# Contemporary Challenges in Artificial Intelligence

Louis Kirsch
http://louiskirsch.com

*University College London*

September 2018

Building systems that exhibit human or super-human intelligence has been a long-standing challenge in computer science [1]. While there has been a lot of excitement around the possibilities and future impact, a major issue has always been to actually define intelligence. It is obvious that such a definition is critical in order to measure our progress towards this goal and to devise strategies to reach it. The situation has changed dramatically with the introduction of universal artificial intelligence by Hutter [2]. Based on this theory Legg [3] has been able to devise a formal measurement of intelligence that fits with most existing treatments and definitions.

In this chapter, we summarize this theory and present the maximally intelligent agent AIXI [2]. We will see that AIXI is incomputable but might nevertheless be a guiding theory for future practical approximations. Based on this insight, we investigate how this affects deep learning and propose a critical set of challenges that should be addressed to practically implement intelligent agents.

Our contributions in this chapter are

- A summary of universal AI [2] and Shane Legg's measure for intelligence [3]
- A novel proposal of the most critical challenges that should be addressed in order to achieve intelligent agents

This thesis relies on fundamental concepts from deep learning [4, 5], probability theory and probabilistic learning [6, 7], computability theory [8], information theory [9, 10], and reinforcement learning [11, 12]. Please refer to the cited reviews for background knowledge.

## 1 Measuring Intelligence

In this section, we define what intelligence is and how we can measure it. We will begin with an intuitive informal definition and then develop a precise formal replacement. The definition is related to universal artificial intelligence and AIXI by Marcus Hutter [2] and was originally defined by Shane Legg [3]. For a more in-depth introduction to universal AI, please refer to our review [13].

Defining intelligence is a major challenge. Colloquially, intelligence is defined in many ways. In order to refine our understanding of the term and measure it, humanity has developed many intelligence tests (IQ tests) that can predict future academic or commercial performance very well [14, 15, 16, 17]. But these tests are limited in scope because they only work for humans and are prone to cultural and time-dependent biases. Additionally, we would like to have a measurement that can be applied to both machines and humans without modification. This makes finding a universal measurement a hard and long-standing challenge [3]. To design the most intelligent agent, we should define the term as precisely and general as possible.

Among many definitions of intelligence one can extract two very common aspects: Firstly, intelligence is seen as a property of an actor interacting with an external environment. Secondly, intelligence is related to this actor's ability to succeed, implying the existence of some kind of goal. Therefore, it is reasonable to state that the greater the capability of an actor to reach certain goals in an environment, the greater the individual's intelligence. Additionally, most contemporary definitions of intelligence focus on adaptation, learning, and experience. Shane Legg argues that this is an indicator that the true

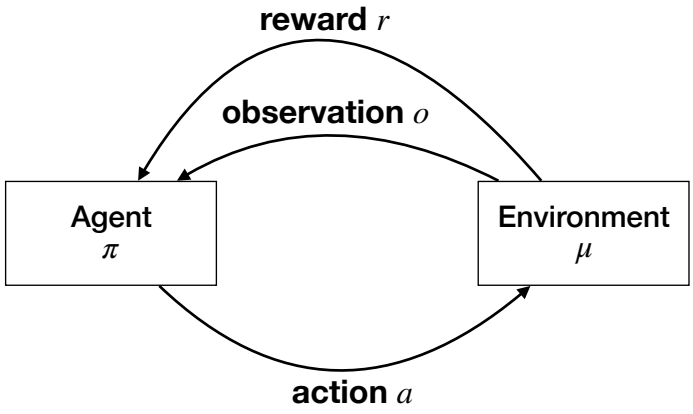


Figure 1: The standard reinforcement framework with agent $\pi$ and environment $\mu$.

environment in which these goals are pursued is not fully known and needs to be discovered first. The actor, therefore, needs to quickly learn and adapt in order to perform as well as possible in a wide range of environments. This leads us to a simple and informal working definition that intuitively covers all aspects of intelligence:

Intelligence measures an agent's ability to achieve goals in a wide range of environments.

Based on the theory of universal AI [2] we can formalize our working definition. We will need four ingredients, namely

1. A formal definition of the set of environments $\mathcal{E}$ and goals.
2. A definition of an agent $\pi$.
3. A definition of the ability of an agent $\pi$ in an environment $\mu \in \mathcal{E}$, namely $V_\mu^\pi$.
4. A measure of complexity of such an environment $K(\mu)$ with $\mu \in \mathcal{E}$.

We will use the standard reinforcement framework in Figure 1 based on an agent $\pi$ that interacts with environment $\mu \in \mathcal{E}$ by emitting actions $a \in A$ and receiving observations $o \in O$ and rewards $r \in R$. For improved readability, we write symbols in sequence to index all of them, e.g.

$$aor_{1:t} = (a_{1:t}, o_{1:t}, r_{1:t}) \tag{1}$$

The environment $\mu$ is a probability distribution over environment transitions based on the interaction history $(aor_{1:t-1}, a_t)$ given by

$$\mu(or_t | aor_{1:t-1}, a_t) \tag{2}$$

Thus, by defining the optimization problem to maximize the return, the sum of discounted (by $\gamma_t$) future rewards

$$R = \sum_t r_t \gamma_t \tag{3}$$

we implicitly define both the environment dynamics as well as the goal of the agent through $\mu$.

We further define our agent $\pi$ (also called the policy) to stochastically select actions

$$\pi(a_t | a_{1:t-1}, o_{1:t-1}, r_{1:t-1}) \tag{4}$$

Simplified agents may make Markov assumptions such that actions are chosen only based on the current observation $o_{t-1}$ or a compressed version of the interaction history.

Finally, we define the ability of an agent $\pi$ to maximize return in an environment $\mu$. This is just the value of the initial state (the interaction history only contains the initial observation $o_1$ and a null-reward and action) according to Bellman's equation

$$V_\mu^\pi(aor_{1:t-1}) = \sum_{aor_t} [\gamma_t r_t + V_\gamma^{\pi\mu}(aor_{1:t})] \pi(a_t | aor_{1:t-1}) \mu(or_t | aor_{1:t-1}, a_t) \tag{5}$$

An optimal agent $\pi$ in environment $\mu$ would then be defined as

$$\pi^\mu := \underset{\pi}{\operatorname{argmax}} V_\mu^\pi \tag{6}$$

It is important to note that this would require the agent $\pi$ to know the dynamics of the environment $\mu$, which is in general not the case. We will discuss this issue later.

Finally, we require a complexity measure for environments $\mu \in \mathcal{E}$. Intuitively, even less intelligent agents should be able to solve simple environments while more intelligent agents should still be able to solve these simple environments but additionally also perform well in more complex environments. For this, we will make use of the Kolmogorov complexity $K(\omega)$ of a sequence $\omega$ which is central to universal AI [2]. Intuitively, it is defined as the length of the shortest possible description of this sequence $\omega$. For instance, the binary string '1111111111' can be described by '10 times 1' while the string '0101101001' would require a longer description. Thus, the first string has a smaller Kolmogorov complexity compared to the second string. More formally, the description is a program $p$ running on a universal Turing Machine $U$. We write $U(p)$ for the output of $U$ when fed with program $p$. The length of program $p$ is given by $l(p)$. The Kolmogorov complexity of an infinite binary sequence $\omega \in \mathbb{B}^\infty$ is the length of the shortest program that produces $\omega$. It is given by

$$K(\omega) := \min_{p \in \mathbb{B}^*} \{l(p) : U(p) = \omega\} \tag{7}$$

Similarly, the Kolmogorov complexity over finite strings $x \in \mathbb{B}^*$ is the length of the shortest program that outputs $x$ and then halts.

The Kolmogorov complexity is thus a great way to measure complexity. Clearly, we must be able to express our environments $\mu \in \mathcal{E}$ as binary strings to apply the above definition. This is possible by defining an enumeration over environments and using the environment's index into this enumeration as the environment description. The precise formalities can be found in our review [13] or in Hutter [2] and Legg [3].

Finally, we can define a measure of intelligence $\Upsilon(\pi)$ for agent $\pi$. It is the sum of performance $V_\mu^\pi$ over all environments $\mu \in \mathcal{E}$ weighted by an environment complexity term $2^{-K(\mu)}$:

$$\Upsilon(\pi) := \sum_{\mu \in \mathcal{E}} 2^{-K(\mu)} V_\mu^\pi \tag{8}$$

The complexity term $2^{-K(\mu)}$ weights simpler environments with a smaller Kolmogorov complexity stronger than harder environments. The better any of these environments can be solved by an agent $\pi$, the larger is its intelligence measure $\Upsilon(\pi)$. We commonly use base 2 here because we choose the universal Turing machine $U$ to determine $K(\mu)$ to be prefix-free, meaning that no valid program for $U$ is a prefix of any other. This ensures that the weights for all valid programs $\mathcal{V}$ sum to 1.

$$\sum_{p \in \mathcal{V}} 2^{-l(p)} = 1 \tag{9}$$

Now that we are able to quantify intelligence, we can apply this measure to biological agents as well as artificial agents. Intelligence tests based on the Kolmogorov complexity have already been successfully applied to humans and positive correlation with IQ test scores has been established [18, 19, 3, 19]. Furthermore, we can define agents that directly optimize or maximize this criterion. The optimal agent that directly maximizes Equation 8 is AIXI [2]. We will discuss this agent in the next section.

# 2 Universal AI

In this section, we will construct an agent AIXI $\pi^\xi$ that maximizes the previously introduced intelligence measurement. Recall that we have introduced the Bellman optimal agent for a particular environment $\mu$ that maximizes the value function $V_\mu^\pi$ in Equation 6:

$$\pi^\mu := \underset{\pi}{\operatorname{argmax}} V_\mu^\pi \tag{10}$$

It is straightforward to see that we can maximize the Bellman equation to yield the action $a_t^{\pi^\mu}$ at time $t$ for agent $\pi^\mu$:

$$a_t^{\pi^\mu} := \arg\max_{a_t} \lim_{m\to\infty} \sum_{or_t} \max_{a_{t+1}} \sum_{or_{t+1}} \ldots \max_{a_m} \sum_{or_m} [\gamma_t r_t + \ldots + \gamma_m r_m] \mu(or_{t:m}|aor_{1:t-1}, a_{t:m}) \tag{11}$$

For environments with finite time horizon $T$ we choose $m = T$ and remove the limit from this equation.

The problem arises when the true environment $\mu$ is unknown. Recall that our intelligent agent should be able to deal with an arbitrary environment. Thus, we can not precode $\mu$ into the agent. In the following, we will construct the universal agent $\pi^\xi$ that uses Occam's razor to act in unknown environments. We will implement Occam's razor by making use of the previously introduced Kolmogorov complexity. We begin with defining a hypothesis space of possible environments

$$E := \{v_1, v_2, \ldots\} \tag{12}$$

To implement Occam's razor, we then define the universal prior probability for an environment $v \in E$

$$P_E(v) := 2^{-K(v)} \tag{13}$$

where $K(v)$ is the Kolmogorov complexity that computes the environment's index. The next step is to construct a prior over the agent's observations using a mixture of all environments

$$\xi(or_{t:n}|aor_{1:t}, a_{t:n}) := \sum_{v\in E} 2^{-K(v)} v(or_{t:n}|aor_{1:t}, a_{t:n}) \tag{14}$$

If we now replace the environment $\mu$ with $\xi$ we yield the universal AIXI agent $\pi^\xi$

$$a_t^{\pi^\xi} := \arg\max_{a_t} \lim_{m\to\infty} \sum_{or_t} \max_{a_{t+1}} \sum_{or_{t+1}} \ldots \max_{a_m} \sum_{or_m} [\gamma_t r_t + \ldots + \gamma_m r_m] \xi(or_{t:m}|aor_{1:t}a_{t:m}) \tag{15}$$

While it is impossible for $\pi^\xi$ to match the performance of $\pi^\mu$ for any environment $\mu \in \mathcal{E}$, we can show that some guarantees can be established. The fact that we can not match $\pi^\mu$ is due to irreversible interaction with the environment. To give a very simple example: Let's imagine an environment where an agent has to pick between two doors. One door leads to hell (very little reward), the other to heaven (plenty of rewards). Crucially, after having chosen one door, the agent can not return. But because the agent does not have any prior knowledge about what is behind the doors, there is no optimal behavior that guarantees that heaven is chosen. Conversely, if $\mu$ was known to the agent, it could infer which door leads to larger rewards. Nevertheless, for any subset of environments $\mathcal{S} \subseteq \mathcal{E}$ on which we define $\pi^\xi$ by replacing references to $\mathcal{E}$ with $\mathcal{S}$, balanced pareto optimality can be shown [2]. Balanced pareto optimality requires that any increase in performance in some environment due to switching to another agent than $\pi^\xi$ is compensated by an equal or greater decrease in performance in some other environment. In other words, we can not construct a universally better agent. Furthermore, for some environment classes, such as ergodic Markov Decision Processes (MDPs), it can be shown that $\pi^\xi$ converges to $\pi^\mu$. In fact, it can be shown that for all environments that permit this behavior $\pi^\xi$ converges [2].

Finally, we can relate $\pi^\xi$ to Shane Legg's definition of intelligence. By the linearity of expectation Equation 8 is just the expected future discounted return $V_\xi^\pi$ under the universal prior probability distribution $\xi$

$$\begin{aligned}
V_\xi^\pi &= V_\xi^\pi(\epsilon) \qquad\qquad \epsilon \text{ is the empty sequence} && (16)\\
&= \sum_{aor_1} [\gamma_t r_t + V_\xi^\pi(aor_1)] \xi(or_1)\pi(a_1|or_1) && (17)\\
&= \sum_{aor_1} [\gamma_t r_t + V_\xi^\pi(aor_1)] \sum_{\mu\in\mathcal{E}} 2^{-K(\mu)} \mu(or_1)\pi(a_1|or_1) && (18)\\
&= \sum_{\mu\in\mathcal{E}} 2^{-K(\mu)} \sum_{aor_1} [\gamma_t r_t + V_\xi^\pi(aor_1)] \mu(or_1)\pi(a_1|or_1) && (19)\\
&= \sum_{\mu\in\mathcal{E}} 2^{-K(\mu)} V_\mu^\pi && (20)
\end{aligned}$$

We have previously seen that AIXI maximizes $V_\xi^\pi$, therefore, by construction, the upper bound on universal intelligence is given by $\pi^\xi$

$$\overline{\Upsilon} := \max_\pi \Upsilon(\pi) = \Upsilon(\pi^\xi) \tag{21}$$

We have seen that AIXI [2] is balanced pareto-optimal and directly maximizes our criteria for intelligence according to Equation 8. Unfortunately, it is not computable and approximations have to be used in practice. In particular, we can not

- exactly compute the Kolmogorov complexity $K(v)$ for $v \in \mathcal{E}$
- sum over the non-finite set of environments $\sum_{v \in \mathcal{E}}$
- compute the value function $V_v^\pi$ over an infinite time horizon (assuming infinite episode length)

Even finite time horizons result in an exponentially deep search tree (width: actions, depth: time).

Instead of directly approximating the Kolmogorov complexity, the set of all environments, and the computation of the value function we might be able to devise learning agents that become efficient at a specific subset of environments, for instance, the kind of environments we as humans encounter every day. We propose that intelligent agents need to compress and store their knowledge about environments they are likely to encounter in order to efficiently act. Crucially, access to this stored knowledge needs to be cheap in terms of computational resources. Deep neural networks and associative lookups might be a good method to achieve that. We will investigate our proposal of the most critical challenges in the next section.

# 3 Critical Challenges for Intelligent Agents

In the previous section, we have investigated the AIXI agent, a maximally intelligent agent. Due to its inherent incomputability, we now propose a general direction to achieve efficient intelligent agents in the context of deep learning and the reinforcement learning framework. We will outline what we believe to be the key challenges to achieve this.

Deep learning allows searching the space of programs by learning functions efficiently using backpropagation or evolutionary strategies [4, 20, 5]. Given that neural networks implement Occam's razor through regularization and modeling assumptions, we could then approximate the Kolmogorov complexity by directly predicting the most likely continuation of the environment given the previous interaction history, i.e. build an Occam's razor based model of the environment. This idea is closely related to model based reinforcement learning [12] and the controller / model framework [4, 21].

Learning about the environment will require to compress the interaction history in a way that makes it useful for future predictions. This essentially means extracting useful relationships and concepts from the interaction history that make a good predictor. Essential to this is that our agent can store relevant previously extracted patterns for a sufficiently long amount of time. Due to an effect called catastrophic forgetting, for current deep learning architectures, this poses a challenge. We will look at this process of accumulating knowledge as life-long learning [22] in subsection 3.1.

Furthermore, learning increasing amounts over the lifetime of an agent will require much larger neural networks compared to today's largest implementations. One major problem is that the energy consumption does not scale with the number of firing neurons in an artificial neural network but with the size of the matrix-matrix-multiplications. We will look at this issue in the context of scalability in subsection 3.2.

In order to allow generalizing predictions beyond the previous interaction history, we have to look at generalization in the context of neural networks. This is often achieved through a Bayesian treatment of parameters or regularization. We will further investigate this in subsection 3.3.

When implementing an agent with neural networks we will have to make some modeling assumptions about its structure (e.g. probabilistic modeling) and optimization procedures. Often we also make architectural assumptions such as LSTMs [23] or convolutional neural networks [24]. One challenge will be to reduce the assumptions to a minimum. Even better, the agent should be self-referential, i.e. be

able to change its own architecture, modeling assumptions and optimization. We will briefly look at this in subsection 3.4.

Finally, we will investigate what kind of environments we would need to bootstrap an intelligent system in subsection 3.5.

## 3.1 Life-Long Learning

We have defined life-long learning as the process of accumulating knowledge. In the context of neural networks, we can refine this definition as the process of learning tasks $T_1, T_2, \ldots$ in sequence without considerably decreasing the performance in previous tasks and transferring knowledge from previous tasks to later tasks [22]. In the reinforcement learning context, we would like to learn from the interaction history to improve our future performance.

One fundamental problem with standard gradient descent to optimize deep neural networks is the phenomenon called catastrophic forgetting [25]. When updating the weights of a neural network with the gradients of a mini-batch on a new task $T_i$, performance on previous tasks is reduced. One obvious solution to this problem is to store all data from the previous tasks and include random samples in each mini-batch. This has two downsides: Firstly, storing all data from previous tasks requires a lot of storage capacity. For instance, recording all sensory experiences of a human being might be feasible today, but not necessarily efficient. Secondly, in order to prevent degradation of performance on any of the previous tasks $T_{1:i-1}$ the batch size would linearly grow with the number of tasks. Thus, different methods need to be developed. A popular recent strategy is to approximate the posterior on the weights $P(W|D_{1:i-1})$ and use it as a prior for the weights on the next task $T_i$ [26, 27, 28, 29, 30]. The major challenge with these approaches is to find a good posterior approximation that is still tractably computable. Furthermore, while catastrophic forgetting is considerably reduced, it is not yet comparable to the ability of humans to remember relevant knowledge.

A different strategy has been introduced with progressive neural networks [31]. The idea is to separate out each task into its own neural network whenever a new task should be learned. Then the parameters of previous tasks are frozen (no longer update through gradient descent) and lateral connections are added in order for any new tasks to leverage the functionality and representations of the previous tasks. This has been shown to improve performance (compared to learning from scratch) in several reinforcement learning tasks, such as Atari. Though, due to this strategy, the size of the entire architecture grows linearly with the number of tasks. Furthermore, existing learned functions are not updated in the light of new data, thus not allowing further compression.

An extensive review of lifelong learning can be found in Chen and Liu [22].

## 3.2 Scalability

While the human brain has around 150 trillion synapses [32] our largest neural networks only achieve a few billion parameters at most [33]. Furthermore, a single synapse is known to be more complex than a single parameter in an artificial neural network. Thus, our goal should be to scale our neural networks to sizes at least as big as the human brain. This also fits well with our previous argument that we need increasingly large neural networks to compress the entire interaction history an agent has collected.

The problem with training really large neural networks lies both in memory and compute. The number of parameters increases both memory and compute resources requirements linearly. In the case of neurons, the growth is even quadratic! This is very unlike the brain, where connectivity is very sparse. In the brain, computation scales linearly with the number of connections, and due to the sparse connectivity also linearly in the number of neurons, instead of quadratic with the number of neurons. Furthermore, energy is mostly required proportional to the number of times a neuron fires. Contemporary GPU architectures cannot leverage this kind of sparsity. Our recent report investigates how this kind of sparsity can be leveraged [34]. We have also proposed the Modular Networks [35], a training algorithm to discover modularity in tasks to enable conditional computation.

Other aspects of scalability involve efficient planning for the future. We have seen this challenge in the context of AIXI where we ended up with an exponential search tree. To overcome this problem new methods have been developed that distill discovered policies through explicit tree search into neural networks. Not only does this make future planning more efficient, the distillation process also allows the neural networks to generalize plans to similar future situations. These methods include learning fast and slow by Anthony, Tian, and Barber [36] and alpha go zero [37].

### 3.3 Generalization

Generalization is not only important for planning but has a direct connection to Occam's razor. Models that are better compressible, and thus simpler, correspond to lower Kolmogorov complexity and thus are more likely according to the universal prior probability as defined in section 2 [3].

It has been recently shown that deep learning with stochastic gradient descent already incorporates a kind of Occam's razor through implicit regularization [38, 39, 40]. Nevertheless, given large enough neural network architectures, maximum likelihood learning still has a tendency to overfit, even with stochastic gradient descent. Explicit regularization can be added in the form regularizing terms, priors in the Bayesian view, dropout [41], or structuring the probabilistic model. This often leads to much manual tuning due to no explicit guarantees. We are currently working on an information theoretic framework for regularization that allows to quantify and regulate the capacity of neural networks in a straight-forward fashion (to be published).

### 3.4 Self-referential Algorithms

When implementing an intelligent agent we inadvertently have to make modeling assumptions on both software and hardware. This includes probabilistic modeling, network architecture, optimization, and execution framework (such as TensorFlow [42]). To be precise, the fact that we use neural networks to build intelligent agents is already a strong inductive bias. That said, we can show that recurrent neural networks in the parametric limit are universal. While we can aim to reduce all assumptions to a minimum, we will probably not be able to get rid of all assumptions. Thus, it would be beneficial to allow our intelligent agent to be self-referential, i.e. being able to modify its own code and structure.

One possible venue to achieve is to create a meta optimization that optimizes these assumptions. This could be viewed as an analog to evolution for biological species. In the case of neural networks, we could introduce architecture optimization such as population-based training [43] that evolves hyperparameters and architecture throughout the training of the agent. Though, this methodology is not truly self-referential because it is not itself accessible by the agent. The main problem with neural networks being truly self-referential is that there is a mismatch between the continuous, differentiable workings of neural networks and their discrete structure. Optimizing discrete objects using backpropagation is much harder than differentiable functions.

If we leave the realm of deep learning, Schmidhuber has been able to construct a truly self-referential agent called the Gödel Machine [44]. It remains to be seen how these ideas can be transferred to deep learning. It is conceivable that neural network based agents will output actions that modify their own structure in an arbitrary fashion in order to optimize themselves. An analogy to this is how humans investigate their own workings in order to cure diseases, enhance their physical and cognitive ability and in the future manipulate their genetics.

### 3.5 Benchmarks

Finally, we will need environments to train and test our agents in. Ultimately we can then apply an approximation of the intelligence measure from Equation 8. In principle, we could generate environments with increasing Kolmogorov complexity [3]. Though, it is unlikely that the learned concepts will be very related to the real world or the specific task the agent is supposed to be useful. Therefore, we might have to explicitly build environments that have a certain structure that is useful to us.

In general, we will have the option to either expose our agents to the real world or construct a virtual one. Our agents in the real world could, for instance, take on the form of robots or be more abstract with

actuators and sensors much different from the ones humans have. Many robotic labs in industry and academia are already following this approach. Alternatively, training in virtual environments can be much faster and less error-prone relying on fewer human error corrections. If we want to transfer intelligence from simulated environments to the real world, the main challenge will be to design environments that have the same kind of complexity and compositionality that can be observed in the real world. Examples of initial simple virtual environments are computer games like Atari [45], Starcraft [46] and Dota 2 [47]. Benchmarks exist that measure abstract reasoning [48] or give access to computer applications like the browser and flash games using the same mouse and keyboard inputs a human uses [49].

# References


[1] Pamela McCorduck. *Machines who think: A personal inquiry into the history and prospects of artificial intelligence*. AK Peters/CRC Press, 2009.

[2] Markus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Berlin: Springer, 2005.

[3] Shane Legg. "Machine super intelligence". PhD thesis. Università della Svizzera italiana, 2008.

[4] Juergen Schmidhuber. "On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning Controllers and Recurrent Neural World Models". In: *arXiv preprint arXiv:1511.09249* (2015).

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] P Murphy Kevin. *Machine learning: a probabilistic perspective*. 2012.

[7] Christopher M Bishop. *Pattern recognition and machine learning*. 1. 2006.

[8] Zohar Manna. *Mathematical theory of computation*. Dover Publications, Incorporated, 2003.

[9] Claude Elwood Shannon. "A Mathematical Theory of Communication (Part {III})". In: *Bell System Technical Journal* XXVII (1948), pp. 623–656.

[10] Claude Elwood Shannon. "A Mathematical Theory of Communication (Parts {I} and {II})". In: *Bell System Technical Journal* XXVII (1948), pp. 379–423.

[11] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[12] Yuxi Li. "Deep Reinforcement Learning: An Overview". In: *arXiv preprint arXiv:1701.07274* (Jan. 2017).

[13] Louis Kirsch. *Theories of Intelligence (1/2): Universal AI*. 2018. URL: `http://louiskirsch.com/ai/universal-ai`.

[14] Alan S Kaufman. "Tests of intelligence". In: *Handbook of intelligence* (2000), pp. 445–476.

[15] William Stern. *Die psychologischen Methoden der Intelligenzprüfung und deren Anwendung an Schulkindern*. Vol. 5. JA Barth, 1912.

[16] Lewis M Terman and Maud A Merrill. *Stanford-Binet Intelligence Scale: Manual for the third revision, Form LM*. Tech. rep. 1960.

[17] Alfred Binet. *Les idées modernes sur les enfants*. Tech. rep. 1910.

[18] José Hernández-Orallo and Neus Minaya-Collado. "A formal definition of intelligence based on an intensional variant of algorithmic complexity". *Proceedings of International Symposium of EngineeringofIntelligentSystems (EIS'98)*. 1998, pp. 146–163.

[19] Jose Hernandez-Orallo. "Beyond the Turing test". In: *Journal of Logic, Language and Information* 9.4 (2000), pp. 447–466.

[20] Tim Salimans et al. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In: *arXiv preprint arXiv:1703.03864* (Mar. 2017).

[21] David Ha and Jürgen Schmidhuber. "World Models". In: *arXiv preprint arXiv: 1803.10122* (2018).

[22] Zhiyuan Chen and Bing Liu. "Lifelong Machine Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10.3 (2016), pp. 1–145.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[24] Yann LeCun and Yoshua Bengio. "Convolutional Networks for Images, Speech, and Time-Series". *The Handbook of Brain Theory and Neural Networks*. Ed. by M A Arbib. MIT Press, 1995.

[25] Michael McCloskey and Neal J. Cohen. "Catastrophic Inference in Connectionist Networks: The Sequential Learning Problem". In: *Psychology of Learning and Motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[26] Hippolyt Ritter, Aleksandar Botev, and David Barber. "Online Structured Laplace Approximations For Overcoming Catastrophic Forgetting". In: *arXiv preprint arXiv:1805.07810* (2018).
[27] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A Scalable Laplace Approxiation for Neural Networks". *International Conference on Machine Learning*. 2018.
[28] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017), p. 201611835.
[29] Cuong V Nguyen et al. "Variational Continual Learning". *International Conference on Learning Representations*. 2018.
[30] Jonathan Schwarz et al. "Progress & Compress: A scalable framework for continual learning". In: *arXiv preprint arXiv:1805.06370* (May 2018).
[31] Andrei A. Rusu et al. "Progressive Neural Networks". In: *arXiv preprint arXiv: 1606.04671* (2016).
[32] Bente Pakkenberg et al. "Aging and the human neocortex". In: *Experimental gerontology* 38.1-2 (2003), pp. 95–99.
[33] Adam Coates et al. "Deep learning with COTS HPC systems". *International Conference on Machine Learning*. 2013, pp. 1337–1345.
[34] Louis Kirsch. *Scaling Neural Networks Through Sparsity*. Tech. rep. 2018.
[35] Louis Kirsch, Julius Kunze, and David Barber. "Modular Networks: Learning to Decompose Neural Computation". In: *Neural Information Processing Systems* (2018).
[36] Thomas Anthony, Zheng Tian, and David Barber. "Thinking Fast and Slow with Deep Learning and Tree Search". *Advances in Neural Information Processing Systems*. 2017, pp. 5360–5370.
[37] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.
[38] Naftali Tishby and Noga Zaslavsky. "Deep Learning and the Information Bottleneck Principle". *Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.
[39] Ravid Shwartz-Ziv and Naftali Tishby. "Opening the Black Box of Deep Neural Networks via Information". In: *arXiv preprint arXiv:1703.00810* (2017).
[40] Stephan Mandt, Matthew D Hoffman, and David M Blei. *Stochastic Gradient Descent as Approximate Bayesian Inference*. Tech. rep. 2017, pp. 1–35.
[41] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.
[42] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *White Paper* (2015).
[43] Max Jaderberg et al. "Population Based Training of Neural Networks". In: *arXiv preprint arXiv:1711.09846* (Nov. 2017).
[44] Jürgen Schmidhuber. "Gödel machines: Self-referential universal problem solvers making provably optimal self-improvements". *In Artificial General Intelligence*. Citeseer. 2005.
[45] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: (2013).
[46] Oriol Vinyals et al. "Starcraft ii: A new challenge for reinforcement learning". In: *arXiv preprint arXiv:1708.04782* (2017).
[47] Trapit Bansal et al. "Emergent complexity via multi-agent competition". In: *arXiv preprint arXiv:1710.03748* (2017).
[48] Adam Santoro et al. "Measuring abstract reasoning in neural networks". *International Conference on Machine Learning*. 2018, pp. 4477–4486.
[49] Greg Brockman et al. *Openai universe*. 2016.