Fiyori Demewez
Timothee Tran

Introduction:

This project explores reinforcement learning on an Atari environment using Deep Q-Learning. We chose ALE/MarioBros-v5 because it has a sparse reward structure, requires fine movement control, and shows clear performance differences between naïve DQN and improved algorithms like Double DQN. Out of curiosity, we also want to experiment with a simpler game, such as ALE/Pong-v5, to see how it performs after using a similar DQN extension implementation.

The project includes:

- A **baseline DQN** for both games
- An extended run using **Double DQN (DDQN)** at two training lengths: **205k frames** and **305k frames** for Mario.
- An extended run using **Double DQN (DDQN)** at two training lengths: **100k frames** and **50k frames** for Pong.
- The goal is to compare stability, learning quality, and gameplay behavior across these models.

Environment Description:

Environment: ALE/MarioBros-v5

- **Observation shape:** 84×84 grayscale, 4-frame stack
- **Action space:** Discrete Atari joystick (movement + jump)
- **Reward structure:** Sparse; points are only awarded for survival time, stomping enemies, and clearing platforms.
- **Key difficulty:** Delayed rewards and the need for precise timing make exploration challenging.

Environment: ALE/Pong-v5

- **Observation shape:** 84×84 grayscale, 4-frame stack
- **Action space:** Discrete Atari paddle movements (up, down, no-op)
- **Reward structure:** Sparse and binary; +1 for scoring a point, -1 for conceding a point, 0 otherwise
- **Key difficulty:** Requires learning long-term strategy and precise paddle positioning to return the ball, with rewards delayed until a point is scored

Model & Training:

Baseline DQN (Required)

The baseline architecture follows the standard Atari DQN:

- Convolutional head → Fully connected → |A (action)|
- Experience replay
- Target network
- Epsilon-greedy policy
- Adam optimizer

Double DQN Extension (Required)

For the second run, we enabled **Double DQN**, where:

- The **online network selects the action**, and
- The **target network evaluates it**.

This reduces Q-value overestimation and significantly improves stability.

Hyperparameters:

	Mario-v5	Pong-v5
Learning rate:	adam	1e-4
Optimzer:	Gamma 0.99	adam
Replay size:	100k	50,000
Batch size:	32	32
Target sync:	10k(DQN), 5k(DDQN)	Every 1,000 frames
Frame Stack:	4	4
Epsilon Schedule	1.0->0.1 over-training	Linear decay from 1.0 to 0.01 over 100,000 frames

Experimentation Log:

Change number	What we try	Why
1	Increased warmup steps 10k-50k	To stabilize DQN replay Samples early on
2	Reduced target sync for DDQN	Better stability for a large training window
3	Extended DDQN run from 205k-305k	To check whether performance recovers from local minima
4	Changed epsilon decay speed	To balance exploration vs. exploitation
5	Adjusted replay buffer size	To reduce correlation in samples

Fiyori Demewez
Timothee Tran

ALE/Mario-v5
Learning Curves:

DQN Learning Curve (200k Frames)

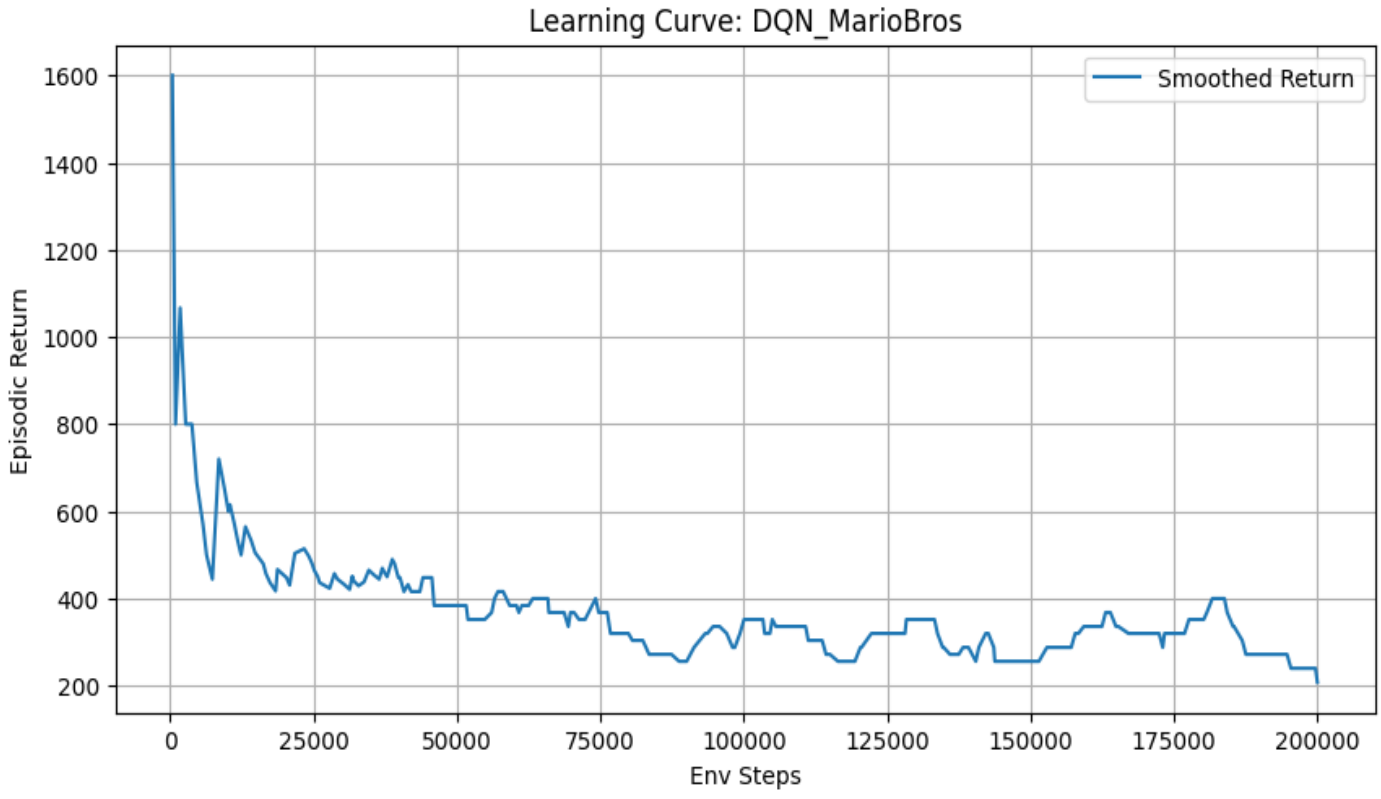
- Early spike due to random exploration
- Rough early learning (returns 450–600)
- Collapse to ~300 due to overestimation
- Final plateau at ~250–350
- Behavior: unstable, hesitant, short survival time

DDQN Learning Curve (205k Frames)

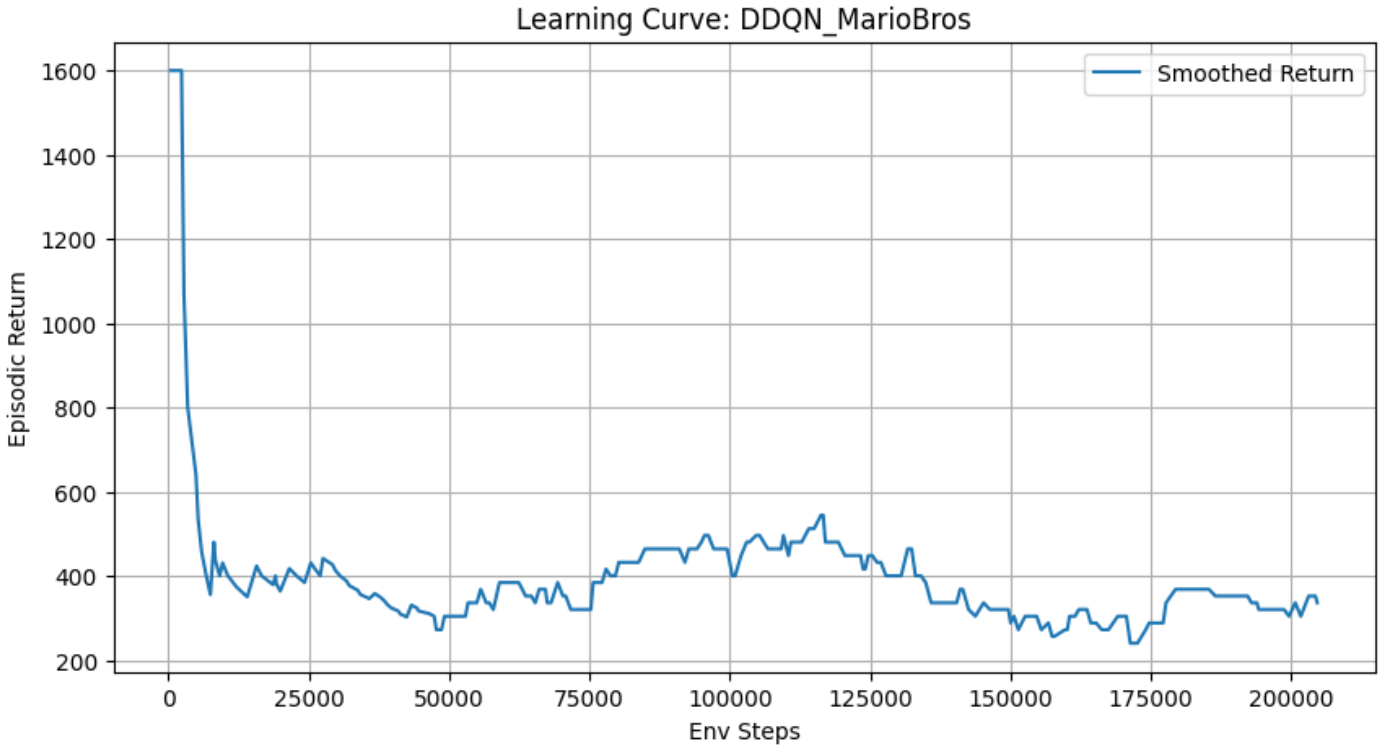
- More stable than DQN
- Still suffers mid-training oscillation
- Moderate improvement over DQN

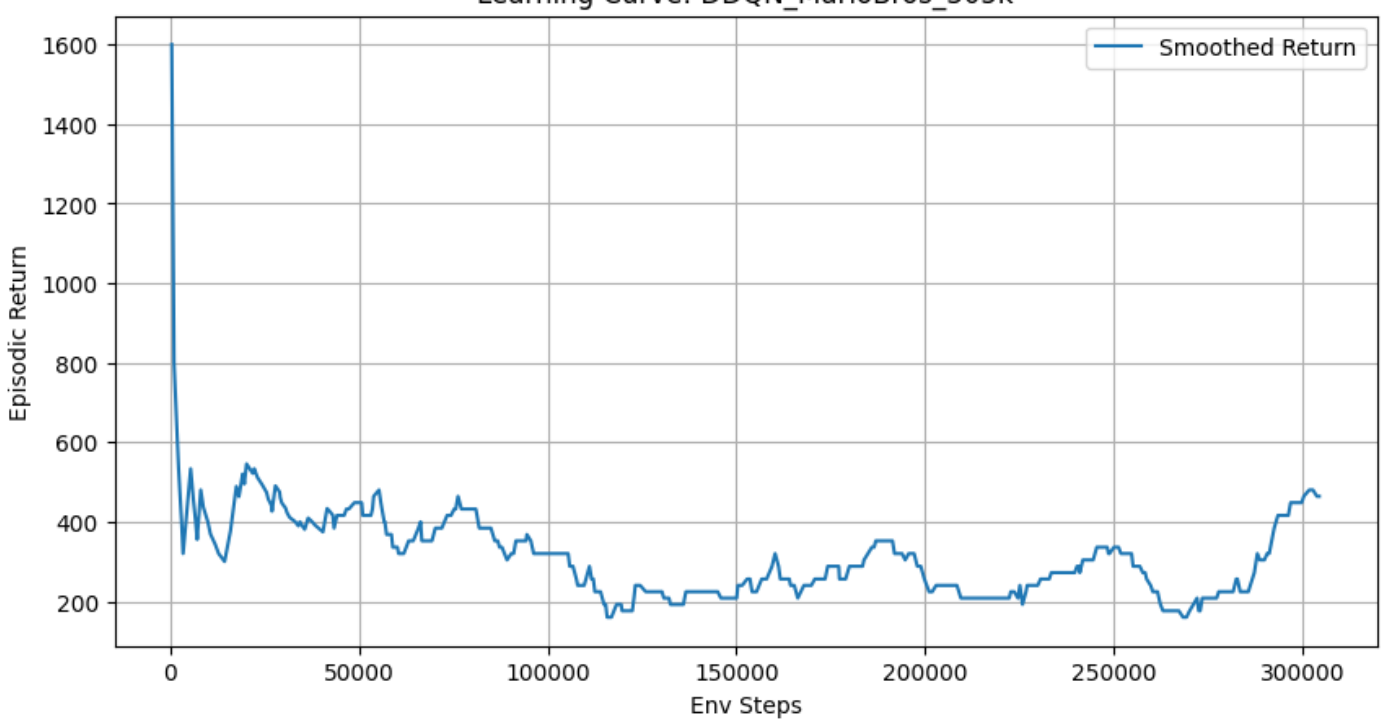
DDQN Learning Curve (305k Frames)

- Recovers after a local minimum
- Reaches ~400–450 return
- Best overall run: smooth movement + longest survival



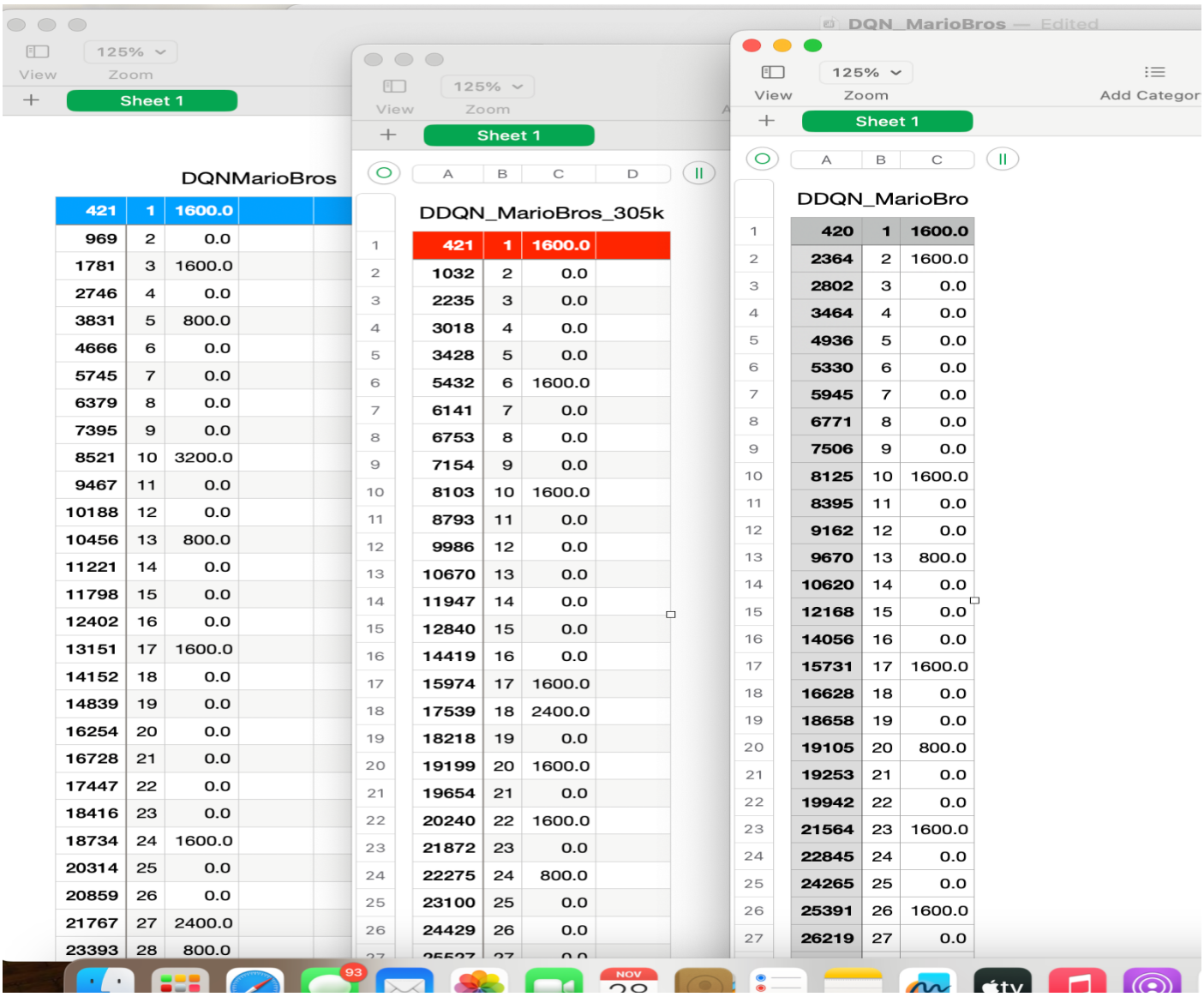
Fiyori Demewez
Timothee Tran





Fiyori Demewez

Timothee Tran



Video Evidence:

Per project requirement, we recorded two videos for DQN and DDQN:

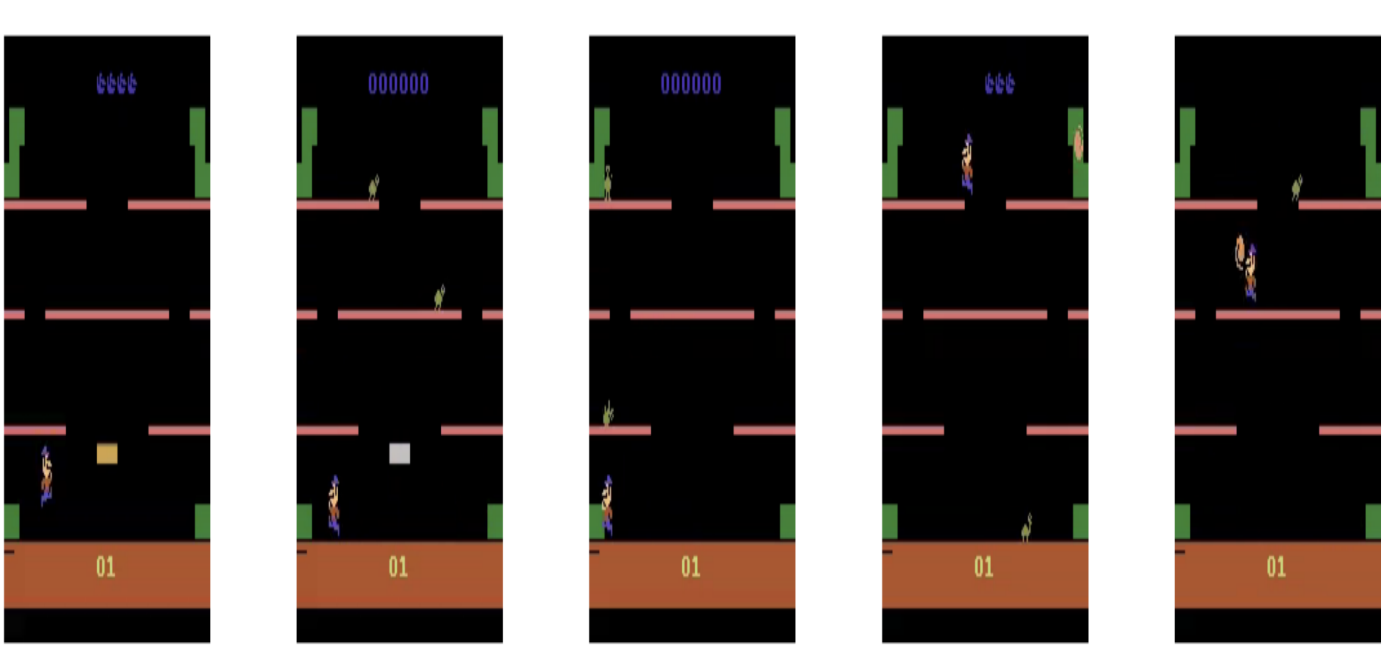
- **Early/randomish episode (ε high)**
- **Learned episode (ε low)**

Videos are in my GitHub repo:

Fiyori Demewez
Timothee Tran

Frame Comparisons:

DQN Early vs. Learned



Observations
Early

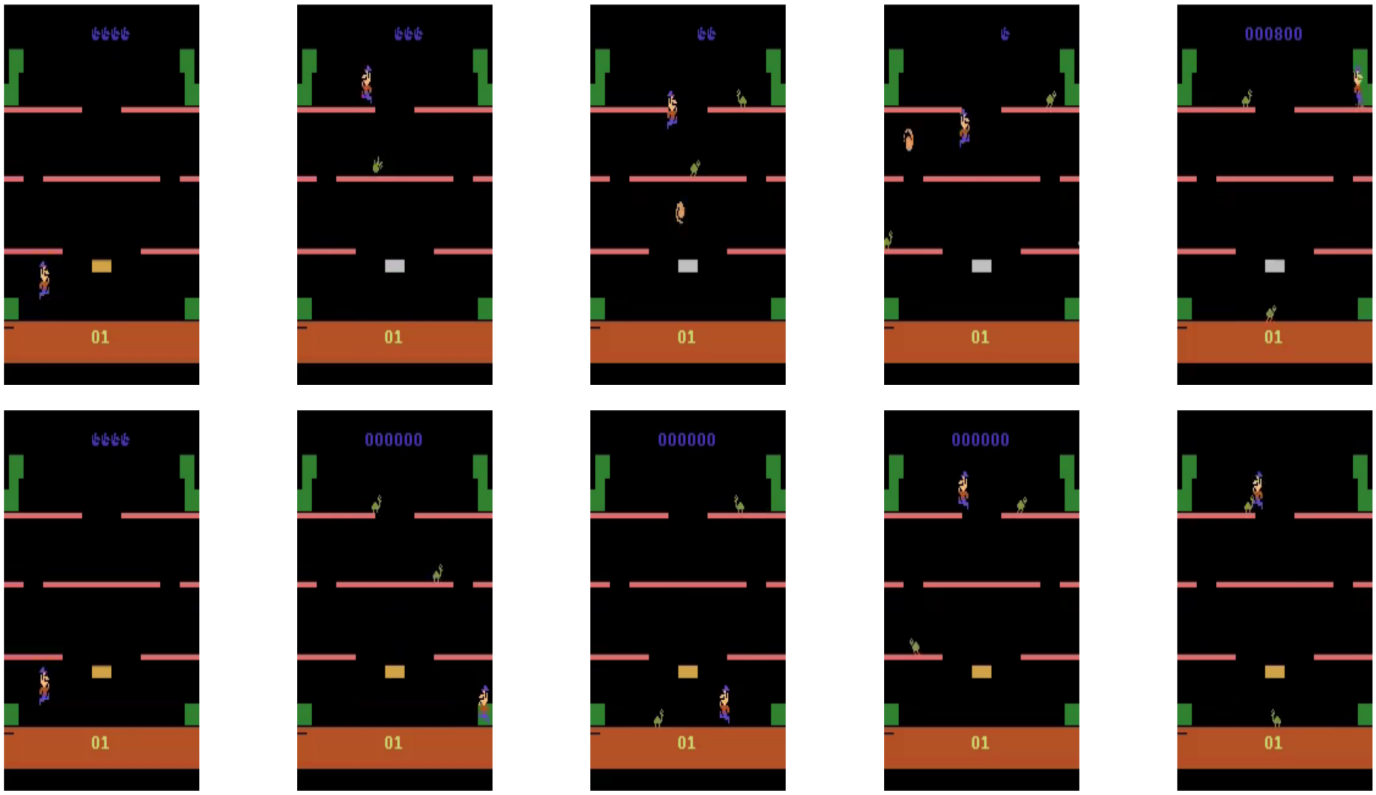
- Hesitant movement
- Weak enemy avoidance
- Poor jump timing
- Frequent backward steps

Learned

- Slightly smoother movement
- Survives somewhat longer
- Still unstable and inconsistent
- Clearly weaker than DDQN

Fiyori Demewez
Timothee Tran

DDQN 205k Frames (Early vs. Learned)

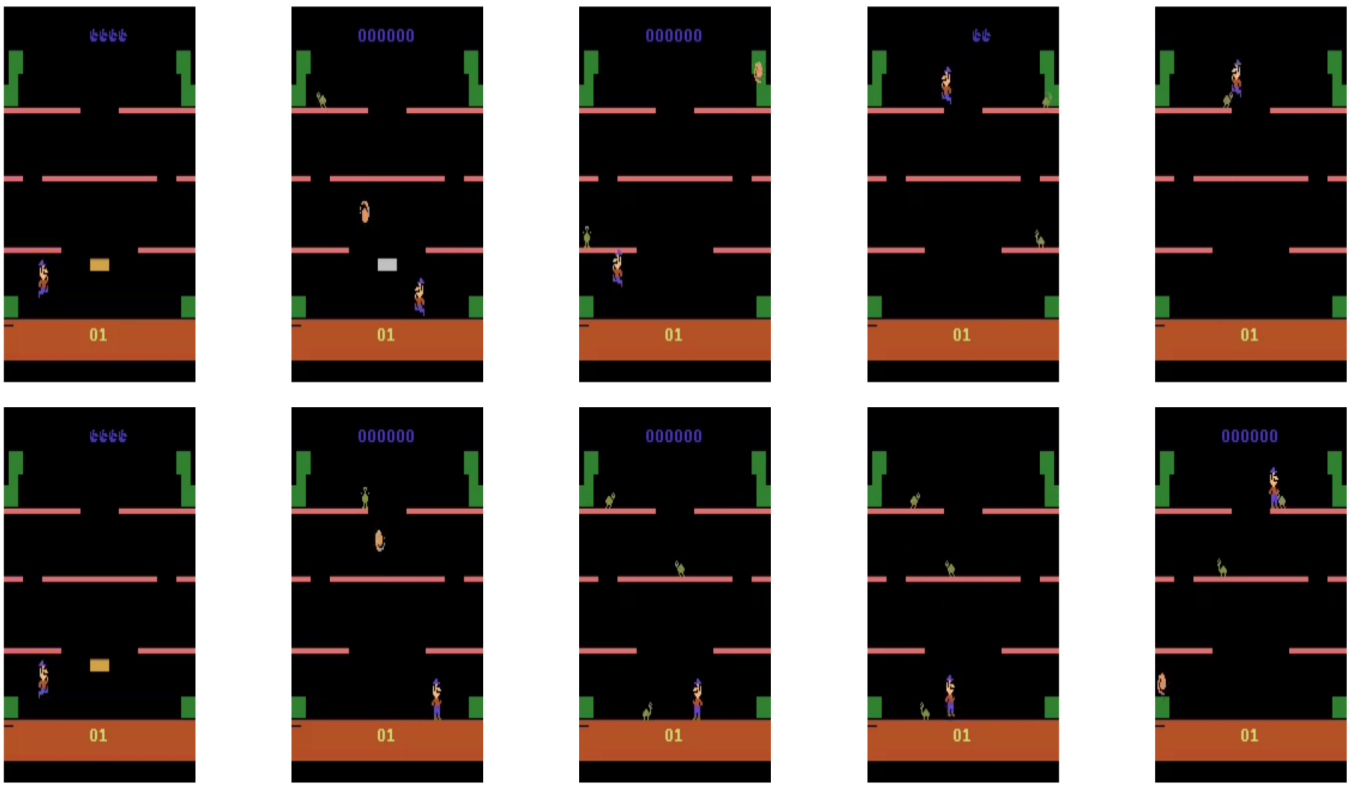


Observations

- Shows moderate improvement
- Still some hesitation
- Not fully stable

Fiyori Demewez
Timothee Tran

DDQN 305k Frames (Early vs. Learned)



Observations

- Strongest and smoothest agent
- Almost no hesitation
- Learns platforming behavior
- Clear, confident movement
- Longest survival by far

DDQN 305k shows the best policy quality of all runs.

Fiyori Demewez
Timothee Tran

9. Quantitative Comparison

Metric	Baseline DQN (200k)	205k Model	305k Model
Avg. episodic return	250–400	200–350	350–450
Stability	low	Low	high
Survival time	Frequent	short	longer
Hesitation	poor	Frequent	rare
Enemy avoidance	weak	Inconsistent	More reliable
Overall policy quality	basic	Moderate	Stronger and more stable

Summary

- Baseline DQN underperforms both DDQN models.
- DDQN is significantly more stable due to reduced Q-value overestimation.
- The 305k DDQN model is the strongest overall, showing recovery after 250k frames.

This aligns with existing reinforcement learning literature: **DDQN almost always outperforms vanilla DQN on Atari environments.**

Reflection

We selected *MarioBros* because it offers clear, sparse rewards and requires precise movement, making it a good environment for observing differences between DQN variants. Improvement mainly looked like increased survival time, smoother movement, and fewer backward or random steps.

Some key challenges included sparse feedback, difficulty with exploration, and long CPU training times, which slowed hyperparameter tuning. DQN, in particular, suffered from Q-value overestimation, which caused unstable returns and a noticeable performance collapse around the mid-training stage. Switching to Double DQN significantly improved convergence stability.

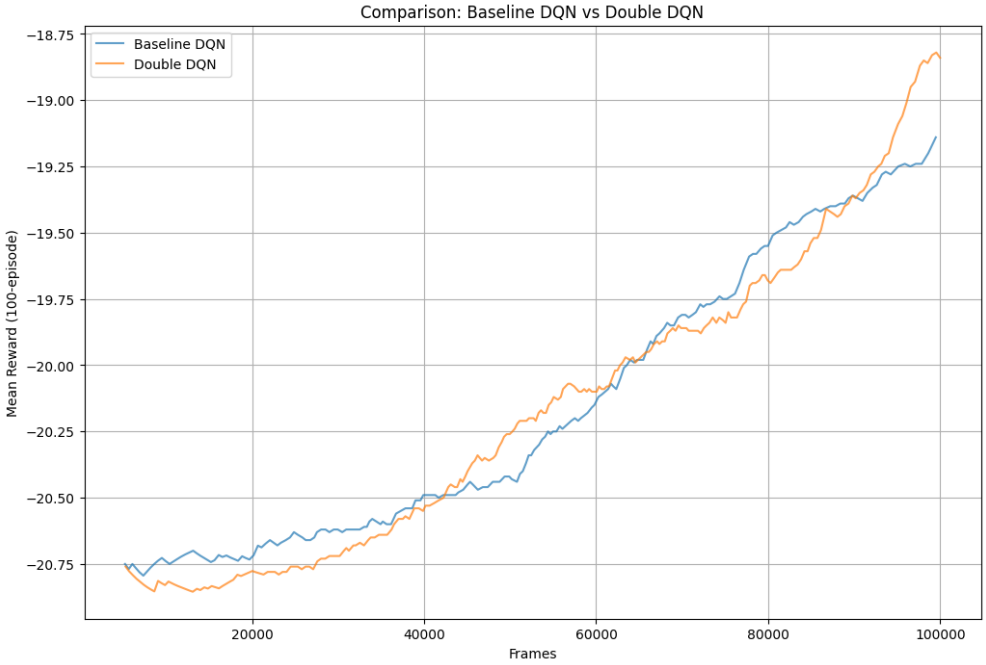
If we continued this project, we would explore N-step returns, sticky actions, prioritized replay, or a slower epsilon decay. We would also train for 1–3 million frames (as in the original Atari DQN paper) to allow the agent to learn more advanced platform behavior.

Fiyori Demewez
Timothee Tran

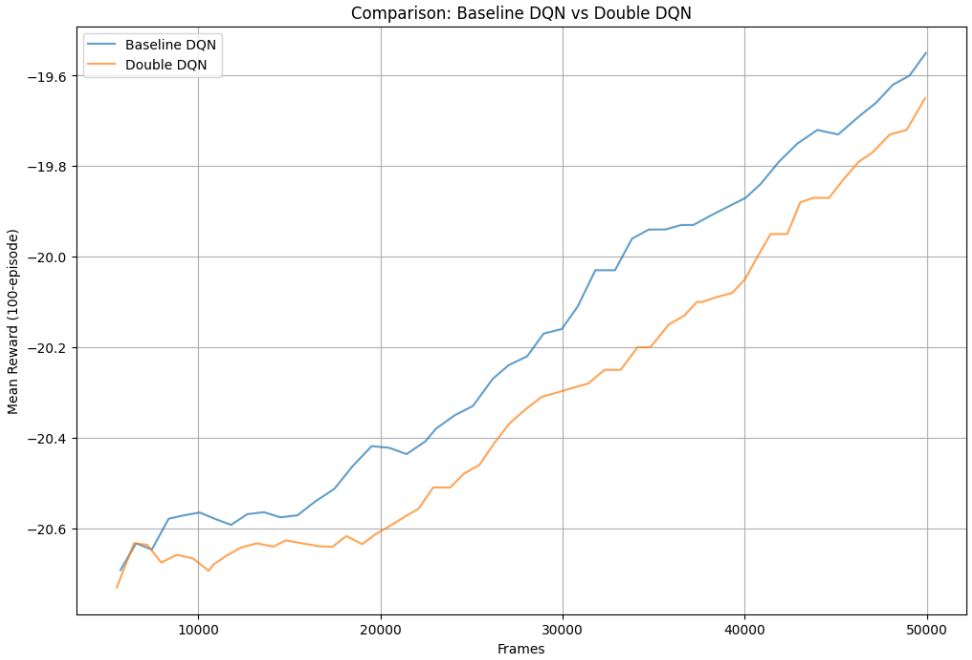
ALE/Pong-v5

Learning Curves:

100k Frame: (Original)

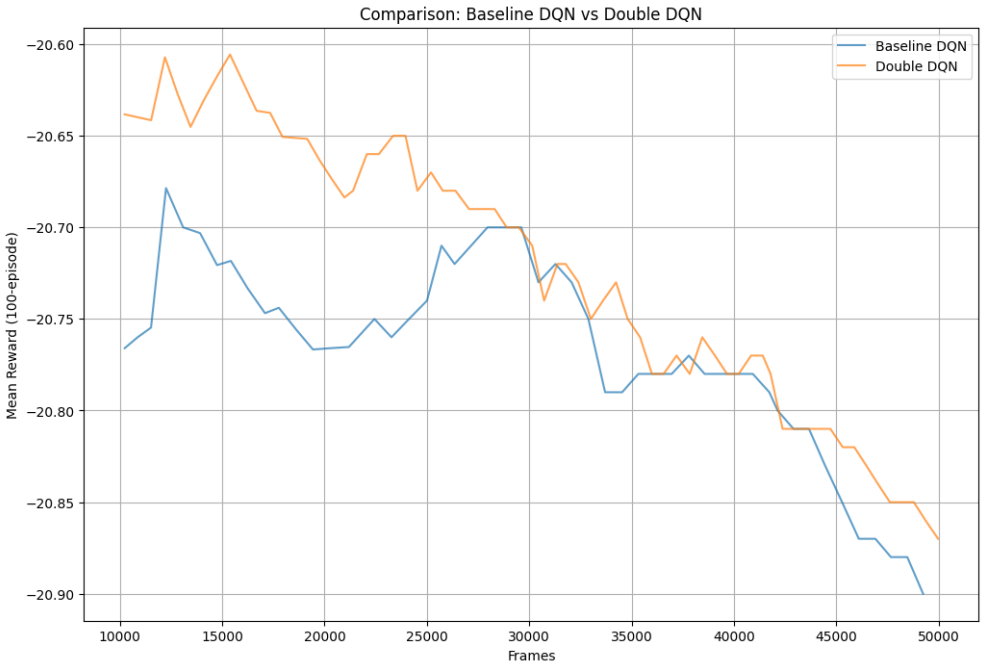


50k Frame: (Epsilon Decay)



Fiyori Demewez
Timothee Tran

50k Frame: (Epsilon Decay + Change in learning rate + start time)



Experimentation and Hyperparameter Analysis

Reduced Epsilon Decay Frames (100k → 50k):

- Goal: Speed up the shift from exploration to exploitation.
- Outcome: Training time dropped significantly (≈175min → ≈30min) with only a small drop in final performance.
- Observation: Double DQN began outperforming baseline DQN after 90k frames, suggesting stronger long-term gains if training continued.

Increased Learning Rate (1e-4 → 0.01):

- Goal: Accelerate convergence with a more aggressive learning rate.
- Outcome: Performance stalled near -20.8 (vs. -19.5 with 1e-4), indicating instability and overshooting.
- Conclusion: A lower, stable learning rate is more effective for this task.

Key Takeaways:

- Faster epsilon decay offers a good trade-off between training time and performance.
- Double DQN shows promise for long-term improvement, even if short-term gains are modest.
- Learning rate is sensitive; too high prevents learning entirely.

Future Directions:

- Tune epsilon decay schedules for better exploration/exploitation balance.
- Test advanced techniques like prioritized experience replay or dueling networks.
- Experiment further with learning rates and batch sizes for optimized training.

Conclusion

This project successfully implemented and compared baseline DQN against extended Double DQN across two distinct Atari environments. The results demonstrate clear differences in learning behavior and algorithm effectiveness between game complexities.

In Mario Bros, the environment proved particularly challenging for reinforcement learning with our base code implementation. Mario is not an easy game to tackle for both DQN & DDQN, as its complex mechanics, sparse rewards, and precise timing requirements presented significant hurdles that limited substantial policy improvement. Although DDQN does show more stable learning trends, the fundamental architecture struggled to achieve meaningful progress in this demanding environment.

On the other hand, Pong provided a clearer demonstration of DQN capabilities and the advantages of Double DQN. The simpler state space and more frequent rewards allowed both algorithms to show measurable improvement, with DDQN particularly demonstrating its value by beginning to outperform baseline DQN after approximately 90,000 frames. This performance divergence suggests that with extended training, Double DQN would achieve significantly stronger long-term results, confirming its theoretical advantage in reducing Q-value overestimation and producing more stable policies.

The comparative analysis through learning curves, gameplay videos, and performance metrics validates Double DQN as a meaningful enhancement over baseline DQN, while also highlighting how environmental complexity directly impacts reinforcement learning algorithm effectiveness and the importance of matching algorithm sophistication to game difficulty.