Deep Science Publishing, 2025
https://doi.org/10.70593/978-93-7185-061-2_6

DeepScience
Open Access Books

Chapter 6: Building and Training Autonomous Deep Learning Agents

6.1. Introduction to Autonomous Agents

In the broadest sense, an autonomous agent is anything capable of acting on the environment autonomously. More precisely, it is an entity that operates in the environment without direct action from humans or agents. Most actions of an autonomous agent are sequentially correlated: the current action affects the next action. Many autonomous agents exhibit some form of goal-seeking behaviour; typically these goals are defined through some notion of a reward function or a score function.

Autonomy exists in all forms of life on this planet. People perceive particular events in the environment, often converting the data to an abstract state space. An intelligent person will also choose an appropriate action for the current state. This action may have a long-term effect on a reward score, such as physical health. Of course, people are usually uncertain about their environment and will have a belief, rather than a concrete conclusion, about the best action. Nevertheless, people are capable of choosing their own actions based on their environment.

6.1.1. Overview of Autonomous Agents

An autonomous agent is a software entity that makes decisions on its own, shifting the focus from the classic AI question “How do we train computers to be smart?” to a distribution question: “How do we design and train smart agents to perform a variety of tasks?” Recent advances have yielded methods and tools that allow virtually anyone to build intelligent agents, including three key developments: effective agent architectures (i.e., neural networks that support agenthood), efficient training methods that yield generalist agents with good liabilities, and fast methods for collecting and leveraging training data.

Together, these advances have executed the pipeline for building and training autonomous deep learning agents, thereby migrating AI research questions from lab prototypes to applications in the wild. As such, they have ushered in the rise of autonomous AI. For readers just getting started with deep learning for autonomous agents and those eager to deepen their understanding, the subsequent sections provide brief introductions to each of these core areas. The resulting narrative is necessarily broad and shallow, and the interested reader is encouraged to consult the pointers to related books and articles scattered throughout the text.

6.2. Fundamentals of Deep Learning

Deep learning, a subfield within machine learning, investigates and develops learning mechanisms within artificial neural networks and their variants—convolutional,

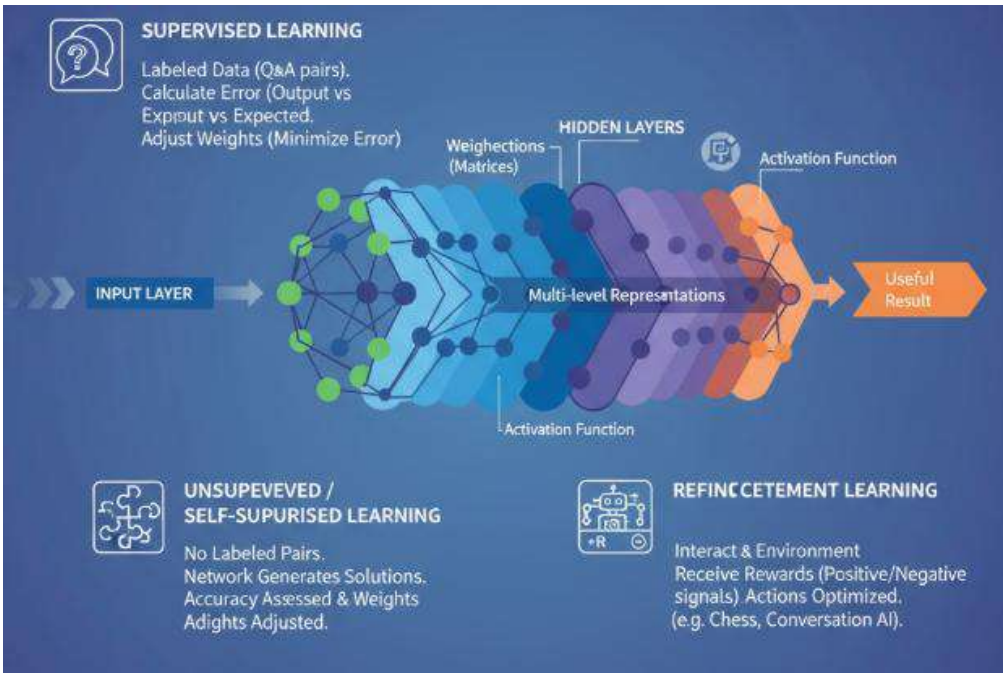

Fig 6 . 1 : Deep Learning: Unraveling Neural Networks

recurrent, graph, variational, and attention-based models among others—through application to large data sets, creating multi-level representations of the incoming data [1-3]. Artificial neural networks consist of elements banded into layers, interconnected using matrices of coordinate-related weights, whose individual values are adjusted over the course of the training process so that the network produces useful outputs. Input nodes, or artificial neurons, convey the incoming signal to the first hidden layer. Hidden nodes operate on the signal, multiplying it first by the weight vector and summing,

adding a bias, and then applying a nonlinear activation function before sending the output on to the nodes of the subsequent layer. In this way, multiple levels of abstraction are learned, displayed by each succeeding hidden layer in the network, until the final layer produces a result matching requirements for a downstream task.

Training involves the presentation of many labeled examples, each followed by an evaluation and adjustment of the hidden connections. The supervised paradigm makes use of datasets of question-answer pairs for the task being modeled, assessing network accuracy after each presentation by calculating the distance between output and expected answer and determining how to adjust the weights to minimize this error. In unsupervised and self-supervised learning, there are no such labeled pairs; solutions to the problem being addressed are generated by the network itself, with accuracy assessed and weights adjusted accordingly. In training by reinforcement learning, the network interacts with an environment, receiving positive and negative signals that measure its occasional actions. These signals, or rewards, can also be generated by other machine learning models, trained for their own purposes—for playing chess, for example, or maintaining conversation in a particular style.

6.2.1. Core Concepts of Deep Learning

Autonomous agents are systems capable of independent action to achieve predetermined goals in complex and dynamic environments. The rapid proliferation of these agents is driven largely by their ability to be trained using deep learning techniques. Having established what autonomous agents are and why deep learning methods are suitable for their development, attention now turns to the core concepts of deep learning as presented in "Fundamentals of Deep Learning."

Autonomous agents are complex data-processing devices that transform raw information into meaningful knowledge. In the context of deep learning, neural networks are employed to implement such devices, although the underlying principles apply to other network types as well. Each processing element in a network corresponds to a neuron in a biological brain. A neuron's state is determined by its activation, which depends on the type of network—feedforward or recurrent [2,4-5]. The internal structure of a neuron within a network is outlined in Figure 2.4; feedforward networks contain neurons of either all the same type or two types. In the feedforward scheme, the activation of units in a given layer is independent, whereas, in recurrent networks, it exhibits temporal evolution.

6.3. Agent Architecture Design

Implementing an autonomous agent in a modern computer or robot is much like building any other deep learning application: first, choose the architecture of the neural network that acts as the decision-making core of the agent. Any type of network can be used, including multilayer perceptrons or feed-forward networks, convolutional networks, recurrent networks, or a hybrid of several networks. Networks with different types of layers and structures can also be combined. Together they define the agent's architecture. When building an agent, modular architectures, in which several neural networks each serve a distinct role, are often very useful.

Next, choose a training paradigm. Most deep learning architectures are trained with supervised learning, meaning that the training data consists of input-output pairs, where the outputs are labels or classes associated with the input data. Supervised learning is commonly applied in computer vision problems, in which the networks are trained on images or video frames that have been labeled with the name of the object that is depicted, for example. This can also be considered teacher-forcing. The teacher forces, or enforces, that the many hidden layers of the neural network must funnel their information in a way that can be used to produce the right outputs for the given training inputs. This particular strategy points at two others that are also widely used for training autonomous agents: reinforcement learning and unsupervised learning.

6.3.1. Neural Network Types

Three fundamental deep-learning approaches to building autonomous systems—agent architectures, training paradigms, and data synthesis—are examined in the context of contemporary research. Agent architectures, commonly implemented as deep neural networks, encompass diverse designs tailored to specific tasks; these representational choices underpin decisionmaking processes that direct the agent's actions in complex environments. Fundamentally, the architecture serves as a mapping function that converts the agent's observations—typically high-dimensional sensory inputs such as RGB images—into subsequent actions, with the network's weights encoding the strategic policy learned from experience.

Neural-model-based agents, exemplified in run-time path-planning algorithms, make decisions without resorting to trial-and-error, exhibiting high reactivity by responding immediately to new environmental patterns. These agents are often modular, comprising function-specific components each associated with a distinct data column; this modularity affords design flexibility, enabling the integration of functions from various methods and allowing additional data sources to contribute new functionality.

Fig 6 . 2 : Deep Learning: Autonomous Systems Approaches

The autonomic hierarchy within this framework recursively encodes constraints on lower-level control elements, expediting autonomous decision-making. In contrast, neural-model-free agents rely on reinforcement learning and trial-and-error strategies, wherein an adaptive feedback system evaluates the agent's accrued experiences, sources data from these experiences, and supplies target values for training a corresponding state-action decision-making neural model.

6.3.2. Modular Architectures

Modular structures can be found throughout nature. Indeed, the modularity of the brain had a great influence on special architectures proposed in the deep learning literature [99]. For example, convolutional networks [98] are inspired by the topological structure of the visual cortex, and translation invariance in the visual system (objects maintain their identity when their position in the visual field is shifted) is exploited through translation parameter sharing in the convolution operation. Modular structures are desirable to study and design, since such structures may emerge from the task complexity. For instance, a task can often be decomposed into manageable subtasks that, in turn, can be solved via a hierarchical structure, where each sub-network solves a particular subtask, and combines and forwards its solution to upper levels.

Modular structures can be effortlessly added to convolutional nets, thanks to the well-known, scalable and modular design of conv layers, which can be flexibly combined to

construct shallow or deep convolutional nets. However, small changes in the behavior of the object being detected (or, roughly speaking, in its statistical properties) require, in the case of a classification network, re-training the entire network. By contrast, the modular nature of the architectures proposed next, allows changes in the statistical properties of some particular features to be addressed by re-engineering only some specific sub-parts of the complete network.

6.4. Reinforcement Learning Basics

Reinforcement learning enables an agent to understand how to interact with its environment to achieve maximum reward. The agent takes an action based on the current state, after which the environment transitions to another state according to some underlying transition model and returns a reward. The agent's goal is to maximize the discounted sum of all future rewards, for which it requires a belief about the value of an action given a state. This value is encapsulated in the state-action value function $Q\pi(s, a)$, the expected discounted sum of future rewards given that the agent took action _a_ in state _s_ and followed policy π afterward. From the state-action value function, the agent's optimal policy $\pi*(a \mid s)$ can be derived by choosing the action that maximizes $Q*(s, a)$, thus guiding the agent's decision-making in pursuit of maximum reward.

An autonomous agent comprises an environment coupled with an agent. The environment generates the inputs and reward signals that the agent model processes, and the agent's outputs influence both the agent's future actions and the environment's state. The training process involves iteratively sampling input, output, and reward signals. The agent's action must encompass all information that might influence future actions, and the reward function should quantify the payoff of an action in a specific state. The computational core of the agent is a state-action value function $Q\theta(s, a)$, parameterized by θ and realized as a feedforward neural network.

6.5. Training Strategies for Agents

Autonomous agents provide a modern approach to designing intelligent systems. By using deep learning algorithms to approximate the policy, value function, or model of the environment, the agent represents the source of intelligence and then justifies the input data to enable decision making. This is an opposite idea from traditional control theory, where the agent is the step-by-step algorithm designed to perform a given task behavior. This strategy has many advantages: the agent can be highly specialized for a specific task, the agent can be designed to operate in multiple tasks, the agent can learn to interact with other agents, and the agent can learn to act from multiple input sources.

Fig 6 . 3 : Deep Learning: Autonomous Agents and Decision Making

An agent interacts with the environment and continually updates its knowledge. An agent that achieves perfect performance can solve unknown tasks based only on interactions with the environment. In simple cases, the agent can be trained by imitation learning. This strategy is followed when the agent does not interact with the environment but can learn from high-quality datasets covering all situations of the problem. More commonly, the agent is updated using self-play, an approach based on the pre-training of the behavior and then using reinforcement learning by fooling an agent with itself. However, this generates a chicken-and-egg problem. Without training the relationship between states and actions, the agent is not able to get useful experiences.

6.5.1. Supervised Learning

Recent advances in artificial intelligence (AI) have brought machines closer to human-level problem-solving abilities. One interesting way to approach AI is by constructing autonomous agents based on deep learning models.

Fundamentals of Deep Learning Only a small subset of deep learning methods has been implemented in autonomous agents. Selecting the conceptually simpler ones leads to a

clearer understanding of the principles and also produces better-based systems. An autonomous agent is a goal-directed entity. Successful action completion requires interaction with the environment and adaptation to the changing circumstances. Interaction with the environment provides the feedback to evaluate the agent's state and guide its next actions. An autonomous agent can be considered a system consisting of an observation function S = Obs(E), which assigns a mark allocation or response based on the activities of the environment E; a reward function R = Rew(E), which assigns a utility, usually a fold increase or decrease in the capital of the agent, based on the activities of the environment E; a state transition function that maps the agent's current state and action to the next state; and finally, a policy π: S $\rightarrow$ A, which prescribes the best action in each state, in order to maximize the expected cumulative rewards in the long run. Action selection by the agent is typically driven by policies that are revised constantly based on updated states and rewards, and on the final objective pursued by the agent. The designer of the agent decides the operating logic of the reward function during the development phase of the agent: the reward function is triggered by internal states of the environment as well as actions of the agent itself, and in opposition to each other. In the context of supervised learning, the agent needs to discover and estimate the relationship between two or more variables presented in the form of samples from a data distribution. For example, a data scientist can design a model that predicts the price of assets based on past records by considering the relationship between the price of an asset at different points in time [1,3-5]. Training employed to develop such a model is referred to as supervised. Training often involves solving a complex optimization problem by following an iterative procedure; however, these details are not vital in the process of building an autonomous agent. As a result, the algorithm is treated as a black box that is capable of producing predictive models. The considerations that matter are, therefore, (a) the right data collection and preparation, and (b) choosing the best training labels and features for the problem at hand.

6.5.2. Unsupervised Learning

Having stated the motivation behind reinforcement learning, it is safe to wonder about its relative advantages and disadvantages when compared to more traditional forms of machine learning such as supervised learning and unsupervised learning. Reinforcement learning, compared to supervised learning, is more akin to how humans are trained into new skills, and is therefore much more flexible (i.e., it can be applied to a wider range of problems). It also enables a concept known as transfer learning, whereby the neural network training in one environment/with one task optimizes also for how fast it can learn other tasks. All this being said, supervised learning remains much more popular and much easier to apply.

Collecting data sets and creating environments that can provide the necessary supervised signals is much easier than defining the "reward" system for a reinforcement learning scenario. Furthermore, there also exist of pre-packaged data sets performing thousands of tasks that can be used to train deep learning models. It is easy to understand the difficulties of combining the advantages of RL and SL through the problem of training a robotic dog to fetch and fetch interactively rewarding the robot dog every time it does something right. Or, to reverse the roles, supposing that the researcher is willing to dog-sit the robot dog every day for months on end, it is not so easy to create the environment in such a way that when the robot dog is rewarded for actually fetching does not simultaneously "break" the training by grabbing the ball and playing NOT FETCH. Instead, one can create a data set of correct actions, and then feed the labeled images / states over so that the robot dog can learn the basic rudiments of the task [6-8].

6.5.3. Reinforcement Learning

Additionally, neural networks allow for context-sensitive behavior, by training a network to choose a response based not only on what is sensed at the moment but also on temporal context. This information can be encoded as internal state of a recurrent net of sigmoidal neurons. Such networks can be trained with a specialized algorithm for recurrent nets or with a reinforcement algorithm that simultaneously learns choice of actions and updates the internal state to remember past events and act on them appropriately.

A different approach is to learn the value of states rather than learning policies directly. States are defined so as to be Markovian, that is, the best action to take in a given state depends only on the state itself. The value of a state is the expected future reward after entering it. By learning the values of states rather than the values of particular actions, agents can attend first to exploring the state space in a systematic fashion. Once the value function is known, an agent can follow a stable policy consisting of the actions that lead to the best successes. Several reinforcement learning algorithms, including temporal difference, Q-learning, and SARSA, are centered on learning value functions of states. By combining these with deep nets of different architectures, it is possible to build very powerful autonomous agents with wide-ranging capabilities.

6.6. Data Collection and Preprocessing

When training autonomous deep learning agents, data collection and preprocessing play foundational roles. Data can be collected through interaction with a simulator or real-world environment, from online sources, or manually created for specific tasks. Careful

data preprocessing can significantly improve the quality of an agent's training experience.

Preprocessing projects the raw data onto a new space where noise is reduced and meaningful relations become more apparent. Properly chosen feature representations can enhance the ability of subsequent modules to extract meaningful information. The discussion draws on previous coverage of training strategies for agents.

6.6.1. Data Sources

Training a neural network, regardless of its nature, requires an appropriate quantity of relevant data, preferably within a proper environment in which the agent will operate. Since supervised learning aims at explicitly providing the function the network must learn, the training data must contain input-output pairs that, when displayed, correctly represent the behavior of the network. However, providing input-output pairs typically has some drawbacks. First, supervising the input-output pairs requires that someone capable of performing the task also runs the demonstrations. Second, supervisors are frequently only able to demonstrate how to behave in simpler scenarios, limiting the complexity of the collected data. Third, many states are never visited while simply running the environment; therefore, no information is collected for the controller to learn from.

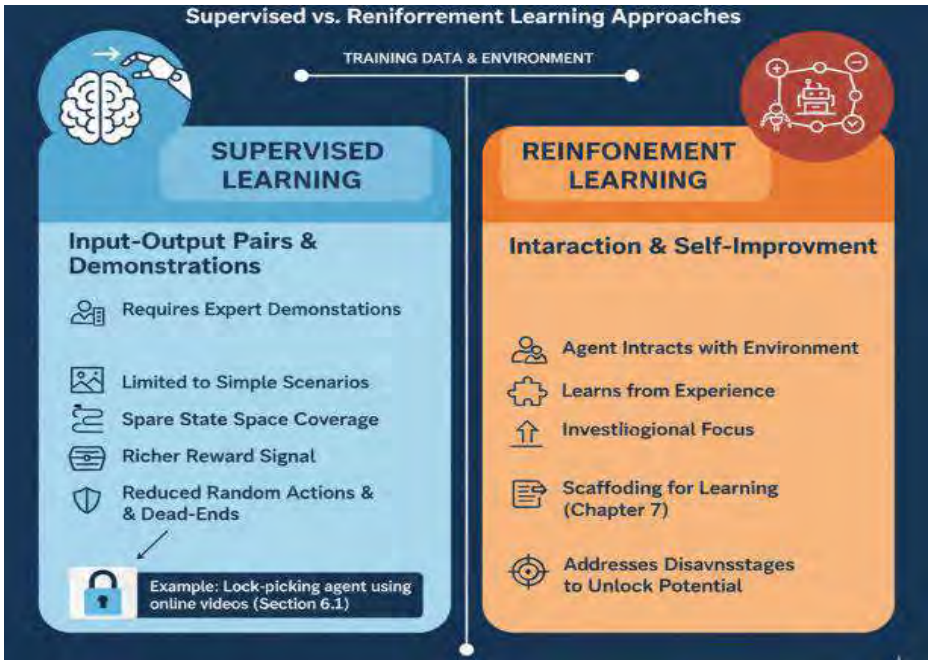

Fig 6 . 4 : Neural Network Training: Data & Environment Paradigms

On the other hand, the advantage of supervised learning is that it provides a richer reward signal, reduces the probability of taking random actions, and reduces the risk of ending in a dead-end during training. Although reinforcement learning suffers from some of the drawbacks of supervised learning, its advantages encourage researchers to continue investigating ways to reduce these disadvantages. Subsection 6.1 describes a lock-picking autonomous agent built using supervised deep learning, which leverages the abundant availability of videos on the task to learn from. Chapter 7, on the back cover of Fundamentals of Deep Learning, uses the ideas behind the agent to build and scaffold agents so that they can be easily trained using reinforcement learning.

6.6.2. Data Cleaning Techniques

Another crucial task for creating autonomous agents capable of handling unstructured data is data cleaning. Although autonomous agents can be trained to recognize and clean data, data cleaning in itself is a widely researched topic in machine learning. For example, Duan et al. [32] solved a specific practical data cleaning problem called data deduplication or entity matching. Recently, Krishnan et al. [55] introduced a multi-agent approach for data cleaning, and in particular data imputation—the filling of missing values in a data set.

Beyond automatically cleaning data, it is also critical to guarantee that the data collected by an autonomous agent is not biased or influenced by the agent itself. Thus, researchers from different areas of machine learning recently started to investigate data collection approaches that guarantee privacy, debiasing, or causality.

6.7. Future Directions in Agent Development

The field of autonomous deep learning agents has achieved remarkable progress recently, yet numerous research avenues await exploration. The examination of agent architecture remains a rich domain, including a detailed study of modular networks, their benefits, and their ability for lateral transfer. Building upon the previous discussion on Training Strategies for Agents, an in-depth analysis of data collection and preprocessing techniques is equally vital. Training strategies themselves offer fertile ground for research; for example, recently advanced supervised approaches, such as procedural generation and strategic policy distillation, have demonstrated promising results. Further exploration of methods that employ diverse task sets for specialized subgoal mastery, followed by distillation into a generalist network, could yield significant improvements.

Equally compelling are the directions presented by generative modeling. Generative models lend themselves naturally to environment design for reinforcement learning tasks

and might also be harnessed to generate new task descriptions. Conversely, methods from planning and classical AI—particularly search and optimization techniques—merit integration into the training of generalist agents capable of operating in complex environments. Developing agents with a modular architecture reinforces this thrust: employing advanced linguistic and logical models for natural language processing as adaptable services enables their application to a broad spectrum of tasks through lateral transfer.

6.7.1. Emerging Technologies

Narrow AI systems are usually trained in specific ways. Supervised learning methods are often used, in which a system is trained on input and output pairs provided by human teachers. "Cats-dogs" classifiers are the most popular example; training samples are labeled pictures of cats and dogs. Unsupervised learning methods are those in which the system trains only on the inputs; for example, by labeling clusters of input samples that are similar to each other. The dominant contemporary learning paradigm for autonomous agents, however, is reinforcement learning, in which an agent interacts with a world and learns the sequence of actions needed to maximize future reward signals.

Techniques are emerging that combine the advantages of all three approaches — supervised, unsupervised, and reinforcement learning. For example, an agent can first gather unsupervised knowledge about percepts and possible actions using a method such as curiosity-based learning [5,9-10]. Then, during a second phase, it tries to learn behavior strategies that maximize externally assigned goals, using that initial preprocessed knowledge as a starting point. Current research is also investigating the transfer of knowledge from one agent to another, which in a training perspective can be framed as moving from supervised learning between different agents. In addition, multi-modal data sources can be combined, permitting agents to train on several types of data simultaneously (e.g., audio and image information). These strategies are of interest because they can either reduce the amount of training time required for an agent to master a task, or make the system much more general.

6.7.2. Interdisciplinary Approaches

Interdisciplinary approaches are equally important, as game-playing decision-making agents with self-awareness, advanced reasoning, and explicit communication will likely incorporate various principles from cognitive science and linguistics, such as deep models of theory of mind, probabilistic programming, and grounding. It has been proposed that Bayesian inference with generative adversarial networks can bridge the gap between probabilistic programming and deep learning, and significant progress has

already been made in leveraging GANs for probabilistic inference. Although such research directions remain largely unexplored, a thoughtful integration of deep reinforcement learning with several core topics in AI and cognitive science may ultimately lead to smarter and more trustworthy agents.

The discussion thus far has focused on deep reinforcement learning alone. The presented building blocks, however, pertain more generally to design and training of versatile deep architectures for autonomous agents. Even when the final agent does not employ reinforcement learning as its main optimization architecture, it will still likely need to leverage reinforcement learning for key training tasks: generating simulated experience to fine-tune an agent trained on supervised or unsupervised learning from real-world experience or shaping an agent's behavior in an environment allowing fast training. Hence, interconnections with other training paradigms, such as supervised learning, unsupervised learning, and generative adversarial learning, will remain a central aspect of the future development of autonomous decision-making agents.

6.8. Conclusion

Artificial intelligence (AI) is evolving towards a state where deep learning–trained agents transform ideas into actions autonomously. Such agents independently explore ideas, generate scenarios for addressing them, and execute the most suitable scenario to solve the problem or deliver the AI service. Their ability to translate questions into answers without human assistance ushers in a new AI era of extraordinary growth.

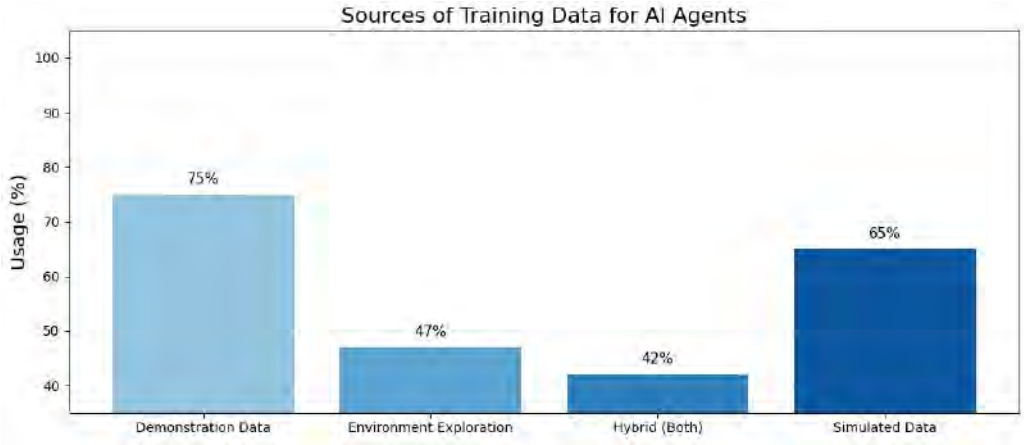

Fig 6 . 5 : Sources of Training Data for AI Agents

The first step in preparing an agent for operation involves selecting an appropriate neural network type. Common models include deep neural networks, feed-forward neural networks, convolutional neural networks, recurrent neural networks, long short-term

memory networks, and transformer networks. Leveraging a modular design allows each specific neural network type to be trained through supervised learning, unsupervised learning, or reinforcement learning. Agents can utilize demonstration data, experiences from exploring the environment, or a combination of both for training purposes.

6.8.1. Summary and Insights on Autonomous Agent Advancements

Autonomous agents must be able to independently perform activities such as building and training. Building agents involves integrating architectural components—sometimes realized as neural networks—of certain types into an agent architecture. Training includes exposure of the architecture to various data-input–transform-target tuples that teach the agent what to do and when. These high-level definitions allow the creation of agents capable of constructing their own building and training processes, enabling flexible and diverse functionality.

Agent-building methods are not restricted to neural networks. Given an autonomous-agent architecture, various components—such as planning, attention, modulation, and memory—can be organized to yield any type of functionality. Training methods encompass supervised learning, for example, training convolutional networks like Inception-ResNet-v2. The proposed method for training an agent with arbitrary components broadens the scope of possible data and training paradigms beyond the supervised-learning training of convolutional networks. Notably, training does not necessarily require data labeled with the actions the agent should take, allowing reinforcement learning training with rewards for good behavior. The subsections that follow tackle architectural considerations and training options for autonomous agents next.

References

[1] Erhan L, Ndubuaku M, Di Mauro M, Song W, Chen M, Fortino G, Bagdasar O, Liotta A. (2020). Smart Anomaly Detection in Sensor Systems: A Multi-Perspective Review. arXiv preprint.

[2] AI-Powered Fraud Detection Systems in Professional and Contractors Insurance Claims. (2024). IJIREEICE, 12(12). https://doi.org/10.17148/ijireeice.2024.121206

[3] Mohammadi M, Al-Fuqaha A, Sorour S, Guizani M. (2017). Deep Learning for IoT Big Data and Streaming Analytics: A Survey. arXiv preprint.

[4] Recharla, M. (2024). Advances in Therapeutic Strategies for Alzheimer's Disease: Bridging Basic Research and Clinical Applications. American Online Journal of Science and Engineering (AOJSE)(ISSN: 3067-1140), 2(1).

[5] Palem Gopalakrishna. (2013). Condition-Based Maintenance using Sensor Arrays and Telematics. arXiv preprint.

[6] AI-Based Testing Frameworks for Next-Generation Semiconductor Devices. (2025). MSW Management Journal, 34(2), 1272-1294.
[7] Kim B-S, Park H-S, Kim K-H, Godfrey D. (2017). A Survey on Real-Time Communications in Wireless Sensor Networks. Wireless Communications and Mobile Computing.
[8] Sriram, H. K., ADUSUPALLI, B., & Malempati, M. (2021). Revolutionizing Risk Assessment and Financial Ecosystems with Smart Automation, Secure Digital Solutions, and Advanced Analytical Frameworks.
[9] Zhang X, Li Y, et al. (2023). A Predictive Analytics Framework for Sensor Data using Time Series and Deep Learning Techniques. Neural Computing and Applications.
[10] Kaulwar, P. K. (2020). Leveraging Data Warehousing for Compliance Analytics in Tax Advisory and Financial Auditing. Global Research Development (GRD) ISSN: 2455-5703, 5(12), 208-225.