RL Final Project - Work Split (Complete Final Version)

Dor & Sara

Quick Summary

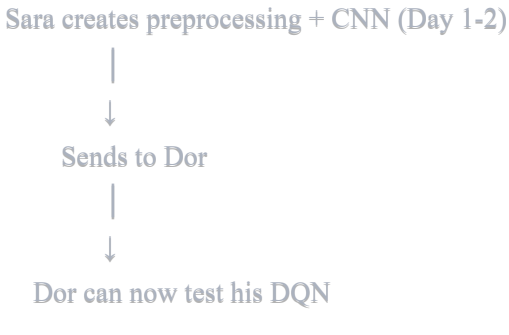
	Dor	**Sara**
Algorithm	DQN	Policy Gradient
Main Environment	KeyDoorBallEnv	SimpleGridEnv
Shared Code	—	Creates preprocessing + CNN
Special Work	Reward shaping, weight init testing	Admin files, comparison graphs, notebook merge
Report Pages	~4.5	~5.5

Why This Split is Fair

Task	**Effort**	**Who**
DQN algorithm (replay buffer, target network, epsilon)	High	Dor
Policy Gradient algorithm	Medium	Sara
KeyDoorBallEnv (4-step sequence, sparse rewards)	High	Dor
SimpleGridEnv (simple navigation)	Low	Sara
Reward shaping design + tuning	Medium	Dor
Weight initialization experiments	Low	Dor
Preprocessing + CNN architecture	Medium	Sara
Comparison graphs (overlay both algorithms)	Medium	Sara
Admin files + notebook merge	Low	Sara

Balance: Both do roughly equal total work.

Coordination: Who Waits for Who?



Dor waits for Sara's preprocessing + CNN before testing.

Dor can write these while waiting:

- Replay buffer class
- DQN training loop structure
- Epsilon decay logic
- Reward shaping logic

Sara's Tasks

S1: Preprocessing + CNN (Do First — Dor Needs This)

Build preprocessing function:

```python
def preprocess(observation):
    # Resize to chosen size (recommend 84x84 or 56x56)
    # Normalize pixels to 0-1 range
    # Convert to PyTorch tensor
    # Return: tensor ready for CNN
```

Build CNN feature extractor:

```python
class CNN(nn.Module):
    def __init__(self):
        # Conv layers to process image
        # Output: feature vector (recommend 256 dimensions)

    def forward(self, x):
        # Return feature vector
```

Decisions to make:

- Image size (recommend 84x84)
- Color or grayscale (recommend grayscale for speed)
- CNN output dimension (recommend 256)

Send to Dor when done.

S2: Policy Gradient Algorithm

Choose REINFORCE or Actor-Critic

Build:

- Policy Network (uses your CNN + FC layers → action probabilities)
- If Actor-Critic: Value Network (uses your CNN + FC layers → state value)
- Training loop with episode logging

Hyperparameters to try (document in table):

Parameter	Values to Try
Learning rate	0.0001, 0.0005, 0.001
Gamma	0.95, 0.99
Entropy bonus	0.0, 0.01
Max steps per episode	100, 200

S3: Solve SimpleGridEnv

Deliverables:

- [ ] Training graph: rewards vs episodes
- [ ] Training graph: steps vs episodes
- [ ] Video: mid-training
- [ ] Video: after convergence
- [ ] Metric: average steps over last 100 episodes
- [ ] Metric: **episode number when first solved** (e.g., first time agent reaches goal in under 20 steps consistently)
- [ ] Hyperparameter experiments table

S4: Run Policy Gradient on KeyDoorBallEnv

Test your algorithm on Dor's environment for comparison data.

Ask Dor for his reward shaping code.

Deliverables:

- [ ] Training graph
- [ ] Last 100 episodes metric
- [ ] Episode number when first solved (if solved)
- [ ] Notes on performance

S5: Comparison Graphs (NEW - Required)

Create these overlay graphs:

1. **SimpleGridEnv Comparison**
   - X-axis: Episodes
   - Y-axis: Rewards (or steps)
   - Two lines: DQN (from Dor) vs Policy Gradient (yours)
   - Same axes, same scale
2. **KeyDoorBallEnv Comparison**
   - X-axis: Episodes
   - Y-axis: Rewards (or steps)
   - Two lines: DQN (from Dor) vs Policy Gradient (yours)
   - Same axes, same scale

Include these in the Algorithm Comparison section of the report.

S6: Report Sections

Section	Content	Pages
Title Page	Project title, both names, both IDs	0.5
Policy Gradient Algorithm	How it works, your implementation, network architecture	1.5
SimpleGridEnv Results	Graphs, hyperparameter table, episode solved, final metric	1
Hyperparameter Analysis	Which parameters mattered, sensitivity	0.5

Section	Content	Pages
Algorithm Comparison	DQN vs PG on both envs, **comparison graphs**, data from both of you	1
Pros and Cons	Good/bad of each approach	0.5
Conclusion	Summary, limitations, what you'd change	0.5

Total: ~5.5 pages

Title page must include:

```
Reinforcement Learning Final Project - 2026

Dor [LastName] - ID: [DorID]
Sara [LastName] - ID: [SaraID]
```

S7: Administrative Files

details.txt:

```
link to the notebook:
https://colab.research.google.com/drive/XXXXX?usp=sharing

Full name student number 1: Dor [LastName]
ID student number 1: [ID]

Full name student number 2: Sara [LastName]
ID student number 2: [ID]
```

explainer.txt:

```
How to run:
1. Open Colab link
2. Runtime → Run all
3. Expected runtime: ~X minutes

Final hyperparameters:
- DQN: lr=X, buffer=X, target_update=X, init=X
- Policy Gradient: lr=X, gamma=X
```

S8: Final Notebook Assembly

1. Combine all code into one clean notebook
2. Add text cells explaining each section
3. Verify all outputs visible
4. **Verify comparison graphs are included**
5. Run once, then **NEVER TOUCH AGAIN**

Sara's Checklist

Shared Code (Do First):

- [ ] Preprocessing function → send to Dor
- [ ] CNN module → send to Dor

Algorithm:

- [ ] Policy Network
- [ ] Value Network (if Actor-Critic)
- [ ] Training loop

SimpleGridEnv Results:

- [ ] Rewards graph
- [ ] Steps graph
- [ ] Mid-training video
- [ ] Converged video
- [ ] Last-100 metric
- [ ] **Episode number when solved**
- [ ] Hyperparameter table

KeyDoorBallEnv Results (for comparison):

- [ ] Training graph
- [ ] Last-100 metric
- [ ] Episode number when solved

Comparison Graphs:

- [ ] DQN vs PG on SimpleGridEnv (overlay)
- [ ] DQN vs PG on KeyDoorBallEnv (overlay)

Report:

- [ ] **Title page with names and IDs**

- [ ] Policy Gradient Algorithm section
- [ ] SimpleGridEnv Results section
- [ ] Hyperparameter Analysis section
- [ ] Algorithm Comparison section (with comparison graphs)
- [ ] Pros and Cons section
- [ ] Conclusion section

Admin:

- [ ] details.txt
- [ ] explainer.txt
- [ ] Final notebook merge
- [ ] Final PDF assembly

Dor's Tasks

D1: DQN Algorithm

Build these components:

1. **Replay Buffer**
    - Store (state, action, reward, next_state, done)
    - Sample random batches
    - Size: 10,000 - 100,000
2. **Q-Network**
    - Uses Sara's CNN + FC layers
    - Output: Q-value per action
3. **Target Network**
    - Copy of Q-Network
    - Update every N steps
4. **Epsilon-Greedy**
    - Start: 1.0, End: 0.01
    - Decay over training

Hyperparameters to try (document in table):

Parameter	Values to Try
Learning rate	0.0001, 0.0005, 0.001
Batch size	32, 64
Replay buffer size	10000, 50000
Target update frequency	100, 500, 1000 steps
Epsilon decay episodes	500, 1000, 2000
Gamma	0.95, 0.99
Weight initialization	Xavier, Kaiming, Default

D2: Weight Initialization Experiments (NEW - Required)

Test these initializations on your Q-Network:

Initialization	PyTorch Code
Xavier Uniform	`nn.init.xavier_uniform_(layer.weight)`
Kaiming (He)	`nn.init.kaiming_uniform_(layer.weight)`
Default	Don't change anything

Document:

- Which initialization converged fastest
- Which gave best final performance
- Include in hyperparameter table

D3: Reward Shaping for KeyDoorBallEnv

Default reward is too sparse. Add:

Event	Reward
Pick up key	+0.2
Open door	+0.3
Pick up ball	+0.3
Reach goal	+1.0
Each step	-0.01

Share this code with Sara when she tests on KeyDoorBallEnv.

D4: Solve KeyDoorBallEnv

Deliverables:

- [ ] Training graph: rewards vs episodes
- [ ] Training graph: steps vs episodes
- [ ] Video: mid-training
- [ ] Video: after convergence
- [ ] Metric: average steps over last 100 episodes
- [ ] Metric: **episode number when first solved** (when agent completes full sequence consistently)
- [ ] Hyperparameter experiments table (including weight init)

D5: Run DQN on SimpleGridEnv

Test your algorithm on Sara's environment for comparison data.

Deliverables:

- [ ] Training graph
- [ ] Last 100 episodes metric
- [ ] Episode number when first solved
- [ ] Notes on performance

Send all results to Sara for comparison graphs and comparison section.

D6: Report Sections

Section	Content	Pages
Introduction	MiniGrid, two environments, project goal	0.5
DQN Algorithm	How it works, replay buffer, target network, your implementation	1.5
KeyDoorBallEnv Results	Reward shaping rationale, graphs, **episode solved**, hyperparameters (including init), final metric	1.5
Exploration vs Exploitation	Epsilon-greedy, what schedule worked	0.5
Environment Considerations	Why KeyDoorBallEnv needs shaping, action space differences	0.5

Total: ~4.5 pages

Dor's Checklist

Algorithm (can do while waiting for CNN):

- [ ] Replay buffer class
- [ ] Q-Network structure (plug in CNN later)
- [ ] Target network logic
- [ ] Epsilon-greedy with decay
- [ ] Training loop structure
- [ ] Reward shaping logic
- [ ] **Weight initialization options**

After receiving CNN from Sara:

- [ ] Integrate CNN into Q-Network
- [ ] Test different weight initializations
- [ ] Test and debug
- [ ] Train on KeyDoorBallEnv
- [ ] Train on SimpleGridEnv

KeyDoorBallEnv Results:

- [ ] Rewards graph
- [ ] Steps graph
- [ ] Mid-training video
- [ ] Converged video

- [ ] Last-100 metric
- [ ] **Episode number when solved**
- [ ] Hyperparameter table (with weight init results)

SimpleGridEnv Results (for comparison):

- [ ] Training graph
- [ ] Last-100 metric
- [ ] Episode number when solved
- [ ] **Send all to Sara**

Report:

- [ ] Introduction section
- [ ] DQN Algorithm section
- [ ] KeyDoorBallEnv Results section
- [ ] Exploration vs Exploitation section
- [ ] Environment Considerations section

Timeline Suggestion

Day	Sara	Dor
1-2	Build preprocessing + CNN	Write replay buffer, DQN structure
2	Send CNN to Dor	Receive CNN, integrate
3-5	Train Policy Gradient on SimpleGridEnv	Train DQN on KeyDoorBallEnv + test inits
6	Test PG on KeyDoorBallEnv	Test DQN on SimpleGridEnv
6	Collect Dor's data	Send data to Sara
7	**Create comparison graphs**	Write report sections
8	Write report sections	Review
9	Merge notebook + report	Final review
10	Final check, freeze notebook	Final check

Complete Assignment Coverage

Requirement	Who	Status
Solve SimpleGridEnv	Sara	✅
Solve KeyDoorBallEnv	Dor	✅
Image-based observation	Sara (preprocessing)	✅
Different Deep RL algorithms	DQN + PG	✅
Discuss advantages/disadvantages	Sara (Pros/Cons)	✅
Graphs COMPARING approaches	**Sara (overlay graphs)**	✅
Preprocessing on input image	Sara	✅
Hyperparameter: Learning rate	Both	✅
Hyperparameter: Epsilon	Dor	✅
Hyperparameter: Replay buffer size	Dor	✅
Hyperparameter: Target network update	Dor	✅
Hyperparameter: Initializations	**Dor**	✅
Different considerations for envs	Dor	✅
Exploration-Exploitation tradeoff	Dor	✅
Training graphs	Both	✅
Number of steps/episode to SOLVE	**Both**	✅
Last 100 episodes average steps	Both	✅
Evaluate good AND bad points	Sara	✅
No existing RL libraries	Both	✅
Convergence graph (Rewards)	Both	✅
Convergence graph (Steps)	Both	✅
Video: mid-training	Both (4 total)	✅
Video: after convergence	Both (4 total)	✅
Google Colab notebook	Sara assembles	✅

Requirement	Who	Status
Clean code + text cells	Sara assembles	✅
Don't change after submission	Sara handles	✅
Self-contained report	Both write	✅
Title page with names + IDs	**Sara**	✅
details.txt	Sara	✅
explainer.txt	Sara	✅
Report filename format	Sara	✅
Max 10 pages	~10 pages	✅

All requirements covered: 100%

Risks

Risk	Who	Prevention
Sara's CNN doesn't work	Sara	Test early, keep simple
KeyDoorBallEnv won't converge	Dor	Reward shaping, 3000+ episodes
Dor blocked too long	Sara	Send CNN within 2 days
Policy Gradient unstable	Sara	Actor-Critic, save checkpoints
Comparison graphs missing	Sara	Get Dor's data by Day 6
Weight init not tested	Dor	Do this early, easy to forget
"Episode solved" not recorded	Both	Add logging from start
Notebook modified after deadline	Sara	Final run → never reopen
Report over 10 pages	Sara	Keep concise
Title page missing names/IDs	Sara	Check before PDF export

Final Submission Checklist

Files:

- [ ] details.txt
- [ ] report_[DorID]_[SaraID].pdf
- [ ] explainer.txt

Report contains:

- [ ] Title page with project title, both names, both IDs
- [ ] Introduction (Dor)
- [ ] DQN Algorithm (Dor)
- [ ] Policy Gradient Algorithm (Sara)
- [ ] SimpleGridEnv Results (Sara)
- [ ] KeyDoorBallEnv Results (Dor)
- [ ] Exploration vs Exploitation (Dor)
- [ ] Environment Considerations (Dor)
- [ ] Hyperparameter Analysis (Sara)
- [ ] Algorithm Comparison with **comparison graphs** (Sara)
- [ ] Pros and Cons (Sara)
- [ ] Conclusion (Sara)
- [ ] ≤ 10 pages total

Notebook contains:

- [ ] Clean code with text cells
- [ ] All training outputs visible
- [ ] All graphs visible
- [ ] **Comparison graphs visible**
- [ ] 4 videos embedded (2 per environment)
- [ ] Last-100 metrics printed
- [ ] **Episode-solved metrics printed**
- [ ] NOT modified after deadline