

AI e Teachable Machine

2022-03-28

AI e Teachable Machine

L'intelligenza artificiale è un ramo della computer science che si occupa di creare sistemi in grado di svolgere compiti che normalmente richiederebbero intelligenza umana. Questi compiti possono includere il riconoscimento vocale, la comprensione del linguaggio naturale, la visione artificiale e la presa di decisioni.

Possiamo dire che gli algoritmi di intelligenza artificiale simulano il comportamento umano.

Si basa sul machine learning (apprendimento automatico), un campo di studio che consente ai computer di imparare senza scrivere esplicitamente il programma per risolvere il problema, bensì utilizzando dei dati dai quali apprendere.

Le tecniche di machine learning si dividono in supervisionate o non supervisionate.

Apprendimento supervisionato

L'apprendimento supervisionato si riferisce ad algoritmi che imparano dai dati etichettati, mappando l'ingresso (X) all'output (Y). Ciò significa che l'algoritmo è fornito con esempi da cui imparare prima di fare previsioni.

L'apprendimento supervisionato è classificato in due tipi principali: regressione e classificazione.

- La regressione viene utilizzata quando l'obiettivo è quello di “prevedere” o “calcolare” un numero reale partendo dai dati forniti. Può essere utilizzata per fare previsioni, stime, calcoli statistici. Un esempio sono i sistemi di AI che, in base ai dati satellitari e delle stazioni meteorologiche terrestri stimano che al **77,40%** possa piovere in una determinata località ad una determinata ora.
- La classificazione viene utilizzata per prevedere le categorie, il risultato di output è finito e limitato. Un esempio sono i sistemi di AI che, in base ai dati satellitari e delle stazioni meteorologiche terrestri stimano che una tromba d'aria sia di categoria **A** oppure **B**.

Apprendimento non supervisionato

L'apprendimento non supervisionato funziona con dati che non hanno etichette. L'algoritmo deve trovare modelli e struttura nei dati da solo.

Tra le tecniche di apprendimento non supervisionato troviamo:

- il clustering, ovvero si raggruppano i dati nei cluster (negli insieme) in base alle somiglianze;
- gli alberi decisionali, utilizza una struttura ad albero per categorizzare i dati in base a degli attributi dei dati. Le foglie dell'albero rappresentano le classi finali in cui sono divisi i dati.

Teachable Machine

Teachable Machine è un ambiente didattico per apprendere, insegnare e realizzare lavori con l'intelligenza artificiale, in un contesto creativo e multimediale.

In questo ambiente possiamo creare progetti che sfruttino il riconoscimento audio, video e della posa del corpo.

Riconoscimento delle immagini

L'ambiente guida l'utente nella creazione delle categorie da riconoscere, per poi effettuare il training e visualizzare l'esito.

Nella parte sinistra dell'interfaccia, l'utente deve scegliere le categorie di immagini che il sistema di AI dovrà riconoscere e quindi caricare le immagini per ogni categoria.

Cliccando sul pulsante centrale di addestramento, viene effettuato il **training** del sistema di AI, un processo attraverso il quale il computer impara a riconoscere i vari modelli e a catalogarli come indicato dall'utente.

Di seguito una schermata d'esempio:

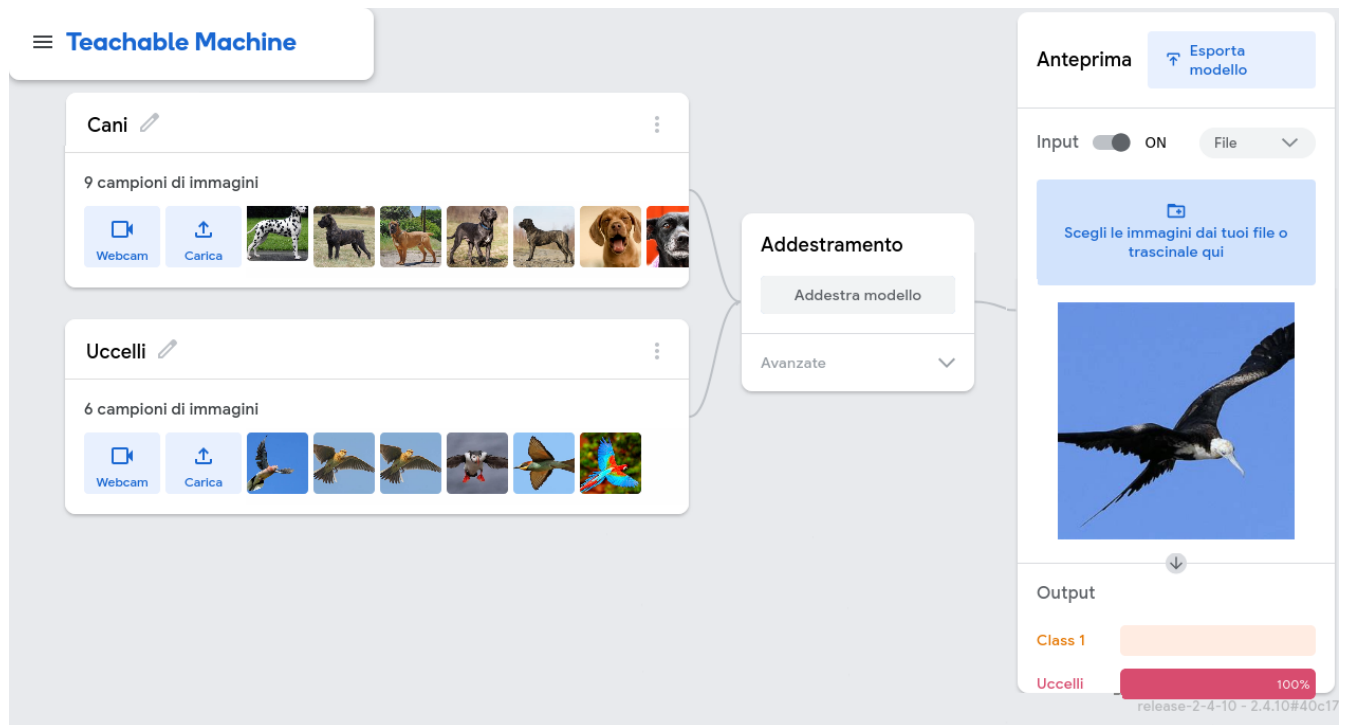


Figure 1: Teachable Machine

Sulla destra dell'interfaccia si può verificare, attraverso la webcam oppure un'immagine caricata, che il sistema effettivamente abbia imparato a catalogare i modelli indicati in fase di training.

Ovviamente se il sistema di AI è stato addestrato su cani e gatti, non potrà catalogare delfini o giraffe.

Riconoscimento dell'audio

Un sistema audio deve tener conto del rumore di fondo, per cui questo tipo di progetti prevede la registrazione di almeno 20 secondi di solo rumore di fondo.

I progetti di Teachable Machine non permettono di caricare .mp3, ma solo di registrarli e di scaricarli successivamente per riutilizzarli.

Il processo di catalogazione è simile a quello delle immagini.

Riconoscimento della posa

Il riconoscimento della posa si basa su sistemi già addestrati al riconoscimento dei punti del corpo ed a stimare per ogni punto la posizione rispetto all'immagine.

Al momento in cui si scrive, il progetto non è esportabile per il linguaggio p5.js.

I sistemi di AI su cui si basa sono due:

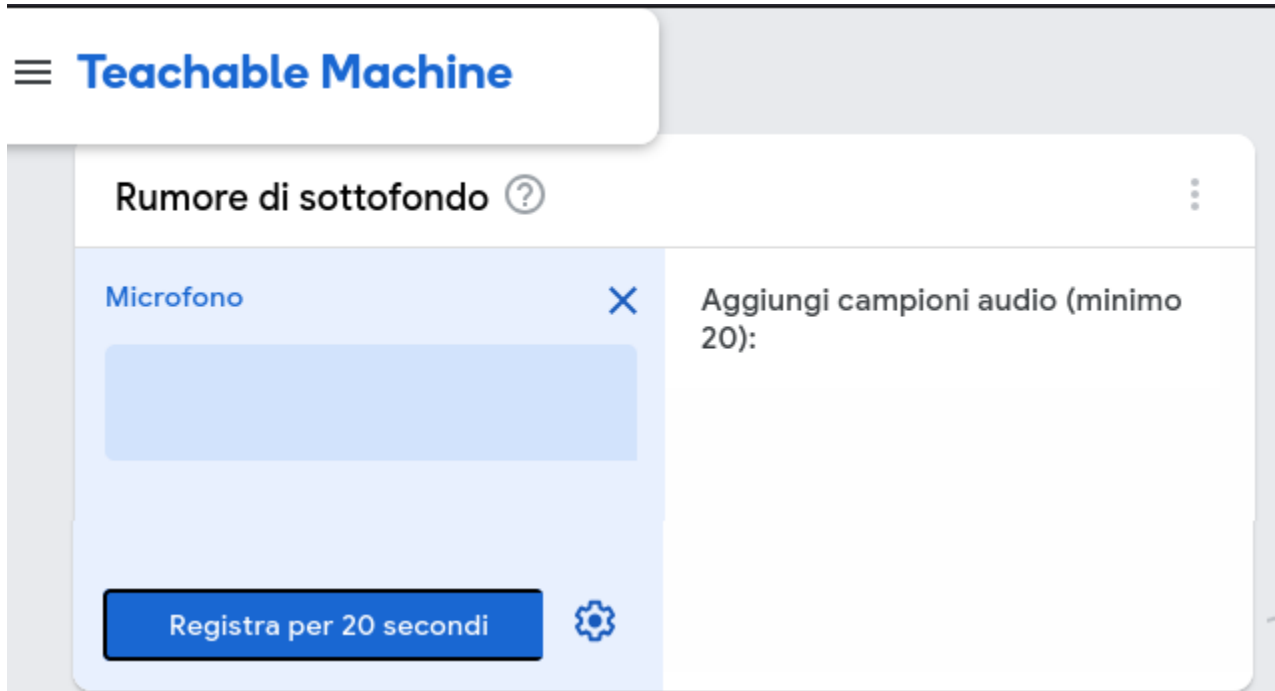


Figure 2: Teachable Machine

- il sistema **MoveNet** che riconosce 17 punti del corpo e ne fornisce le coordinate in uno spazio 2D;
- il sistema **BlazePose** che riconosce 33 punti del corpo e ne fornisce le coordinate in uno spazio 3D di 2 metri cubi.

Cliccando sul pulsante centrale di addestramento, viene effettuato il **training** del sistema di AI, un processo attraverso il quale il computer impara a riconoscere i vari modelli e a catalogarli come indicato dall'utente.

Effettuato il training, è possibile revisionare le immagini caricate e visualizzare i punti riconosciuti.

E' consigliabile cancellare le immagini sulle quali non è stato riconosciuto quasi nessun punto ed effettuare nuovamente il training su quelle rimanenti.

Il sistema di AI come da documentazione è utilizzabile nel seguente modo:

- la funzione `bodyPose.detectStart(video, gotPoses)`; è invocata per il riconoscimento di un modello presente nel video;
- la funzione `gotPoses(results)` è invocata per restituire la posa, in particolare:
 1. `poses[0].keypoints` contiene l'insieme di punti, ognuno accessibile con la sintassi `poses[0].keypoints[0]`, `poses[0].keypoints[1]`, ..., oppure accessibile per nome, come ad esempio `poses[0].left_eye`, `poses[0].right_shoulder`;
 2. ogni punto è identificato da una coppia di coordinate (x, y) utilizzabili con entrambe le sintassi evidenziate al punto precedente, sia `poses[0].keypoints[0].x` e `poses[0].keypoints[0].y`, sia `poses[0].left_eye.x` ed `poses[0].left_eye.y`.

In allegato un archivio del progetto in formato compresso con estensione `.zip`.

Progetto di posa

Lista punti noti per MoveNet

0. `nose`;
1. `left_eye`;
2. `right_eye`;
3. `left_ear`;
4. `right_ear`;

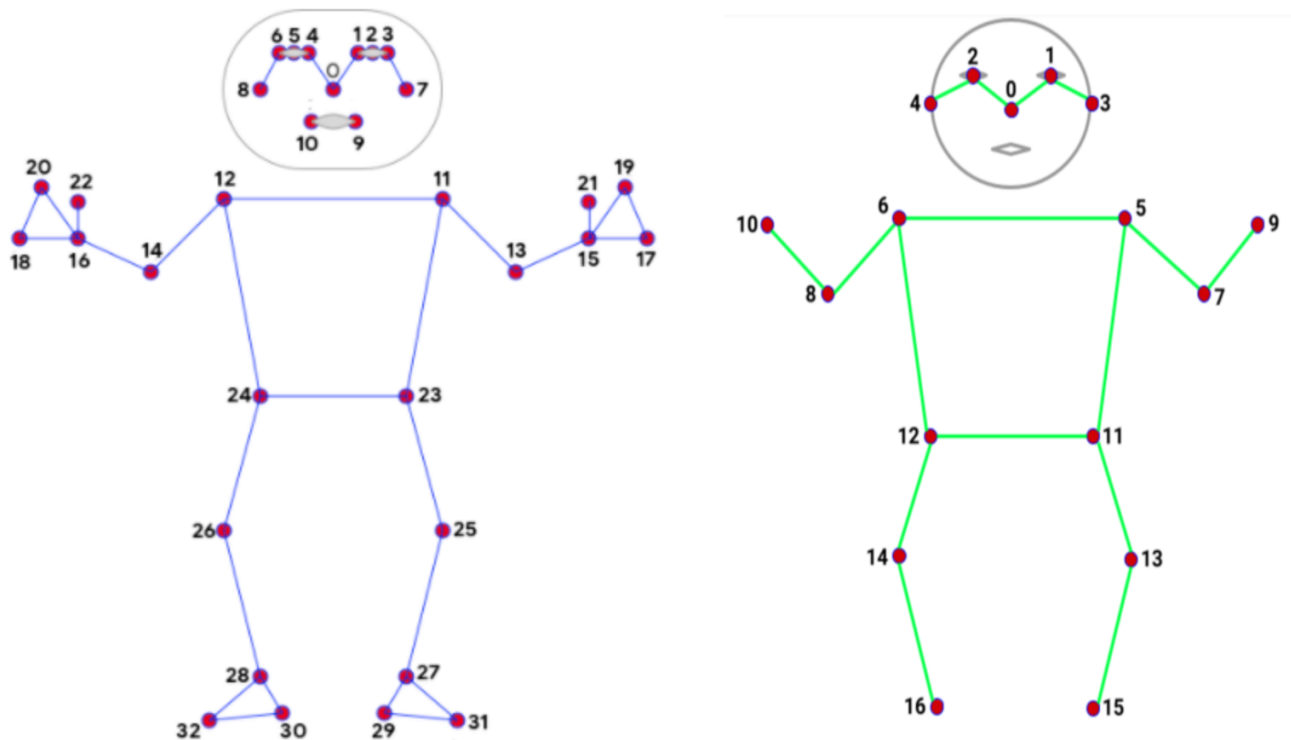


Figure 3: Teachable Machine

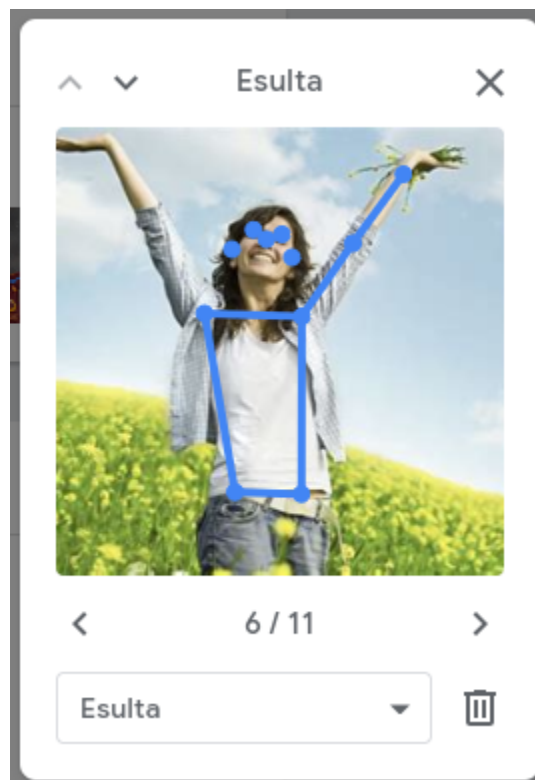


Figure 4: Teachable Machine

- 5. left_shoulder;
- 6. right_shoulder;
- 7. left_elbow;
- 8. right_elbow;
- 9. left_wrist;
- 10. right_wrist;
- 11. left_hip;
- 12. right_hip;
- 13. left_knee;
- 14. right_knee;
- 15. left_ankle;
- 16. right_ankle;

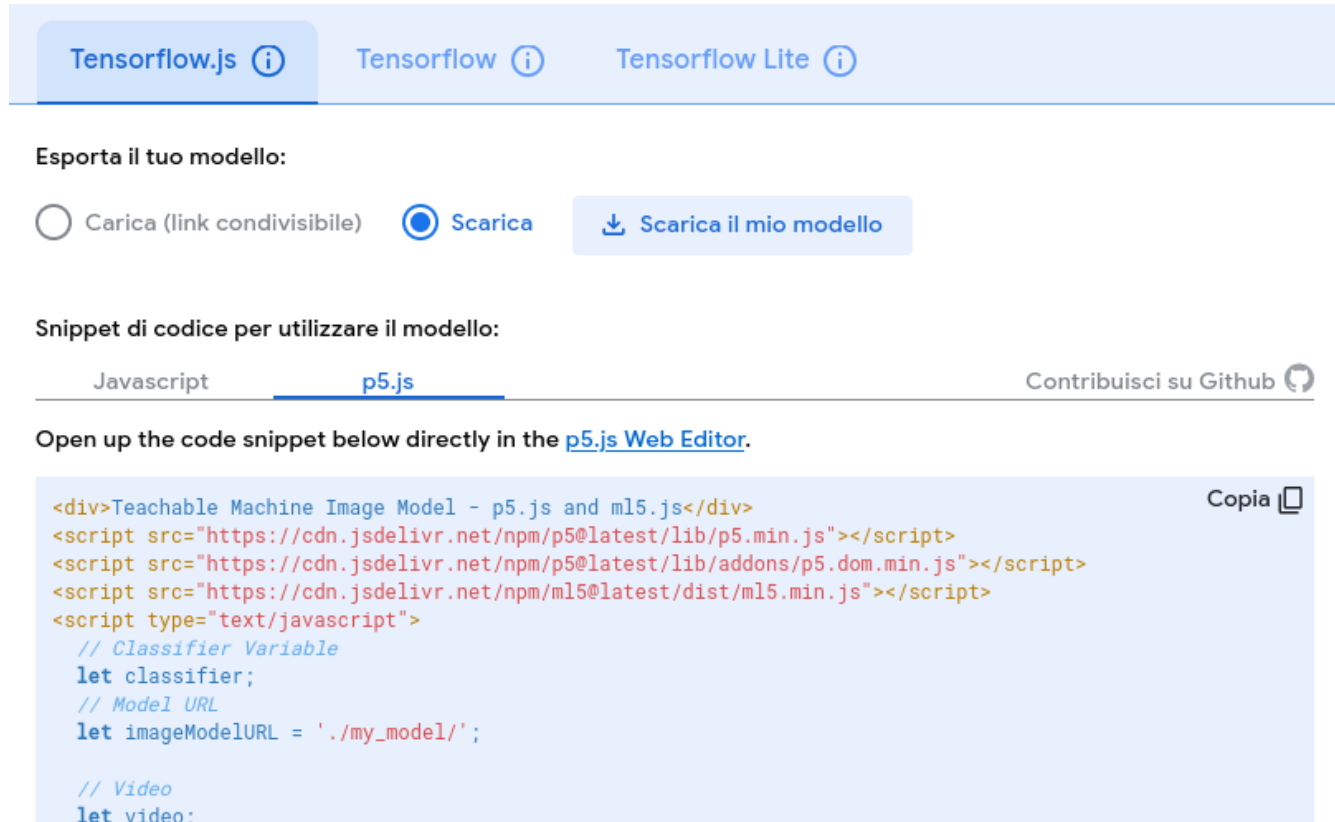
Lista punti noti per BlazePose

- 0. nose
- 1. left_eye_inner
- 2. left_eye
- 3. left_eye_outer
- 4. right_eye_inner
- 5. right_eye
- 6. right_eye_outer
- 7. left_ear
- 8. right_ear
- 9. mouth_left
- 10. mouth_right
- 11. left_shoulder
- 12. right_shoulder
- 13. left_elbow
- 14. right_elbow
- 15. left_wrist
- 16. right_wrist
- 17. left_pinky
- 18. right_pinky
- 19. left_index
- 20. right_index
- 21. left_thumb
- 22. right_thumb
- 23. left_hip
- 24. right_hip
- 25. left_knee
- 26. right_knee
- 27. left_ankle
- 28. right_ankle
- 29. left_heel
- 30. right_heel
- 31. left_foot_index
- 32. right_foot_index
- 33. bodyCenter
- 34. forehead
- 35. leftThumb
- 36. leftHand
- 37. rightThumb
- 38. rightHand

Esportazione“ del modello

Nell’ambiente di Teachable Machine è possibile esportare il modello addestrato per utilizzarlo nello sviluppo dei propri programmi.

Per esportare un modello, bisogna cliccare sul pulsante “Esporta modello” e poi scegliere il linguaggio di programmazione tra quelli proposti e se utilizzarlo online o effettuare il download.



```
<div>Teachable Machine Image Model - p5.js and ml5.js</div>
<script src="https://cdn.jsdelivr.net/npm/p5@latest/lib/p5.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/p5@latest/lib/addons/p5.dom.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/ml5@latest/dist/ml5.min.js"></script>
<script type="text/javascript">
  // Classifier Variable
  let classifier;
  // Model URL
  let imageModelURL = './my_model/';

  // Video
  let video;
```

Figure 5: Teachable Machine

Il progetto JavaScript consiste in 4 file, la pagina web ed i tre file del sistema di AI.

Al momento in cui si scrive, le versioni delle librerie JavaScript sono errate, per poter far funzionare il progetto è necessario sostituirle indicando la versione:

```
<script src="https://cdn.jsdelivr.net/npm/p5@1.2.0/lib/p5.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/p5@0.7.3/lib/addons/p5.dom.min.js"></script>
<script src="https://unpkg.com/ml5@0.12.2/dist/ml5.min.js"></script>
```

Ulteriore attenzione deve essere posta nell’indicare la cartella in cui si trovano i file di AI:

```
let imageModelURL = './cartella/'; // FIX MODEL PATH

classifier = ml5.imageClassifier(imageModelURL + 'model.json');
```

Una volta effettuato ciò è necessario avviare un proprio server locale sul quale caricare il progetto (oppure caricarli su un vero e proprio server online) e verificare che il progetto funzioni correttamente.

Il progetto realizzato utilizza il sistema di AI nel seguente modo:

- la funzione `classifier.classify(flippedVideo, gotResult);` è invocata per il riconoscimento di un modello presente nell’immagine (il video frame corrente);
- la funzione `gotResult(error, results)` è invocata per restituire un errore o una serie di risultati, in particolare:

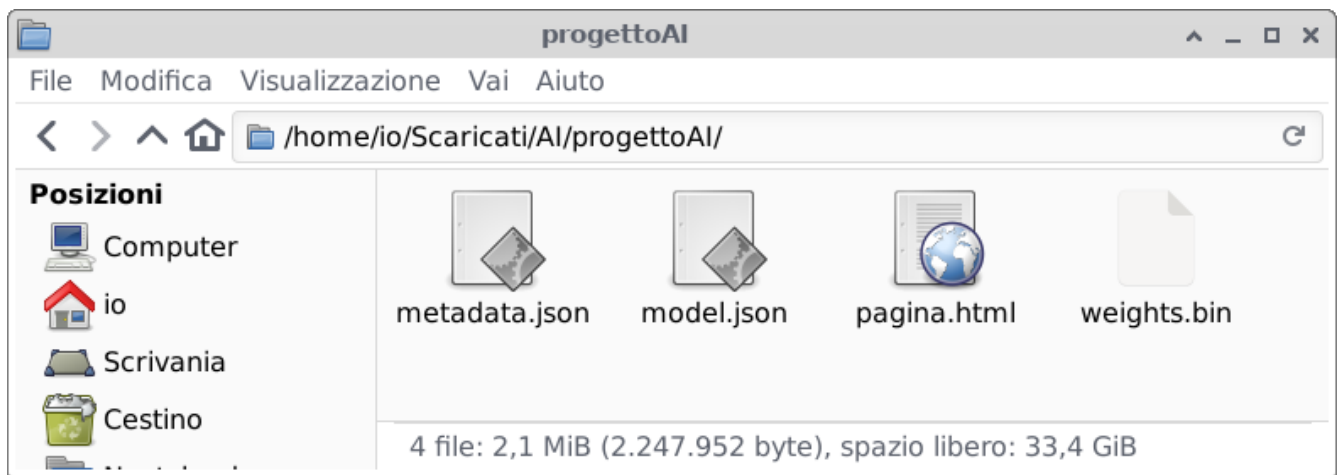


Figure 6: Teachable Machine

1. `results[0].label` contiene la catalogazione del modello;
2. `results[0].confidence` contiene l'accuratezza del riconoscimento (in percentuale).

Si può quindi personalizzare la logica del progetto in base alla classificazione ottenuta dal sistema di AI, ad esempio nella funzione `draw` si può aggiungere:

```
if (label == 'Cani') {
  fill('red');
  circle(200, 200, 20);
}
if (label == 'Uccelli') {
  fill('green');
  rect(200, 200, 40, 20);
}
```

In allegato un archivio del progetto in formato compresso con estensione `.zip`.

Teachable Machine

Playing Mortal kombat

Il progetto didattico per utilizzare l'AI con mortal kombat al link:

<https://github.com/mgechev/mk.js>

Il progetto prevede 4 tipi di modalità di gioco:

- **basic**: un solo giocatore si muove utilizzando la tastiera;
- **multiplayer**: due giocatori in movimento utilizzando entrambi la tastiera;
- **network**: si gioca online, è necessario avere il server;
- **webcaminput**: due giocatori in movimento utilizzando la webcam.

Il progetto è presentato dalla pagina web `index.html` che contiene il codice di avvio del gioco. In particolare in questa pagina è impostata la variabile `option` che indica la configurazione con la modalità di gioco:

```
var options = {
  arena: {
    container: document.getElementById('arena'),
    arena: mk.arenas.types.THRONE_ROOM
  },
  fighters: [{ name: 'Subzero' }, { name: 'Kano' }],
  callbacks: { ... }
  isHost: 'no',
```

```

gameName: "mortalKombat",
gameType: 'basic'
}

```

In particolare la variabile `gameType` indica la modalità di gioco sopra elencate, la variabile `isHostsi` imposta solo se la modalità di gioco è online.

L'arena può essere TOWER oppure THRONE_ROOM, mentre i nomi dei combattenti sono indicati.

Il progetto non prevede l'utilizzo di `p5.js` ne di `Teachable Machine`, ma lo sviluppatore può aggiungere facilmente questa casistica.

Struttura del codice

Controller del gioco

Nel file `mk.js` è definita la funzione di avvio del gioco `start()` che a seconda della modalità di gioco configurata (nella variabile `option.gameType`), crea il controller di movimento specifico. La funzione è realizzata in questo modo:

```

mk.start = function (options) {
  var type = options.gameType || 'basic',
      promise = new mk.Promise();
  type = type.toLowerCase();
  switch (type) {
    case 'basic':
      mk.game = new mk.controllers.Basic(options);
      break;
    case 'network':
      mk.game = new mk.controllers.Network(options);
      break;
    case 'multiplayer':
      mk.game = new mk.controllers.Multiplayer(options);
      break;
    case 'webcaminput':
      mk.game = new mk.controllers.WebcamInput(options);
      break;
    default:
      mk.game = new mk.controllers.Basic(options);
  }
  mk.game.init(promise);
  return promise;
};

```

E' facile intuire che a seconda della modalità di gioco configurata, viene **creato** uno specifico controller del gioco. In particolare il codice è strutturato per **definire** i controller di movimento `Base`, `Basic`, `WebcamInput`, `Multiplayer` e `Network`, ognuno con un metodo `prototype()` e un metodo `prototype._initialize()`, di seguito elencati.

```

mk.controllers.Base
mk.controllers.Basic
mk.controllers.Basic.prototype
mk.controllers.Basic.prototype._initialize
mk.controllers.WebcamInput
mk.controllers.WebcamInput.prototype
mk.controllers.WebcamInput.prototype._initialize
mk.controllers.Multiplayer
mk.controllers.Multiplayer.prototype
mk.controllers.Multiplayer.prototype._initialize
mk.controllers.Network

```



```
mk.controllers.Network.prototype
mk.controllers.Network.prototype._initialize
```

Mosse dei combattenti

I combattenti sono 2 e sono definiti nell'array di gioco chiamato `fighters`. E' possibile indicare il primo o il secondo combattente per posizione: `fighters[0]` e `fighters[1]`.

Le “mosse” che un combattente può effettuare sono definite nel file `mk.js` nella seguente sezione di codice:

```
mk.moves.types = {
  STAND      : 'stand',
  WALK       : 'walking',
  WALK_BACKWARD : 'walking-backward',
  SQUAT      : 'squatting',
  STAND_UP   : 'stand-up',
  HIGH_KICK  : 'high-kick',
  JUMP       : 'jumping',
  FORWARD_JUMP : 'forward-jump',
  BACKWARD_JUMP : 'backward-jump',
  LOW_KICK   : 'low-kick',
  LOW_PUNCH  : 'low-punch',
  HIGH_PUNCH : 'high-punch',
  FALL       : 'fall',
  WIN        : 'win',
  ENDURE     : 'endure',
  SQUAT_ENDURE : 'squat-endure',
  UPPERCUT   : 'uppercut',
  SQUAT_LOW_KICK : 'squat-low-kick',
  SQUAT_HIGH_KICK : 'squat-high-kick',
  SQUAT_LOW_PUNCH : 'squat-low-punch',
  KNOCK_DOWN : 'knock-down',
  ATTRACTIVE_STAND_UP : 'attractive-stand-up',
  SPIN_KICK  : 'spin-kick',
  BLOCK      : 'blocking',
  FORWARD_JUMP_KICK : 'forward-jump-kick',
  BACKWARD_JUMP_KICK : 'backward-jump-kick',
  BACKWARD_JUMP_PUNCH : 'backward-jump-punch',
  FORWARD_JUMP_PUNCH : 'forward-jump-punch'
};
```

Il metodo per far effettuare una mossa ad un combattente è `._moveFighter(f, mossa)`, questa funzione prende in input un combattente ed una mossa e realizza l'animazione grafica.

Ogni controller definisce un proprio metodo per selezionare la mossa tra le mosse indicate precedentemente, selezionare il combattente dall'array indicato precedentemente ed applicare l'azione.

Aggiungere P5JS e Teachable Machine

Tenendo presente la struttura del codice descritta precedentemente, per aggiungere un controller per P5JS dobbiamo modificare il file `mk.js` creando:

1. un opzione nel metodo `start` per creare un controller:

```
case 'p5js':
  mk.game = new mk.controllers.P5JS(options);
  break;
```

2. il controller P5JS, conservando la stessa struttura degli altri controller di gioco:

```

mk.controllers.P5JS = function (options) {
  mk.controllers.Basic.call(this, options);
  mk.classifier = options.classifier;
};

mk.controllers.P5JS.prototype = new mk.controllers.Basic();

mk.controllers.P5JS.prototype._initialize = function () {
  this._player = 0;
  this._callbacks.setMK(mk);
  this._callbacks.setP5JS(this);
};

```

3. il metodo per selezionare la mossa tra le mosse indicate precedentemente, selezionare il combattente dall'array indicato precedentemente ed applicare il movimento al combattente:

```

mk.controllers.P5JS.prototype.movimento = function (categoriaAI) {
  // per ora non fa nulla
  let mossa = mk.moves.types.STAND;

  // se trovo il movimento, lo fa
  if (categoriaAI == 'pugno') {
    mossa = mk.moves.types.HIGH_PUNCH;
  }
  if (categoriaAI == 'calcio') {
    mossa = mk.moves.types.HIGH_KICK;
  }

  var self = this;
  let f = this.fighters[0];
  this._moveFighter(f, mossa);
  return mossa;
}

```

4. Creato il controller, dobbiamo modificare il file `index.html` per aggiungere sia il progetto di `p5.js` sia quello di Teachable Machine.

Aggiungiamo le librerie di `p5.js` e di AI per farle convivere con `mk`:

```

<!-- FIX VERSION FOR ALL PATH -->
<script src="https://cdn.jsdelivr.net/npm/p5@1.2.0/lib/p5.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/p5@0.7.3/lib/addons/p5.dom.min.js"></script>
<script src="https://unpkg.com/ml5@0.12.2/dist/ml5.min.js"></script>

```

Nello script di Mortal Kombat aggiungiamo le variabili per il gioco e per la modalità specifica di gioco P5JS:

```

let mortalKombat;
let p5js;

```

Aggiungiamo le funzioni per controllare il gioco dall'esterno, tramite configurazione:

```

var options = {
  arena: {
    container: document.getElementById('arena'),
    arena: mk.arenas.types.THRONE_ROOM
  },
  fighters: [{ name: 'Subzero' }, { name: 'Kano' }],
  callbacks: {
    attack: function (f, o, l) {
      if (o.getName() === 'kano') {

```

```

        setLife($('player2Life'), o.getLife());
    } else {
        setLife($('player1Life'), o.getLife());
    }
},
setMK : function(g) {
    mortalKombat = g;
},
setP5JS : function(g) {
    p5js = g;
}
},
isHost: 'no',
gameName: "mortalKombat",
gameType: 'p5js'
};

```

Aggiungiamo lo script standard di p5.js

```

function setup() {
    createCanvas(320, 260);
}

function draw() {
    background(0);

    circle(200, 200, 20);
}

```

Verifichiamo che funzioni aggiungendo l'istruzione `p5js.movimento(key)`

```

function draw() {
    background(0);

    circle(200, 200, 20);
}

```

A questo punto è necessario aggiungere il riconoscimento dei movimenti con **Teachable Machine**, incollando il codice necessario ed utilizzando il metodo `getResult(error, results)` per recuperare il movimento:

```

let categoriaAI = results[0].label;
let movimentoMK = p5js.movimento(categoriaAI);

```

Un esempio di progetto è in allegato.