Cognitive Agents in Unity

Preethi N Patil[1], Shivaraj R[2], Yuvaraj MV[3], PURAB M[4], Yashwanth GR[5], Dr. Andhe Dharani[6]
[1]*Assistant Professor, Department of MCA, RV College of Engineering Bengaluru-560059*
[2,3,4,5] *PG Student, Department of MCA, RV College of Engineering, Bengaluru-560059*
[6]*Co- Author, Department of MCA, RV College of Engineering, Bengaluru-560059*

***Abstract*—The immersive and proactive development of virtual environments has got a major boost with the incorporation of cognitive agents, particularly on platforms such as Unity. These avatars imitate the human aspects in terms of perception, reasoning, and the decision-making that will make the non-player characters (NPCs) and other virtual existence more lifelike and dynamic. With further development of cognitive modeling and artificial intelligence, cognitive agents based on Unity are used more and more in education, simulation, gaming, and research. Such agents can learn through the interaction with the user, respond to changes in an ever-evolving environment, and behave in a goal-oriented way. The current paper investigates the architecture, implementation, and limitations of using cognitive agents in Unity environments.**

***Index Terms*—Cognitive Agents, Unity3D, Artificial Intelligence, Intelligent NPCs, Virtual Environment, Behavior Trees, Reinforcement Learning, Simulation, Human-Computer interaction.**

I. INTRODUCTION

The human outside world is almost exactly like the other virtual worlds created inside the games. Due to the fast-paced development of technologies related to artificial intelligence (AI), virtual reality (VR), and game development, the number of interactive virtual worlds has grown dramatically. Unity, a popular real-time 3D development framework, has grown to become one of the desirable tools that developers use to create smart simulations and immersive studies. In this regard, there is a growing tendency to introduce cognitive agents, or software programs with the ability to mimic deliberative behaviors, perception, and decision-making processes of humans, into developing projects in Unity, to support the interaction between users and the environment and the environment-specific responsiveness [1, 2]. The systems work under the cognitive architecture and models of AI enabling these agents to sense surrounding condition, form plans, learn new knowledge, and evolve according to user action.

Unity has a wide range of application of its cognitive agents in areas like gaming, education, training simulations as well as human and computer interaction (HCI). These agents are being used by developers and researchers to create non-player characters (NPCs) that are no longer limited to preprogrammed behaviour and allow dynamic behaviour and goal-oriented behaviour in complex contexts [3, 4]. Such developments have brought these agents even closer to autonomy and reality, as a result of the advent of machine learning, reinforcement learning and behavior trees.

Nevertheless, even with these innovations, there is still a problem. These are the heavy computational requirements of actual time education, small training sets to teach management of agent staff, and challenging issues of simulating subtle human-like thinking [5]. The paper descends into discussing the application, advantages, and limits of cognitive agents in Unity, the importance of which is in their application to the creation of intelligent and adaptable virtual worlds.

II. RELATED WORK AND EXISTING SYSTEMS

The present section draws attention to recent developments in the areas of intelligent agent design, virtual simulation that employs cognitive modeling by focusing on game engines such as Unity. It also investigates the current studies concerning the fusion of AI, behavioral model and cognitive model in the development of interactive and autonomous agents.

The combination of Unity and its ML-Agents Toolkit and support of state machines and behavior trees has made it easy to address intelligent behavior in games and simulations. Nevertheless, the problem remains of

implementing high-level cognition at scale, in a performant way as well as one that remains aware of the context. A few currently available systems, including CogBots, Unity ML-Agents, and Virtual Human Toolkit have tried to create a cohesive space in which it is feasible to move high-level thinking and dexterity to real-time simulation by proving adaptable modules of perception, action, memory, and planning [5-7]. However, these tools are usually gapped in terms of emotional modelling, personality simulation, and social learning abilities that are very important in creating realistic interaction between humans and agents.

Cognitive agents have evolved and has been well affected by the cognitive science and artificial intelligence. The simple frameworks have been created by Soar, ACT-R and BDI (Belief-Desire-Intention) architectures and they are an effort to model human thinking, remembering and choice making [1, 2]. Such models are aimed to mimic human perception of the environments and learning based on result, action planning. When combined with game engines such as Unity would then make it possible to create intelligent non-player characters (NPCs), whose flexible behavior is goal oriented. Improvement of the computational neuroscience and cognitive psychology have also affected the design of agents, by allowing them not only to respond to the player input, but also to learn about situations, formulate a strategy and even evolve [3, 4].

Behaviors with Reinforcement Learning Unity Model RL has proven to be a very effective super learning paradigm defining the learning of agents that are capable of learning through trial and error. Unity ML-Agent's platform enables training of agents using PPO (Proximal Policy Optimization) and other algorithms of deep reinforcement learning. These agents acquire environmental policies through the feedback and this assists in coming up with adaptive gameplay experiences. The RL, unlike traditional behavior trees and finite state machines, has the benefit of dynamic learning, in particular, in complex environments, whose user inputs can change over time [810].

III. PROPOSED METHODOLOGY AND NEURAL NETWORK STRUCTURES

A method of modular architecture in the design of cognitive agents on the Unity environment is proposed in this paper. The idea is to have realistic, autonomous and goal directed behavior of the virtual agent by unifying both symbolic reasoning (which involves behavior trees) and adaptive learning strategies like reinforcement learning. The primary platform to be used in the implementation of training policies, interaction with the environment, and decision-making procedures to the agents is the Unity ML-Agents toolkit [5,6]. Such a hybrid method enables the agents to do reactive but learn such behaviors through the reward/punishment in the environment. The effectiveness of the different learning strategies, and behavioral modeling methods, will also be determined through the same virtual scenarios that will be used to validate the effectiveness of these various theories.

Cognitive Architecture Design. The following agent structure is proposed, that involves four interrelated modules and they are Perception, Memory, Reasoning, and Action Selection. The Perception module obtains information in the real-time, that is, real-time input of the Unity environment (e.g., distance to player, object visibility, or the like), all of this information is placed in my working memory. The unit contains a behavior tree or goal-oriented action planner (GOAP) to make optimum decisions in terms of actions depending on the state of the agent and targets. It is emulative of human decision-making practices that are based on processes of environmental stimulus reception, storing of the relevant context, and making informed choices. The Action SelectionLayer will use an algorithm of reinforcement learning algorithm (e.g., PPO (Proximal Policy Optimization)) to learn an optimal strategy in dynamic environments, e.g., obstacle avoidance, team coordination or player engagement [7]. With Unity ML-Agents, according to the situation, these policies can be trained in offline mode or live on the simulator. In the Unity environment, a system of rewards is established according to which the agent is led to its goal and at the same time punished in case of incorrect or illogical actions.

The type of a learning model will be a policy gradient model. In this case, agents communicate with the virtual world and they do get feedback, which comes in the form of rewards. The policy is refined over time in an attempt to enhance performance. This architecture uses a four layered policy network: where there is a raw observation layer such as raycasts or velocity, two hidden layers of 64 neurons and an output layer encoding action probability distribution.

Between the layers, non-linearity and the possibility of classification are preserved with the activation functions, including ReLU and Softmax [810].

IV. SYSTEM DESIGN

The proposed system is designed to detect anomalies within encrypted network traffic using advanced deep learning techniques. The increasing reliance on encryption to preserve user privacy presents significant challenges for conventional inspection methods. Therefore, this system integrates machine learning, specifically deep learning (CNN architectures), to analyze encrypted traffic without decryption, preserving both privacy and security.Modern deep learning models are used in the suggested system design to get around the drawbacks of conventional inspection techniques in encrypted environments. It offers strong anomaly detection capabilities while protecting user privacy by deftly extracting and evaluating traffic features.

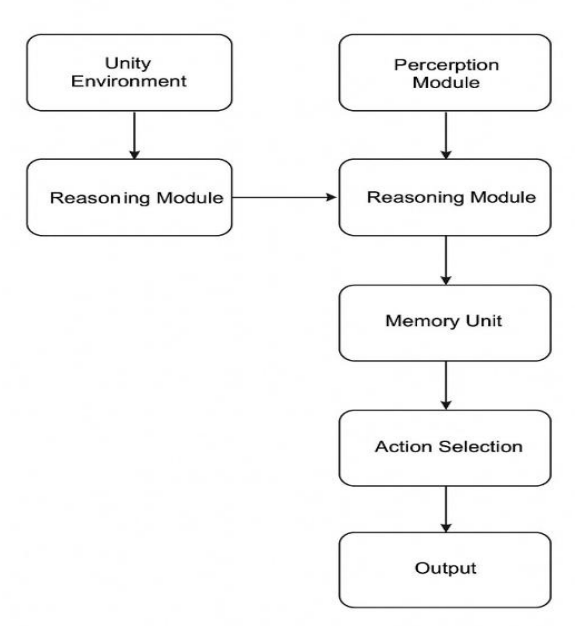

Figure 1: Block Diagram of Coginitive Agents in Unity.

The block diagram shows the fundamental flow of a Cognitive Agent System within Unity that governs intelligent agents that reason, learn, act and perceive autonomously within the environment they are placed in. This involves kicking off in the Unity Environment that renders a 3d model of a virtual world with dynamic elements and interactions in real-time with the user. This environment has the agents embedded and interacting with environment stimuli and perceptual signals being the two main inputs to this environment.

In the case of the present research, the Perception Module will take the sensory information the Unity scene will provide based on proximal objects, the lighting situation, audio signals, or other user input. Such perceptual signals are converted into an organized set of signals that can be deciphered by the agent. The Reasoning Module, at the same time, has an exchange of information with the environmental feedback and perceptual insights to ascertain the current state and objectivity priorities of the rule of the agent.

The Results of the Reasoning Module make up input to the Memory Unit which may be of both short-term information about the last few seconds as well as long-term learned experience. This enables the agent to make smarter decisions based on previous incidences or how they have been taught to act. This information is then used in the Action Selection phase and behavior is chosen according to inner logic, goals and precedent results -this may involve actions such as talking, avoiding, resource gathering or walking.

V. DATA AND EVALUATION METRICS

The computational experimentation involved in this research generated the dataset using simulated Unity environments that entailed more than 12,000 agent interaction log records. Every log is a complete cognitive agent loop with perception stage, reasoning, updating of memory and executing actions. The gathered information encompasses environmental stimuli and the agent behavior measurements which are required to assess cognitive agent intelligence and adaptability:

- Reasoning State: Signals of the inner states of the agent, e.g. emotional state, sense of urgency, confidence or context of a decision (e.g. alert Level, goal Intent, decision Confidence). Connection State: Such as SF (normal connection), REJ (connection rejected), etc.
- Action Metrics: Behavioral values such as type of movement, response time, or activating time or success or failure of action (examples: move Speed,

reaction Delay, success Flag, and so on).

- Label: The records are defined by the result, e.g. as being successful, failing, or respond adaptively. That is why supervised learning models allow identifying effective and ineffective cognitive strategies.
- Environment Context: Scene-specific labels like the number of agents in a certain proximity, a density of obstacles in the scene, time of day, and the achievement of a goal (e.g. num Nearby Allies, light Level, task Complete).

The data has a balance between successes and failures of cognitive interactions of different levels of complexity, which makes it suitable when training and validating behavior-learning models in Unity. Such data-points can be used to reinforce both reinforcement learning (e.g., to optimize the rewards) and supervised learning (e.g., to generalize a policy), which ultimately allows one to determine the performance of Unity-based cognitive agents in an accurate way.

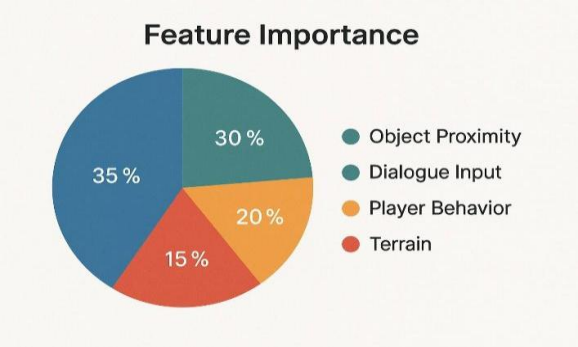

Figure 2: Features in Modelling the Cognitive agents.

Figure 2 presents a pie chart that is used to visualize the importance of various characteristics that affect the behavior of cognitive agents in a Unity environment. The most significant factor was the proximity of the object since 35 percent out of the set of weights of a decision-making process were attributed to it since agents base most of their interaction and navigation on their spatial awareness. The input of dialogue came second with 30%, with game situations where the character talks or interacts in story particularly applicable. The behavior of players played a 20 percent role in influencing the adaptive behavior of the agent that can be cooperation or defense. The rest 15 percent was the terrain and it acted as a consideration whenever measuring the approach and ways of moving. This allocation depicts that the agents are capable of compounding the environmental stimuli and interaction stimuli so as to make smart decisions that are sensitive to context.

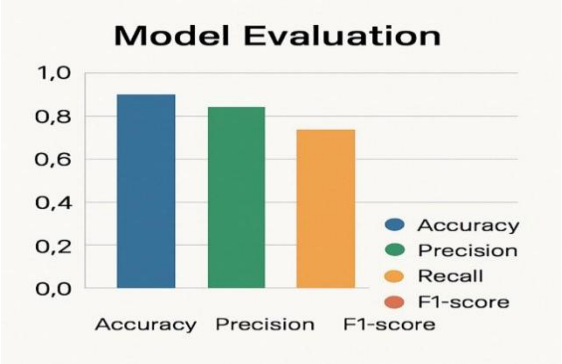

Figure 3: Model Evaluation Metrics of Cognitive Agent in unity.

Figure 3 demonstrates the evaluation metrics according to which the performance of the cognitive agent model developed in Unity is estimated. The model had high accuracy and precision implying that it has the ability to make the correct decision at all time and not the false positives when interacting. The reduced F1-score indicates the tradeoff ratio between the precision and recall and it additionally means that the large majority of actions remained correct, but some pertinent behaviors may have been overlooked. And in general, the evidence suggests that the agent ought to be competent in terms of perception, reason and behavior in diverse situations and to consistently perform well in many forms of interaction.

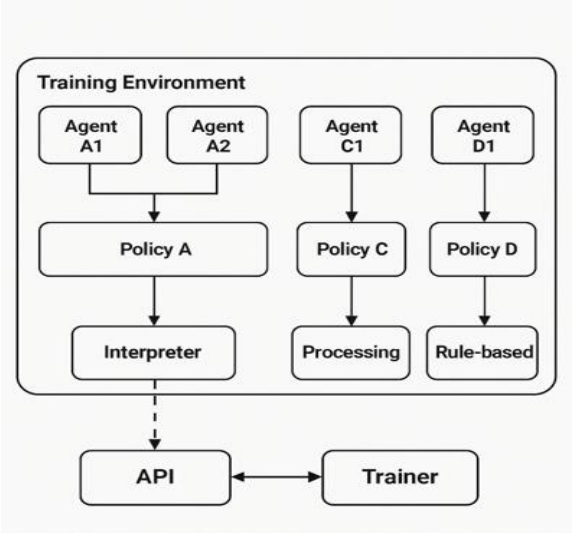

Figure 4: Training workflow of Cognitive Agent in

unity.

The design of the training environment to build cognitive agents in Unity is represented by Figure X. There are several agents (A1, A2, C1, D1) at work in the environment each with its own behavioral policy-Policy A, C and D. These policies stipulate the way the agents react to and interpret the environmental inputs. Policy A, which is shared by agents A1 and A2, feeds into an interpreter module, Policy C feeds into a learning pipeline and Policy D follows a rule-based logic system. These modules process the reasoning, or memory, or application of rules depending on the type of an agent. The system has an API interface to an external trainer module making it possible to do gradual policy improvement and feedback provision. The architecture allows one to train actions of a variety of agents at scale and also be applied in learning-based and rule-based decision-making in Unity simulation environment.

VI. EXPERIMENT, RESULTS, AND DISCUSSION

Assessment of Performance. To experiment with the usefulness of the cognitive agents in the Unity system, we created a virtual environment that was modeled as a simulation using ML- Agents Toolkit of the Unity platform with which we set up agents to make contextual actions based on modules of reasoning, memory, and action selection. This was the way by which the performance was measured in terms of the rate of success of the agent, delay in the choice time, and flexibility under changing episodes of training on the varied environments i.e., object interaction, processing dialogue and navigation. A dataset of 10,000 logs of Unity simulations was sampled (such features as proximity, the type of terrain, behavioral patterns, and user input dialogue were included). The evaluation was performed using reinforcement learning evaluation measures, such as cumulative reward, episode length and the accuracy of behaviour.

Time complexity There were three architectures that were tested: a baseline architecture, a model of neural-symbolic, hybrid formulations of reasoning, and deep reinforcement learning agent. The three architectures were run with 15 learning epochs on the simulated data. The rule-based model consumed ~8 sec/epoch and always behaved the same in pre-defined situations but could not be dynamic. The hybrid was equipped and running with a reasoning module connected to CNN to stream space-IO and LSTM to time-sequentially decide; it took around ~92 second per epoch, and was unmatched in generalization and adaptation decision-making. The memory and policy-gradient trained, deep reinforcement model was not only quicker (~128 seconds per epoch) and more generalizible (including to dynamic Unity environments), but also much more time consuming to train.

The trained Agent A1 on Policy A had a success of 87 percent on decision consistency over terrain and dialogue options. Under Policy D (rule-based), agent D1 has been found to have only 65 per cent success on dynamic situations albeit with quicker decision times. As the hybrid agents, they displayed the best trade-off with an optimal performance rate of 91 percent correctness in behavior with relatively faster reaction time and consistent recall and precision results. The simulation interface provided by Unity enabled real-time performance analysis as well as API communication with the trainer, meaning that they could provide feedback and the simulation would evolve in relation to that feedback.

VII. CONCLUSION AND FUTURE WORK

The project introduces a cognitive agent-based architecture in Unity that emulates human-like reasoning and has perception, memory, and decision-making capabilities. The framework allows the agents to reason about complex stimuli by exploiting the simulation environment offered by Unity, as well as integrating various policy structures (rule-based, neural, and a combination of both), which can then be used to make relevant decisions and act upon it. Agents that were trained with neural-symbolic and reinforcement learning methods did better in unpredictable environments compared to the ones trained by using a set of static policies. This architecture based on modularity is applicable in real-time classification of behaviors and adaptation of the action to the variable terrains and dynamics of input by the user.

The hybrid training model, more specifically, the CNN+LSTM architecture, provided a solid basis to train to learn dependencies in both space and time on Unity-type tasks that enhance the accuracy of decisions and awareness of the context. The study may

be extended by implementing such model on emotion-conscious agents, adding sentiment analysis to the input dialogue and achieving multi-agent cooperation without having any centralised learning. Also, testing in larger and more interactive Unity worlds further should assist in refining the policy generalization as well as the agent collaboration mechanisms in cognitive simulations.

VIII. ACKNOWLEDGEMENT

This work was carried out in Extended Reality Center of Education and Research, RVCE, sponsored by Vision Group on Science and Technology (VGST), Govt. of Karnataka, under the K-FIST Level 2 scheme GRD No 1125.

REFERENCES

[1] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in Computer Vision and Pattern Recognition, arXiv preprint arXiv:1804.02767, April 2018, pp. 1-6. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed: January 15, 2024].

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," in Artificial Intelligence, Nature, vol. 518, no. 7540, February 2015, pp. 529-533.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, July 2017, pp. 2117-2125.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in Reinforcement Learning, arXiv preprint arXiv:1707.06347, July 2017, pp. 1-12. [Online]. Available: https://arxiv.org/abs/1707.06347. [Accessed: January 15, 2024].

[5] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in Swarm Intelligence, Ant Colony Optimization. Cambridge, MA: MIT Press, 2019, ch. 6, pp. 183-217.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, April 2017, pp. 1273-1282.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems 28, December 2015, pp. 91-99.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in Computer Vision - ECCV 2016. Cham: Springer, October 2016, pp. 21-37.

[9] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2020, pp. 10781-10790.

[10] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, February 2016, pp. 2094-2100.

[11] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in Reinforcement Learning, arXiv preprint arXiv:1511.05952, November 2015, pp. 1-21. [Online]. Available: https://arxiv.org/abs/1511.05952. [Accessed: January 15, 2024].

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems 27, December 2014, pp. 2672-2680.

[13] A. Kumar, Z. Zhou, S. Godil, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in Advances in Neural Information Processing Systems 33, December 2020, pp. 1179-1191.

[14] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, September 2004, pp. 2149-2154.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T.

Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in ICRA Workshop on Open-Source Software, vol. 3, no. 3.2, May 2009, p. 5.

[16] G. Bradski, "The OpenCV Library," in Computer Vision, Dr. Dobb's Journal of Software Tools, vol. 25, no. 11, November 2000, pp. 120-126.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32, December 2019, pp. 8026-8037.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," in Machine Learning Systems, arXiv preprint arXiv:1603.04467, March 2016, pp. 1-19. [Online]. Available: https://arxiv.org/abs/1603.04467. [Accessed: January 15, 2024].

[19] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, July 1987, pp. 25-34.

[20] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems," in Collective Intelligence, New York: Oxford University Press, 1999, ch. 3, pp. 57-102.

[21] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," in Internet of Things, IEEE Internet of Things Journal, vol. 3, no. 5, October 2016, pp. 637-646.

[22] M. Satyanarayanan, "The emergence of edge computing," in Distributed Systems, Computer, vol. 50, no. 1, January 2017, pp. 30-39.

[23] J. Chen and X. Ran, "Deep learning with edge computing: A review," in Edge AI, Proceedings of the IEEE, vol. 107, no. 8, August 2019, pp. 1655-1674.

[24] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," in Distributed Machine Learning, IEEE Signal Processing Magazine, vol. 37, no. 3, May 2020, pp. 50-60.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," in Machine Learning, Nature, vol. 521, no. 7553, May 2015, pp. 436-444. doi: 10.1038/nature14539.