DEEPRACER: OUR JOURNEY TO THE TOP TEN!

AWS DeepRacer | ML | re:MARS

beSharp | 14 Giugno 2019

Las Vegas è diventata in questi anni il punto di riferimento per gli eventi legati al Cloud di AWS: abbiamo visto in prima persona il re:Invent crescere dai 6.000 partecipanti del 2012 agli oltre 40.000 dell'anno scorso. Una manifestazione oceanica, nella quale è diventato difficile anche semplicemente orientarsi nello scegliere le session a cui partecipare! Deve essere anche per questo che AWS, da quest'anno, ha deciso di affiancare al loro main event alcune conferenze con un focus più specifico, la prima delle quali, **l'AWS re:MARS,** è stata realizzata intorno ai topic più hot del momento: **M**achine Learning, **A**utomazione, **R**obotica e **S**pazio.

beSharp – ovviamente – non poteva mancare.

Tanti grandi nomi per i keynote: **Jeff Bezos, Werner Vogels,** il cofondatore di Coursera **Andrew Ng,** il CEO e fondatore di IRobot **Colin M. Angle** e... **Robert Downey Jr!** Chi meglio di "Iron Man" per parlare delle meraviglie tecnologiche che cambieranno radicalmente la nostra vita già dai prossimi anni? Lo stesso Robert è, tra le altre cose, il co-finanziatore di Footprint Coalition, un'organizzazione privata creata con lo scopo di ripulire il nostro pianeta mediante robotica e tecnologie all'avanguardia.

Moltissime le sessioni curate da aziende disruptive che hanno presentato innovazioni rese possibili dall'intelligenza artificiale: compagnie oil&gas, enti spaziali privati per il lancio di satelliti artificiali e,

soprattutto, l'incredibile Amazon GO, la catena di negozi Amazon in cui è possibile fare la spesa ed uscire senza passare dalle casse. Come dice il motto, **"no lines, no checkout. NO seriously!"**: grazie a tecniche di machine learning e simulazioni in ambienti 3D, chiunque entri in uno store viene etichettato all'ingresso, così da tenere traccia delle azioni e degli articoli prelevati dagli scaffali: all'uscita dal negozio, il sistema di **Amazon GO** elabora il "carrello" ed invia la fattura direttamente sul profilo Amazon personale dell'utente. Un'esperienza incredibile!

Mentre le sessioni ufficiali sarebbero iniziate solo il 5 giugno, già dal primo giorno era possibile seguire dei **workshop** su alcune tematiche specifiche; noi ne abbiamo individuata subito una che stuzzicava in modo particolare le nostre fantasie di nerd: **un deep-dive su AWS DeepRacer!**

Il workshop ci ha veramente colpiti: introdotto al keynote del re:Invent 2018 da Andy Jassy, questo **modellino 4WD** con asse da monster truck è in grado di imparare mediante **Reinforcement Learning** come muoversi autonomamente su percorsi prestabiliti. Descritta da AWS come il modo più semplice per imparare il Machine Learning, AWS DeepRacer mantiene tutte le promesse: la serie di passaggi per scendere in pista e veder correre la propria macchinina è davvero minima. È possibile avere **un modello trainizzato per la guida in poco meno di un'ora,** anche se – ovviamente – per ottenere buoni risultati sono necessari più esperimenti e molto più tempo.

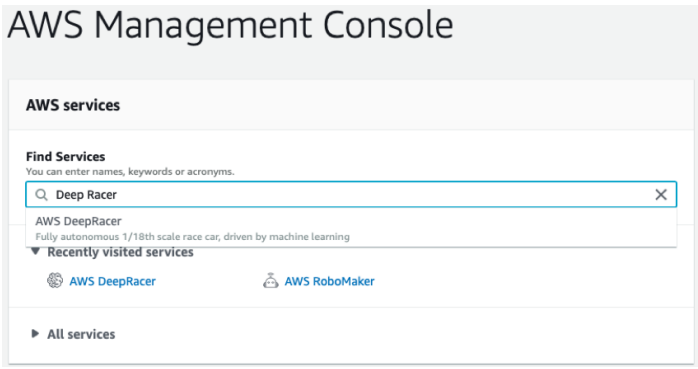
Noi abbiamo subito sperimentato quante più opzioni possibili per migliorare di volta in volta il nostro tempo in pista. Tra le altre cose, il re:MARS è una delle tappe della **DeepRacer League,** una competizione che si svolge in concomitanza con i principali eventi di AWS.

Quale migliore occasione per imparare direttamente sul campo?

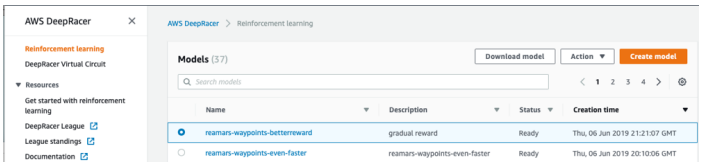
Come funzionano l'AWS DeepRacer e il Reinforcement Learning

Prima di iniziare a parlare di corse e tempi da record è bene dare uno sguardo all'**interfaccia del servizio DeepRacer di AWS,** che è lo strumento per il training del modello. Ci sembra inutile specificarlo, ma è indispensabile possedere un account AWS! 🙂

Appena entrati nella vostra console, cliccate sulla barra dei servizi e cercate "DeepRacer"

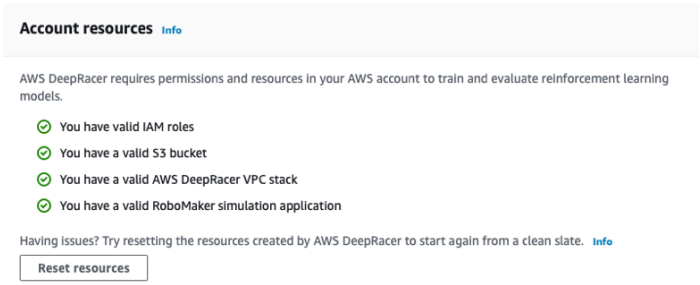


Dalla schermata iniziale è possibile vedere i nostri modelli, verificarne lo stato del training e crearne di nuovi.



Per cominciare, creiamo un nuovo modello cliccando su **“Create model”.**

Questa schermata ci presenta le caratteristiche del modello, oltre a verificare se abbiamo tutti i permessi sull’account per poterlo salvare correttamente.



Nel caso ci fosse qualcosa da sistemare, AWS provvederà a segnalarvelo e ad aiutarvi nel correggerlo.

Inseriamo un nome e una descrizione: scegliete un nome facile da ricordare e soprattutto univoco perché, se vorrete competere in una gara ufficiale, vi verrà richiesto di trasferire il vostro modello su una macchina attraverso una chiavetta USB, e quindi di richiamarlo tra quelli caricati attraverso una app dall’iPad del marshall di pista.

Scegliamo una pista dove trainizzare il modello: noi abbiamo selezionato la prima, che è il circuito ufficiale per la DeepRacer League, “re:Invent 2018”. Potete provare con qualsiasi percorso a disposizione.

Una volta selezionata la pista per il training, **è il momento di creare la funzione di reward** con cui addestreremo il modello. Questo passo è fondamentale per ottenere un macchina performante e

ottenere buoni punteggi nelle gare.

Prima di raccontarvi la nostra esperienza, è utile ricordare brevemente come funziona **il Reinforcement Learning.**

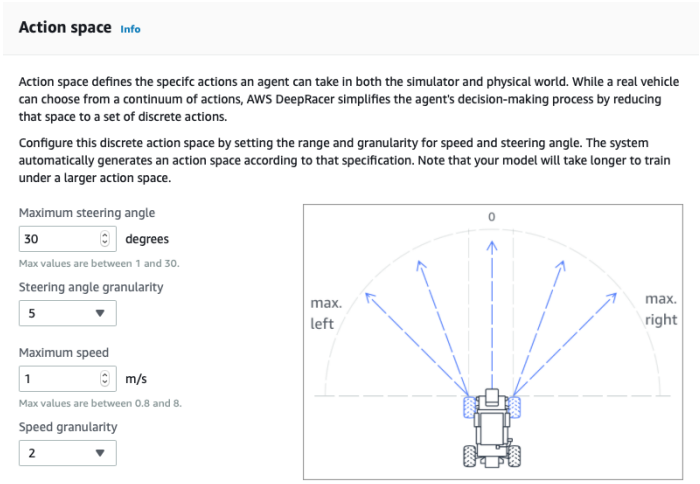
Il Reinforcement Learning **è un sistema di training di reti neurali unsupervised,** ovvero che non necessitano di una ground truth iniziale con la quale adattare i propri pesi. Il Reinforcement Learning effettua invece diverse misurazioni dell'ambiente circostante per massimizzare la propria funzione di reward. Durante questo processo, che viene ripetuto ad oltranza fino al raggiungimento di una soglia di cutoff, i pesi della rete vengono aggiornati volta per volta, andando così ad ottimizzare la rete stessa.

Nel caso della DeepRacer Car abbiamo cominciato con una funzione di reward molto semplice, il cui obiettivo è quello di insegnare alla macchina a rimanere in mezzo alla pista; questo significa far ritornare un valore di reward più alto se, al momento della misurazione, la distanza dal centro della carreggiata è meno della metà della larghezza della strada. In tutti gli altri casi la reward viene ridotta.

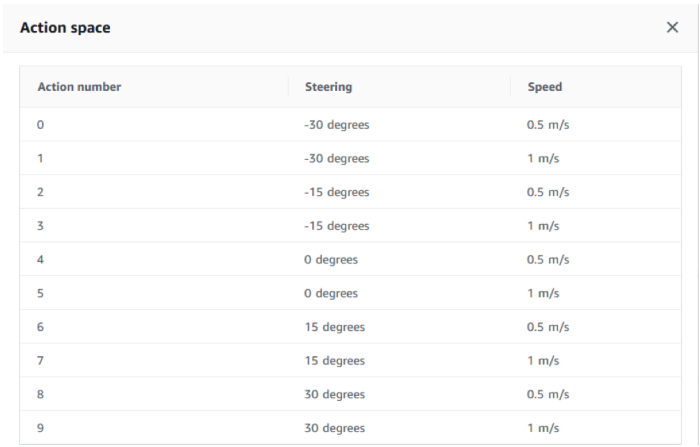
Di seguito un esempio di come costruire la funzione.

```
import math

def reward_function(params):

    '''

   Use square root for center line

    '''

    track_width = params['track_width']

    distance_from_center = params['distance_from_center']

    reward = 1 - math.sqrt(distance_from_center / (track_width/2))

    if reward < 0:

        reward = 0

    return float(reward)
```

Scegliamo i gradi di libertà della nostra 4WD: **velocità massima, angolo di sterzata e livelli di velocità possibili.** La combinazione lineare di queste informazioni definisce quante variazioni la macchina è in grado di gestire, sia nel caso di sterzate che di cambi di velocità.



Questa operazione è fortemente dipendente dalla funzione di training e viceversa: spesso alterazioni nei gradi di libertà sulla funzione di reward producono risultati molto diversi tra di loro.

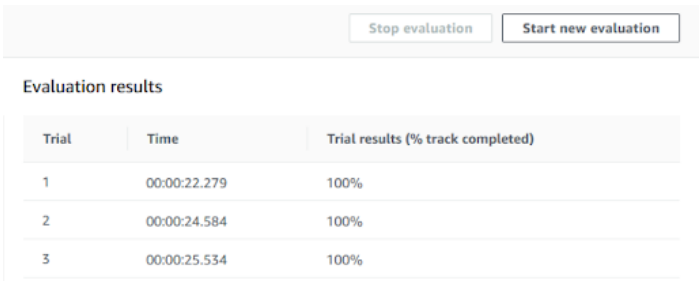


Action number	Steering	Speed
0	-30 degrees	0.5 m/s
1	-30 degrees	1 m/s
2	-15 degrees	0.5 m/s
3	-15 degrees	1 m/s
4	0 degrees	0.5 m/s
5	0 degrees	1 m/s
6	15 degrees	0.5 m/s
7	15 degrees	1 m/s
8	30 degrees	0.5 m/s
9	30 degrees	1 m/s

Inserite queste informazioni, è possibile decidere per quante ore addestrare il modello, fino ad un massimo di 8 ore per singola operazione.

È utile sapere che **è possibile ri-addestrare ulteriormente lo stesso modello** per aumentare il grado di confidenza: quello che abbiamo verificato è che, con un tempo di training di circa 8 – 10 ore, è possibile dare alla macchina una certa confidenza sulla pista, a patto di mantenere un modello semplice.

Effettuiamo alcuni test di confidenza sulla funzione descritta precedentemente: dalla schermata principale del modello clicchiamo su **“Start new evaluation”** e scegliamo il numero di “trial” sulla pista; con tre prove il risultato è il seguente:



Trial	Time	Trial results (% track completed)
1	00:00:22.279	100%
2	00:00:24.584	100%
3	00:00:25.534	100%

Niente male come primo risultato ma non potevamo certo fermarci a 23 secondi! Ecco quindi che entrano in gioco le diverse variabili che DeepRacer fornisce per manipolare la propria funzione di

reward.

```
{

    "all_wheels_on_track": Boolean,    # flag to indicate if the vehicle is on the tra
ck

    "x": float,                        # vehicle's x-coordinate in meters

    "y": float,                        # vehicle's y-coordinate in meters

    "distance_from_center": float,     # distance in meters from the track center

    "is_left_of_center": Boolean,      # Flag to indicate if the vehicle is on the lef
t side to the track center or not.

    "heading": float,                  # vehicle's yaw in degrees

    "progress": float,                 # percentage of track completed

    "steps": int,                      # number steps completed

    "speed": float,                    # vehicle's speed in meters per second (m/s)

    "steering_angle": float,           # vehicle's steering angle in degrees

    "track_width": float,              # width of the track

    "waypoints": [[float, float], … ], # list of [x,y] as milestones along the track
 center

    "closest_waypoints": [int, int]    # indices of the two nearest waypoints.

}
```

Proviamo ad aggiungere alcune di queste informazioni alla nostra funzione di reward:

```
import math

def reward_function(params):

    '''

    Use square root for center line

    '''

    track_width = params['track_width']

    distance_from_center = params['distance_from_center']

    steering = abs(params['steering_angle'])

    speed = params['speed']

    all_wheels_on_track = params['all_wheels_on_track']
```

```
    ABS_STEERING_THRESHOLD = 15



    reward = 1 - (distance_from_center / (track_width/2))**(4)
    if reward < 0:
        reward = 0


    if steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8


    if not (all_wheels_on_track):
        reward = 0


    return float(reward)
```

In particolare, abbiamo aggiunto lo “steering_angle”, la “speed” e la variabile Booleana “all_wheels_on_track” che ci indica se in un determinato momento la macchinina ha tutte le ruote fuori dal tracciato.

Se osserviamo il codice, vediamo che la funzione di reward, dopo essere stata calcolata rispetto alla posizione relativa al centro della pista, viene modificata come segue:

- se stiamo sterzando per più di 15 gradi rispetto al nostro asse centrale, riduciamo del 20% la reward;
- se usciamo di pista la reward è 0.

Così facendo, **chiediamo alla macchina di sforzarsi a rimanere il più possibile in centro pista,** penalizzando fortemente le uscite di strada ed evitando sterzate eccessive.

Ecco il risultato di 4 ore di training:

Evaluation results

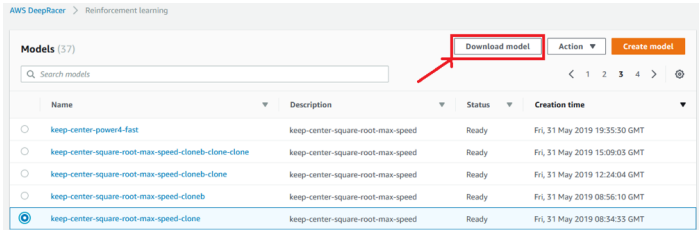
Trial	Time	Trial results (% track completed)
1	00:00:13.870	100%
2	00:00:13.232	100%
3	00:00:03.464	24%
4	00:00:13.218	100%
5	00:00:03.242	21%

Osserviamo come il modello non abbia completato sempre il circuito ma, le volte in cui ci è riuscito, ha mostrato **un notevole miglioramento** dei tempi sul tracciato, portando la media da 22 a poco più di 13 secondi.

Con questo modello ci siamo presentati il primo giorno alla competizione.

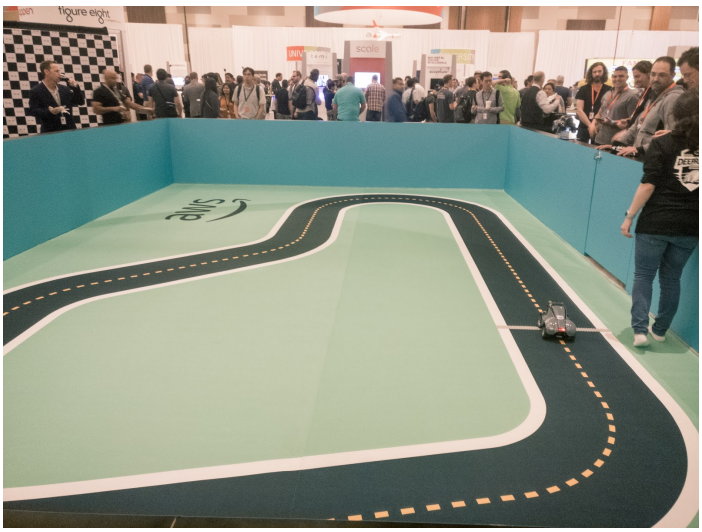
1° giorno: warm-up

Arriviamo al circuito e ci uniamo ad un nutrito gruppo di persone che si preparano a gareggiare. Gli assistenti ci forniscono una chiavetta USB dove caricare il nostro modello. Ecco come fare dalla console di AWS:



Dalla lista dei modelli addestrati, **selezioniamo il modello da scaricare e premiamo "Download model".** Il modello viene scaricato in formato compresso e va copiato nella chiavetta USB, in una directory denominata "Models".

Fatto questo siamo pronti a gareggiare! Aspettiamo il nostro turno osservando le prestazioni dei nostri avversari: alcuni staccano tempi eccellenti, altri escono fuori strada di continuo. Qualche curioso si limita a provare la macchina con i modelli di esempio di AWS.



È il nostro turno: compiliamo il nostro profilo e ci registriamo per la prossima corsa. **Scegliamo "beSharp" e "beSharp-2"** come nome delle nostre macchine: adesso siamo obbligati a fare bella figura!

La chiavetta USB con il modello viene caricata nella DeepRacer Car e un operatore sincronizza il tutto sull'iPad che viene utilizzato dal "pilota" per controllare il comportamento della macchina.

Sull'iPad abbiamo a disposizione 3 comandi:

- Accendi veicolo;
- Spegni veicolo;
- Imposta la velocità: è un input numerico con cui si può aumentare percentualmente la velocità della DeepRacer Car in qualsiasi momento del giro.

Parte il countdown, premiamo “Start” e il veicolo si muove da solo seguendo il tracciato con discreta sicurezza. Le regole sono semplici: abbiamo **quattro minuti per completare quanti più giri possibile,** possiamo uscire di strada solo tre volte pena l’annullamento del giro e possiamo sbagliare tutte le volte che vogliamo. Ai fini della classifica conterà il giro completo più veloce.

Alla fine del nostro primo tentativo, il nostro miglior tempo è di 13 secondi e 964 millesimi: siamo quinti! Ma la strada è ancora lunga, i primi tre in classifica girano intorno ai 9 – 10 secondi e sappiamo che presto molti si avvicineranno a quei tempi: infatti, man mano che il modello viene addestrato per una pista specifica, la macchina performerà sempre meglio su quel tracciato, anche aumentando le velocità.

Parlando con alcuni dei nostri avversari, capiamo che la strategia più diffusa è quella di massimizzare la velocità a discapito della stabilità complessiva in pista, recuperando però questo fattore con giorni interi di training sul tracciato. Qualcuno parla anche di **Waypoint,** un parametro che non avevamo ancora considerato.

A questo punto decidiamo di sperimentare anche noi: mentre continuiamo a provare nuovi settaggi sugli algoritmi già sviluppati, cominciamo a preparare un nuovo modello che vada a sfruttare i waypoint, punti caratteristici del tracciato. In pratica, in fase di training, il sistema si baserà su dei punti ottimali definiti nel percorso e noi andremo a massimizzare la funzione di reward in base alla vicinanza della macchina rispetto a quei punti.

Come approccio estemporaneo, in attesa di un training ottimale sui waypoint, ottimizziamo la funzione come da suggerimenti di alcuni concorrenti:

```
import math

def reward_function(params):

    '''

    Use square root for center line

    '''

    track_width = params['track_width']

    distance_from_center = params['distance_from_center']

    speed = params['speed']

    progress = params['progress']

    all_wheels_on_track = params['all_wheels_on_track']
```

```
    SPEED_TRESHOLD = 6


    reward = 1 - (distance_from_center / (track_width/2))**(4)
    if reward < 0:
        reward = 0


    if speed > SPEED_TRESHOLD:
        reward *= 0.8


    if not (all_wheels_on_track):
        reward = 0



    if progress == 100:
        reward += 100


    return float(reward)
```

In questa nuova funzione andiamo a **ridurre la reward se la velocità della macchina diminuisce e diamo un punteggio molto più alto se la macchina riesce a completare il tracciato.**

2° e 3° giorno: qualche progresso sui tempi, ma la competizione si fa sempre più dura...

Grazie all'algoritmo migliorato **i tempi si riducono** e le nostre macchine riescono a stabilirsi intorno agli 11 – 12 secondi. Non è molto, ma in queste gare anche una frazione di secondo può fare la differenza. Tuttavia, i più preparati cominciano a scalare la classifica: assistiamo ai primi giri da 8 secondi!

Un concorrente giapponese, l'attuale primo in classifica, ci confessa di aver addestrato il modello per più di due giorni consecutivi sullo stesso tracciato, pur di rendere la macchina performante.

La competizione è davvero coinvolgente e intorno alla pista si raduna sempre una nutrita folla di persone. Tra di loro, ci confrontiamo con una ragazza che si rivela essere nientemeno che il capo progetto di AWS DeepRacer! Ci da un paio di consigli su come migliorare le prestazioni del nostro modello e, cosa più importante, è subito d'accordo con noi sulla validità dell'uso dei waypoint.

Al momento siamo dodicesimo e sedicesimo: **è il momento di risalire la classifica e assicurarci un post nella Top Ten!**

4° giorno: la svolta!

All'ultimo giorno presentiamo il **modello appena addestrato che utilizza i waypoint:**

```
import math



def reward_function(params):


    track_width = params['track_width']

    distance_from_center = params['distance_from_center']

    steering = abs(params['steering_angle'])

    direction_stearing=params['steering_angle']

    speed = params['speed']

    steps = params['steps']

    progress = params['progress']

    all_wheels_on_track = params['all_wheels_on_track']

    ABS_STEERING_THRESHOLD = 15

    SPEED_TRESHOLD = 5

    TOTAL_NUM_STEPS = 85


    # Read input variables

    waypoints = params['waypoints']

    closest_waypoints = params['closest_waypoints']

    heading = params['heading']


    reward = 1.0


    if progress == 100:

        reward += 100
```

```
    # Calculate the direction of the center line based on the closest waypoints
    next_point = waypoints[closest_waypoints[1]]
    prev_point = waypoints[closest_waypoints[0]]

    # Calculate the direction in radius, arctan2(dy, dx), the result is (-pi, pi) in radians
    track_direction = math.atan2(next_point[1] - prev_point[1], next_point[0] - prev_
point[0])
    # Convert to degree
    track_direction = math.degrees(track_direction)

    # Calculate the difference between the track direction and the heading direction of the car
    direction_diff = abs(track_direction - heading)

    # Penalize the reward if the difference is too large
    DIRECTION_THRESHOLD = 10.0

    malus=1

    if direction_diff > DIRECTION_THRESHOLD:
        malus=1-(direction_diff/50)
        if malus<0 or malus>1:
            malus = 0
        reward *= malus

    return reward
```

Ci presentiamo la mattina dell'ultimo giorno, saltando il pranzo (per la gloria questo e altro!) e cominciamo a correre: l'algoritmo sembra funzionare bene ad alte velocità ma la stabilità complessiva non è delle migliori. Dopo i primi giri incerti, capiamo come regolarci con la velocità: penultimo giro perfetto, **il commentatore urla: 10 secondi e 272 millesimi!** Nella Top Ten di nuovo, ma mancano diverse ore alla fine della gara e la nostra decima posizione è traballante.

Seconda run: ora sappiamo come gestire manualmente la velocità della macchina nei vari punti del tracciato e possiamo venire incontro al nostro modello per sfruttarlo al meglio.

Dopo qualche tentativo ci rimane giusto il tempo per un ultimo giro: 9.222 secondi, ottava posizione a quattro ore dalla fine.

Torniamo a seguire le sessioni con un occhio sulla classifica in real-time: qualche nuovo concorrente in fondo alla lista e i soliti cinque che si contendono le prime posizioni. Solo giovedì il record mondiale sul tracciato viene infranto per due volte, scendendo sotto i 7.7 secondi!

Il nostro tempo regge! Conquistiamo un onorevole ottavo posto e una DeepRacer Car!

Cosa ci portiamo a casa da questa esperienza? Se lo scopo principale di DeepRacer è quello di insegnare il Machine Learning in modo facile e divertente, missione compiuta! Tra testacoda, colpi di scena e sorpassi in classifica abbiamo avuto modo di studiare e di mettere alla prova alcune semplici ma efficaci nozioni di Reinforcement Learning.

Sperando che ci siano presto nuove occasioni per scendere in pista, **vogliamo condividere con voi alcuni nostri appunti:**

- È meglio utilizzare un modello di Machine Learning specifico per il tracciato su cui vogliamo gareggiare;
- Non è strettamente necessario addestrare per più di 8 ore consecutive un modello ma, per ottenere tempi da record, diventa fondamentale;
- È sempre possibile aumentare la confidenza andando a modificare i gradi di libertà della macchina;
- Utilizzare i Waypoint permette di delineare il percorso ideale;
- Per guadagnare quei millesimi di secondo che fanno la differenza, è possibile variare manualmente la velocità della macchina durante i giri.

Vi è venuta voglia di sfidarci? Controllate sul sito ufficiale di AWS l'elenco dei prossimi eventi, e cominciate ad allenarvi!

https://aws.amazon.com/it/deepracer/

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it