Bullet Dodge: An Environment For Testing The Impact Of Different Neural Architectures

S. A. M. Zahin Abdal Shanila Nehlin Fahrin Hossain Sunaira
Department of Electrical and Computer Engineering, North South University, Dhaka, 1229, Bangladesh

***Abstract*—This study investigates the learning capabilities of an agent in the dynamic and challenging environment of "Bullet Dodge", where the agent must simultaneously avoid incoming bullets and destroy a target. The primary objectives are to assess how varying environmental complexities, rule modifications, reward structures, and the agent's speed influence learned policies. Furthermore, the impact of different neural network architectures, including variations in the number of neurons and layers, on training stability and reward maximization is explored.**

Introduction

Reinforcement learning (RL) is the branch of machine learning that is concerned with making sequences of decisions [1]. RL has a rich mathematical theory and has found a variety of practical applications [2]. Thereby, in the realm of reinforcement learning, the pursuit of training agents to navigate dynamic and challenging environments has become a focal point of research and innovation [3].

This study delves into the intricate landscape of agent learning within the confines of "Bullet Dodge", an environment that demands a delicate balance between evasive maneuvers from the bullets by the agent and strategic target destruction also performed by the agent in return. The essence of the research lies in unraveling the nuanced interplay between environmental complexities, rule modifications, reward structures, and the velocity at which the agent operates, all of which converge to shape the learned policies.

The core objective is to gain a comprehensive understanding of how agents adapt to diverse environmental intricacies, particularly in the context of dodging bullets and annihilating a target simultaneously. The environment introduces an element of unpredictability by incorporating randomness in the placement of obstacles in the forms of walls, which are a different set for each level, compelling the agent to cultivate an innate ability to navigate the perilous landscape while staying true to its mission. An in-depth exploration into various neural network architectures is undertaken, delving into the influence of factors such as the number of neurons and layers on training stability and the maximization of rewards [4].

The incorporation of speed, environmental rules, and neural network architectures adds a layer of depth to the investigation, opening avenues for refining training methodologies and advancing our understanding of how agents navigate complex, real-world scenarios [5]. Thus, we can say that the significance of this research extends beyond the confines of "Bullet Dodge", reaching into the broader landscape of reinforcement learning.

Code can be found at https://github.com/ZahinSam1/CSE_465.git

Related Works

Our research focuses on developing a simple reinforcement learning agent within a dynamic and challenging environment. Reinforcement Learning (RL) is a learning paradigm concerned with learning to control a system so as to maximize an objective over the long term [6]. Through extensive training spanning thousands of episodes, the agent exhibits a growing proficiency in solving complex game tasks. This work highlights the emergence of diverse representations of its assigned tasks, all achieved through learning with indirect supervision from a reward function. Numerous studies have delved into the realm of reinforcement learning agents, exploring their capabilities in dynamic and challenging scenarios. Works such as [7], [8] and [9] have paved the way for understanding how agents adapt and evolve over training episodes. Our emphasis on the automatic emergence of task-specific representations aligns with recent advancements in representation learning. Studies by [10] provide insights into how agents can autonomously develop representations that capture the intricacies of their assigned tasks. The utilization of indirect supervision from a reward function mirrors approaches in works like [11], showcasing the efficacy of guiding agents through learning trajectories rather than explicit instruction. In summary, our work not only contributes to the ongoing discourse on reinforcement learning in dynamic environments but also underscores the significance of task-specific representation learning and the influence of indirect supervision from reward functions.

Bullet Dodge

Fig: 1 and Fig: 2 represents a 2D reinforcement learning setting featuring an RL agent, a turret, and obstacles in the form of walls. The agent's objective is straightforward: dodge incoming bullets, leverage the surroundings for strategic advantages, and reach the endpoint — a shooting turret. Success is achieved when the agent successfully dodges bullets and reaches the turret, marking a win and triggering a level reset. This concise definition sets the stage for developing a targeted reinforcement learning approach tailored to the agent's core tasks.

For this experiment we used an NVDIA RTX 2060 GPU.

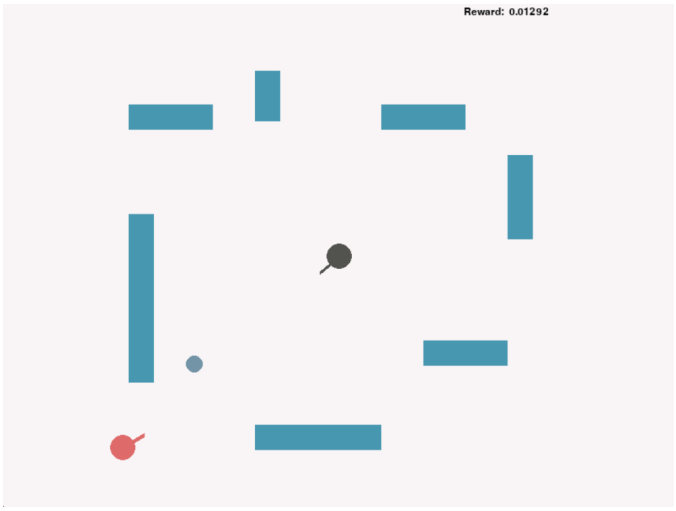

Fig. 1: Agent training in Angular movement rules

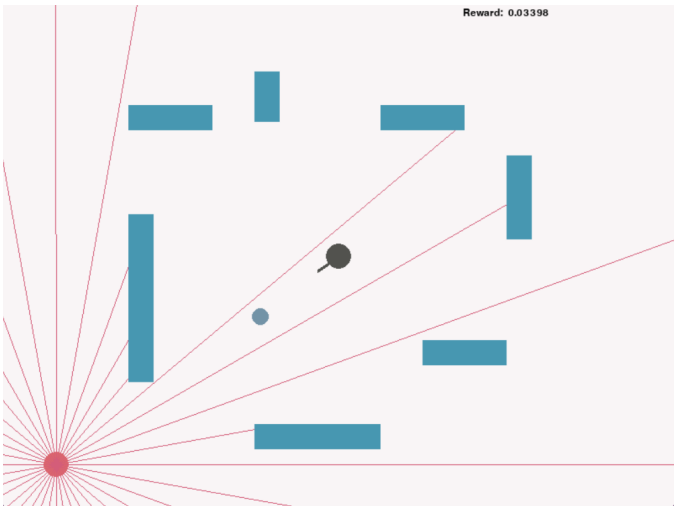

Fig. 2: Agent training in Directional movement rules with ray castings

Environment Setup

The environment plays a pivotal role in training a reinforcement learning agent, serving as the virtual space where the agent learns and makes decisions. In the context of a "Bullet Dodge", agent facing a turret and walls, the environment must accurately capture the dynamics of dodging bullets. This involves setting up the physics of bullet trajectories, defining the agent's starting position, incorporating a turret that fires bullets one at a time until it is destroyed, and establishing the rules governing movement and interactions with walls. A well-designed environment should be realistic enough to provide meaningful training scenarios for the agent, considering the turret's firing patterns, the presence of walls, and the need for the agent to navigate strategically [12].

Algorithm Selection

For this specific task, we initially employed Deep Q Networks [13] (DQN) as our primary reinforcement learning algorithm. DQN, known for its effectiveness in handling complex state spaces, provided a solid foundation for training the agent to strategically navigate and engage the shooting turret. The inherent nature of a Markov decision process [14] allowed the agent to make decisions based solely on its current state, optimizing its actions for the immediate task at hand. To explore potential improvements [15] [10], we also experimented with Proximal Policy Optimization (PPO) during the training phase. PPO [16] is recognized for its stability and sample efficiency, making it a compelling alternative for tasks with continuous action spaces. Upon thorough evaluation, we found that DQN served the purpose more effectively, offering a robust solution for the agent to learn optimal strategies for dodging bullets and reaching the turret.

State Representation

- Agent's Position: By including the agent's position(co-ordinates x,y), the model can understand its location within the environment.
- Bullet Position: Incorporating the center position(co-ordinates x,y) of incoming bullet allows the agent to dynamically respond to the immediate threat, predicting trajectories and optimizing evasion strategies.
- Turret Position: The turret's position(co-ordinates x,y) is a critical element, guiding the agent towards the ultimate goal.
- Ray Casts for Wall Distances: Utilizing ray casts to measure distances from walls enhances the agent's spatial awareness(a total of 36 lengths).

All these state elements are meticulously crafted with a data type of float32, ensuring numerical precision suitable for reinforcement learning algorithms. Once gathered, these states are flattened into a list and transmitted to the model.

Action Space

The action space is tailored to the specific demands of the bullet-dodging environment, featuring distinct sets of actions for different scenarios:

Environment 1:

- Left: Move the agent to the left.
- Right: Move the agent to the right.
- Up: Move the agent upward.
- Down: Move the agent downward.

Environment 2:

- Forward: Move the agent forward.
- Clockwise Rotate: Rotate the agent in a clockwise direction.

- Anti-clockwise Rotate: Rotate the agent in an anti-clockwise direction.

In this experiment, our focus remained exclusively on Environment 1. The decision to prioritize Environment 1 over Environment 2 was driven by the increased complexity inherent in angular movement. The rich variability of angular motion posed a more intricate challenge for the agent, demanding a higher level of adaptability and nuanced decision-making [17]. However, it's important to note that the extensive complexity of Environment 1 resulted in prolonged training times. Regrettably, due to hardware limitations, we were unable to extend our experimentation to Environment 2.

Reward Design

- Movement Points: Positive rewards are assigned for each successful movement, encouraging the agent to explore and navigate effectively within the environment. The reward given here is 0.01 points.

- Progress Towards Turret: Increasing rewards are granted as the agent makes progress towards the turret, reinforcing the importance of advancing towards the ultimate goal. The reward given here is:

$$Reward = \frac{10}{\text{distance between the agent and the goal}}$$

- Turret Engagement Bonus: A substantial reward is given upon reaching and engaging the turret, amplifying the significance of the primary objective. The reward here is 200 points.

- Penalty for Bullet Hit: Negative penalties are imposed for each instance of a bullet hitting the agent, discouraging sub-optimal defensive strategies. For every bullet hit, the agent is penalized by 50 points.

- Time Expiry Penalty: Negative penalties are applied when time expires without reaching the turret, encouraging the agent to act efficiently. By imposing a slightly lower penalty for time expiration, the agent is encouraged to focus on learning the fundamental aspects of its task without the added urgency. This optimization approach acknowledges that a gentler time penalty provides the agent with a conducive learning environment, allowing it to explore and understand the complexities of the simulated scenario more thoroughly [18].

Iterative adjustments to these reward components was essential for fine-tuning the learning process. Striking the right balance ensuring the agent learns to navigate strategically, prioritize turret engagement, and effectively evade bullet.

Implementation Details

- Python: The core programming language utilized for its versatility and extensive libraries, facilitating efficient development and integration.
- Pygame: Employed for graphical rendering and interactive visualization of the bullet-dodging environment. Pygame provides a convenient framework for creating game-like simulations.
- Stable Baselines: Leveraged for reinforcement learning algorithms, offering a set of high-quality implementations. The choice of Stable Baselines ensures access to a variety of algorithms, simplifying experimentation and comparison [19].

One of the training is done on the default network architecture (layers: 2, perceptrons per layer: 64, learning rate: 0.0001) and the other for comparison (layers: 4, perceptrons per layer: 256, learning rate: 0.003). Each of them has been trained for about 16 million time steps.

EXPERIMENTS

The goal of our experiments is to analyze the learned capabilities of the policies, exhibit what behaviors are emerged, and characterize why different strategies emerge in the simulated environment, and also to figure out the differences in learning for different network architectures. Overall, the agent in this simulated environment learns to traverse the environment, use the pre-existing objects to it's advantage, and learn new policies such as dodge, wait and going around to reach it's ultimate goal.

Does the agent learn to solve it's training task?

Our initial evaluation focuses on the model's proficiency in its training task: avoiding bullet objects. Impressively, the agent demonstrated successful task completion after 16 million timesteps, achieving an average episode length of 129. Notably, the agent attained a peak score of 101, showcasing its adeptness in mastering the core objectives of the task. However, it's crucial to acknowledge the subsequent dip in the score, a phenomenon attributed to the agent's exploration of new strategies. This intentional exploration, marked by a willingness to deviate from initially learned patterns, is a pivotal aspect of reinforcement learning. While the score temporarily regressed, this phase of exploration is indicative of the agent's adaptability and its endeavor to uncover more effective strategies for navigating the task environment. This dynamic interplay between task mastery and exploration underscores the nuanced learning process, where the agent continually refines its understanding and refashions its approach. In the initial phases, the agent embarked on a journey of observation, meticulously scrutinizing the surrounding environment to decipher the cause-and-effect relationship between its actions and changes in the environment. As the learning process unfolded, the agent, driven by an innate capacity for adaptability, honed a defensive strategy. It initially sought refuge behind walls, leveraging them as shields to dodge incoming

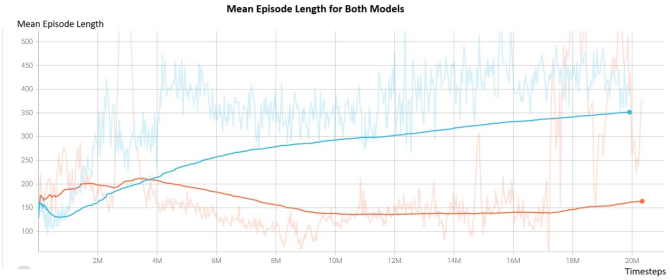

Fig. 3: Average episode length during training(smoothed)

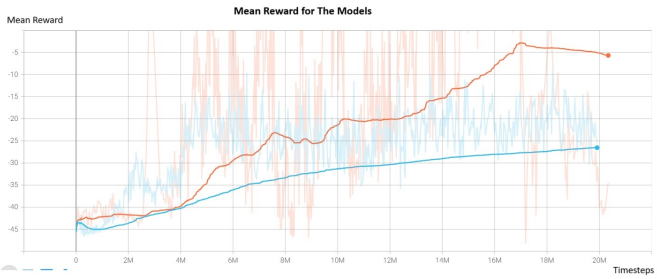

Fig. 4: Average reward achieved by the agent every episode(smoothed)
Blue(layer: 4, perceptrons: 256, lr: 0.003)
Orange(layer: 2, perceptrons: 64, lr: 0.0001)

bullets. An intriguing turning point emerged when the agent began discerning not just the immediate threat but also the subtleties of bullet trajectories. Over time, it acquired an understanding of the spatial dimensions, precisely gauging the radius of both the bullets and its own presence. However, this refined strategy faced challenges, with a period of suboptimal performance lasting for an extended duration. The learning process, characterized by trial and error, persevered. Notably, at the 14-million-step mark. This pivotal adjustment marked a breakthrough, leading to a remarkable achievement: a high score of 101 within a cumulative 17 million steps. During the evaluation phase fig: 6, the scores ranged from 60 to 68, with an average of 60∼68 $\pm$ 5.7 points. The observed variation in scores is attributed to a limited number of training steps. Notably, both models indicate that optimizing hyperparameters can address this issue, leading to more efficient and improved results.

The two graphs from Fig: 3 and Fig: 4 comparing Mean Episodic Training Reward against the Number of Timesteps provide valuable insights into the performance of two training models in our reinforcement learning scenario. The first model colored **orange** represents the architecture with 2 convolution layers with 64 neurons each and a modest learning rate of 0.0001. In contrast, the second model colored **blue** corresponds to the architecture of 4 convolution layers boasting 256 neurons each, and a higher learning rate of 0.003.

The first model(**orange**) exhibits higher fluctuations. This trend suggests a higher degree of instability in the learning process, and the modest learning rate implies a slower convergence over time. The **orange** configuration appears to compromise the speed of learning, raising concerns about its adaptability and efficiency in dynamic environments. The sensitivity of the **orange** line to variations in episodic training reward indicates that the model with 2 neurons may struggle with convergence and be susceptible to oscillations during training. That is clear in the Fig: 4.

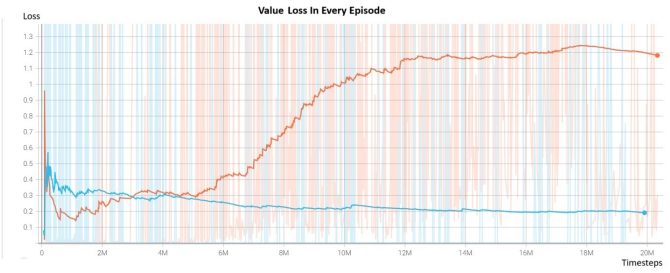

Fig. 5: Agent training in Directional movement rules with ray castings(smoothed)

Contrarily, the second model(**blue**) line's episodic training rewards suggests not only stability but also a faster convergence rate. Despite the higher learning rate, the **blue** configuration maintains a stable trajectory, indicating a successful balance between speed and steadiness in the learning process. This observation positions the blue configuration as a more effective and efficient model in our reinforcement learning scenario, which is clearly described by the Fig: 4.

The second model(**blue**) achieved comparable average steps per day with a shorter training duration compared to the first one(**orange**). Despite the **orange** model taking more total steps, the time-normalized analysis revealed the **blue** model's efficiency in achieving similar daily progress. This indicates that the **blue** model exhibited a swifter learning process, achieving outcomes within a slightly shorter time frame, highlighting its efficiency in converging towards an optimal solution Fig: 4.

Even in terms of value loss, it is a proof that the **blue** is more stable than the **orange** model Fig: 5.

Impact of different network architectures

In our research, we introduced a deliberate contrast in neural architectures within the DQN algorithm, exemplified by a second agent with a distinct structure. The alternative architecture featured four neural layers, each comprising 256 neurons. This intentional diversification provided valuable insights into the impact of architectural nuances on various facets of the learning process. The four-layer architecture showcased a remarkable proficiency in quickly mastering the primary task goal. Throughout the intermediate phase of training, this architecture demonstrated an ability to sustain a consistent reward level. This stability indicates a robust adherence to initially learned strategies, suggesting that the agent maintained a reliable performance in executing the core task over extended periods Fig. [3, 4, 5].

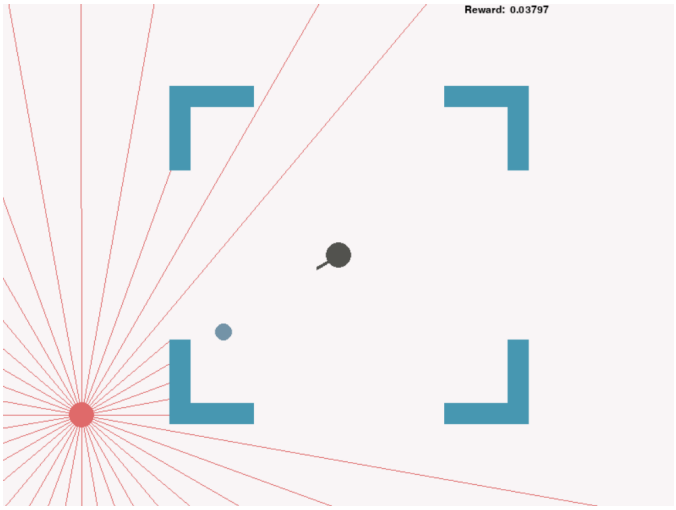

Fig. 6: Agent's testing Environment(Evaluation)

Discussion and Future Work

Our experiments illuminate the adaptability and versatility of reinforcement learning, unveiling a spectrum of strategies within our simulated environment. The current framework, centered around a single agent and entity, has yielded valuable insights into adaptive capabilities. However, our vision transcends these achievements, charting a course for future work and enhancements. In forthcoming iterations, we envisage a pivotal shift with the introduction of multiple agents. This evolution injects a new layer of complexity into the environment, unveiling interactions that range from cooperation to competition. Understanding the dynamics of collaborative and adversarial learning becomes a focal point, unraveling intricate patterns in agent behaviors and strategic decision-making.

As part of our future enhancements, we are exploring the integration of attention mechanisms into our algorithm. Attention mechanisms have proven instrumental in capturing and emphasizing relevant information during the learning process [20]. By incorporating attention mechanisms, we anticipate an enriched ability for the agent to focus on salient features within the environment [21], potentially enhancing its decision-making and adaptability.

Conclusion

In conclusion, our exploration into training a reinforcement learning agent for the intricate task of bullet dodging has provided valuable insights into the interplay of environment, algorithm selection, state representation, action space, reward design, and implementation details. The meticulously crafted environment, featuring dynamic bullet trajectories, spatial awareness and strategic turret engagement, served as a challenging arena for our agent to learn optimal strategies.

Our comparative analysis of reinforcement learning algorithms, specifically favoring Deep Q Networks (DQN) for this task, underscores the importance of algorithmic choices in achieving robust learning outcomes.

This study not only sheds light on effective strategies for bullet dodging but also provides a foundation for future advancements in training agents for complex, real-world tasks.

References

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[2] D. P. Bertsekas, "Dynamic programming and optimal control," *Journal of the Operational Research Society*, vol. 47, no. 6, pp. 833–833, 1996.

[3] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

[4] A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Yazdanbakhsh, and H. Esmaeilzadeh, "Releq: A reinforcement learning approach for deep quantization of neural networks," *arXiv preprint arXiv:1811.01704*, 2018.

[5] A. Iranfar, M. Zapater, and D. Atienza, "Multiagent reinforcement learning for hyperparameter optimization of convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1034–1047, 2021.

[6] S. Kapoor, "Multi-agent reinforcement learning: A report on challenges and approaches," *arXiv preprint arXiv:1807.09427*, 2018.

[7] S. Zhang and X. Zhuan, "Two-dimensional car-following control strategy for electric vehicle based on mpc and dqn," *Symmetry*, 2022.

[8] D. S. Volodymyr Mnih, Koray Kavukcuoglu, D. W. Alex Graves, Ioannis Antonoglou, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[9] S. Jagtap, A. Thomas, S. Gadiwan, and S. Mishra, "Multi-agent reinforcement learning-implementation of hide and seek," pp. 1–7, 2022.

[10] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee, "Diversity-driven exploration strategy for deep reinforcement learning," 2018.

[11] A. Karalakou, D. Troullinos, G. Chalkiadakis, and M. Papageorgiou, "Deep reinforcement learning reward function design for autonomous driving in lane-free traffic," *Systems*, 2023.

[12] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Transactions on Games*, pp. 400–412, 2018.

[13] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[14] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms," *Management Science 28(1):1-16*, 1982.

[15] P. Rodrigues and S. Vieira, "Optimizing agent training with deep q-learning on a self-driving reinforcement learning environment," pp. 745–752, 2020.

[16] P. D. A. R. O. K. John Schulman, Filip Wolski, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[17] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine, "Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability," 2021.

[18] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," pp. 4045–4054, 2018.

[19] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," vol. 22, 2021.

[20] L. S. P. R. M. M. Wang, Heng, "Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning," 2019.

[21] H. Itaya, T. Hirakawa, T. Yamashita, H. Fujiyoshi, and K. Sugiura, "Visual explanation using attention mechanism in actor-critic-based deep reinforcement learning," pp. 1–10, 2021.