ML AGENTS
METAL SLUG

GRUPPO 7

BEDETTA
ALESSANDRO

ASCANI
CHRISTIAN

RECCHI
GIOVANNI

OVERVIEW

- Obiettivi e richieste
- Descrizione dell'agente
- Diogrommo delle classi e delle sequenze
- Problemotiche riscontrote e soluzioni
- Iperporometri e configurozione migliore
- Altre configurozioni provote
- Sviluppi futuri e conclusioni

INTRODUZIONE

Interesse sempre maggiore per Deep Reinforcement Learning nell'ambito videoludico

→ ML-Agents: implementazione del DRL in ogni gioco sviluppato in Unity

Obiettivo:

- Implementazione di ML-Agents nel gioco «Metal Slug» in modo che l'agente sia in grado di completare il livello

Richieste:

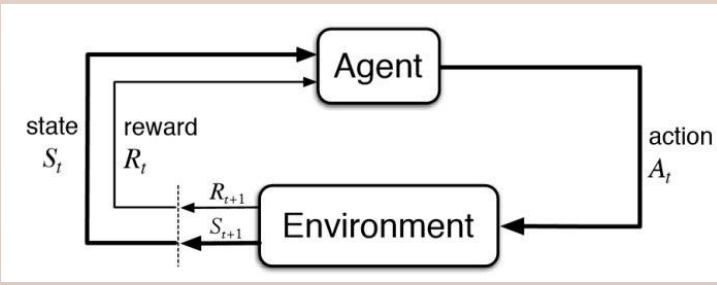
- Modifica del codice per adattarlo a ML-Agents
- Creazione della classe per la gestione dell'agente
- Configurazione degli iperparametri per il training
- Training dell'agente

DESCRIZIONE AGENTE

RL

Agente: entità capace di percepire l'ambiente circostante

➡ Agent ancorato al Player (Marco)



DRL

- Observations: informazioni dall'ambiente
- Decision: preso da una policy
- Action: esecuzione di una decisione
- Reward: ricompenso data all'agente

OBSERVATIONS E ACTIONS

Osservazioni:

- Osservazione vettoriale: posizione dell'agente (float)
- Osservazione raycast: tramite dei raggi emessi, l'agente è in grado di osservare elementi specifici che si trovano attorno

Azioni:

- 1 azione continua: movimento verticale (a soglio)
- 3 azioni discrete:
  - Movimento orizzontale (0:verso sx 1:fermo 2:verso dx)
  - Fuoco (0:riposo 1:sparo 2:lancio della granata)
  - Salto (0:a terra 1:il salto)

REWARDS E PENALTIES

Ricompense

- Nemico colpito: 1
- Uccisione di un nemico: 10
- Raccolto di un collectible: 30
- Raggiungimento di un checkpoint: 20

Penalità

- Player colpito: -50
- Game over: -500
- Direzione a sx: -1
- Salto a vuoto: -0.5
- Contatto con il bordo dello schermo: -0.02

DIAGRAMMA DELLE CLASSI

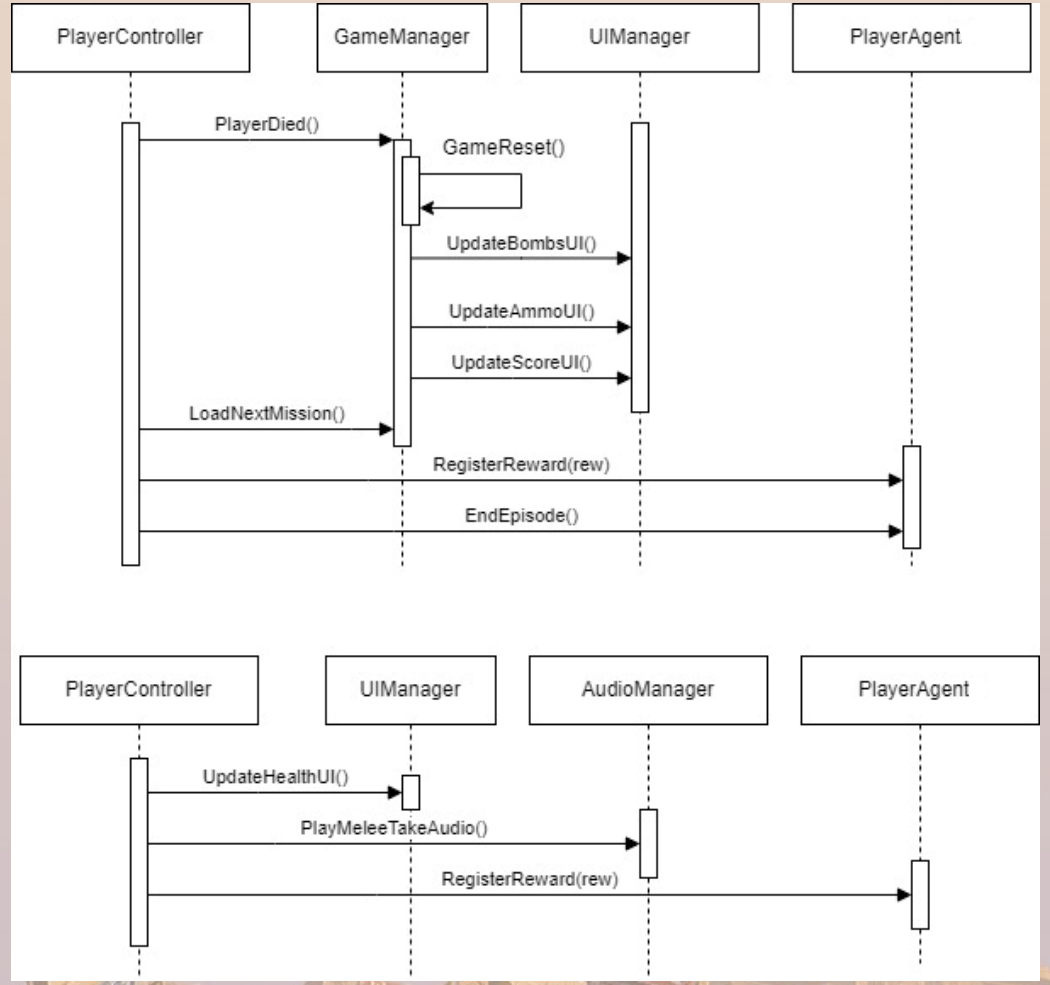
PlayerController
float wasFiring float fireDelta float jumpDelta GameObject bottom GameObject top Animator topAnimator Animator bottomAnimator CinemachineBrain cinemachinebrain Health health PlayerAgent _playerAgent
void Start() void registerHealth() void Update() void OnDead(float damage) void OnHit(float damage) void Jump(int jump) void Fire(int fire) voiid ThrowGranate(int granate) void MoveHorizontally(int moveH) void MoveVertically(int moveV)

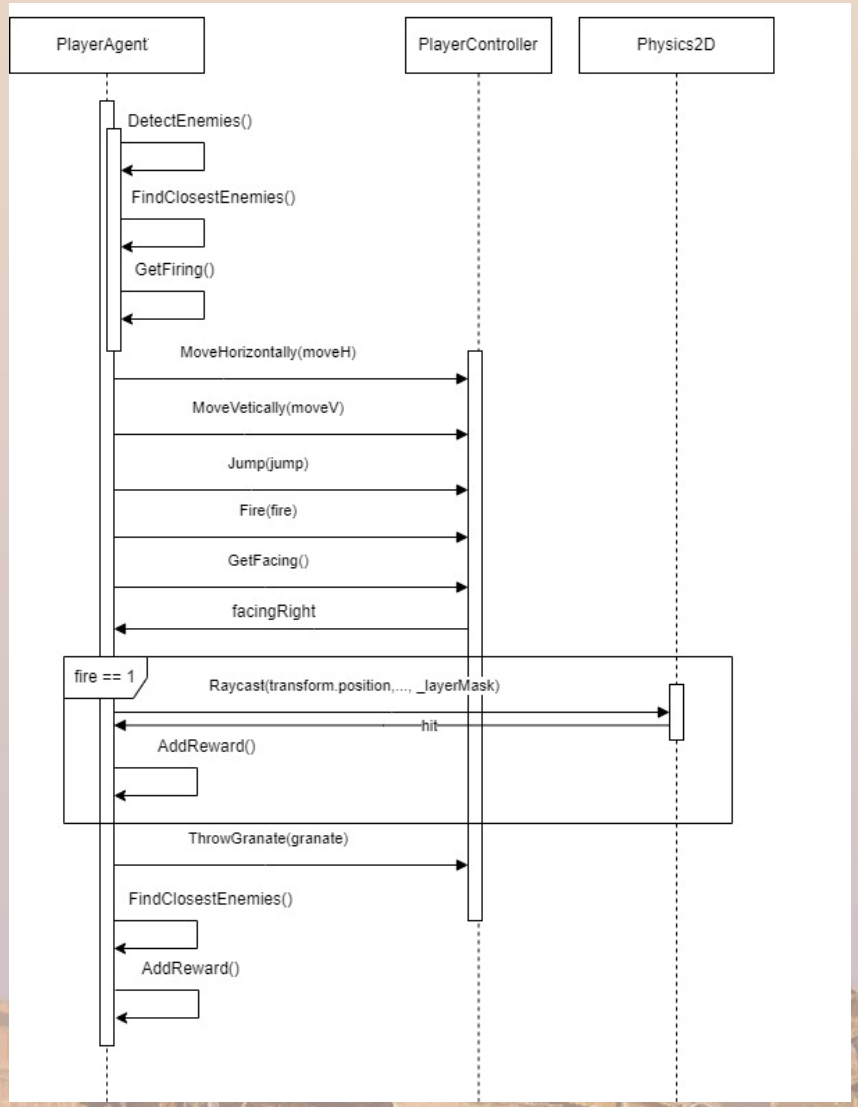
PlayerAgent
PlayerController _playerController bool flagJump bool flagEnemy bool firing RayCastHit2D hit
void OnEpisodeBegin() void Update() void RegisterReward(float rew) void CollectObservations(VectorSensor sensor) void OnActionRecived(ActionBuffers actions void OnCollisionEnter2D(Collision2D collision) int FindClosestEnemies() void CameraAction()

CameraController
PlayerAgent _playerAgent Camera camera2 GameObject player Vector2 playerVPPos Vector2 oldPosition
void Start() void LateUpdate()

GameManager (Singleton)
enum Difficulty (1 2 3) enum Missions (0, 1, 2, 3, 3Boss) bool isGameOver int bombs Missions currentMission LayerMask enemyLayer LayerMask playerLayer LayerMask walkableLayer LayerMask buildingLayer
void Start() void Update() void PlayerDied() void LoadNextMission()

DIAGRAMMA DELLE SEQUENZE





PROBLEMI RISCONTRATI

- Gestione del reset dell'episodio
- Animazioni (loncio dello gronoto/sporo)
- Input system
- Loncio dello gronoto
- Altri Bug del codice (come elicotteri)
- Level design complesso

SOLUZIONI

- Modifico del codice: Comero e GomeManager
- Inserimento di un parametro in input
  - int per indicare il movimento a sinistra o a destra (MoveHorizontally)
  - float per indicare la direzione dello sguardo (MoveVerticolly)
  - int per indicare il salto (Jump)
  - int per indicare lo sparo (Fire)
  - int per indicare il lancio della granata (ThrowGranate)
- Timer e flag per lancio della granata
- Modifico del codice dello spawn dell'elicottero

CONFIGURAZIONE

Numero di esperienze da raccogliere prima di oggiornore il modello di policy

Influenza la rapidità con cui la policy evolve durante il training

Numero di unità per ogni layer fully connected

Numero di strati nascosti della rete neurale

```
behaviors:
  Player Behaviour:
    trainer_type: ppo
    hyperparameters:
      batch_size: 512
      buffer_size: 5120
      learning_rate: 0.00003
      beta: 0.001
      epsilon: 0.3
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
      beta_schedule: linear
      epsilon_schedule: linear
    network_settings:
      normalize: true
      hidden_units: 512
      num_layers: 4
      vis_encode_type: simple
      memory: null
      goal_conditioning_type: hyper
      deterministic: false
```

Numero di esperienze assunte in ogni iterazione

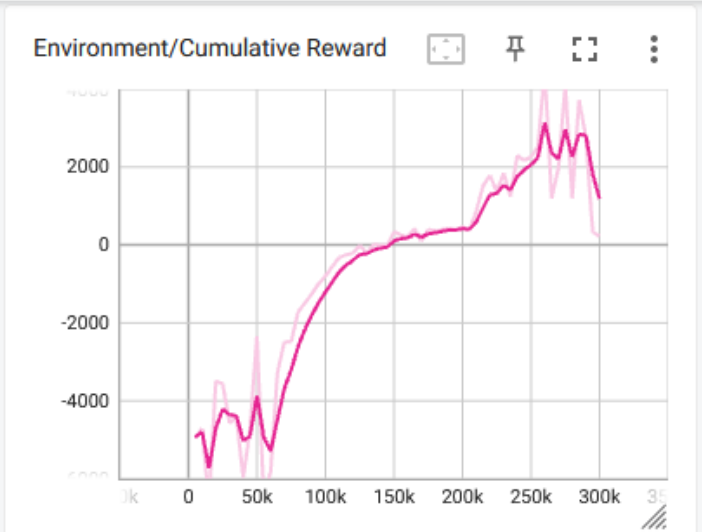
Forza di regolarizzazione dell'entropia

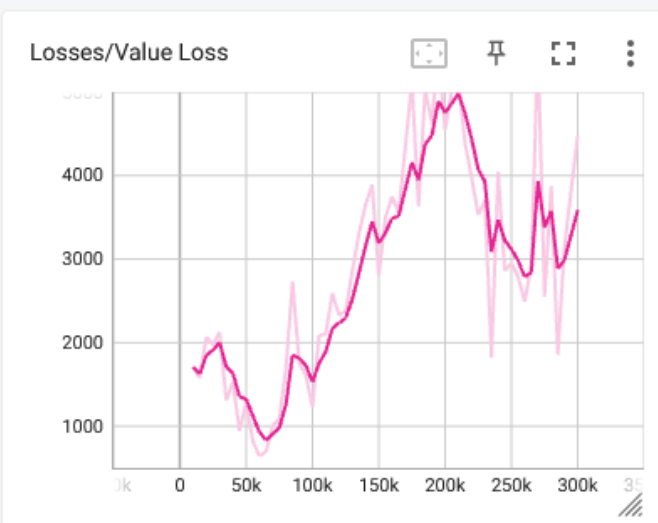
La misura con cui l'agente si affida allo stimo del valore corrente quando calcola una stima aggiornata

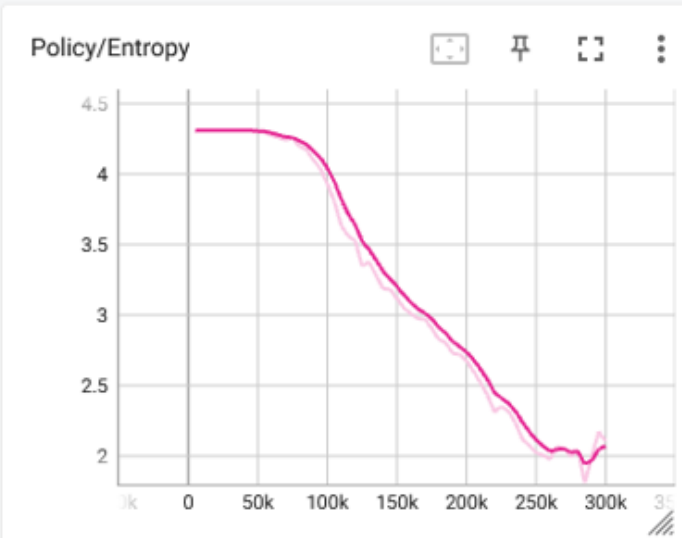
Decision period: 15

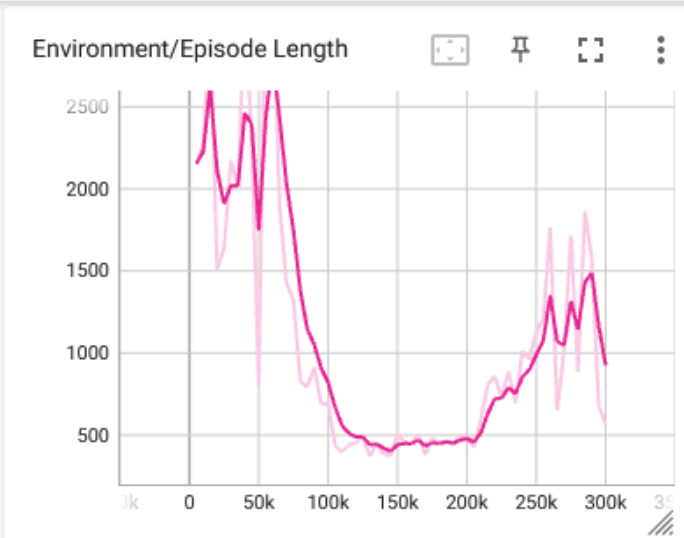
Frequenza con cui l'agente prende una decisione

RISULTATI



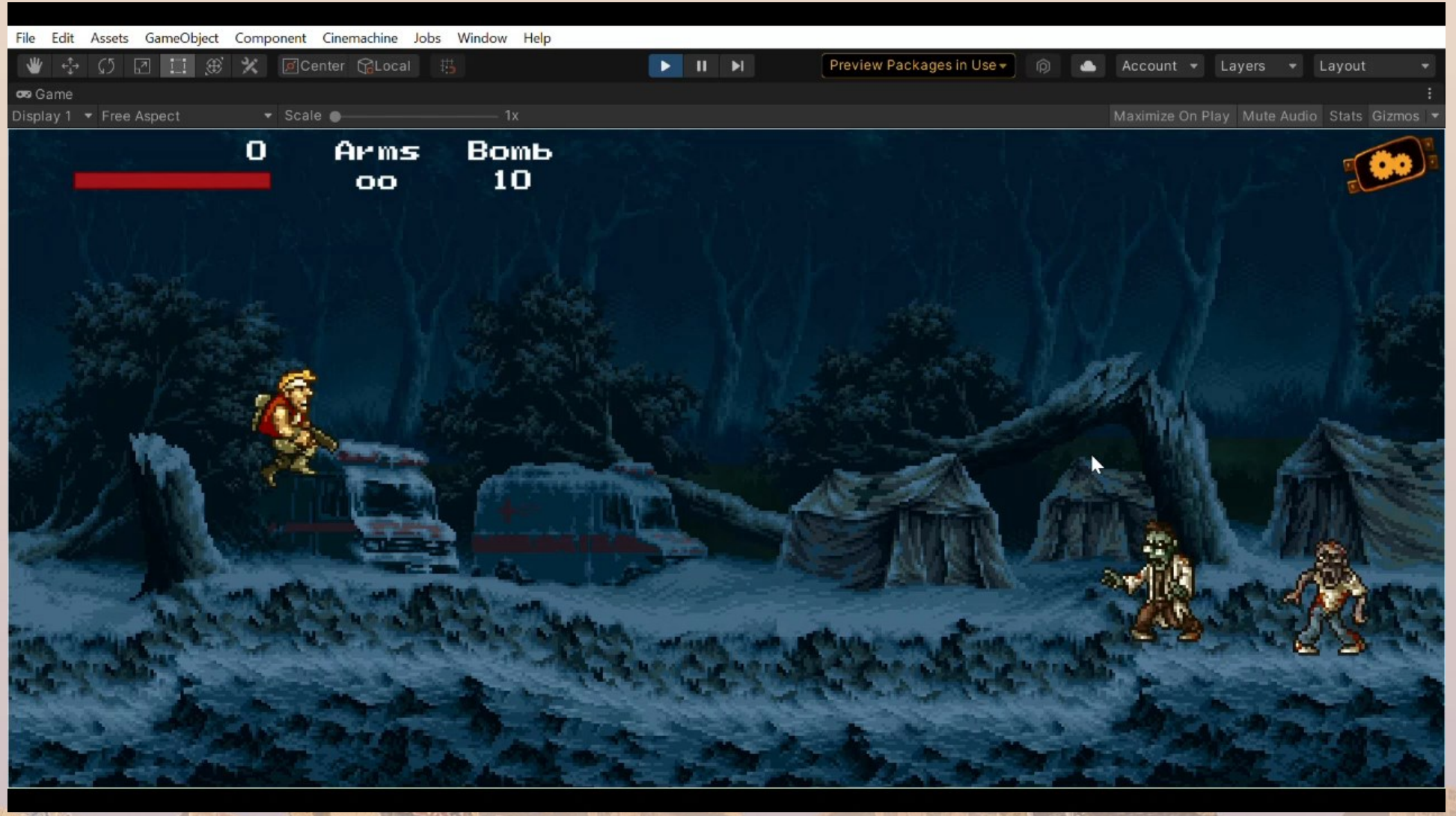






Passi totali: 300'000
Tempo di training: 4h 45m

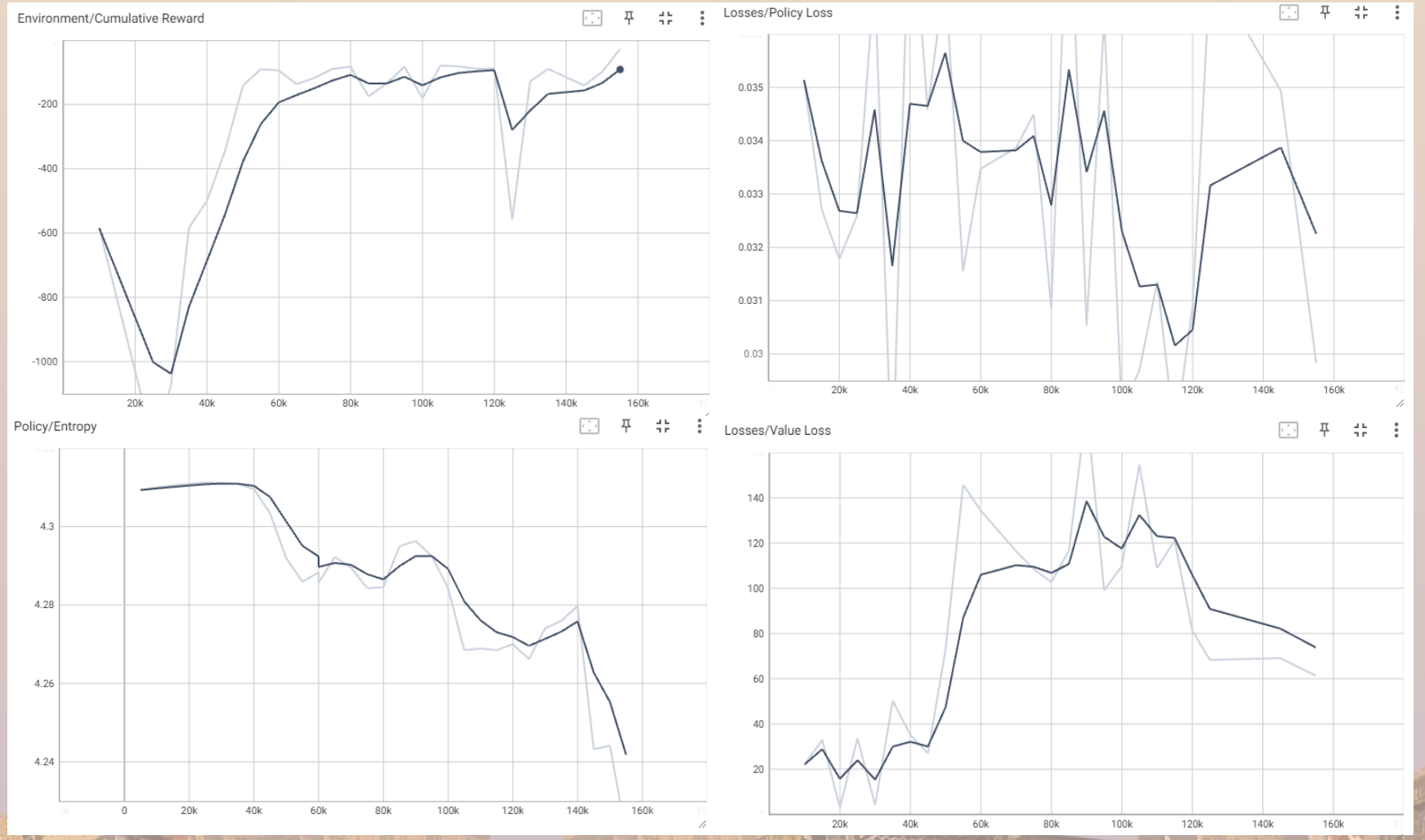
DIMOSTRAZIONE



CONFIGURAZIONI ALTERNATIVE

Differenze

- Colpo a vuoto
- Reward e penalty
- Checkpoint
  - HardCheckPoint
  - DeadPoint

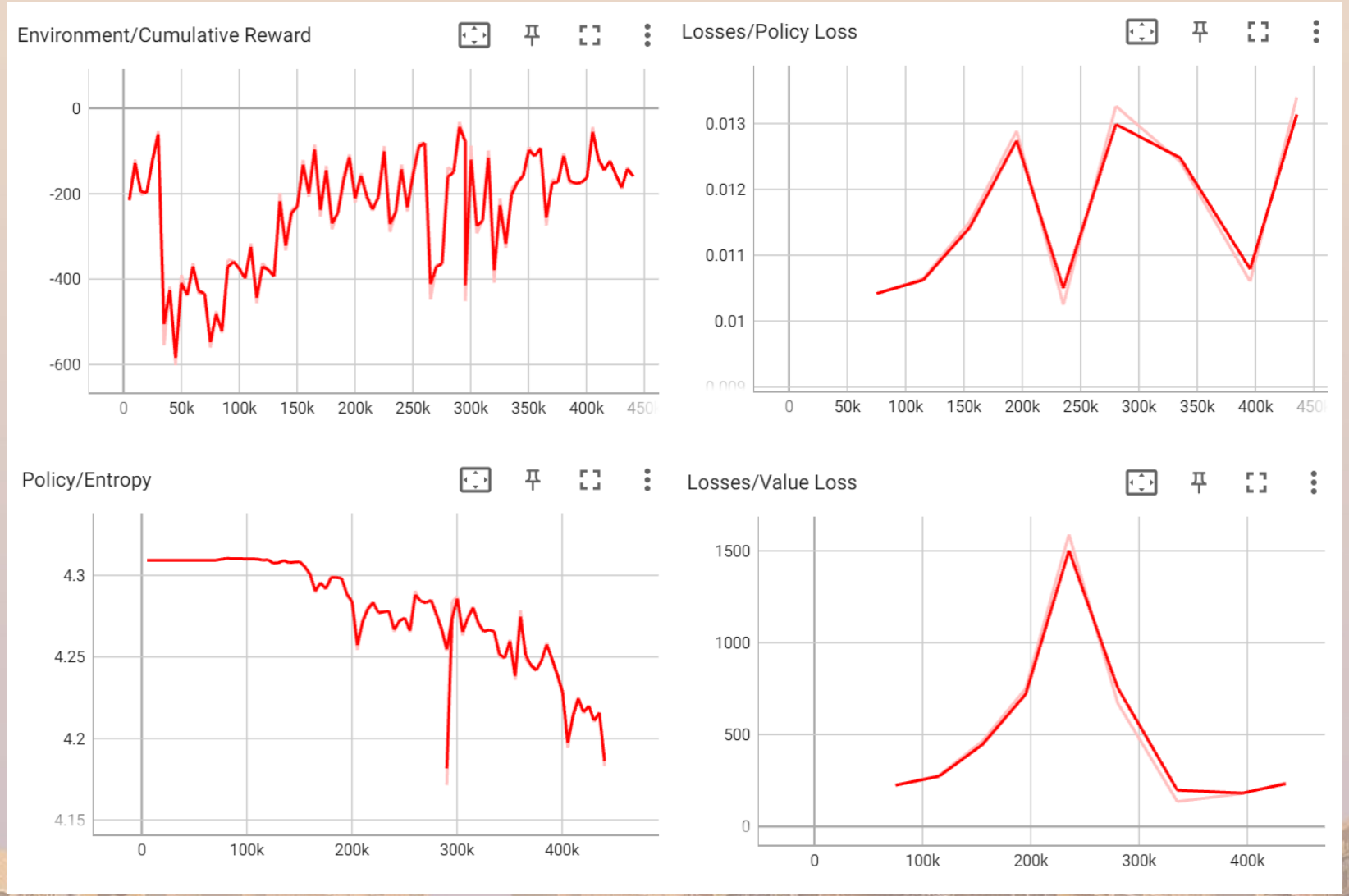


Passi totali: 150'000
Tempo di training: 4h

CONFIGURAZIONI ALTERNATIVE

Differenze

- Batch size: 4096
- Beta: 0.01
- Epoche: 5
- Hidden units: 256
- Colpo a vuoto
- Reward e penalty
- Checkpoint



Passi totali: 440'000
Tempo di training: 8h

SVILUPPI FUTURI

- Diverso impostazione di reword/penolty
- Maggiore potenzo computazionole
- Tempo di oddestromento più lungo
- Modifico del codice per un oddestromento puntuole
- Imitotion leorning

MISSION COMPLETE!

A curo di:

Asconi Christion (S1107369)

Bedetto Alessondro (S1107621)

Recchi Giovanni (S1108636)

Docenti:

Zingoretti Primo

Balloni Emonuele