Q-Learning FrozenLake – Debugging & Implementation Documentation

# 🔰 Project Objective

To implement and debug a Q-Learning agent to solve the FrozenLake-v1 environment from OpenAI Gym, analyze computational bottlenecks, and explore a potential hardware accelerator implementation in SystemVerilog for the most critical operation.

# 🧠 Initial Prompt

User Request:  
"Ask your favorite LLM to identify ‘computational bottlenecks’ in the FrozenLake code from https://github.com/ronanmmurphy/Q-Learning-Algorithm..."

Goals:

1. Identify bottlenecks in Q-Learning code.  
2. Evaluate suggestions for accuracy.  
3. Propose a hardware solution to the main bottleneck.  
4. Generate SystemVerilog code for the hardware accelerator.

# 🧮 Q-Learning Overview

Q[state, action] = Q[state, action] + α \* (reward + γ \* max(Q[next\_state, :]) - Q[state, action])

# 🧩 Problems Encountered

❌ 1. ModuleNotFoundError: No module named 'gym'  
Cause: gym was not installed  
Fix: Run pip install gym in terminal

❌ 2. IndexError: only integers, slices (:)... are valid indices  
Cause: env.reset() returned a tuple  
Fix: state, \_ = env.reset()

❌ 3. AttributeError: module 'numpy' has no attribute 'bool8'  
Cause: NumPy 1.24+ removed bool8  
Fix: Downgrade NumPy or upgrade Gym

# ✅ Final Working Code

Python code implementing a working Q-learning agent and printing progress during training.

# 🔍 Bottleneck Analysis

Bottlenecks identified by LLM:  
- np.argmax(Q[state, :]) – selects best action  
- np.max(Q[new\_state, :]) – for updating Q

# ⚙️ Hardware Accelerator Proposal

Operation: Max/argmax finder across action space  
Approach: Comparator tree in SystemVerilog  
Design Goal: Take 4 Q-values, output max value + index

# 💾 SystemVerilog Code – Max Finder

module max\_finder\_4 #(  
 parameter WIDTH = 16  
)(  
 input logic [WIDTH-1:0] q0, q1, q2, q3,  
 output logic [WIDTH-1:0] max\_val,  
 output logic [1:0] max\_idx  
);  
  
 logic [WIDTH-1:0] max0, max1;  
 logic [1:0] idx0, idx1;  
  
 always\_comb begin  
 if (q0 >= q1) begin max0 = q0; idx0 = 2'd0; end  
 else begin max0 = q1; idx0 = 2'd1; end  
  
 if (q2 >= q3) begin max1 = q2; idx1 = 2'd2; end  
 else begin max1 = q3; idx1 = 2'd3; end  
  
 if (max0 >= max1) begin max\_val = max0; max\_idx = idx0; end  
 else begin max\_val = max1; max\_idx = idx1; end  
 end  
endmodule

# 🖨️ Printing and Visualization in VS Code

Track training progress using print statements and visualize output in the VS Code terminal.

# 🎯 Final Results

The agent gradually learns to maximize rewards. Training prints every 100 episodes. Final Q-table shows learned values.

# 📝 Questions Asked (Chronological)

1. Ask LLM to find bottlenecks in FrozenLake Q-learning code  
2. Validate if bottlenecks make sense  
3. Propose hardware for max/argmax operation  
4. Generate SystemVerilog for hardware  
5. How to fix ModuleNotFoundError: gym  
6. How to fix IndexError due to tuple unpacking  
7. How to fix AttributeError: numpy has no bool8  
8. How to print results in VS Code