlocal back-propagation algorithm. This has led to better than human performance in narrowly-defined tasks; however, the future of ML will likely depend on learning model-based, generative methods that can retain knowledge and extract new lessons from that knowledge and its context. A DL method called generative adversarial networks uses generative models to produce unparalleled levels of accuracy and is part of the standard DL toolkit. However, back-propagation is still the workhorse used to train these architectures.

DL over the last five years has met, or exceeded, human capabilities in object recognition, language translation, navigation, and other narrowly-defined tasks. Personal assistants use DL in speech recognition, synthesis, and language understanding and translation. Driverless cars use DL for scene understanding, navigation, and control. Photo services and social networks use DL for face recognition and identification. The results have propelled AI to the forefront of public consciousness and had large impacts on industry as well. Experts in AI command superstar salaries in large internet, automotive, financial, and other companies.

Before about 2010, NN methods had largely been confined to two layers of adjustable weights and had declined in popularity and somewhat in performance, relative to other ML statistical methods, especially SVMs. SVMs were attractive because they came with some mathematical guarantees of optimality given the input features (usually carefully hand-coded) to be used, and output classes to be learned. DL extended the number of layers able to be learned by a variety of new methods, by using massive datasets for training, and by the steadily increasing computing power of microchips, especially GPUs that can accelerate the types of computation needed. These improvements in algorithms, data, and hardware enabled learning better features than those from hand-coding. For example, in figure 3, a DL system learns layers of a visual hierarchy for training images of only well-posed faces.

The first layer of the hierarchy is composed mostly of edge detectors of various orientations. This is similar to features constructed by hand for image recognition. However, the higher layers capture more

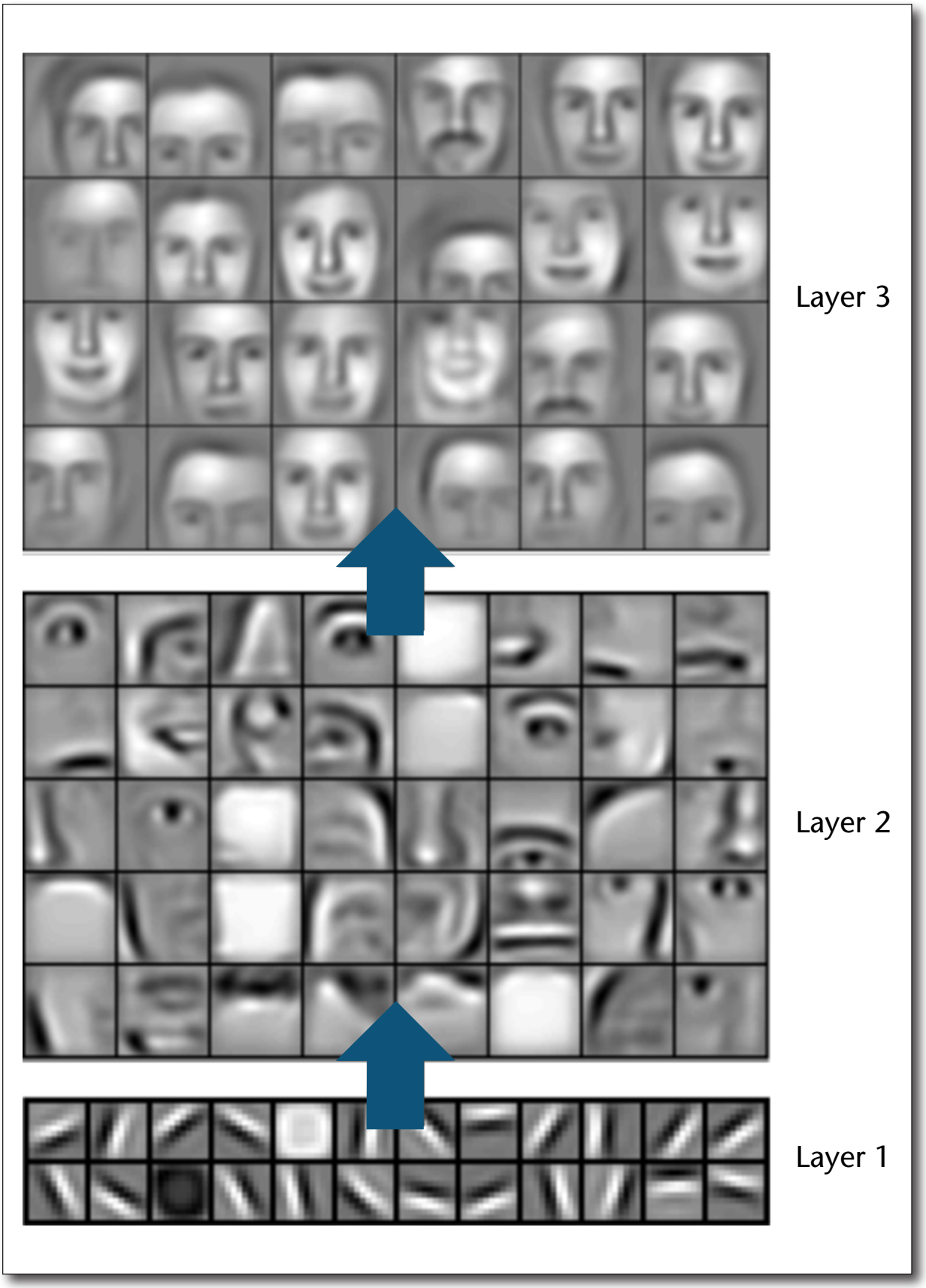

Figure 3. A Layered Hierarchy of Features Learned from Face Images.

Adapted with permission from Andrew Ng.[1]

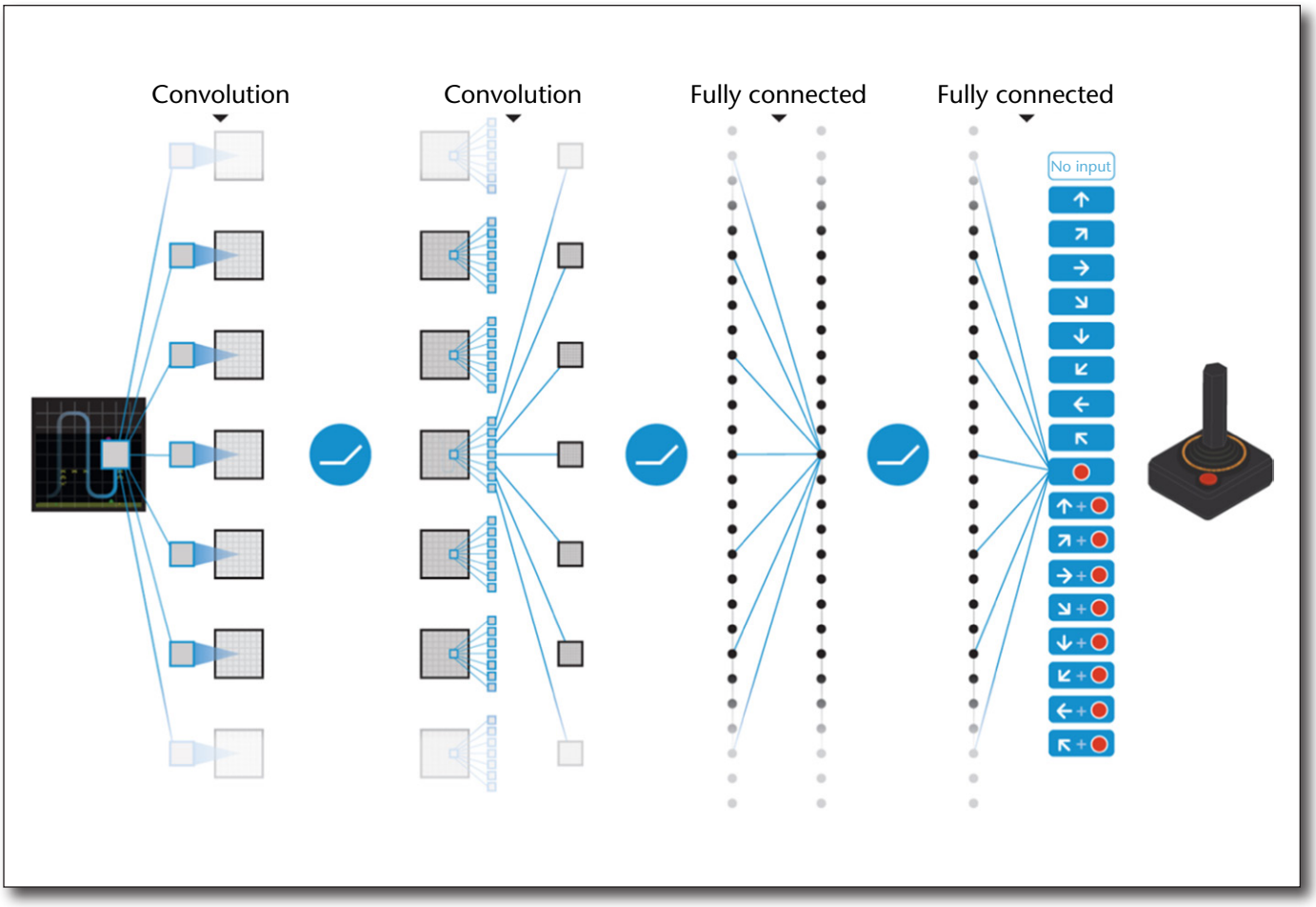

Figure 4. A Deep Network for Control of a Joystick in Atari Games.

Reprinted by permission from Springer Nature (Mnih et al. 2015).

face-specific features of the data presented for training, unlike hand-coded techniques, making face identification easy. These improved features from end-to-end learning led to spectacular improvements in accuracy and other measures of performance. DL systems match human performance in object identification in images and word identification in speech. These systems surpass human capabilities in games such as Atari and Go. In the Atari system pictured in figure 4 (Mnih et al. 2015), a DL network learns by trial-and-error, from playing millions of games, to control a joystick using reinforcement learning. The only inputs are the screen pixels and whether it won or reached its goal after a very long sequence of moves. This general method of learning control can be used to train autonomous systems such as cars or drones.

DARPA investment in the basic-research DL program bore fruit after the 2012 ImageNet result (Krizhevsky et al 2012), when DL methods began revolutionizing important applications. Current DARPA AI programs include Probabilistic Programming to Advance Machine Learning, Cortical Processor, Explainable AI, Lifelong Learning Machines, Radio Frequency Machine Learning Systems, Learning with Less Labels, Guaranteeing AI Robustness against Deception, and Machine Common Sense. DARPA will continue to fund $2B worth of AI programs over the next five years as part of its "AI Next" initiative.

More Methods of ML and Future Directions

ML builds a model of the environment based on data collected. As evidenced by the great variety of ML methods discussed, the field is not yet mature. It has swung from structured symbolic methods to data-driven neural methods, and this diversity will probably continue in the future. It is likely that more-semantic and symbolic methods will be incorporated into data-driven methods, as we have seen in recent years. Context and knowledge will certainly be added to ML, and DARPA programs such as Machine Common Sense and Lifelong

Learning Machines are exemplars of this trend. Specialized hardware for ML had been a feature of early neural net research and this trend is now reappearing; GPU processors are increasingly used for the fast multiply-adds needed in the inner loop of DL computations, ML hardware startups are being acquired by chip companies, and some companies, such as Intel, are developing their own specialized chips again (Davies and Srinivasa 2018). DARPA has funded some of this hardware work in programs such as Cortical Processor.

Now that AI and ML methods have become mainstream and commercially valuable, observers of progress in DL have questioned whether these systems will eventually lead to a superintelligence beyond human control. It is important to note that DL systems, while surpassing humans in the narrow domain for which they are trained, such as face recognition, are very far from human capabilities in general intelligence across the breadth and depth of human enterprise. Estimates of when DL systems might enable artificial general intelligence range from a decade to a century, and many people argue that the notion of artificial general intelligence is not even well-defined. Questions surrounding how to manage the development of artificial general intelligence are of high current interest. DARPA is addressing many aspects of AI safety with the programs for Assured Autonomy, Guaranteeing AI Robustness against Deception, Competence-Aware Machine Learning, and Science of Artificial Intelligence and Learning for Open-world Novelty.

ML and DL systems have revolutionized the way in which we interact with technology and, increasingly, the world. New applications are emerging. For example, recent progress in prosthetic control and cognitive science have benefited from ML technology (Savage 2019). As ML technologies improve, along with advances in hardware, the number and diversity of tasks to which they are applicable will grow. DARPA has played a key role in developing the technologies we have today, with the prime example being Siri, and will likely continue to advance AI technologies in the future.

Acknowledgments

We gratefully acknowledge the support of DARPA, including Dave Gunning, the program manager who helped organize this project along with Steve Cross and Scott Fouse. The DARPA program timeline was adapted from work with Yifty Eisenberg, deputy director of the Microsystems Technology Office at the time. There are countless others who have influenced and aided the authors over the decades, including many of those referenced in the article.

Note

1. Taken from Andrew Ng's talk, Unsupervised Feature Learning and Deep Learning www.csee.umbc.edu/courses/pub/www/courses/graduate/678/fall14/visionaudio.pdf

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A Learning Algorithm for Boltzmann Machines. *Cognitive Science* 9(1): 147–69. doi.org/10.1207/s15516709cog0901_7.

Alspector, J.; Allen, R. B.; Jayakumar, A.; Zeppenfeld, T.; and Meir, R. 1991. Relaxation Networks for Large Supervised Learning Problems. In *Advances in Neural Information Processing Systems, Vol 3*. Touretzky, D. S.; Moody, J.; and Lippmann, R., editors. San Mateo, CA: Morgan Kaufmann.

Baker, J. K. 1975. The DRAGON System—An Overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing (ASSP)* 23(1): 24–9. doi.org/10.1109/TASSP.1975.1162650.

Carnegie Mellon University Computer Science Speech Group. 1977. *Speech Understanding Systems: Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University*. Pittsburgh, PA: CMU Department of Computer Science. August. archive.org/details/DTIC_ADA049288.

Cortes, C., and Vapnik, V. 1995. Support-Vector Networks. *Machine Learning* 20(3): 273–97. doi.org/10.1007/BF00994018.

DARPA. 2003. *DARPA Fact File: A Compendium of DARPA Programs, April 2003*. Arlington, VA: DARPA. www.hsdl.org/?view&did=440746.

Davies, M., and Srinivasa, N. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38(1): 82–99. doi.org/10.1109/MM.2018.112130359.

Dietterich, T. G., and Bao, X. 2008. Integrating Multiple Learning Components Through Markov Logic. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (Association for the Advancement of Artificial Intelligence, AAAI 2008),* 622–7. Palo Alto, CA: AAAI Press.

Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; and Reddy, D. R. 1980. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys* 12(2): 213–53. doi.org/10.1145/356810.356816.

Getoor, L.; Friedman, N.; Koller, D.; and Pfeffer, A. 2001. Learning Probabilistic Relational Models. *In Relational Data Mining*. Berlin, Germany: Springer. doi.org/10.1007/978-3-662-04599-2_13.

Graf, H., and Henderson, D. 1990. A Reconfigurable CMOS Neural Network. In *Institute of Electrical and Electronics Engineers International Solid-State Circuits Conference (IEEE 1990) Digest of Technical Papers*. San Francisco, CA: IEEE. 144–5.

Griffin, M.; Tahara, G.; Knorpp, K.; Pinkham, R.; Riley, R.; Hammerstrom, D.; and Means, E. 1991. An 11-Million Transistor Digital Neural Network Execution Engine. In *Institute of Electrical and Electronics Engineers International Solid-State Circuits Conference (IEEE 1991) Digest of Technical Papers*. 180–181. Castine, ME: John H. Wuorinen.

Hinton, G. E.; Osindero, S.; and Teh, Y. W. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 18(7): 1527–54. doi.org/10.1162/neco.2006.18.7.1527.

Intel. 1990. *80170NW Electrically Trainable Analog Neural Network*. Intel Information Sheet E358. May. Santa Clara, CA: Intel Corporation.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)*. 1097–105. New York, NY: Association for Computing Machinery.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)* 86(11): 2278–324. doi.org/10.1109/5.726791.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; Krishnamurthy, J., et al. 2015. Never-Ending Learning. In *Proceedings of the Twenty-Ninth Association for the Advancement of Artificial Intelligence Conference (AAAI 2015)*. Palo Alto, CA: AAAI Press.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; and Graves, A. 2015. Human-Level Control Through Deep Reinforcement Learning. *Nature* 518(7540): 529–33. doi.org/10.1038/nature14236

Newell, A.; Barnett, J.; Forgie, J.; Green, C.; Klatt, C.; Licklider, J. C. R.; Munson, J.; Reddy, R.; and Woods, W. 1971. *Speech Understanding Systems: Final Report of a Study Group*. Amsterdam, Netherlands: Elsevier North-Holland.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning Internal Representations by Error Propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Rumelhart, D. E.; McClelland, J. L.; and the PDP Research Group, editors. 318–62. Cambridge, MA: The MIT Press.

Samuel, A. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 44: 1.2.

Savage, N. 2019. How AI and Neuroscience Drive Each Other Forward. *Nature* 571: S15–S17. 10.1038/d41586-019-02212-4.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. Presented at the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA. arxiv.org/pdf/1706.03762.pdf.

Josh Alspector received a PhD in Physics from the Massachusetts Institute of Technology. His physics career included the first direct measurement of the velocity of the neutrino. He spent much of his career at Bell Labs and Bellcore, where he developed the first NN learning microchip. While at University of Colorado – Colorado Springs, he founded a company acquired by AOL to create adaptive antispam filters in the first large-scale email service to use this technology. In 2009, at DARPA, he initiated the DL program. He is currently an adjunct Research Staff Member at the Institute for Defense Analyses, often consulting at DARPA.

Thomas Dietterich is professor emeritus of computer science at Oregon State University. He is one of the cofounders of the field of ML and has served the community in many roles including executive editor of {\it Machine Learning}, cofounder of the *Journal of Machine Learning Research*, founding president of the International Machine Learning Society, and president of the Association for the Advancement of Artificial Intelligence. He currently serves on the Steering Committee of DARPA's Information Science and Technology study group, and he is the lead moderator of the arXiv ML category.