

Chapter 2

Advancements in the Field

Abstract This chapter is divided into three sections, i.e., machine learning, neuroscience, and technology. This distribution corresponds to the main driving factors of the new AI revolution, meaning algorithms and data, knowledge of the brain structure, and greater computational power. The goal of the chapter is to give an overview of the state of art of these three blocks in order to understand what AI is going toward.

AI is moving at a stellar speed and is probably one of most complex and present sciences. The complexity here is not meant as a level of difficulty in understanding and innovating (although of course, this is quite high), but as the degree of interrelation with other fields apparently disconnected.

There are basically two schools of thought on how an AI should be properly built: the Connectionists start from the assumption that we should draw inspiration from the neural networks of the human brain, while the Symbolists prefer to move from banks of knowledge and fixed rules on how the world works. Given these two pillars, they think it is possible to build a system capable of reasoning and interpreting.

In addition, a strong dichotomy is naturally taking shape in terms of problem-solving strategy: you can solve a problem through a simpler algorithm, which though it increases its accuracy in time (iteration approach), or you can divide the problem into smaller and smaller blocks (parallel sequential decomposition approach).

Up to date, there is not a clear answer on what approach or school of thoughts works the best, and thus I find appropriate to briefly discuss major advancements in both pure machine learning techniques and neuroscience with an agnostic lens.

2.1 Machine Learning

Machine learning techniques can be roughly divided into supervised methods and unsupervised methods, with the main difference of whether the data are labelled (supervised learning) or not (unsupervised). A third class can be introduced when we talk about AI: reinforcement learning (RL). RL is a learning method for machines based on the simple idea of reward feedback: the machine indeed acts in a specific set of circumstances with the goal of maximizing the potential future (cumulative) reward. In other words, it is a trial-and-error intermediate method between supervised and unsupervised learning: the data labels are indeed assigned only after the action, and not for every training example (i.e., they are sparse and time-delayed). RL usually comes with two major problems, namely the credit assignment problem and the explore-exploit dilemma—plus a series of technical issues such as the curse of dimensionality, non-stationary environments, or partial observability of the problem. The former one concerns the fact that rewards are, by definition, delayed, and you might need a series of specific actions in order to achieve your goal. The problem is then to identify which of the preceding action was actually responsible for the final output (and to get the reward then), and if so to what degree. The latter problem is instead an optimal searching problem: the software has to map the environment as accurately as possible in order to figure out its reward structure. There is an optimal stop problem—a sort of satisficing indeed: to what extent the agent should keep exploring the space to look for better strategies, or start exploiting the ones it already knows (and knows that work)?

In addition to the present classification, machine learning algorithms can be classified based on the output they produce: classification algorithms; regressions; clustering methods; density estimation; and dimensionality reduction methods.

The new AI wave encouraged the development of innovative ground-breaking techniques, as well as it brought back to the top a quite old concept, i.e., the use of artificial neural networks (ANNs).

Artificial Neural Networks are a biologically-inspired approach that allows software to learn from observational data—in this sense sometimes is said they mimic the human brain. The first ANN named Threshold Logic Unit (TLU) was introduced in the Forties by McCulloch and Pitts (1943), but only forty years later Rumelhart et al. (1986) pushed the field forward designing the back-propagation training algorithm for feed-forward multi-layer perceptrons (MLPs).

The standard architecture for any ANNs is having a series of nodes arranged in an input layer, an output layer, and a variable number of hidden layers (that characterize the depth of the network). The inputs from each layer are multiplied by a certain connection weight and summed up, to be compared to a threshold level. The signal obtained through the summation is passed into a transfer function, to produce an output signal that is, in turn, passed as input into the following layer. The learning happens in fact in the multiple iterations of this process, and it is quantitatively computed by choosing the weighting factors that minimize the input-output mapping error given a certain training dataset.

ANNs do not require any prior knowledge to be implemented, but on the other side, they can still be fooled because of it. They are often also called *Deep Learning* (DL), especially for the case in which there are many layers that perform computational tasks. There exist many types of ANNs up to date, but the most known ones are Recurrent Neural Networks (RNNs); Convolutional Neural Networks (CNNs); and Biological Neural Networks (BNNs).

RNNs use the sequential information to make accurate predictions. In traditional ANNs, all the inputs are independent one from the other. RNNs perform instead a certain task for every element of the sequence, keeping a sort of *memory* of the previous computations. CNNs try instead to mirror the structure of the mammalian visual cortex and they have every layer working as detection filters for detecting specific patterns in the original data (and this is why they are really suitable for object recognition). Finally, BNNs are more a sub-field of ANNs rather than a specific application. The best example of this class is, in my opinion, the Hierarchical Temporal Memory (HTM) model developed by Hawkins and George of Numenta, Inc, which is a technology that captures both the structural and algorithmic properties of the neocortex.

In spite of the big hype around deep learning possibilities, all that glitters is not gold. DL is for sure a great step ahead toward the creation of an AGI, but it also presents limitations. The greatest one is the exceptional amount of data required to work properly, which represents the major barrier to a wider cross-sectional application. DL is also not easy to debug, and usually, problems are solved by feeding more and more data into the network, which creates a tighter big-data-dependency. Furthermore, DL is quite useful to bring to light hidden connections and correlations but is not informative at all regarding the causation (the why of things).

The data need imposes a considerable amount of time to train a network. In order to reduce this time, networks are often trained in parallel, either partitioning the model across different machines on different GPU cards (model parallelism) or reading different (random) buckets of data through the same model run on different machines to tune the parameters (data parallelism).

Because of the limitations just mentioned, a series of other tools have been developed over the years. **Particle Swarm Optimization (PSO)** is a computational method that iteratively improves candidate solution to optimize a certain problem (Kennedy and Eberhart 1995). The initial population of candidates (namely *dubbed particles*) is moved around in the search-space, and it has single particles that optimize their own position both locally and with respect to the entire search-space—creating then an optimized swarm. **Agent-based Computational Economics (ACE)** is an additional tool that lets agents interacting according to pre-specified rules into simulated environments (Arthur 1994). Starting from some initial condition imposed by the modeler, the dynamic systems evolves over time as interactions between agents occur (and as they learn from previous interactions).

Evolutionary Algorithms (EA) are instead a broad class of techniques that find solutions to optimization problems through concepts borrowed from natural

evolution, i.e., selection, mutations, inheritance, and crossover. An example of EA is the **Genetic Algorithm (GA)**, which is an adaptive search heuristic that attempts to mimic the natural selection process (Holland 1975). It is an evolutionary computing search optimization method that starts from a base population of candidate solutions and makes them evolving according to the “survival of the fittest” principle. **Genetic Programming (GP)** is an extension of GA (Koza 1992) because it basically applies a GA to a population of computer programs. It creates the chromosomes (i.e., the initial population of programs) made by a predefined set of functions and a set of terminals, and it randomly combines them into a tree-structure. In this context, the previous terminology acquires a slightly different connotation: reproduction means copying another computer model from existing population; cross-over means randomly recombining chosen parts of two computer programs, and mutation is a random replacement of chosen functional or terminal node. **Evolutionary Polynomial Regressions (EPRs)** are instead hybrid regressions that use GA to select the exponents of the polynomial, and a numerical regression (i.e., least square regression) to compute the actual coefficients (Giustolini and Savic 2006). A final interesting model is called **Evolutionary Intelligence (EI)** or **Evolutionary Computation (EC)**, and it has been recently developed by Sentient Technologies, LLC. It begins randomly generating trillions of candidate solutions (called genes) that by definition would probably perform poorly. They are then tested against training data, and a fitness score allowed the software to rank the best solutions (and eliminates the worst). Parts of the emerging candidates are then used to reassemble new populations, and the process restarts until a convergence is achieved.

To conclude this section, two additional approaches are worthy to be acknowledged. First, **Generative Models (GMs)** have been initially proposed by Shannon (1948), but recently brought back to the top by OpenAI, a non-profit AI research institute based in San Francisco (Salimans et al. 2016; Chen et al. 2016). This class of models is intuitively defined as those models we can randomly generate data for, assumed some hidden parameters. Once the data are feed, the system specifies a joint probability distribution and label sequences of data.

Second, Cao and Yang (2015) proposed a new method that converts the learning algorithm into a summation form, instead of proceeding directly from each training data point. It is called **Machine Unlearning (MU)**, and it allows the systems to “forget” unwanted data. They actually introduce an intermediate layer of summation between the algorithm and the training data points, such that they will not depend on each other anymore, but only on the summations themselves.

In this way, they learning process is much faster, and it can be updated incrementally without training again the model from scratch—which is quite time-intensive and costly. Hence, if some data and its lineage want to be eliminated, the system does not need to recompute the entire lineage anymore—a term coined by the two authors to indicate the entire data propagation network—but it can simply recompute a small number of summations.

2.2 Neuroscience Advancements

Along with the advancements in pure machine learning research, we have done many steps ahead toward a greater comprehension of the brain mechanisms. Although much has still to be understood, we have nowadays a slightly better overview of the brain processes, and this might help to foster the development of an AGI. It seems clear that try to fully mimic the human brain is not a feasible approach, and is not even the correct one. However, drawing inspiration from how the brain works is a completely different story, and the study of neuroscience could both stimulate the creation of new algorithms and architectures, as well as validate the use of current machine learning research toward a formation of an AGI.

More in detail, according to Numenta's researchers AI should be inspired to the human neocortex. Although a common theoretical cortical framework has not been fully accepted by the scientific community, according to Numenta a cortical theory should be able to explain: (i) how layers of neurons can learn sequences; (ii) the properties of SDRs; (iii) unsupervised learning mechanism with streaming temporal data flows; (iv) layer to layer connectivity; (v) how brain regions model the world and create behaviors; and finally, (vi) the hierarchy between different regions. These can be seen then as the six principles any biological or artificial intelligence should possess to be defined as such. Intuitively, it sounds a reasonable model, because the neocortex learns from sensory data, and thus it creates a sensory-motor model of the world. Unfortunately, we do not fully comprehend how the neocortex works yet, and this demands a machine intelligence be created flexible as much as robust at the same time.

In a more recent work, Hawkins and Ahmad (2016) turned their attention on a neuroscientific problem who is though crucial to the development of an AGI. They tried to explain how neurons integrate inputs from thousands of synapses, and their consequent large-scale network behavior. Since it is not clear why neurons have active dendrites, almost every ANNs created so far do not use artificial dendrites at all, and this would suggest that something is probably missing in our artificial structures. Their theory explains how networks of neurons work together, assumed all the many thousands of synapses presented in our brain. Given those excitatory neurons, they proposed a model for sequence memory that is a universal characteristic of the neocortical tissue, and that if correct would have a drastic impact on the way we design and implement artificial minds.

Rocki (2016) also highlighted few aspects specifically relevant for building a biologically inspired AI—specifically, the necessary components for creating a general-purpose learning algorithm. It is commonly assumed that humans do not learn in a supervised way, but they learn (unsupervised) to interpret the input from the environment, and they filter out as much data as possible without losing relevant information (Schmidhuber 2015). Somehow, the human brain applies a sort of Pareto's rule (or a Minimum Description Length rule otherwise) to information it gathers through sensory representations, and keeps and stores only the information that can explain the most of what is happening. According to Rocki, unsupervised

learning regularizes and compresses information making our brain a data compactor (Bengio et al. 2012; Hinton and Sejnowski 1999).

In addition to being unsupervised, Rocki hypothesizes that the architecture of a general-learning algorithm has to be compositional; sparse and distributed; objectiveless; and scalable. Human brain learns sequentially, starting from simpler patterns and breaking up more complex problems in terms of those simpler bricks it already understood—and this type of hierarchy and compositional learning is indeed well captured by deep learning. As already pointed out by Ahmad and Hawkins (2015), sparse distributed representations are essential, and they are much more noisy-resistant than their dense counterparts. However, there are much more peculiarities that make SDRs preferable: there are no region-specific algorithms in the brain, but the cortical columns act as independent feature detectors. Each column becomes active in response to a certain stimulus, and at the same time, it laterally inhibits other adjacent columns, forming thus sparse activity patterns. Since they are sparse, it is easier to reverse engineer a certain external signal and extract information from it (Candès et al. 2006). The property of being distributed helps instead in understanding the causes of patterns variations. SDRs also facilitates the process described above of filtering out useless information. They represent minimum entropy-codes (Barlow et al. 1989) that provide a generalized learning mechanism with simpler temporal dependencies.

The reason why the learning process should not have a clear stated objective is slightly controversial, but Rocki—and Stanley and Lehman (2015) before him—support this argument as the only way to achieve and form transferrable concepts. Moreover, Rocki states scalability as fundamental for a general-learning architecture. The brain is inherently a parallel machine, and every region has both computational and storing tasks (and this is why GPUs are much more efficient than CPUs in deep learning). This would suggest an AI to have a hierarchical structure that separates local learning (parallel) from higher-order connections (synapses updates), as well as a memory that can itself compute, in order to reduce the energy cost of data transfers.

Rocki eventually concludes with some further functional rather than structural ingredients for the formation of an AI, namely: compression; prediction; understanding; sensorimotor; spatiotemporal invariance; context update; and pattern completion. We discussed the importance of compression and sensorimotor before, and we can think of AGI as a general purpose compressor that forms stable representations of abstract concepts—although this point is controversial according to the *no free lunch theorem* (Wolpert and Macready 1997) that indirectly states that this algorithm cannot exist. We can also see prediction as of a weak form of spatiotemporal coherence of the world, and then we can argue learning to predict to be equivalent to understanding. Finally, we need to incorporate a continuous loop of bottom-up predictions and top-down contextualization to our learning process, and this contextual spatiotemporal concept would also allow for a disambiguation in the case of multiple (contrasting) predictions.

2.3 Technologies

As we explained before, the recent surge of AI and its rapidly becoming a dominant discipline are partially due to the exponential degree of technological progress we faced over the last few years. What it is interesting to point out though is that AI is deeply influencing and shaping the course of technology as well.

First of all, the Graphics Processing Units (GPUs) have been adapted from traditional graphical user interface applications to alternative parallel computing operations. NVIDIA is leading this flow and is pioneering the market with the CUDA platform and the recent introduction of Telsa P100 platform (the first GPU designed for hyperscale data center applications). On top of P100, they also created the first full server appliance platform (named DGX-1), which will bring deep learning to an entirely new level. Very recently, they also released the Titan X, which is the biggest GPU ever built (3584 CUDA cores).

In general, the most impressive developments we observed are related to chips, especially Neuromorphic Processing Units (NPU) ideated to emulate the human brain. Specific AI-chips have been created by major incumbents: IBM has released in 2016 the TrueNorth chip, which it is claimed to work very similarly to a mammalian brain. The chip is made of 5.4 billion transistors, and it is able to simulate up to 1 million neurons and 256 million neural connections. It is equipped with 4000 cores that have 256 inputs lines (the axons) and as much output lines (neurons), which send signals only when electrical charges achieve a determined threshold.

This structure is quite similar to the Neurogrid developed by Stanford, although the academic counterpart is made of 16 different chips instead of the single one proposed by the software colossus.

Google, on the other hand, announced the introduction design of an application-specific integrated circuit (ASIC) thought and tuned specifically for neural networks—the so-called Tensor Processing Unit (TPU). The TPU optimizes the performance per watt specifically for machine learning problems, and it both powers RankBrain (i.e., Google Search) and DeepMind (i.e., AlphaGO).

Intel is working on similar chips as well, i.e., the Xeon Phi chip series, and the latest release has been named Knights Landing (KNL). KNL has up to 72 cores, and instead of being a GPU, it can be a primary CPU that reduces the need to offload machine learning to co-processors.

Even Qualcomm has invested enormous resources in the Snapdragon 820, and eventually into the deep learning SDK Snapdragon Neural Processing Engine and their Zeroth Machine Intelligence Platform.

The cost for all those chips is huge (on the order of billions for R&D, and hundred thousand dollars as selling cost), and they are not viable for retail consumers yet but only thought for enterprise applications. The main exception to this major trend is the mass-scale commercial AI chip called Eyeriss, released earlier in 2016 by a group of researchers at MIT. This chip—made of 168 processing

engines—has been built on a smartphone’s power budget and thus is particularly energy-friendly, but it presents anyway computational limitations.

Even though this is a cost-intensive game, several startups and smaller companies are considerably contributing to the space: Numenta open-source NuPIC, a platform for intelligent computing, to analyze streaming data. Knowm, Inc. has brought memristor chips to the market, which is a device that can change its internal resistance based on electrical signals fed into it (and used as a non-volatile memory). KnuEdge (and its subsidiaries KnuPath) created LambdaFabric, which runs on a completely innovative architecture different not only from traditional GPUs but also from TPUs. Nervana Systems released an ASIC with a new high-capacity and high-speed memory technology called High Bandwidth Memory. Horizon Robotics is another company actively working in the space, as well as krtrl, which has produced a new low-cost dual-core ARM processor (FPGA, Wi-Fi, Bluetooth) named Snickerdoodle.

A final note has to be made in favor of Movidius, which introduced a completely new concept, i.e., an all-in-one USB for deep learning. Codenamed Fathom Neural Compute Stick, it contains a chip called Myriad 2, which has been thought in partnership with Google specifically to tackle down any advanced image recognition issue (but it has been used also to power drones and robots of a diverse kind).

References

- Ahmad, S., & Hawkins, J. (2015). *Properties of sparse distributed representations and their application to hierarchical temporal memory*. [arXiv:1503.07469](https://arxiv.org/abs/1503.07469)
- Arthur, B. W. (1994). Inductive reasoning and bounded rationality. *American Economic Review*, 84(2), 406–411.
- Barlow, H. B., Kaushal, T. P., & Mitchison, G. J. (1989). Finding minimum entropy codes. *Neural Computation*, 1(3), 412–423.
- Bengio, Y., Courville, A. C., & Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. CoRR. [arXiv:abs/1206.5538](https://arxiv.org/abs/1206.5538)
- Candès, E. J., Romberg, J. K., & Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8), 1207–1223.
- Cao, Y., & Yang, J. (2015). Towards making systems forget with machine unlearning. *IEEE Symposium on Security and Privacy, 2015*, 463–480.
- Chen, X., Duan, X., Houthoofd, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). *InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets*. [arXiv:1606.03657](https://arxiv.org/abs/1606.03657)
- Giustolisi, O., & Savic, D. A. (2006). A symbolic data-driven technique based on evolutionary polynomial regression. *Journal of Hydroinformatics*, 8(3), 207–222.
- Hawkins, J., & Ahmad, S. (2016). Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10.
- Hinton, G., & Sejnowski, T. (1999). *Unsupervised learning: Foundations of neural computation*. Cambridge: MIT Press.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Cambridge: MIT Press.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942–1948).

- Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge: MIT Press.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.
- Rocki, K. (2016). *Towards machine intelligence* (pp. 1–15). CoRR. [arXiv:abs/1603.08262](https://arxiv.org/abs/1603.08262)
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). *Improved techniques for training GANs*. [arXiv:1606.03498](https://arxiv.org/abs/1606.03498)
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.
- Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(379–423), 623–656.
- Stanley, K. O., & Lehman, J. (2015). *Why greatness cannot be planned—The myth of the objective*. Berlin: Springer International Publishing.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *Transactions on Evolutionary Computation*, 1(1), 67–82.

Artificial Intelligence and Exponential Technologies:
Business Models Evolution and New Investment
Opportunities

Corea, F.

2017, IX, 45 p. 15 illus. in color., Softcover

ISBN: 978-3-319-51549-6