Teaching intelligence:

Designing modular AI that explicitly learns skills and strategies

Contents

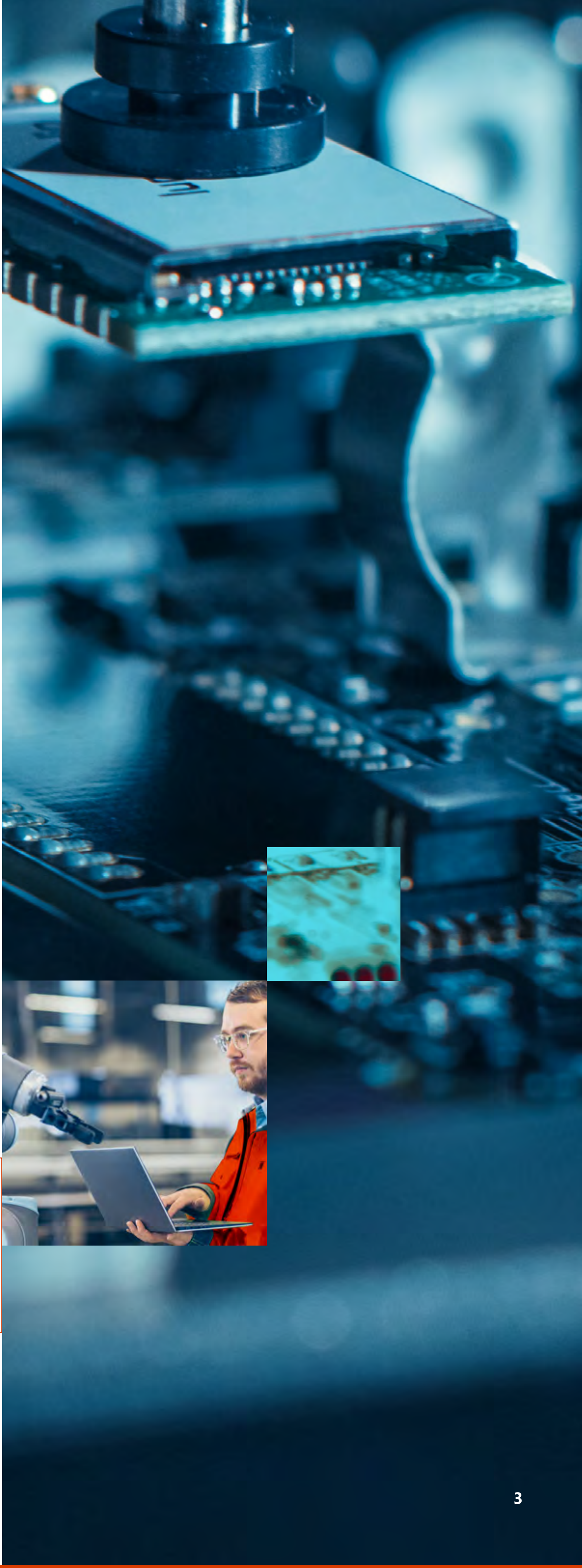
Authored by:

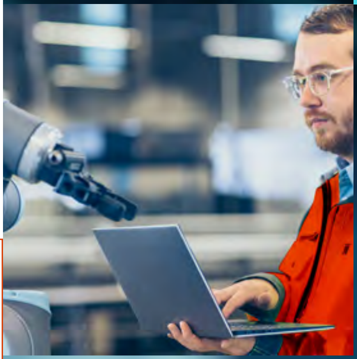
Kence Anderson
Principal Program Manager, Machine Teaching Innovation for Autonomous Systems
Microsoft

Introduction

Most technology that's been realized as a product or program faces a tumultuous transition from applied research and development (usually at a very low level by a small number of innovators) to mass market technology, where it is consumed by a wider audience. The same is true for more advanced technologies and complex systems, where modularity and reusability optimize production and implementation.

Most complex systems from factories to rockets to software programs are broken into modular pieces to separate concerns and reuse design patterns as well as pre-built components. In this paper, we will show that real-time decision-making artificial intelligence (AI) is making the transition from applied research to production-ready AI that is useful in factories, logistics processes, and engineering equipment. Like the control systems that run these factories and logistics processes, and software itself, decision-making AI is easier to build, test, and maintain when decisions are separated into modular concerns. We will demonstrate, through multiple real-world examples and AI research, that learning AI acquires skills more quickly and thoroughly when requisite skills and strategies are explicitly taught and that modular AI can teach skills and strategies explicitly.





Limitations of monolithic AI present opportunities for modular AI

In the early 1990s, there was disagreement in the software community over how to approach development. One side argued that applications should be designed and built as a single monolith, a large and singular vision. The other argued that software should be modularized into distinct, but interoperable, objects. The result of this debate was the software programming paradigm we now know as object-oriented programming (OOP).

The key lesson that came from this debate was the importance of creating a way to ease the transitions between lower-level development and higher-level pieces of technology that can harness multiple smaller processes. This transition includes both an abstraction and a modularization of the lower-level code of technology to make it easier to manipulate. This approach is the same for the development of AI and digital brains made of neural networks.

In his article deep reinforcement learning doesn't work yet, AI researcher Alex Irpan points out that deep reinforcement learning (DRL), the promising AI technology that learns decision-making by practicing in simulation, struggles to learn in complex, real-world environments. He goes so far as to say that he finds it difficult to imagine a real-world application where DRL can and will succeed.

While we at Microsoft Autonomous Systems have helped our customers build AI (using DRL) that makes real-world decisions well, we should explore the reasons why this AI expert finds real-world scenarios so challenging. At first glance, it may seem that Irpan is criticizing DRL, but upon closer inspection, it becomes clear that he is pointing out the limitations of monolithic AI: DRL systems that learn tasks in a single neural network.

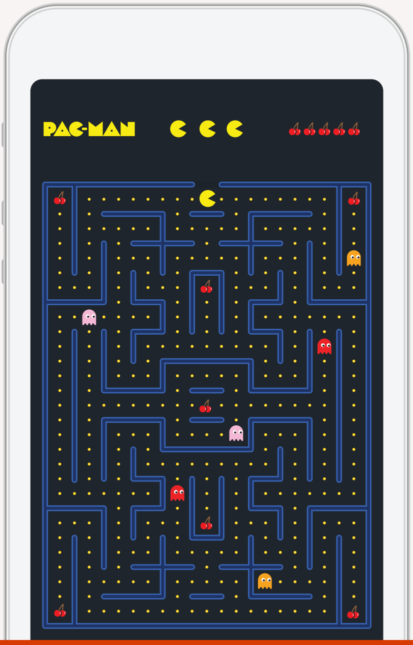
We address three of the points in Irpan's analysis, replacing his references to DRL with monolithic AI.

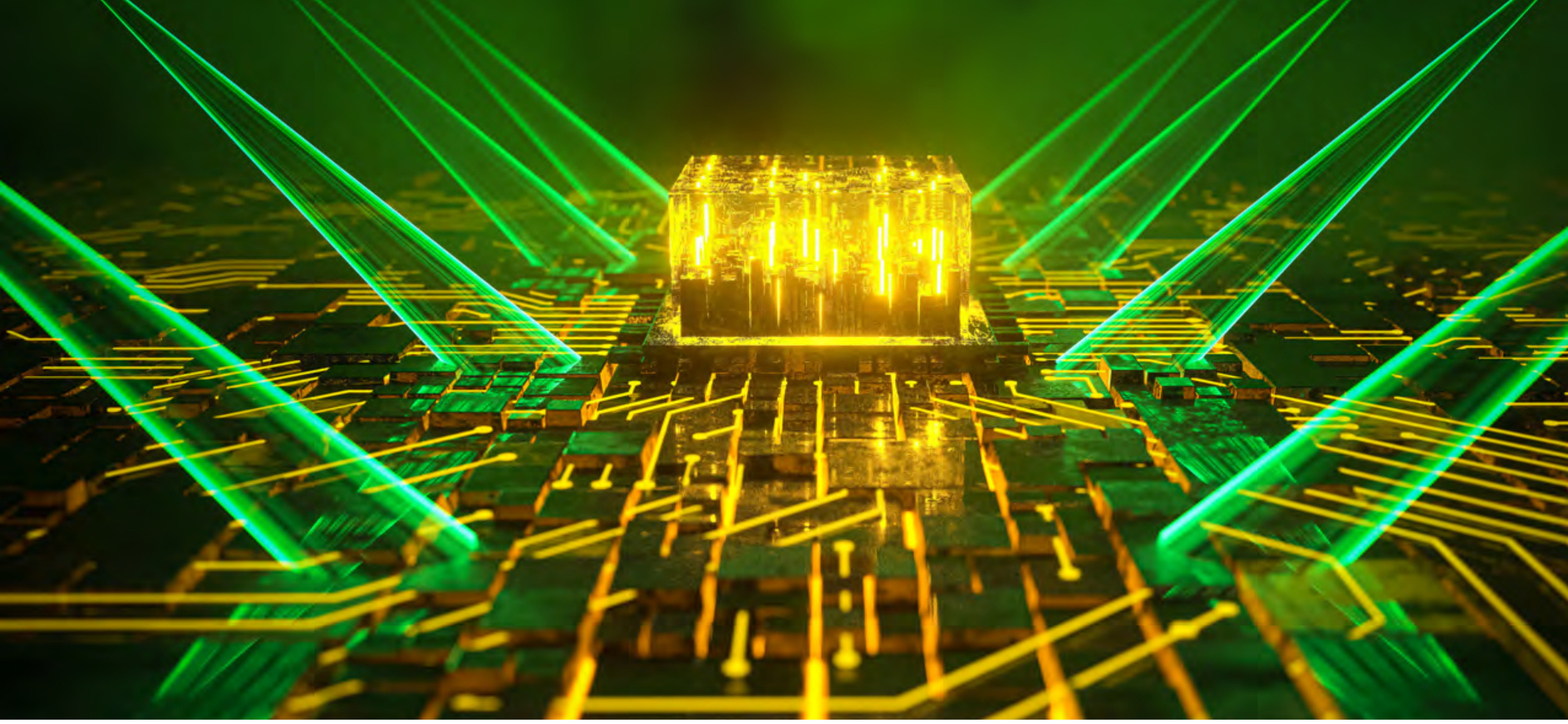
Monolithic AI can be sample inefficient because learning multiple skills simultaneously is challenging for both humans and AI

This is just another way of saying that practice usually happens over a sometimes lengthy period of time. In an equally insightful article published by DeepMind in 2019, the authors argue that neural networks learn skills incompletely and at the expense of each other when algorithms attempt to learn disparate skills simultaneously in a single neural network. In fact, they demonstrate both mathematically and with experimental evidence that it takes exponentially longer for a neural network to learn each additional skill that you try to learn simultaneously.

For example, in the classic arcade game Pac-Man, the player must control the Pac-Man character to achieve four objectives:

1. Eat all the power pellets to complete each maze level
2. Eat fruit to earn points
3. Avoid ghosts that will take one life on contact
4. Eat power pellets, which make you able to eat ghosts



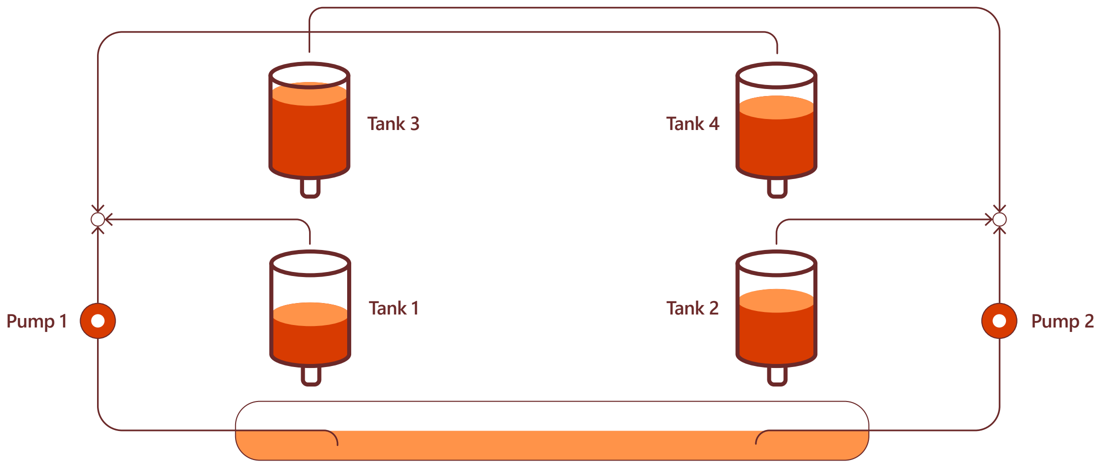


Now, imagine learning to play this game with no understanding of what will happen when you interact with each object. You are rewarded for the score you achieve, and you learn by trying various actions and pursuing higher rewards. When you start playing, you don't know that ghosts will take your life (or even what life is), but you learn from experience that if you come into contact with ghosts on three separate occasions, your opportunity to earn points in the game is permanently capped (you lose). Simultaneously you learn that if you eat a pellet your score increases by 10 points, if you eat a fruit your score increases by 100 points, and if you clear a level by eating all the pellets, a whole new set of pellets (a new game level) appears.

Imagine how much experimentation it takes to learn complex strategies in this environment!

This difficulty in learning multiple skills simultaneously is what DeepMind is referring to when they describe that the performance plateaus observed in Atari might be due to learning multiple interfering skills (such as picking up pellets, avoiding ghosts, and eating ghosts in Ms. Pac-Man).

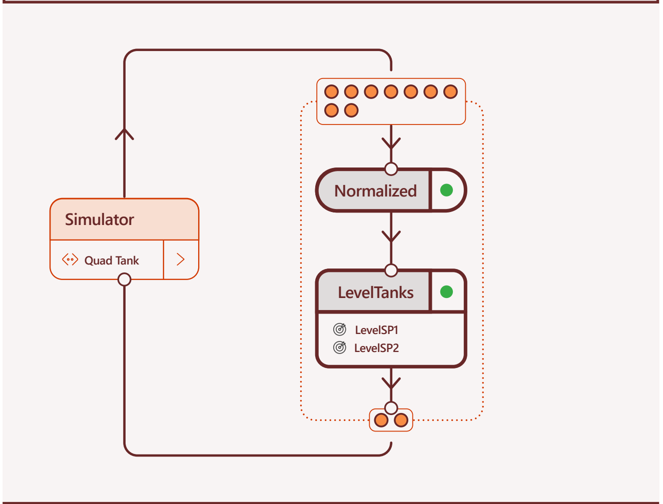
Let's use a more industrial example to show that real-life control tasks also can require multiple skills and strategies and that those skills and strategies are better learned when explicitly taught. The Quad Tank problem is a staple of chemical engineering education where a controller operates two pumps that drive the levels of two tanks to prescribed set points.



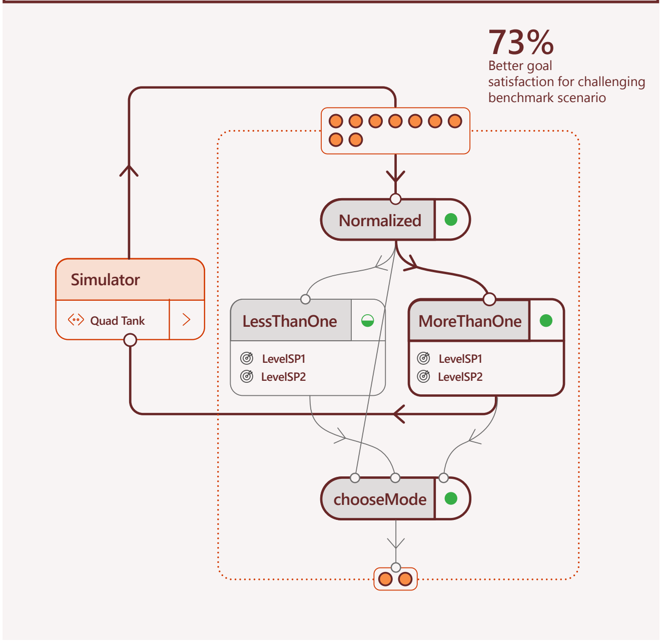
The system behaves completely differently in each of two different regimes: one regime where the overflow travels in one direction and other regime where the overflow water travels in the opposite direction. The values of the variables labeled Gamma1Inv and Gamma2Inv determine which of the two regimes the system is operating in at any time. Each regime requires a different control strategy. Engineers run experiments to show how well the AI brain learns when two control strategies are explicitly taught compared to learning control across both regimes simultaneously.

The following charts compare how well each AI learns to control the system.

Learning performance for the monolithic AI



Learning performance for the modular AI



The monolithic AI is a single module that simultaneously learns to control the system under both regimes. Just as the DeepMind paper predicted, it takes this AI longer to learn the skill, and the two control strategies are not learned well. One is learned at the expense of the other so both skills are not learned completely.

The modular AI explicitly learns each control strategy by allocating a module that will learn each control strategy separately by practicing on the scenario. Designing the AI this way is like teaching the skills separately by forcing them to be learned in separate neural networks that will then be joined together to form one brain. The separate skills are learned more quickly and more thoroughly when they are taught separately.

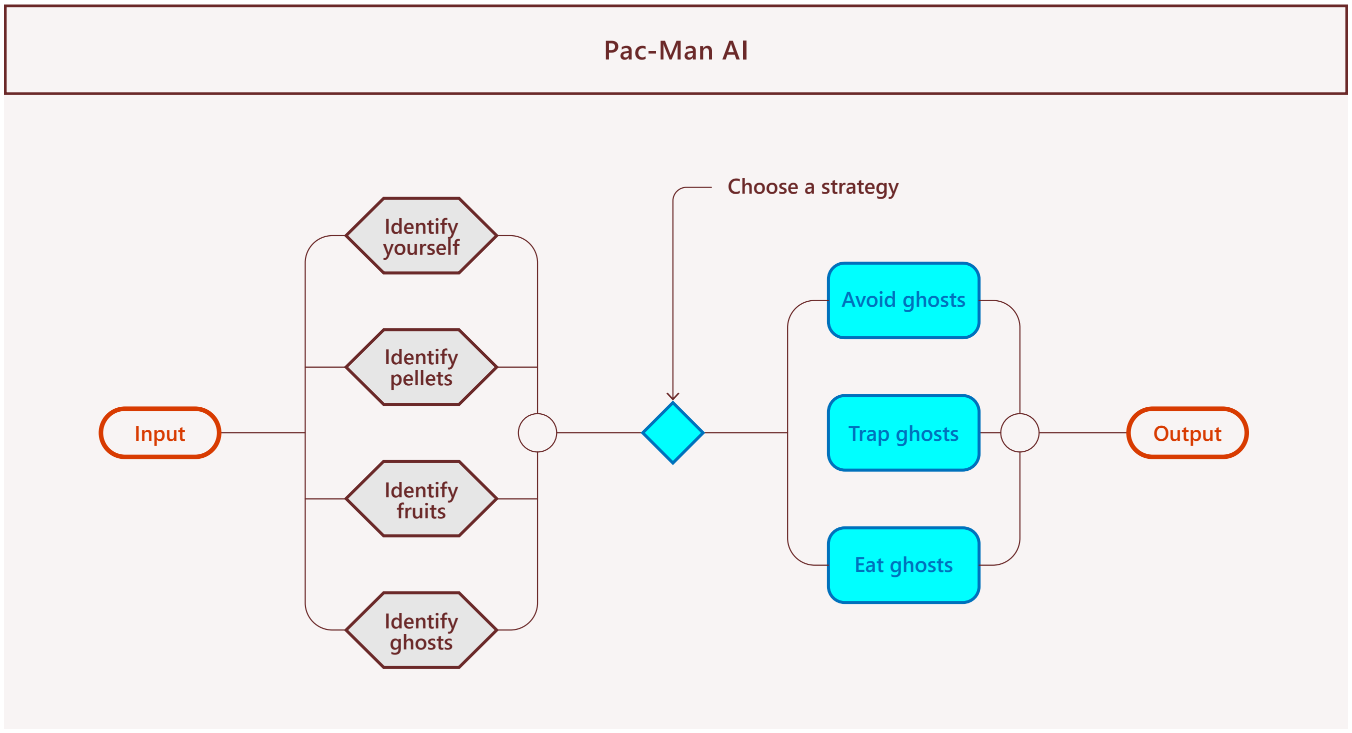
Even when Monolithic AI works, it may just be overfitting to odd patterns in the environment

Here Irpan argues that AI often learns peculiar patterns that match oddities of the environment instead of learning the dynamics of the task, very much like a child might learn an odd technique for bouncing a basketball that has an air bubble that deforms its shape. The ball bounces in an unexpected direction when it lands on the air bubble, so the child learns to dribble in a peculiar way that adapts to the air bubble. This observation is another argument for teaching skills explicitly.

Much like humans, AI takes a very long time to learn multiple skills without being taught those skills properly. When exploring how to perform a task by curious self-discovery, it may be difficult to tell whether you have practiced against all the key learning scenarios or even enough scenarios for you to have learned the task well instead of "overfitting to weird patterns in the environment." A seasoned expert will be able to tell you which scenarios you are likely to face when completing the task. In addition, master teachers can outline all the skills and strategies you will need to complete the task, provide guidance on which strategies and skills will most likely lead to success in each of the key scenarios, and sequence the skills for you to practice in an order that is likely to lead to mastery of the task most quickly.

Modular AI explicitly teaches skills and strategies in each module

Let’s examine what it might look like to explicitly teach skills and strategies for playing Pac-Man to a modular AI. In the same way that an expert teacher outlines each of the skills and strategies that comprise a task, we start with the skills and strategies above and arrange them in a modular brain where they will each perform their assigned function.



The first set of skills to teach is related to perception. Yes, AI can input images directly from a video game, but we may instead want to train a machine learning module to visually identify each of the key game objects and output their location. For example, the “Identify Fruit” module inputs the current game screen and outputs the location and identity of any fruit on the screen.

In the same way that the sequence in which skills are learned and practiced matters for human skill acquisition, the sequence in which we train each module matters for the intelligent system. First, we train perception skills because they will be used by the decision-making part of the “brain,” or decision-making neural network learning from the game. This is analogous to a football quarterback memorizing the structures of opposing defensive teams from flashcards before practicing scoring against any defenses. Next, we train individual strategies.

For example, the goal for an “avoid ghosts” strategy might be to maximize the distance (perhaps expressed as an average or as a total distance) to each ghost on the screen. When trying to execute a “trap ghosts” strategy, the measure of goal success might be monitoring the proximity of the ghosts to each other and to an escape hatch that transports you to another part of the board. Finally, the goal for the “eat ghosts” strategy might be measured as proximity of the ghosts to each other, to a corner, and to some fruit so that you can eat the fruit then trap the ghosts in the corner as you eat them.

Finally, we train a decision-making AI that chooses which strategy to deploy in each situation. This deep reinforcement learning agent uses the system’s perception of where ghost threats, pellet points, and fruit abilities lie on the board to learn when it is best to leverage each strategy as it plays the game.

Teaching framework for designing modular AI

We propose a framework used for teaching skills and designing modular AI at Microsoft Autonomous Systems. This framework first identifies the skills and strategies that you want to teach to an AI, then orchestrates how the skills and strategies will relate to each other while completing the task, and finally narrows in on which decision-making technology will execute each strategy.

One such example is a published robotic application from Microsoft Research that demonstrates the end-to-end framework for teaching skills explicitly to the controlling agent. The specific task for this framework is to teach a brain that can control the Jaco Robotic arm to grasp and stack one block on another block.

Identify the skills and strategies that you want to explicitly teach the AI

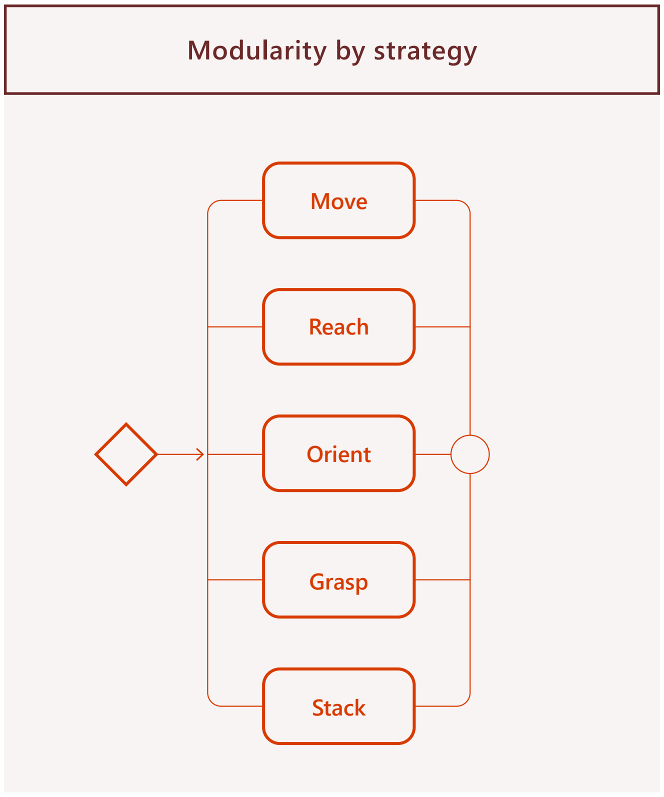
First, we consider the skills necessary to complete the task. Are there separate skills and objectives involved in the same way that we see in the task of playing Pac-Man? Laterally moving the arm back and forth involves mostly the "shoulder" joint. Reaching or extending the arm in a straightening motion involves the "shoulder," "elbow," and "wrist" joints. Orienting the end effector hand around a block is a skill that mostly involves the wrist, and grasping a block is a skill that primarily involves the fingers of the end effector hand.

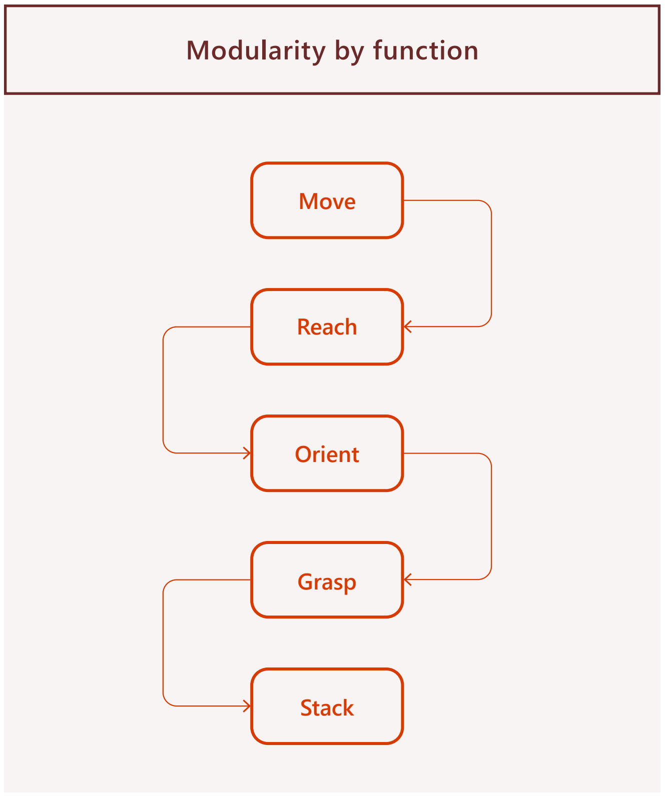
Orchestrate how the skills and strategies interact with each other to complete the task

To demonstrate that the framework allows for machine teachers to design autonomous systems in different ways to perform the same task, we present two ways to design the brain that will teach the same skills differently but explicitly and effectively.

The first design, the one that appears in the academic paper, learns each of the skills separately and then selects from those skills in parallel as if they were strategies to be used, the right strategy at the right time. The second design separates the skills by joint function and arranges them in series to be completed one after the other.

The second design handles the skill sequence by explicitly ordering the skills in the orchestration but introduces rigidity in the process. There is quite a bit of fuzziness in the space between the skills. For example, it's difficult to tell exactly when you are finished reaching and should begin orienting your arm around the block. Similarly, it's difficult to tell exactly when the end effector is in a good orientation and ready to begin grasping. In fact, this transition will depend on multiple factors, including the size and shape of the block. To make this second brain design effective, we will need to set target point goals for the end of each movement.





Select the decision-making technology that will execute each skill and strategy

Now that we have identified the skills required to complete the task and determined how we would like to orchestrate how those skills work together to complete the task, we can choose which technology is best to make the decision. As the paper describes, the actions for the reach and movement skills are commonly calculated using inverse kinematics. So, the researchers trained deep reinforcement learning modules only for the fuzzier skills of orienting, grasping, and stacking, which other technologies have difficulty accomplishing.

The research team compared the learning performance between the first brain design approach and similar work done with advanced transfer learning techniques on a monolithic brain.

The results were telling: the selector module for the modular brain learned how to complete the task well in approximately 22,000 training iterations, while the transfer learning brain took approximately 1,000,000 iterations to learn the entire task from the ground up.

Examples of modular AI in real-life applications

Each of the following brain designs provides additional examples of how to teach AI skills and strategies for completing the task explicitly in separate modules. In each case, the brain designer first interviewed subject matter experts to understand what skills and strategies are used to complete the task.

We borrow from workflow diagrams to express the brain designs in a standard format. The input node represents the environment information coming into the brain from the simulator or the sensors. The output node represents the decisions that the AI brain makes.

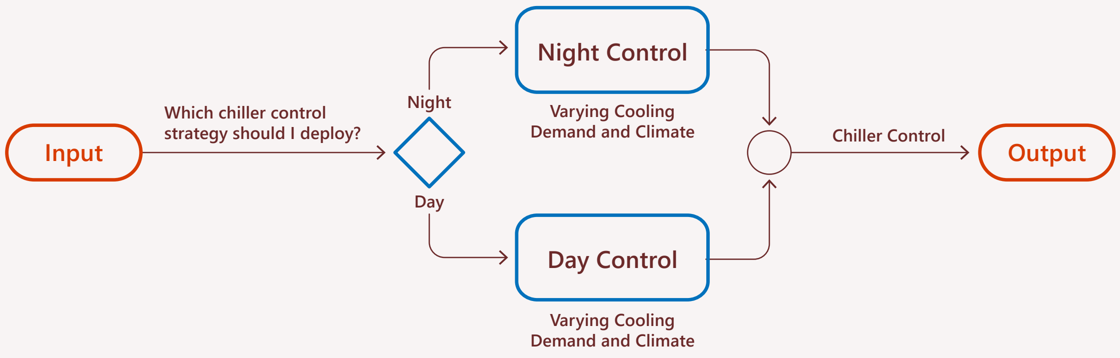
HVAC system control

The HVAC control system was built to control a chiller that cools a commercial building, but imagine this brain as an industrial-strength smart thermostat that you might purchase for your home. Except this smart thermostat brain needs to balance against the outdoor air temperature the fluctuating price of energy during the day and the amount of people occupying the building at any time.

The building occupancy patterns are so different during the day and night that the brain needs to be taught day control and night control separately to learn the task well. During the day the cooling demand is dominated by people inside the building. During the night, there are fewer people in the building, and the cooling demand is dominated by equipment. The cooling strategy required to provide real-time control during the day is different than the cooling strategy required to respond to cooling demands during the night.

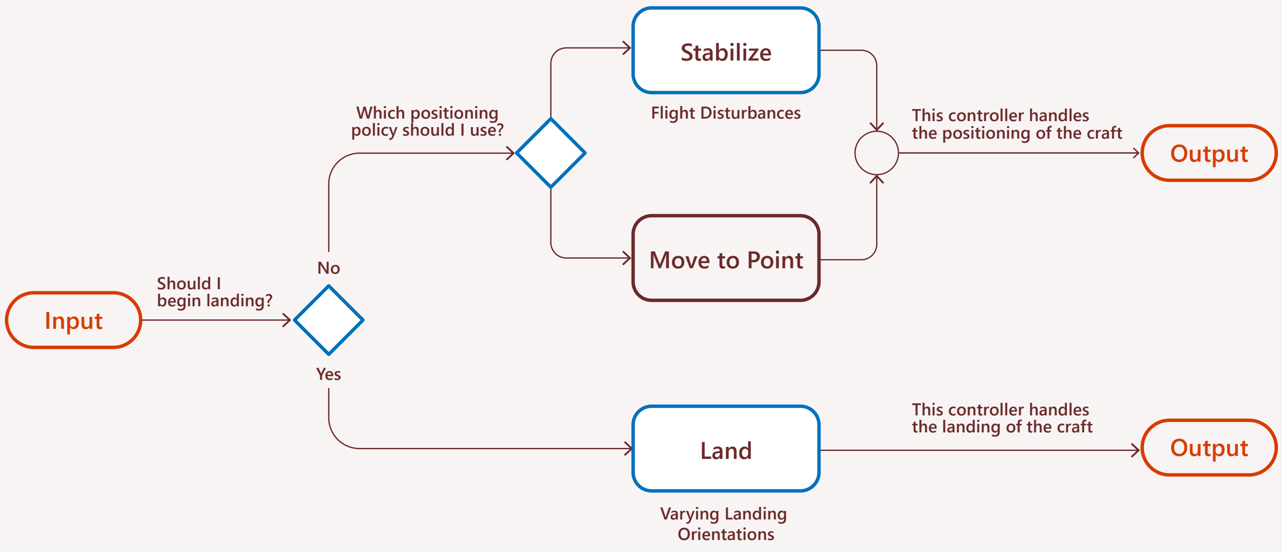
So, we train one brain module that practices controlling the chiller and receiving rewards for responding to changing conditions in simulation during daytime scenarios. We train another brain module that practices controlling the chiller in simulation during nighttime scenarios. The third module is a programmed selector module that activates the day control module during the day and the night control module at night. This selector configuration assumes that there is a stark difference between day and night occupancy; for example, **a company where 99% of employees leave at 5 PM sharp.**

We can imagine an alternative brain design that uses a third AI module that learns when to activate each brain module. This brain design might be useful if the boundary between day and night conditions is fuzzy; for example, if, **at a different company, employees leave anytime between 4 PM and 7 PM.**


Input
Which chiller control strategy should I deploy?
Night
Day
Night Control
Varying Cooling Demand and Climate
Day Control
Varying Cooling Demand and Climate
Chiller Control
Output

Autonomous drone control

This hypothetical example of a brain for autonomous drone control example explicitly learns separate skills for flying from point to point, stabilizing the craft during disturbances, and landing the craft. It is easy to imagine that landing a drone is a separate skill from flying a drone to a destination. The goal is touching down with a slow enough speed to prevent crash damage. Landing is primarily concerned with safe vertical descent, and flying to a destination is primarily involved with horizontal motion. The goal is reaching the destination, avoiding obstacles along the way, and conserving fuel. A brain designer might also teach a separate skill for avoiding obstacles from steady flight.


Input
Should I begin landing?
No
Yes
Which positioning policy should I use?
Stabilize
Flight Disturbances
Move to Point
This controller handles the positioning of the craft
Output
Land
Varying Landing Orientations
This controller handles the landing of the craft
Output

Conclusion

Designing autonomous decision-making AI resembles teaching, and modularity allows us to teach know skills separately and explicitly. We recommend a flexible framework for designing AI brains from subject matter expertise. There are many valid and interesting brain design variations for each application, but allocating separate modules for each skill we desire to teach the AI not only makes the system easier to validate, maintain, and certify but also helps the AI learn more quickly and thoroughly. There may be many ways to build the machine learning layer of these modular systems as well. Just as in software itself, modularity in AI is a powerful tool to make AI more useful.

Learn more about Microsoft Autonomous Systems →

About the Author

Kence Anderson is Principal Program Manager, Machine Teaching Innovation for Autonomous Systems at Microsoft. Kence has designed over 100 autonomous decision-making AI systems for commercial uses, including at PepsiCo, Bell Flight, and Shell. Kence is the son of a master teacher and is trained in mechanical engineering, and he utilizes both aspects by researching how the principles of teaching combined with human expertise can be used to design and build useful AI.