0 Cerințele Proiectului

Atenție! Nota finală depinde 100% de proiectul realizat.

Echipe obligatorii

- Echipe de **minim 3** și **maxim 5** studenți.
- **Lucrul individual nu este permis** (complexitatea proiectului nu permite).

Organizare

- Se va deschide o **listă** unde veți nota:
  - componența echipei
  - tema proiectului pe care o propuneți
- Prezentările se vor desfășura în **ultima săptămână a semestrului**, în mai multe sesiuni (orele de laborator + curs).

Ce trebuie să conțină proiectul?

1. Alegerea unei teme de proiect

Exemple:
- agent RL într-un joc 2D / Unity / Unreal Engine / etc.
- robotică simulată
- optimizarea unui mediu personalizat
- control, navigație, sisteme dinamice etc.

2. Implementarea sau modificarea unui environment

- fie creați un environment propriu
- fie modificați/îmbunătățiți unul existent (Gymnasium, Unity, IsaacGym, etc.)

3. Implementarea a cel puțin 3 algoritmi/agenți

- trebuie comparați în **același mediu**
- recomandare: combinație tabular + deep + policy-based

4. Reglarea hiperparametrilor & experimente multiple

- rulați experimente variate
- analizați instabilități, convergență, overfitting, explorare etc.

5. Evidențierea rezultatelor

Recomandate: - grafice (reward, loss, Q-values, stabilitate) - tabele comparative
- metrici de performanță

6. Documentarea completă a proiectului

Format la alegere: - PowerPoint
- LaTeX
- Word / PDF

Documentarea trebuie să includă: - motivație și idee
- arhitectura agenților
- environment design
- rezultate și discuții
- probleme întâmpinate + soluții

7. Prezentarea finală

- prezentare în echipă
- accent pe evoluția proiectului: **de la idee → implementare → rezultate**

1 Algoritmi de Reinforcement Learning

Această secțiune oferă o listă cuprinzătoare de algoritmi RL, de la metode tabulare clasice până la algoritmi avansați Deep RL, offline RL și multi-agent. Fiecare categorie include link-uri utile.

Algoritmi Clasici / Tabulari

Metode fundamentale, excelente pentru înțelegerea mecanicii RL.

- Dynamic Programming (Value Iteration, Policy Iteration)
  https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf
- Monte Carlo Prediction & Control
- Temporal Difference (TD): TD(0), TD(n), TD( )
- SARSA

- Expected SARSA
- Q-Learning (Watkins, 1992)
- Double Q-Learning
- Weighted Q-Learning
- Dyna-Q
- Prioritized Sweeping
- R-Learning (average reward)

Resursă recomandată:
Sutton & Barto – *Reinforcement Learning: An Introduction*
https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

---

Deep RL (Value-Based)

Algoritmi ce combină funcții de valoare cu rețele neurale.

- DQN – Deep Q-Network
  https://www.nature.com/articles/nature14236
- Double DQN
- Dueling DQN
- Prioritized Experience Replay
- NoisyNets
- C51 – Distributional DQN
  https://arxiv.org/abs/1707.06887
- QR-DQN – Quantile Regression DQN
- Rainbow DQN
  https://arxiv.org/abs/1710.02298
- Deep SARSA / Expected Deep SARSA

Implementări recomandate:
Stable Baselines3 – https://stable-baselines3.readthedocs.io/
CleanRL – https://github.com/vwxyzjn/cleanrl

Policy-Based Methods

Algoritmi ce învață direct o politică.

- REINFORCE (Policy Gradient)
- Actor-Critic (AC)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Trust Region Policy Optimization (TRPO)
  https://arxiv.org/abs/1502.05477
- Proximal Policy Optimization (PPO)
  https://arxiv.org/abs/1707.06347

- Soft Actor-Critic (SAC)
  https://arxiv.org/abs/1812.05905
- Deep Deterministic Policy Gradient (DDPG)
- Twin Delayed DDPG (TD3)
  https://arxiv.org/abs/1802.09477
- Natural Policy Gradient (NPG)

Implementări:
SB3 PPO – https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

---

Offline RL / Batch RL

Algoritmi ce lucrează doar cu date existente (fără environment live).

- Behavior Cloning (BC)
- BCQ – Batch-Constrained Q-Learning
  https://arxiv.org/abs/1812.02900
- CQL – Conservative Q-Learning
  https://arxiv.org/abs/2006.04779
- TD3-BC
- AWAC (Advantage Weighted Actor-Critic)
- Diffusion Policies
  https://arxiv.org/abs/2301.11824

Librărie utilă:
d3rlpy – https://github.com/takuseno/d3rlpy

Model-Based RL

Algoritmi ce învață un model intern al dinamicii.

- Dyna-Q

- MPC-based RL
- MuZero
  https://arxiv.org/abs/1911.08265
- DreamerV2 / DreamerV3
  https://arxiv.org/abs/2010.02193
- PETS – Probabilistic Ensembles with Trajectory Sampling

- MBPO – Model-Based Policy Optimization

Framework:
Brax – https://github.com/google/brax

Multi-Agent RL (MARL)

Algoritmi pentru sisteme cu mai mulți agenți.

- Independent Q-Learning (IQL)
- MADDPG – Multi-Agent DDPG
  https://arxiv.org/abs/1706.02275
- QMIX
  https://arxiv.org/abs/1803.11485
- VDN (Value Decomposition Networks)
- COMA – Counterfactual Multi-Agent Policy Gradients
- MAPPO – Multi-Agent PPO

Suită completă medii de lucru:
PettingZoo – https://pettingzoo.farama.org/

Exploration-Driven RL

Algoritmi pentru medii sparse-reward.

- RND – Random Network Distillation
  https://arxiv.org/abs/1810.12894
- ICM – Intrinsic Curiosity Module
- Go-Explore
  https://arxiv.org/abs/1901.10995
- Count-Based Exploration
- Pseudo-Counts / Density Models

2 Framework-uri de Machine Learning

Această secțiune prezintă principalele framework-uri de Machine Learning folosite în proiectele de Reinforcement Learning. Sunt incluse atât tehnologiile consacrate (PyTorch, TensorFlow), cât și framework-uri moderne (JAX, Lightning, TorchRL). La final se găsește și un ghid rapid de **compatibilitate hardware + OS**, esențial pentru implementările RL.

Framework-uri consacrate

1. PyTorch (cel mai folosit în RL)

- Ușor de folosit, dinamic (define-by-run)
- Recomandat pentru proiecte RL și cercetare

https://pytorch.org/
Documentație GPU: https://pytorch.org/get-started/locally/

---

2. TensorFlow 2 / Keras

- Stabil, suport masiv Google
- Bun pentru prototipuri și LLM fine-tuning

https://www.tensorflow.org/
Keras: https://keras.io/

Framework-uri moderne

3. JAX (Google)

- Ultra-rapid, optimizări XLA
- Recomandat pentru RL de mare performanță, model-based, world models

https://jax.readthedocs.io/
Flax (NN library for JAX): https://flax.readthedocs.io/

---

4. PyTorch 2.x + torch.compile

- Compilare just-in-time pentru accelerare
- Recomandat pentru experimente avansate

https://pytorch.org/docs/stable/generated/torch.compile.html

---

5. Lightning AI (fost PyTorch Lightning)

- Structurează codul ML / RL în modul profesionist
- Logging + training loops standardizate

https://lightning.ai/

---

6. **TorchRL (Meta AI) – *framework RL oficial pentru PyTorch***
   - Zeci de implementări RL
   - Perfect pentru proiecte avansate cu PyTorch

   https://pytorch.org/rl/

---

7. **HuggingFace TRL – RL pentru modele de limbaj**
   - PPO, DPO, GRPO
   - Folosit în RLHF pentru LLM-uri

   https://github.com/huggingface/trl

---

8. **Diffusers (RL + modele de difuzie)**
   - Folosit în robotică, planning și control
   - Difussion Policies (2023–2024 trend major)

   https://github.com/huggingface/diffusers

Framework-uri auxiliare pentru monitorizare & tuning

- **Weights & Biases (wandb)** – vizualizare experimente
  https://wandb.ai
- **TensorBoard** – logging și grafice
  https://www.tensorflow.org/tensorboard
- **Optuna** – hyperparameter tuning
  https://optuna.org/
- **Ray Tune** – tuning distribuit + sweeps mari
  https://docs.ray.io/en/latest/tune/

Compatibilitate hardware și sistem de operare

Compatibilitatea contează mult în RL din cauza costurilor GPU.

Windows

- **PyTorch** — complet suportat (NVIDIA GPU OK)
- **TensorFlow** — suport GPU doar pe Windows 10+ cu CUDA (NVIDIA)
- **JAX** — *nu are suport oficial GPU pe Windows* → doar CPU
- **AMD GPU (ROCm)** — suport limitat pe Windows

Recomandare: dacă ai Windows + NVIDIA → folosește PyTorch.

Linux (Ubuntu / Debian / Arch etc.)

- **PyTorch** — suport complet
- **TensorFlow** — suport complet
- **JAX** — cel mai bun suport (GPU + TPU)
- **NVIDIA GPU** — este platforma standard pentru RL
- **AMD GPU (ROCm)** — suport bun în ultimii ani
  - PyTorch ROCm: https://pytorch.org/get-started/locally/#linux-installation
  - TensorFlow ROCm: https://rocm.docs.amd.com/projects/install-on-linux/en/latest/install/3rd-party/tensorflow-install.html

Recomandat pentru proiecte avansate: Linux + NVIDIA.

---

macOS (Intel & Apple Silicon M1/M2/M3)

macOS Intel

- GPU support: aproape inexistent
- CPU only: PyTorch, TensorFlow

macOS ARM (M1/M2/M3/M4/M5 - nu, nu liniile de metrou...)

- **PyTorch** — suport nativ pentru GPU (Metal) https://pytorch.org/blog/pytorch-1.12-released/

- **TensorFlow-Metal** — suport GPU Apple Silicon
  https://developer.apple.com/metal/tensorflow-plugin/
- **JAX** — doar CPU, suport experimental GPU
- **CUDA** — NU funcționează pe Mac

MacOS M1/M2 este foarte bun pentru RL educațional.

GPU NVIDIA – cea mai importantă platformă în RL

- Suport complet pentru:
  PyTorch
  TensorFlow
  JAX
- Necesită:
  – CUDA Toolkit
  – cuDNN

https://developer.nvidia.com/cuda-downloads

AMD GPU – ROCm (2025)

- **PyTorch ROCm:** suport bun
- **TensorFlow ROCm:** suport bun
- **JAX:** fără suport GPU

https://rocmdocs.amd.com/

Recomandat doar dacă nu ai acces la NVIDIA.

Rezumat compatibilitate

Platformă	PyTorch	TensorFlow	JAX	GPU accel.	Nota utilizare
Windows + NVIDIA			CPU only	NVIDIA	Recomandat pt. începători
Linux + NVIDIA				Cea mai bună	Standard industrial
Linux + AMD (ROCm)				AMD	Ok pentru bugete mici
macOS M1/M2/M3	Metal	Metal		GPU Apple	Bun pentru proiecte educaționale
macOS Intel	CPU	CPU	CPU		Evitat pentru RL serios

---

3 Environment-uri Avansate pentru Proiecte (Real-Life & Simulare)

Această secțiune oferă o listă de environment-uri moderne și aplicabile în scenarii reale. Toate sunt potrivite pentru proiecte intermediare și avansate în Reinforcement Learning.

---

1. Robotică & Control

MuJoCo – robotică, control

- Standard industrial pentru control robotic
- Folosit în DDPG/T D3/SAC/PPO
  https://mujoco.org/

Isaac Gym (NVIDIA) – simulare paralelizată

- Rulează mii de environment-uri pe GPU
- Ideal pentru RL cu rețele mari sau multi-agent
  https://developer.nvidia.com/isaac-gym

DeepMind Control Suite

- Benchmark robotic avansat
  https://www.deepmind.com/publications/dm-control

robosuite (Stanford + NVIDIA)

- Task-uri reale (lifting, manipulation)
  https://robosuite.ai/

ManiSkill / ManiSkill2

- Simulare robotică modernă, foarte realistă
  https://github.com/haosulab/ManiSkill

---

2. Driving, Autonomous Vehicles, Drone Control

CARLA Simulator – self-driving cars

- Folosit de Tesla, Waymo, Uber AI
- Scenarii urbane, trafic, vreme reală
  https://carla.org/

MetaDrive – driving simulators cu date sintetice

- Multi-agent driving, traffic RL
  https://metadrive-simulator.readthedocs.io/

AirSim (Microsoft) – drone & vehicule

- Suport pentru drone autonome, real flight sim
  https://microsoft.github.io/AirSim/

Gymnasium Driving (HighwayEnv)

- Manevre auto, trafic, avoidance
  https://highway-env.readthedocs.io/

---

3. Simulatoare Industriale & Procese Reale

OpenAI Safety Gym / Safety Gymnasium

- RL cu constrângeri de siguranță
  https://github.com/PKU-Alignment/safety-gymnasium

Grid2Op – Energy Grid RL

- Controlul rețelelor electrice în timp real
  https://grid2op.readthedocs.io/

CityLearn – energie urbană, smart grid

- Folosit pentru optimizarea consumului în orașe
  https://www.citylearn.net/

TEP (Tennessee Eastman Process)

- Simularea unui proces chimic industrial
- Folosit în control predictiv și RL
  https://github.com/aspuru-guzik-group/rl-te

EplusGym (EnergyPlus)

- RL pentru control HVAC real
  https://github.com/jingpengwong/EplusGym

---

4. Medii cu Date Reale (Finance, Operations)

FinRL / FinRL-Meta

- Trading bots, portofolii, risc
  https://github.com/AI4Finance-Foundation/FinRL

RL4CO – Reinforcement Learning for Combinatorial Optimization

- VRP (Vehicle Routing Problem)
- Knapsack, TSP, scheduling
  https://github.com/kaist-silab/RL4CO

Recommender Systems (RecSim – Google)

- Personalizare, ranking, RL în recomandări
  https://github.com/google-research/recsim

---

5. Healthcare (simulare & offline RL) – *doar pentru cercetare, nu decizii reale*

MIMIC-III / MIMIC-IV Offline RL datasets

- Seturi clinice reale, doar pentru simulare offline
  https://physionet.org/content/mimiciv/

Open Clinical RL (OCH)

- Proiect academic pentru offline RL
  https://github.com/open-clinical/open-clinical-rl

Sepsis Treatment RL benchmark

https://github.com/microsoft/AIforHealth-SepsisRL

---

6. Multi-Agent Environments

PettingZoo

- Environment-uri multi-agent standardizate
  https://www.pettingzoo.ml/

MAgent

- Mii de agenți în scenarii grid-based
  https://github.com/geek-ai/MAgent

SMAC (StarCraft Multi-Agent Challenge)

- Multi-agent control în StarCraft II
  https://github.com/oxwhirl/smac

---

7. Jocuri Avansate, Simulare & Puzzle

Procgen Benchmark (OpenAI)

- Generare procedurală → generalizare puternică
  https://github.com/openai/procgen

MiniGrid / MiniWorld

- Navigație 2D/3D, recomandate în RL educațional
  https://minigrid.farama.org/
  https://miniworld.farama.org/

Minecraft (MineRL)

- Environment complet, acțiuni complexe
  https://minerl.readthedocs.io/

---

8. Construirea propriului environment (API-based)

Poți crea un environment propriu folosind date reale sau API-uri live:

API-uri utile pentru proiecte RL custom:

- **Google Maps API** – routing, navigație, trafic
  https://developers.google.com/maps
- **OpenWeather API** – modelare HVAC, energie, outdoor RL
  https://openweathermap.org/api
- **Binance/Kraken API** – trading RL
  https://binance-docs.github.io/apidocs/
- **Spotify API** – RL pentru recomandări audio
  https://developer.spotify.com/documentation/web-api
- **GitHub API / Jira API** – scheduling și optimizare task-uri
  https://docs.github.com/en/rest
  https://developer.atlassian.com/cloud/jira/platform/rest/v3/

---

9. Environment-uri RL din suita Farama Foundation

Gymnasium – successor of OpenAI Gym

https://gymnasium.farama.org/

Gymnasium Robotics (fostul Gym Robotics)

https://robotics.farama.org/

Gymnasium Classic Control / Box2D

https://gymnasium.farama.org/environments/classic_control/

Rezumat

Alegeți un mediu care permite: - complexitate moderată
- posibilitatea de a testa **minim 3 algoritmi**
- metrici clare de evaluare

---

4 Librării / Tools

Această secțiune prezintă librăriile esențiale din ecosistemul Reinforcement Learning: framework-uri minimaliste pentru începători, librării industriale distribuite, tool-uri pentru robotică, offline RL și multi-agent.

---

1. Implementări RL pentru standard academic & educațional

Stable Baselines3 (SB3)

- Cea mai folosită librărie RL în educație și prototipare
- Implementări solide: DQN, PPO, A2C, SAC, TD3
  https://stable-baselines3.readthedocs.io/

CleanRL

- Fiecare algoritm = un singur fișier
- Ideal pentru învățarea mecanicii interne
  https://github.com/vwxyzjn/cleanrl

TorchRL (Meta AI)

- Librăria oficială PyTorch pentru RL
- Include environment-uri, rollout collectors, replay buffers
  https://pytorch.org/rl/

---

2. Framework-uri industriale & distribuite

RLlib (Ray)

- Scalare pe clustere, HPC, cloud
- Folosit în industrie pentru aplicații mari
  https://docs.ray.io/en/latest/rllib/

Acme (DeepMind)

- Toolkit modular RL pentru cercetare
- Implementări oficiale ale algoritmilor DeepMind
  https://github.com/deepmind/acme

Tianshou

- Performant, flexibil, foarte activ în comunitate
  https://github.com/thu-ml/tianshou

Coach (Intel AI Lab)

- Framework vechi dar încă util, multe algoritmi incluși
  https://github.com/IntelLabs/coach

---

3. Librării pentru Offline RL (dataset-based RL)

d3rlpy

- Cel mai popular framework pentru offline RL
- Conține implementări pentru: BCQ, CQL, TD3-BC, IQL
  https://github.com/takuseno/d3rlpy

Horizon RL (Meta)

- Folosit în recommender systems
  https://github.com/facebookresearch/Horizon

offline-rl (Berkeley)

- Unelte pentru experimente offline
  https://github.com/aviralkumar2907/offlineRL

---

4. Multi-Agent RL (MARL)

PettingZoo

- Standardul pentru environment-uri multi-agent
  https://www.pettingzoo.ml/

MAgent

- Simulări cu mii de agenți
  https://github.com/geek-ai/MAgent

MARLlib

- Implementări moderne pentru multi-agent: QMIX, MADDPG, MAPPO, QTRAN
  https://github.com/Replicable-MARL/MARLlib

PyMARL / PyMARL2

- Implementări pentru QMIX și alte algoritmi cooperative
  https://github.com/oxwhirl/pymarl

---

5. Librării pentru robotică și simulare

robomimic (Stanford + NVIDIA)

- Combină imitation learning + RL pentru manipulare robotică
  https://robomimic.github.io/

robosuite

- Framework robotic compatibil MuJoCo
  https://robosuite.ai/

ManiSkill / ManiSkill2

- Set masiv de environment-uri realiste
  https://github.com/haosulab/ManiSkill

Brax (Google - JAX)

- Simulări rapide cu JAX
- Perfect pentru Dreamer / model-based RL
  https://github.com/google/brax

dm_control (DeepMind)

https://github.com/deepmind/dm_control

---

6. Tool-uri pentru RL în LLM-uri (RLHF, RLAIF)

HuggingFace TRL

- PPO, DPO, GRPO
- Folosit în fine-tuning LLMs
  https://github.com/huggingface/trl

OpenAI Evals

- Evaluare automată pentru RLHF / alignment
  https://github.com/openai/evals

DeepSpeed-Chat

- Training accelerat pentru LLMs cu RL
  https://github.com/microsoft/DeepSpeed

---

7. Evaluare, logging și tuning pentru RL

Weights & Biases (wandb)

- Logging, dashboards, compararea experimentelor
  https://wandb.ai/

TensorBoard

- Grafice și inspectare modele
  https://www.tensorflow.org/tensorboard

Ray Tune

- Hyperparameter tuning avansat
  https://docs.ray.io/en/latest/tune/

Optuna

- Tuning pentru parametri (simplu, eficient)
  https://optuna.org/

Rezumat rapid (ce să folosești în proiect)

Scop	Librărie recomandată
Proiect educațional / prototip	**Stable Baselines3, CleanRL**
Implementare custom avansată	**TorchRL, Tianshou**
Scalare / cluster / cloud	**RLlib**
Offline RL	**d3rlpy, Horizon**
Robotică	**robomimic, robosuite, ManiSkill2**
Multi-agent	**PettingZoo + MARLlib**
Model-based (MuZero / Dreamer)	**Brax, dm_control, JAX**
RL pentru LLM (PPO / DPO)	**HuggingFace TRL**
