anylogic

ARTIFICIAL INTELLIGENCE AND SIMULATION IN BUSINESS

WHITE PAPER

CONTENTS

INTRODUCTION

Artificial intelligence, machine learning, and deep learning are common terms that you can often hear in everyday conversation. Familiar as they are, before looking at the value of their intersection with simulation, it is worth highlighting some important details.

> **The Encyclopedia Britannica simply defines artificial intelligence (AI) as 'the ability of a computer... to perform tasks commonly associated with intelligent beings'. This is a broad definition for a whole discipline of terms and methods that aim to make computers behave intelligently.**

Machine learning (ML) is a subset of artificial intelligence and a field of study directed at enabling computers to develop their own abilities. A system using ML will have specific algorithms for identifying patterns in data, patterns which can subsequently be used for making predictions and decisions.

For ML, there are three basic types of learning paradigms – supervised, unsupervised, and reinforcement:

SUPERVISED

In supervised learning, labeled training data is used – inputs are provided to an algorithm alongside known solutions. The labeled training data provides examples for algorithms to learn from. And, using the labeled pairs of input and output data, an algorithm will learn how to map any given input to an output.
After training, an algorithm can be used to classify inputs, such as whether email is email, or predict values, such as stock prices.

UNSUPERVISED

Algorithms in unsupervised learning are not given any solutions. Instead, with only unlabeled data for input, they must attempt to find hidden structures and patterns. This approach is mainly used for clustering, such as grouping people based on their purchase history, and association, where preferences and tendencies are identified. The results, for example, could be personalized supermarket discounts sent to your phone, or a tailored homepage for a streaming service.

REINFORCEMENT

In reinforcement learning, software agents are placed in an environment and seek to maximize the cumulative reward for their actions. The significant difference in this approach compared to the previous two is that no training data is needed, the training data comes from the cycle of observation, action, and reward. This type of ML can be applied in areas such as game playing and for control problems.

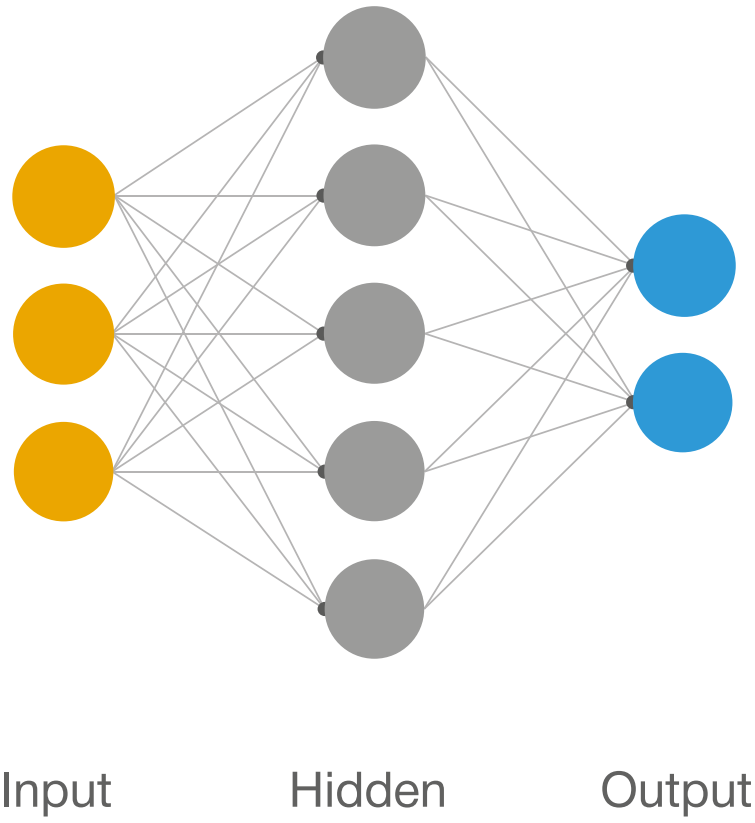
A limitation of ML paradigms is their need for handcrafted input and assumptions about what is important. This handcrafted, manual, element is difficult to scale and is reliant on human intelligence for guidance and informative feature extraction. There is, however, another way and, as an advancing subset of ML, deep learning is dealing with these limitations.

Deep learning is used to solve the same types of problems as the other three ML paradigms but differentiates itself by discovering important features for itself and learning directly from raw data, without the need for human intervention.

Deep learning uses artificial neural networks (ANN) – a system loosely based on how the brain works.

A simple ANN consists of a layer of input nodes which are connected through multiple hidden layers to a layer of output nodes. By using multiple layers, each intermediary layer can represent a different layer of abstraction and contribute to learning by producing a hierarchy of concepts. In areas where manually coding features is not practical, such as dictating which pixels are important to an object-identifying program, deep learning excels.

AN ARTIFICIAL NEURAL NETWORK



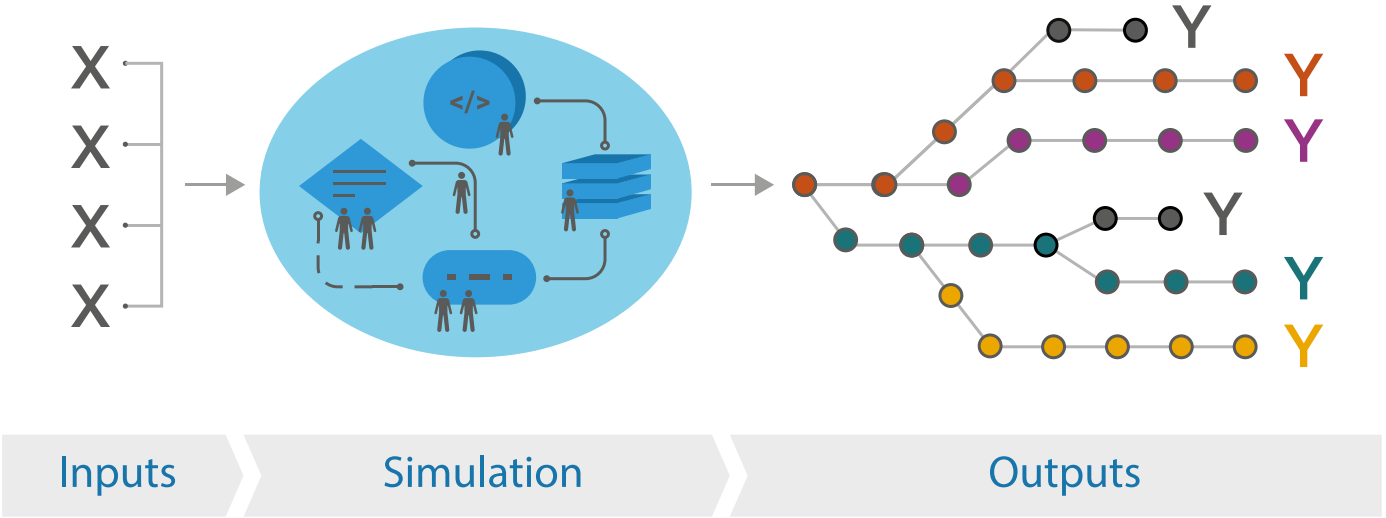
GENERAL PURPOSE SIMULATION TOOLS AND ANYLOGIC

01

GENERAL PURPOSE SIMULATION TOOLS AND ANYLOGIC

Classical modeling tools such as Microsoft Excel allow analytical solutions to be calculated using formulas and scripts. Going further, more sophisticated analytical tools are also available, such as linear or integer programming for optimization.

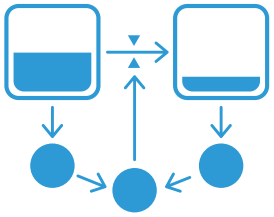
Where analytical tools fall short is their low ceiling of complexity before problems become too difficult to solve, particularly in dynamic, nonlinear, and stochastic systems. For these situations, simulation tools can be used to capture and define the behavior of a system.



The logic of a simulation model is a set of rules that tells the simulation engine how to get from the current state of the system to the next. As the model progresses in virtual time, various trajectories of the system can be observed, and output statistics collected.

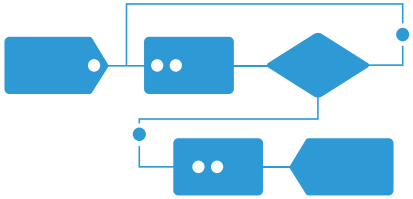
Experiments can be performed to explore possibilities by varying the inputs, running the same model multiple times to see the effect of randomness on the outputs, or a combination of both. This dynamic approach ultimately allows a more detailed analysis of a system, and for problems to be solved, in ways that other tools cannot provide.

There are three primary simulation methodologies for modeling a real-world system – system dynamics, discrete event, or agent-based:



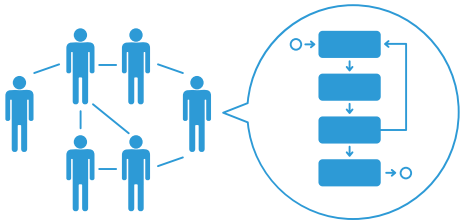
SYSTEM DYNAMICS

System dynamics operates at a high abstraction level (less detail), in a typically deterministic environment, and with a continuous modeling of time. These models use stock-and-flow diagrams to capture causal relationships, feedback loops, and non-linear behaviors of complex systems over time.



DISCRETE EVENT

Discrete event simulation models a system at a medium-low abstraction level and focuses on systems in which a sequence of operations or tasks need to be performed. This type of modelling is built using generic blocks that control the flow of entities in different contexts (e.g., patients in a hospital, cars in a manufacturing facility, packages in a warehouse).



AGENT-BASED

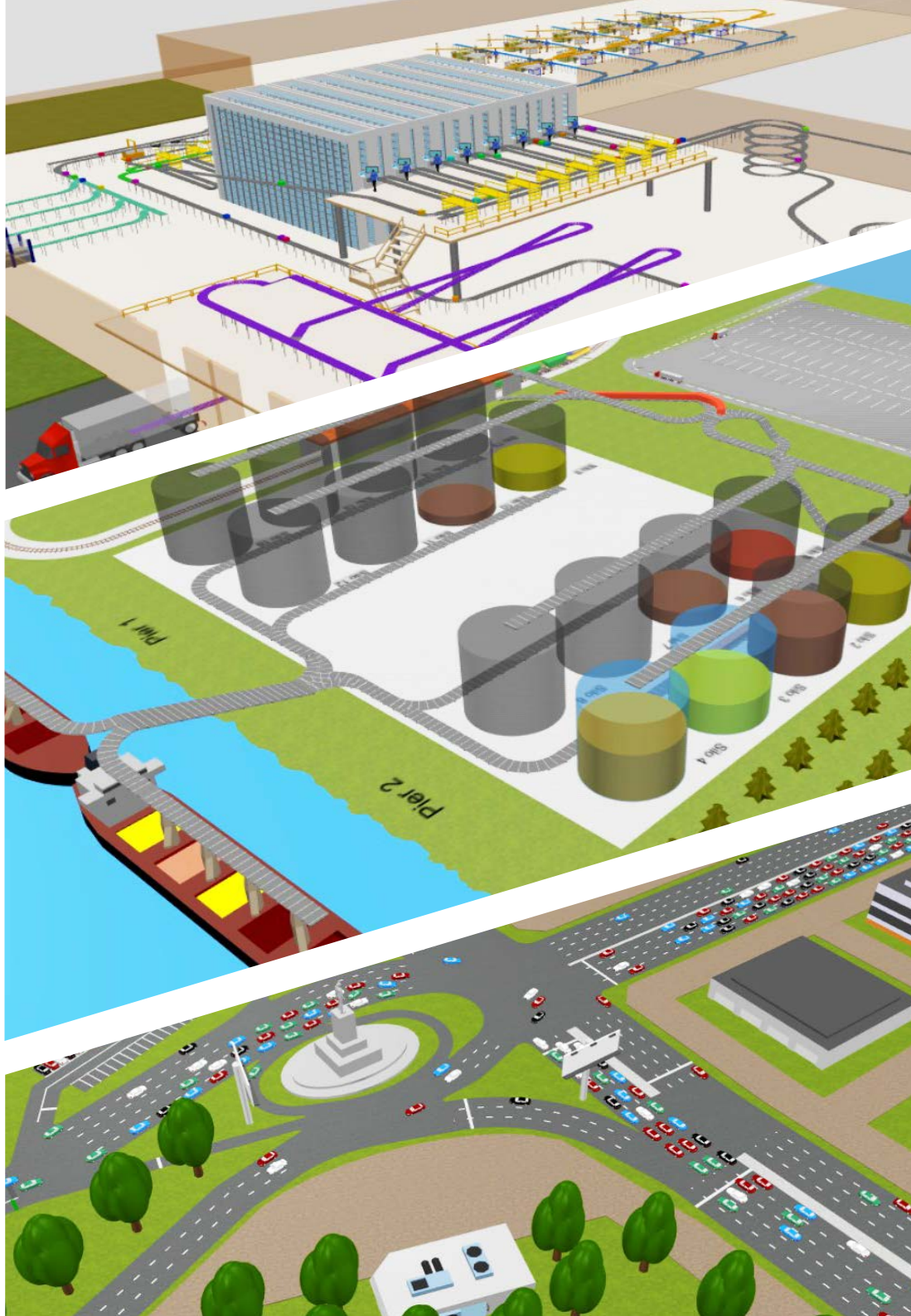
Agent-based modeling can work at any abstraction level and is made up of subsystems (local models) that may work independently or collaboratively. Behaviors are typically described through states and through communication between agents. Both discrete event and agent-based models tend to be stochastic but can also be deterministic.

How does all this relate to AnyLogic simulation software?

While most simulation tools specialize in one simulation methodology, AnyLogic supports them all.

AnyLogic supports using GIS maps, provides database connectivity, and has specific vertical libraries for industry areas such as material handling, fluid, pedestrian, road traffic, and rail – all these features give users a flexible environment for elegantly and efficiently modeling the structure, behavior, and rules of a real system.

Models are built using a drag-and-drop interface, greatly simplifying the creation process. The visual interface is superior to coding the system from scratch, helping to better structure models. Coding specifics are not lost, however. AnyLogic is written in Java and has a rich API, so the flexibility of programming is not abstracted away from the user. Any further customization to a model can be made by writing custom code or importing and making use of external Java libraries. No matter how complicated the system being modeled is, an elegant, flexible model can be built with no workarounds.



HOW ARE SIMULATION AND AI BEING USED TOGETHER?

02

HOW ARE SIMULATION AND AI BEING USED TOGETHER?

There exists immediate potential for AI and simulation to further develop their mutually beneficial relationship, specifically in three key areas:

Synthetic data – simulation generated data for training AI

Testbed – for examining AI behavior in new situations, not just on historical data

Learning environment – for RL when the real world is too costly, dangerous or unavailable

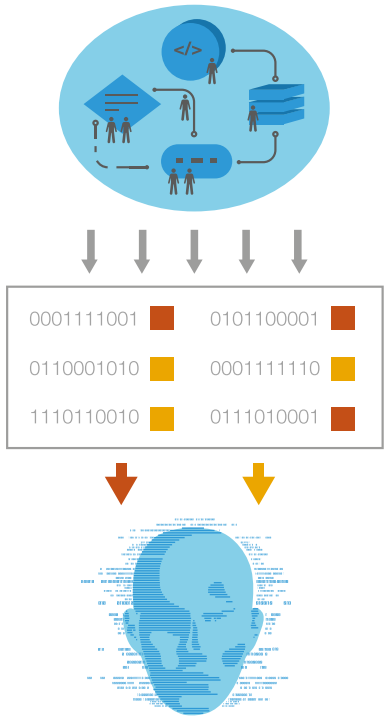
Before examining these areas, it's important to first understand how companies and organizations currently benefit from simulation.

In the process of building a mature simulation model to represent a real-life system, it is necessary to perform verification and validation. Verification is the iterative process of running a model and checking for any errors where the programmed logic does not match assumptions and specifications, then going back to fix those issues. Validation is the subsequent process that checks the model representativeness of the real or proposed system. Once a simulation model has been sufficiently verified and validated, the outputs of simulation and the real world parallel each other.

There are two implications to this that line up with the first two key areas of where AI and simulation intersect: the first is that simulation models can be used as a source to generate an unlimited amount of synthetic, clean, labeled data that is representative of a real system's outputs; second, that a simulation model can be used as a virtual environment to test the implications of incorporating AI into existing systems.

The third area comes from how useful information is extracted from a simulation model. A business or organization that has developed a simulation model will use it to run experiments and learn from the virtual representation of their real system in a risk-free way. These experiments – such as parameter variation, Monte Carlo, and optimization – facilitate the decision-making process and help analysis and strategy development. One shortcoming of these types of experiments is that they have limitations when extracting useful strategies from systems with high stochasticity or with many decision variables. These experiments cannot automatically suggest adaptive dynamic policies and rely on human experts to produce potential policies, algorithms, or heuristics for testing in simulation.

As simulation models are dynamic tools, they may reasonably be expected to propose dynamic solutions – this is the final key area where AI and simulation intersect: a simulation model can be used as an environment for a neural network based learning agent to train in. A trained policy is like a human-curated algorithm or heuristics that adaptively suggest optimal decisions for different situations over time. To test the learned policies before putting them into live production, they can be placed back into a simulation and rigorously examined.



1. SYNTHETIC DATA

Lack of data might be the biggest Achilles heel of machine learning – almost all machine learning algorithms heavily rely on input data being provided in both quantity and quality. Data collection is a significant task that needs many expertly designed, and often costly, processes to be put into place. Even when the process can be streamlined, there is no guarantee that sufficient data can be gathered due to the hard constraints of monitoring a real-world system (e.g., one day of data takes one day to produce). Moreover, machine learning algorithms, especially in supervised learning, need labeled data. The labeling process usually needs to be done by humans, adding to the time, cost, and difficulty of generating the necessary data.

To combat this, there are methods that can be used to create synthetic data from historical data, such as statistical algorithms. However, the main issue with this approach and similar methods, is that they do not capture the behavior of a system or the influence of interacting parts; they only generate new data by inference from historical data. Unlike statistical or mathematical methods, a sufficiently verified and validated simulation model can be used to generate unlimited amounts of relevant, clean, structured, and labeled synthetic input data.

Compared to mathematically generated data, simulation-based synthetic data is not entirely constrained to historical data. Simulation models rely on rules and inherently capture the causal relationships in a system. Since these models uncover the essentials of how the system works, their

outputs may also uncover counterfactuals and scenarios that have never been experienced. This ability gives simulation unique capabilities for generating synthetic data in terms of comprehensiveness, accuracy, and representativeness.

The data generated from a simulation model is almost entirely free of measurement bias. This is true because every aspect of a virtualized system is controlled, and all the recorded outputs are direct readings of values, not estimates or approximated measurements. This feature can be effectively utilized for testing and fine-tuning learning algorithms because there is absolute confidence that the input data, and its labels, are not influenced or diluted by irrelevant external factors or measurement errors. Moreover, unlike the real-world, simulations give you control over model conditions from start to finish. Modelers also have control over the random components of a system because they can set the seed of the random number generator (RNG). In this way, it is possible to tweak input parameters and analyze output data knowing that any differences resulted from an input change and not randomness.

Simulation models can generate an unlimited amount of clean, synthetic data for all parts of a system.

The model can track all levels of outputs and statistics, ranging from broad information – such as the yearly profit of a company – down to the most minute detail – such as a breakdown of the minutes worked by a specific worker. This level of precision allows the data output by a model to have a similar level of precision applied to the labels. Since each data point originates from a specific, known part of a model, both the individual data points and the results are accurately labeled.

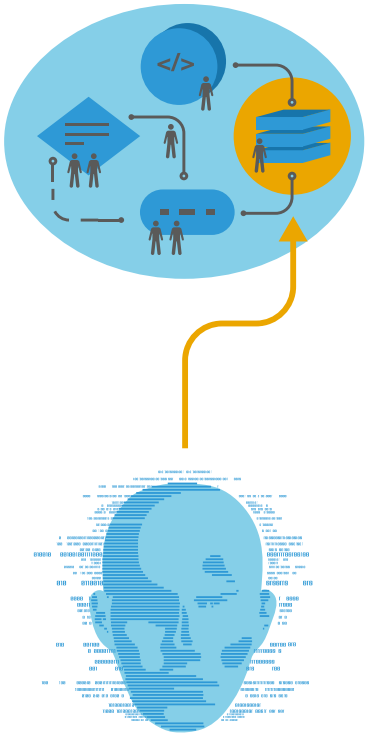
2. TESTBED

As artificial intelligence (AI) and machine learning (ML) solutions become increasingly mainstream, more of these solutions are deployed and integrated with existing systems. In many cases, data is first gathered from real world systems and then fed into ML algorithms for training. Datasets are partitioned, leaving one part for training on and the other for testing, allowing the performance of the algorithm to be evaluated. If the performance of a trained model is satisfactory, it can be deployed and incorporated into a live system.

The objective of adding AI components to a system is to improve the system's overall performance and not just that of the components being substituted by AI. Consequently, testing a trained model on its own does not verify that the performance of the modified system is sufficiently improved.

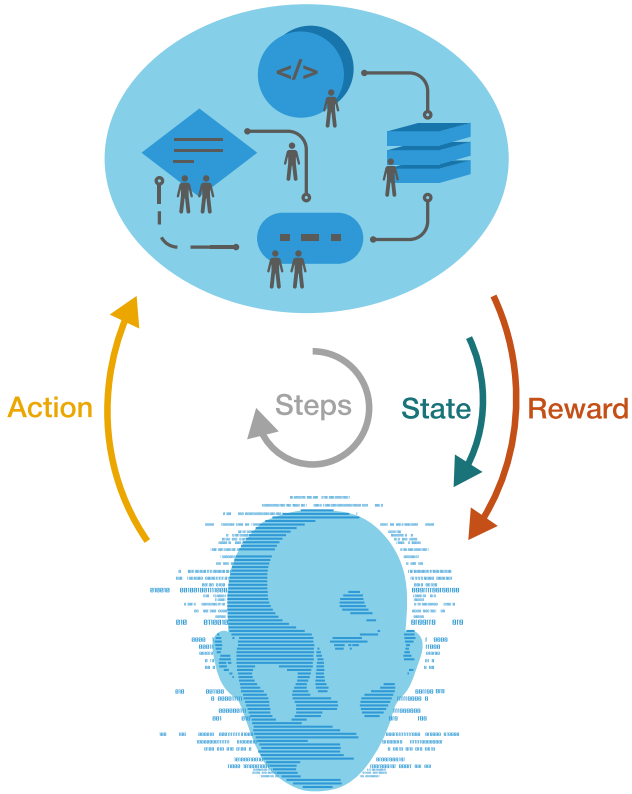
One of the main advantages of simulation is the ability to test changes to a system in a virtual, risk-free environment; this advantage can also be applied when testing the effectiveness of an AI component on the overall performance of a complex system.

For example, imagine a mortgage approval process in a bank. It involves several consecutive steps such as pre-qualification, application, appraisal, credit report, underwriting and closing. To streamline the process, the bank has trained and tested a ML model that automates some part of the pre-qualification process by automatically detecting whether an applicant is considered a high-risk applicant based on their personal information. The policy performs better than the current system, but it is uncertain whether



implementing the new pre-qualification ML model will benefit overall closing rates. This kind of uncertainty means simulation is an ideal way to find the answer. In this specific case, a simulated system incorporating the ML model might uncover a bottleneck which has now moved from the pre-qualification stage to the appraisal stage. In addition, when in the designing phase of any new system that will have AI components, alternative designs can be simulated to select the option with the best overall performance.

Simulation can also be used to test polices that were trained in a virtualized learning environment, as the result of reinforcement learning schemes – a concept further described in the next section. For these cases, the policies learned in a simulated environment can be put back to into a simulation for performance testing.



3. LEARNING ENVIRONMENT

Reinforcement learning is a machine learning paradigm that has shown great potential in solving problems that were previously deemed unsolvable for machines.

RL borrows many ideas from how intelligent beings, like humans, learn from their environment, specifically how experience is gained through trial and error. The only difference is that a computer program, or learning agent, does the learning.

The interesting part about the RL process is that no direct transfer of knowledge to the learning agent takes place; it is only through a reward system that the agent learns to understand what actions in any given situation are most beneficial and lead toward a learning objective. According to Richard S. Sutton[1] (2018), "Reinforcement learning is learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them".

For a learning agent to learn, it needs to interact with an environment or a playground. Although the learning agent itself is a computer program, the environment can either be physical and in the real world, or a virtual representation of it. For example, to learn how to fly a drone, a learning agent can take control of a physical drone and fly it, or it could learn in a computer simulation. Although more realistic and comprehensive, the real-world environment has several problems. Many drones will be needed, as the first few batches will likely be destroyed, and the

1 "Reinforcement Learning: An Introduction" Richard S. Sutton, Andrew G. Barto, 2nd Edition, ch. 1, Nov 13, 2018

implementation costs will be high.

Additional costs from learning in the real world may be incurred through injury to people or damage to property. Legal restrictions can also limit testing and impose further constraints.

In contrast, simulated environments are controllable, reproduceable, and the direct results of slight changes to situations are visible.

Simulations can also run faster than real-time and in parallel, allowing quick and concurrent accumulation of knowledge.

Many current simulated environments used for RL algorithms are either research-oriented toy environments or hyper-specific custom environments made for a single use case. They are often developed from scratch for substantial cost and effort.

The creation of these custom environments is mainly due to an assumption that general-purpose simulation tools are not flexible or comprehensive enough. And this is not far from the truth, since almost all available general-purpose simulation tools are intentionally simplified applications and and restricted as a result. In contrast, AnyLogic is developed with scalability, interoperability, connectivity, and flexibility in mind.

AnyLogic is a multi-scale, multi-method simulation platform. It can model the entire business lifecycle and is the natural choice as the learning environment for real-world use-cases. Furthermore, AnyLogic Cloud provides scalable computational power and the ability to abstract away from a model's implementation language. It also exposes a rich API for

connecting Python, Java, or any other language that can interface via REST. This feature specifically is important for the integration of simulation models into machine learning or deep learning frameworks and their development environments.

DEEP REINFORCEMENT LEARNING IN ACTION

DEEP REINFORCEMENT LEARNING IN ACTION

There are many examples in the past decade that showcase the potential of deep reinforcement learning. A significant moment in the field was the 2016 defeat of Go world champion, Lee Sedol, by DeepMind's AlphaGo.

Although the rules are simple, the game complexity of Go make it formidably difficult and it was seen as the biggest challenge in classical games for artificial intelligence to master. It is estimated that there are more valid ways to play the game to conclusion than there are atoms in the observable universe. Of the five games played, AlphaGo won all but the fourth game.

> "AlphaGo showed anomalies and moves from a broader perspective which professional Go players described as looking like mistakes at the first sight but an intentional strategy in hindsight

Not content with its board game success, Google's self-driving research led to Waymo starting a self-driving taxi service in Phoenix, Arizona, in April 2017. Waymo, as of 2025, operates commercial robotaxi services in seven US states. It offers over 250,000 paid rides per week, totaling over 1 million miles monthly.

Another popular example comes from Dota 2 – a multiplayer online battle arena game where two teams of five players defend their base while trying to destroy the opposing team's. The game demands quick thinking, teamwork, and long-term strategy, making it a tough challenge even for professional players.

Back in 2019, OpenAI's Five – a system of neural networks trained entirely through reinforcement learning – stunned the esports world by defeating the reigning international champions in a best-of-three series. In public matches that followed, it won over 95% of 42,000+ games played against human challengers.

The training for the AI was entirely based on self-play. The system started with no experience and was only provided with an incentive to win. It then played games against itself, experiencing 45,000 years of play over a period of 10 real months – averaging 250 years of simulated gameplay per day.

Fast forward to today, OpenAI is more widely recognized for projects like ChatGPT, which has transformed how millions of people interact with AI in everyday life. Yet the same underlying reinforcement learning techniques that powered OpenAI Five's gaming success continue to shape advances in AI systems—whether they're competing in virtual arenas or holding natural conversations with humans.

EXAMPLE MODEL: TRAFFIC LIGHT TIMING – AI VS OPTIMIZATION

04

EXAMPLE MODEL: TRAFFIC LIGHT TIMING – AI VS OPTIMIZATION

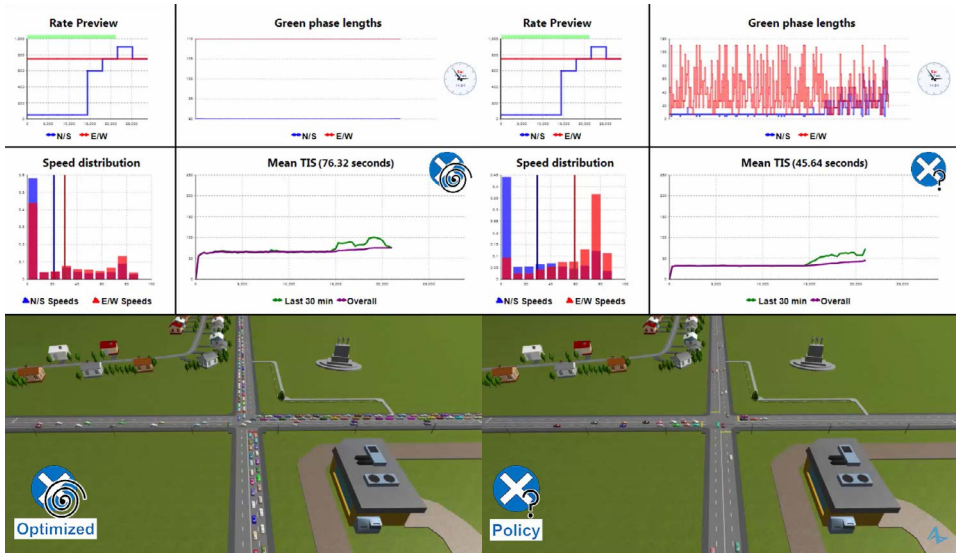
To showcase the capabilities of a powerful general-purpose simulation tool as the training environment, AnyLogic, in partnership with Skymind, developed a simple but illustrative example model based on the simulation of a traffic light-controlled intersection. The objective of this was to train an artificial neural network to be able to optimally control a traffic light. It did this by learning a policy which best controlled the traffic light based on the current status of the traffic.

The model consisted of a very straight forward intersection: cars start at beginning of one of four roads and have their destination set to the end of the opposing road. The single traffic light in the model controls the right of way in each direction. The time that each car spends in the model is affected by the traffic level and how efficiently the traffic light's signal is controlled. The logic of the traffic and intersection was modeled using standard flowchart blocks within the AnyLogic Road Traffic Library.

Two Schedule elements were used to determine the arrival rates for each axis of traffic (one for the north/south, one for the east/west). The simulation starts at 8:00 AM and finishes 8 hours later at 4:00 PM. The cars traveling east/west had a consistently high rate throughout the simulation (500 cars per hour). However, the north/south direction experienced light traffic in the morning (50 cars per hour) and heavier traffic that peaked between 2:00 to 3:00 PM (900 cars per hour).

For this model, the key performance indicator (KPI) was defined as the average time a car spends in the system; this was tracked through a Dataset object which was added to whenever cars left the system. As this model made use of reinforcement learning, the observation and actions taken by the learning agent, and the reward structure also needed to be defined.

Every ten seconds, an observation was sent to the learning agent that consisted of ten values: the first eight described the total amount of time spent in the model for all cars within 150 meters of the intersection for each of the eight lanes; the ninth element was also the total time spent in the model, but for the cars in the intersection; the tenth element was the index of the current traffic light phase. From that information, the learning agent then took one of two actions: do nothing and keep the current phase of the traffic light or move to the next phase.



The model was then advanced another ten seconds. The reward given to the learning agent is a function that was adopted from the work by Arel et al.[1] (2010) and is a numeric value between -1 to 1. For the observations, the sum is taken for the cars in the forward lanes (moving towards the center) and for the cars in the intersection. Initially, the reward is set as the difference between these two values. It is then normalized by dividing the maximum of the values.

The learning agent was trained on 100 episodes using the cyclical process described above and then added back into the model for testing. To give a comparative idea of performance, an Optimization Experiment was also run on the model and output what phase length it determined was best for each axis. When using the optimized values, the mean time in system value was 1.54 minutes. However, using the AI policy in the model, achieved a mean time in system of 1.03 minutes – a 33% improvement!

The power of replacing static values with a dynamic ANN allows dynamic adaptation to any given situation. Even for situations never experienced before, with knowledge from similar situations, a policy can outperform values determined by black-box optimization experiments.

The model is simple for illustrative purposes and, as a result, the policy it learned can be outperformed by a human. However,

the beauty of the policy is that no human assistant was involved in the learning process; the AI learned a meaningful policy on its own, based on its interaction with the simulation model.

If a more realistic case were setup and trained on, the learning agent would start to show its advantages over human curated algorithms.

1 I. Arel, C. Liu, T. Urbanik and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," in IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128-135, June 2010.

CASE STUDY: CORE MOVEMENT RESOLUTION – AN INDUSTRIAL PROBLEM RESOLVED BY AI AND SIMULATION

05

CASE STUDY: CORE MOVEMENT RESOLUTION – AN INDUSTRIAL PROBLEM RESOLVED BY AI AND SIMULATION

Engineering Ingegneria Informatica (EII) developed an automated decision system using reinforcement learning and AnyLogic for Lagor S.R.L. The decision system was applied to resolve the problem of movement planning on a factory shop floor.

Lagor's factory produces ferromagnetic cores used for electrical transformers. The production line consists of several automated workstations, each connected by heavy-duty conveyor systems. Each core is highly customized to meet the specific requirements of each transformer, resulting in production cycles and movements within the production line that are neither unique nor able to be automated.

Since at any given time several cores can be under process, a line manager needs to consider each individual production cycle and plan movements in advance. Despite all efforts, bottlenecks and line blockages occur and result in costly delays.

One possible solution for this problem is to use mathematical optimization. However, the scope of the problem is too large for daily execution of a search algorithm. Indeed, this is a sequential decision-making problem for which reinforcement learning is the best approach. All the required heavy computations only need performing during the training stage and then solutions can be obtained repetitively with relatively no time-cost.

Using a simulation solution that enabled the client to evaluate their production plan, EII developed a reinforcement learning-based agent that could automatically generate the best movement sequence.

Being highly customizable, AnyLogic allowed the utilization of Java-based machine learning packages and provided a robust simulation environment in which to train the learning agent.

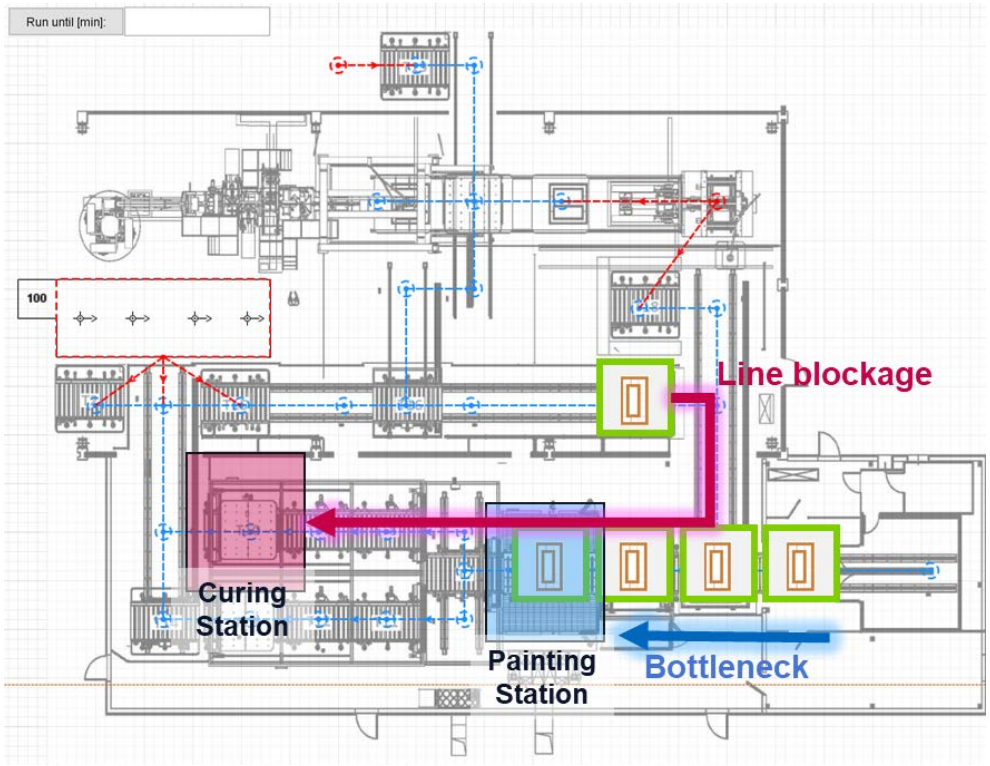
A FERROMAGNETIC CORE AT THE END OF THE MANUFACTURING PROCESS

The adopted learning algorithm was a double deep Q network (DDQN), implemented by Skymind in their DL4J package. Since the problem consisted of different tasks, each one was addressed separately, and the combination of these solutions was used to address the main problem. The result was a policy that replaced the complicated heuristics, which were difficult to maintain and debug.

It was observed that the robustness and flexibility of the AnyLogic simulation platform was critical for the implementation. The learned policy successfully proved that the application of advanced machine learning techniques with simulation modelling helps solve complicated tasks when other solutions are not readily available.

For more information about this project, please see Ell's [AnyLogic Conference presentation]

AN EXAMPLE OF A BOTTLENECK AND BLOCKAGE DURING THE MANUFACTURING OF CORES WITH DIFFERENT PRODUCTION CYCLES



CONCLUSION

We are now at a point where AI is expanding beyond research environments and into the real world. To facilitate this, the power of simulation is increasingly being used to leverage the promises of AI. Companies working with either technology stand to benefit from their combination.

With the origins of machine learning starting in the 1950's, it is not a new topic. However, it wasn't until recently, and the availability of the necessary computing power, that the theorized concepts could start to be applied.

Simulation practitioners will soon notice a dramatic increase in the demand for their expertise. With real-world systems being too complicated for the constant redevelopment of hand-coded environments, there will be a significant need for general-purpose simulation modelers. This will provide simulation practitioners with many opportunities in the upcoming boom.

Concurrently, the benefit to the AI practitioners is that they can use simulation models today; simulation has a rich and mature ecosystem spanning thousands of real projects that are deployed across almost all industries. This will allow an expansion beyond the focus on toy and game simulations and into real-world systems.

Regardless of background, both industries can easily collaborate. However, while both simulation and AI are clearly compatible with one another, there is still a separation of expertise. Overall, the compatibility of the two fields enables easy collaboration and practitioners can benefit without the need for retraining. With AnyLogic's interoperability and flexibility, it provides a natural fit for these two industries to come together.

ADDITIONAL RESOURCES

For up-to-date AnyLogic AI resources visit our dedicated page.

- White Papers
- Case Studies
- Videos
- Books
- Download AnyLogic Simulation Software
- Seminars and Training

CONTACTS

THE ANYLOGIC COMPANY

info@anylogic.com

Contact your local office >>