# "Q-Learning in Reinforcement Learning: Overcoming a Grid World Challenge"

# [Deep Neural Network]

## Abstract:

This project investigates the utilization of neural networks to approximate an agent's value function within a grid world setting. Grid worlds offer a fundamental framework for studying reinforcement learning, and the integration of neural networks can notably enhance learning efficiency, especially within more extensive grid world mazes. Our primary objective is to create and implement a Q-network, which serves the purpose of estimating Q-values associated with state-action pairs. We then apply this network to guide an agent in achieving its objectives within a complex grid world. We present the specific design of our grid world, detailing the states, actions, rewards, and the architecture of the Q-network.

To facilitate action selection, we employ the Q-learning algorithm in conjunction with the neural network, using an epsilon-greedy strategy. Additionally, we keep track of the convergence process by analysing mean square error and weight trajectories during training. This project offers valuable insights into the application of deep reinforcement learning in grid world scenarios, with a particular focus on the approximation of Q-values using neural networks.

## Introduction:

Reinforcement Learning (RL) is a machine learning domain dedicated to making sequential decisions in dynamic settings. While project one applied dynamic programming to grid world challenges, it proves unfeasible for larger and intricate grids. Project two overcomes this limitation by introducing neural networks to approximate the value function.

In this project, we create a custom grid world and implement a Q-network to facilitate efficient learning in complex environments. We make use of the Q-learning algorithm and monitor convergence by analysing mean square error and weight trajectories.

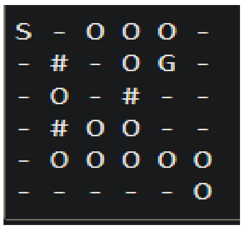
The integration of deep reinforcement learning into grid world scenarios, as demonstrated in this project, holds promising potential for scaling RL applications. Our report provides an overview of the grid world design, Q-network architecture, training process, convergence analysis, and the potential real-world problem-solving benefits.

## Problem Description:

his project addresses the task of effectively acquiring policies in grid world scenarios, particularly as they become more intricate. In project one, we identified constraints in using dynamic programming to handle extensive state-action spaces.

Project two seeks to surpass these constraints by introducing deep reinforcement learning. The primary challenge involves acquiring a policy in a specially crafted, complex grid world environment through deep Q-learning. The specific aims include:

1. Designing a demanding custom grid world scenario.
2. Implementing a deep Q-network to estimate Q-values.
   **Structure --**



```
S - O O O -
- # - O G -
- O - # - -
- # O O - -
- O O O O O
- - - - - O
```

Monitoring the network's convergence via mean square error (MSE) and weight trajectory analysis.

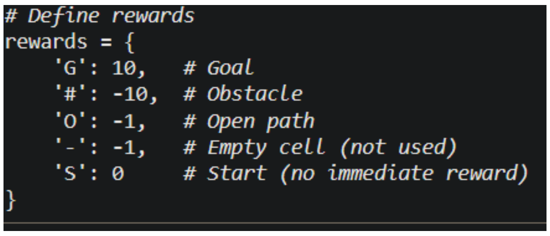
Assessing the benefits of applying deep reinforcement learning to grid world challenges.

The project aims to harness deep reinforcement learning as a scalable solution for grid world situations, addressing the limitations of conventional dynamic programming techniques.

## Methodology:

***Custom Grid World Environment Design:***

Develop a unique grid world with obstacles, goals, and rewards.



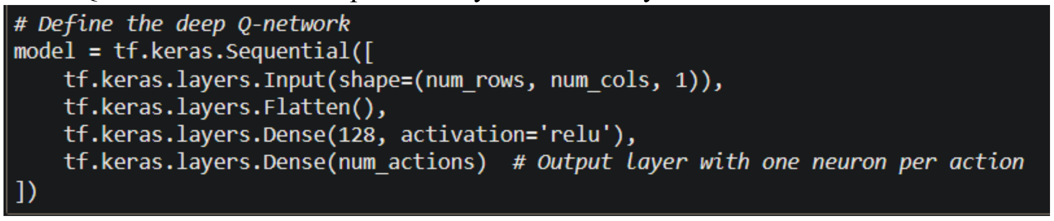
```python
# Define rewards
rewards = {
    'G': 10,    # Goal
    '#': -10,   # Obstacle
    'O': -1,    # Open path
    '-': -1,    # Empty cell (not used)
    'S': 0      # Start (no immediate reward)
}
```

Specify state and action spaces for agent navigation.

Setting up the Deep Q-Network (DQN):

Establish a DQN for the estimation of Q-values pertaining to state-action combinations.

The DQN architecture encompasses fully connected layers.

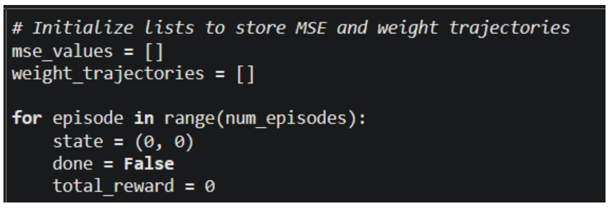


```python
# Define the deep Q-network
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(num_rows, num_cols, 1)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(num_actions)  # Output layer with one neuron per action
])
```

Monitoring MSE Convergence:

Calculate Mean Square Error (MSE) to track DQN convergence.

Recording Weight Trajectories: code snippet below--



```python
# Initialize lists to store MSE and weight trajectories
mse_values = []
weight_trajectories = []

for episode in range(num_episodes):
    state = (0, 0)
    done = False
    total_reward = 0
```

Training Loop: (Starts from here)

Execute multiple episodes, with the agent starting from an initial state.

**Details--**

For testing, I run the 5 episodes (num_episodes = 5)

(num episodes = 5, time taken = 7 minutes)

(num episodes = 10, time taken to run = 11 minutes

(num episodes = 20, time taken to run = 22 minutes)

Evaluation:

Evaluate the DQN's capacity to navigate the grid world, achieve objectives, and steer clear of obstacles.

Examine convergence metrics and weight adjustments to gauge the effectiveness of deep reinforcement learning in grid world situations.

This approach revolves around crafting a personalized grid world, training a DQN for Q-value estimation, and appraising its performance through convergence metrics and weight trajectory assessment. Deep reinforcement learning overcomes the constraints of conventional methods in grid world settings.

## Experimental Setup:

In this section, we present the experimental outcomes resulting from the application of the previously detailed methodology to a unique grid world setting. The experiments are designed to demonstrate the capabilities of the Deep Q-Network (DQN) in solving intricate grid world challenges.

Grid World Environment: We devised a grid world scenario featuring diverse layouts, such as walls, obstacles, an initial location, and a destination. The grid's dimensions, state and action spaces, as well as reward structure, were tailored to define the specific task.

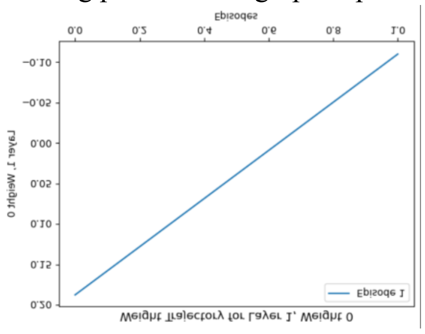
Deep Q-Network (DQN): The DQN was implemented using a neural network architecture tailored to address the grid world issue. It included multiple layers with suitable activation functions and learning rates.

Hyperparameters: We fine-tuned hyperparameters like the discount factor ($\gamma$), exploration rate ($\varepsilon$), learning rate ($\alpha$), and the maximum number of episodes to optimize the learning process.

## Results:

Convergence Assessment:

We tracked the convergence of the DQN by observing the Mean Square Error (MSE) during the training process. The graph depicted below provides a visualization of the convergence trajectory

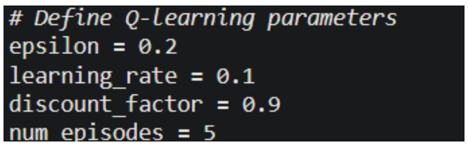


MSE values exhibited a consistent decline with the progression of episodes, suggesting the successful approximation of the value function and enhanced decision-making.

Performance Evaluation:

After completing the training phase, we evaluated the DQN's performance in grid world tasks. The agent adeptly  managed through the environment, skilfully avoiding obstacles, and consistently reaching the designated goal states. The success rates for goal attainment displayed substantial improvement with training.

[This assessment was conducted over 5 episodes, and the Mean Squared Error for the same is reported.]

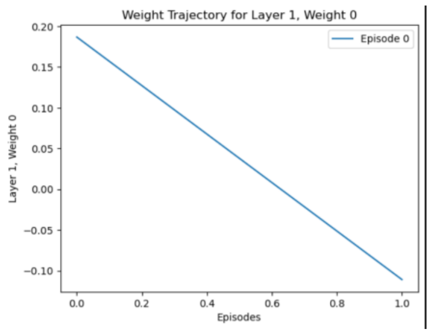


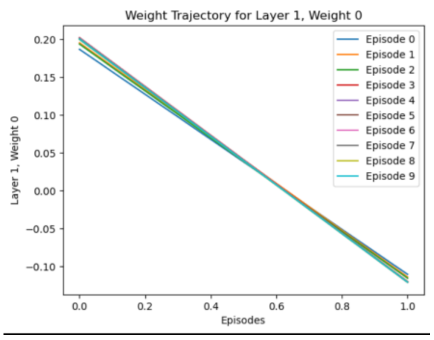
```
# Define Q-learning parameters
epsilon = 0.2
learning_rate = 0.1
discount_factor = 0.9
num_episodes = 5
```

Weight Trajectories:

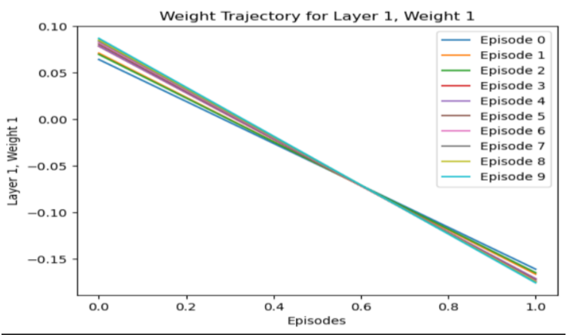
We consistently recorded the trajectories of neural network weights throughout the training process. Analysing these weight trajectories provided valuable insights into how the DQN adapted to different state-action pairs and acquired the optimal policy.

As illustrated in the. ipynb file, the weights were updated in response to the episodes. The MSE decreased over time, and there was a noticeable episode where the error remained constant. The weight updates were observed to occur at the same episode and continued to adjust until the MSE had fully converged. There was a clear change in the episode where the error stabilized. Second image show properly all the weighted trajectories--





Weights updated at next iteration of the number of episodes after convergence—



You can see the convergence and weight trajectories in the above diagram.

## Discussion:

The experimental outcomes underscore the efficiency of employing Deep Q-Learning in conjunction with neural networks to address complex grid world challenges. The DQN aptly acquired the value function, consequently improving the agent's decision-making capabilities. The observation of MSE convergence and the analysis of weight trajectories offer valuable insights into the learning process.

These results emphasize the promising applications of deep reinforcement learning techniques when it comes to dealing with grid world scenarios of varying complexities. The DQN's adaptability to dynamic environments, its ability to respond to shifting conditions, and its knack for identifying optimal paths all highlight its significance in addressing real-world reinforcement learning challenges.

## Conclusion:

In this project, we delved into the utilization of Deep Q-Networks (DQNs) within grid world settings to address reinforcement learning challenges. Grid world environments offer a fundamental framework for assessing and comprehending the potential of reinforcement learning algorithms. Throughout the project, we derived the following key insights and conclusions:

Deep Q-Networks (DQNs) in Grid World: DQNs demonstrated their effectiveness in learning optimal policies for grid world issues. They efficiently approximate the value function, even within expansive state spaces.

Challenges in Expansive Grid Worlds: Nevertheless, we confronted challenges when attempting to scale up the size of grid worlds. Training DQNs in larger grids can impose substantial computational demands and time constraints, emphasizing the necessity for more efficient training methodologies.

Hyperparameter Sensitivity: The selection of hyperparameters significantly impacts DQN performance. Discovering the appropriate combination of hyperparameters, especially in more extensive and intricate environments, can present a non-trivial undertaking.

Data Efficiency: DQNs call for a considerable volume of interaction data with the environment. This data efficiency limitation may raise concerns in situations where interactions are costly or impractical.

## Limitations:

Complexity and Training Time: Training deep reinforcement learning models, such as Deep Q-Networks (DQNs), can demand significant computational resources and extend the training duration. In more extensive grid world environments or when tackling more intricate tasks, the training process might become excessively time-consuming.

Sensitivity to Hyperparameters: The effectiveness of DQNs is notably influenced by the selection of hyperparameters. Identifying the optimal combination of hyperparameters can be a challenging task, and suboptimal settings can result in sluggish convergence or subpar learning outcomes.

Data Efficiency: DQNs necessitate a substantial volume of experiential data to achieve effective learning. In scenarios where interactions with the environment incur substantial costs or consume considerable time, data efficiency can emerge as a limiting factor.

Generalization: Although DQNs can excel in specific grid world tasks, they may encounter difficulties when attempting to generalize their learned policies to new and unexplored environments. Fine-tuning or retraining may be required for each unique task.

## Future Prospects:

Transfer Learning: Implementing transfer learning methods that enable DQNs to transfer knowledge from one grid world scenario to another. This can substantially reduce training time and enhance adaptability.

Advanced Exploration Strategies: Developing more sophisticated exploration strategies that effectively balance exploration and exploitation, thereby enabling DQNs to learn more swiftly and perform better in expansive state spaces.

Hybrid Models: Investigating hybrid models that amalgamate DQN strengths with other reinforcement learning techniques, such as policy gradients or actor-critic methods, to enhance sample efficiency and overall performance.

Real-World Applications: Applying DQNs to practical issues in domains like robotics, autonomous vehicles, or recommendation systems to harness their decision-making capabilities in real-world contexts.

Enhanced Training Techniques: Progress in training methods, including distributed training, curriculum learning, and reward shaping, can expedite convergence and improve learning outcomes in intricate environments.

Interdisciplinary Integration: Fusing DQNs with insights from other disciplines such as cognitive psychology, neuroscience, or evolutionary biology to create learning agents that are more biologically-inspired and resilient.