Simone Tanzi 978586

PWM IMAGE RECOGNITION AI

2023

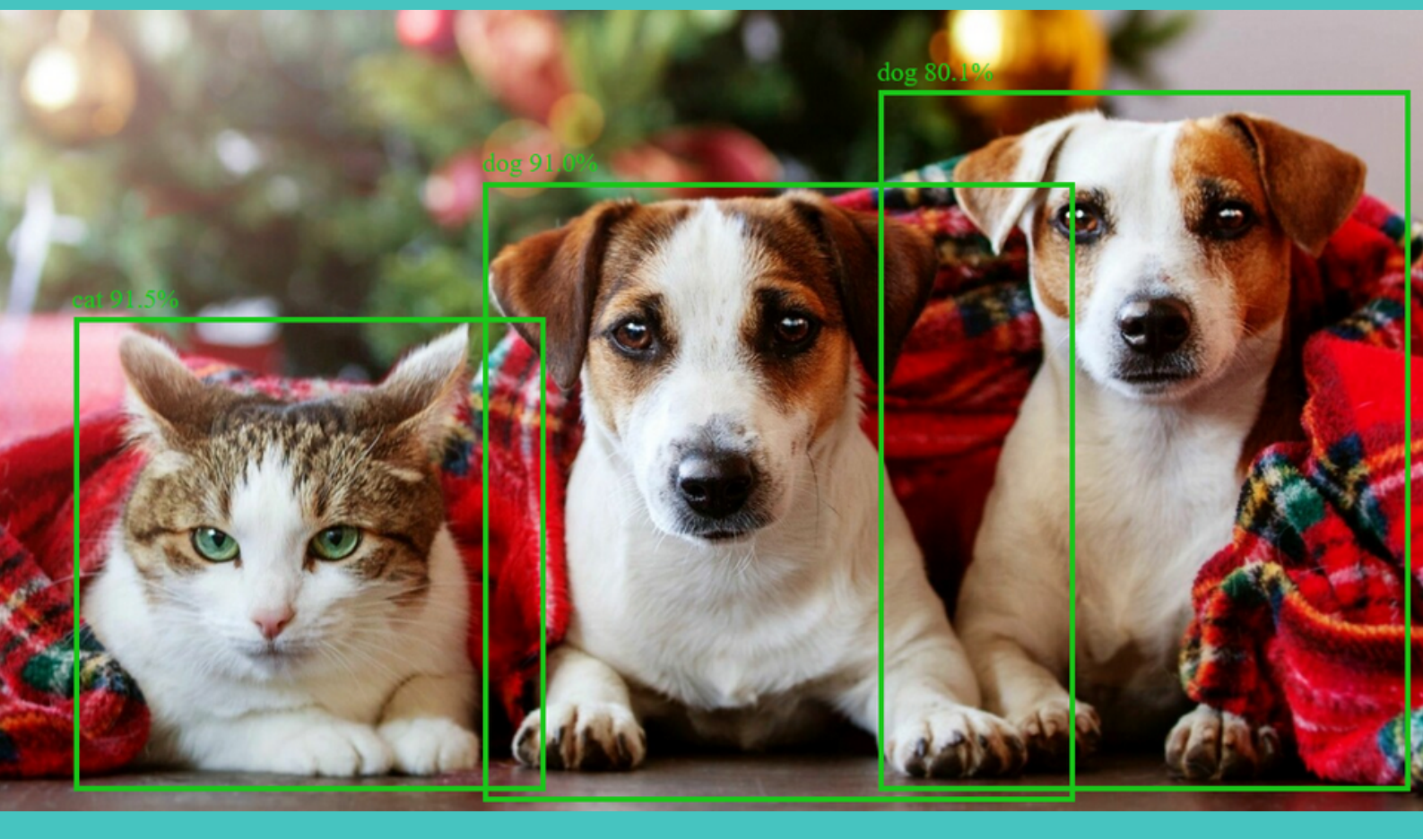


TABLE OF CONTENTS

INTRODUZIONE

Ho sviluppato un'applicazione web che utilizza l'intelligenza artificiale per riconoscere gli oggetti presenti in un'immagine fornita dall'utente. Utilizzando il modulo di rilevamento degli oggetti COCO-SSD basato su Tensorflow e integrato in React.js. COCO-SSD può individuare fino a 80 classi di oggetti, compresi animali e automobili. L'applicazione fornisce una percentuale di accuratezza nella rilevazione degli oggetti

Modello di valore

l'applicazione viene utilizzata principalmente per divertimento o intrattenimento, il modello di valore più adatto potrebbe essere quello pubblicitario o freemium, in cui l'applicazione viene offerta gratuitamente agli utenti e genera profitti attraverso la pubblicità

Destinatari

l'applicazione ha lo scopo principale di dimostrare come è possibile utilizzare strumenti complessi come l'intelligenza artificiale sul web. è possibile utilizzare il prodotto sia da personal computer che da cellulare con un interfaccia principale molto semplice e pulita

Aspetti tecnologici

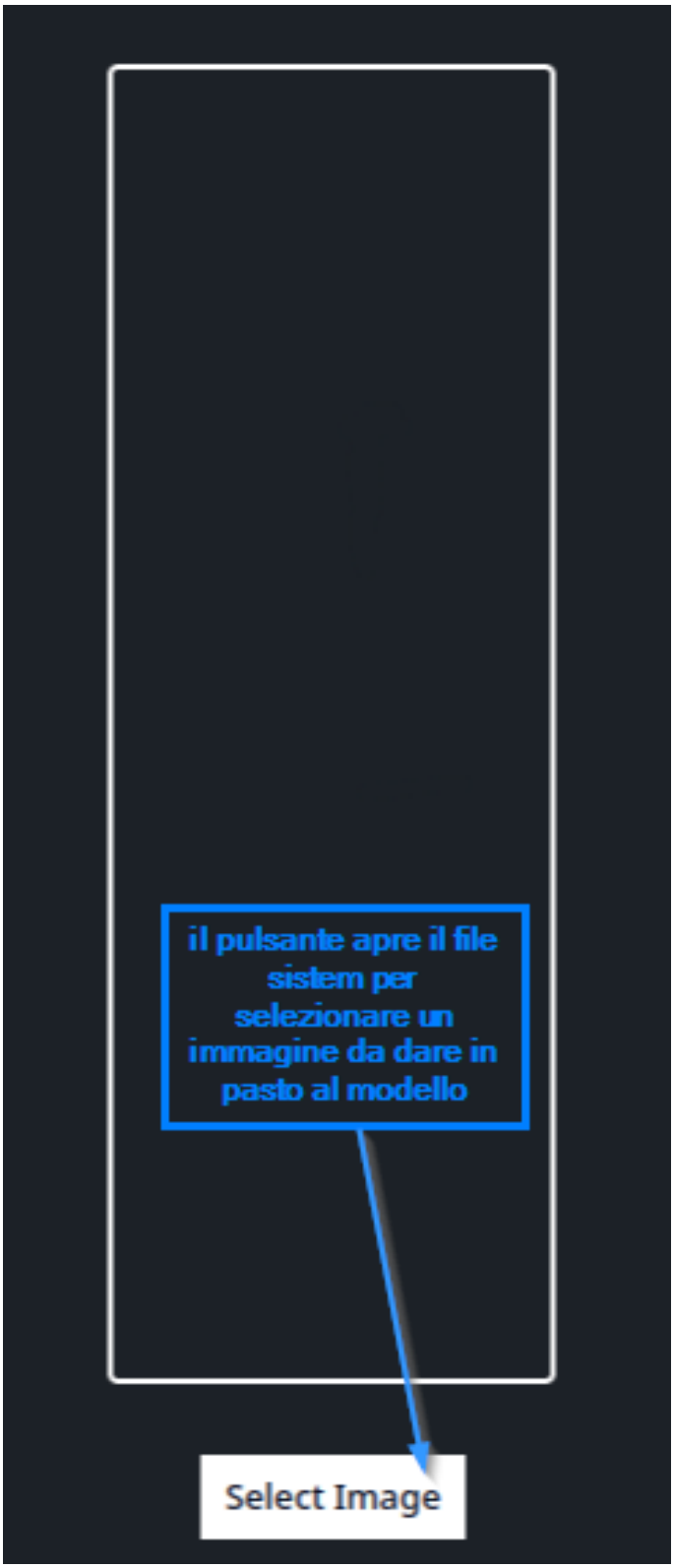
Una piattaforma open source end-to-end per l'apprendimento automatico permette agli sviluppatori di creare e distribuire facilmente applicazioni basate sull'intelligenza artificiale, in modo da rendere l'utilizzo del machine learning accessibile a tutti

TensorFlow

TensorFlows.js è una libreria opensource, basata su WebGL per l'intelligenza artificiale. Essenzialmente fornisce il framework per poter effettuare il train di reti neurali, inoltre è in grado di integrare modelli già "addestrati". TensorFlow.js fornisce sia blocchi low-level per il machine learning sia API di high-level

è un set di dati di rilevamento, segmentazione e sottotitoli di oggetti su larga scala che contiene 80 classi di oggetti.
Questo set di dati è stato progettato per essere utilizzato come strumento di apprendimento per l'addestramento di modelli di machine learning

INTERFACCE



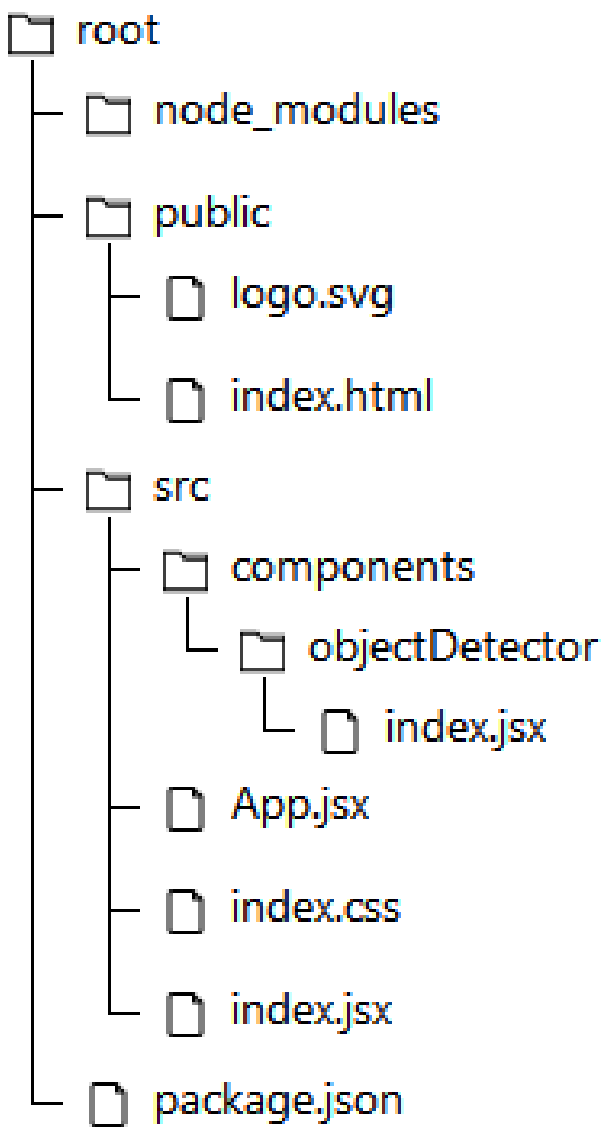
L'importanza di un'interfaccia semplice

Il design di questo sistema è stato pensato per ridurre al minimo gli errori di usabilità commessi dall'utente, al fine di permettere loro di concentrarsi sulle qualità dell'intelligenza artificiale incorporata nel sistema stesso

Simone Tanzi | 978586

ARCHITETTURA

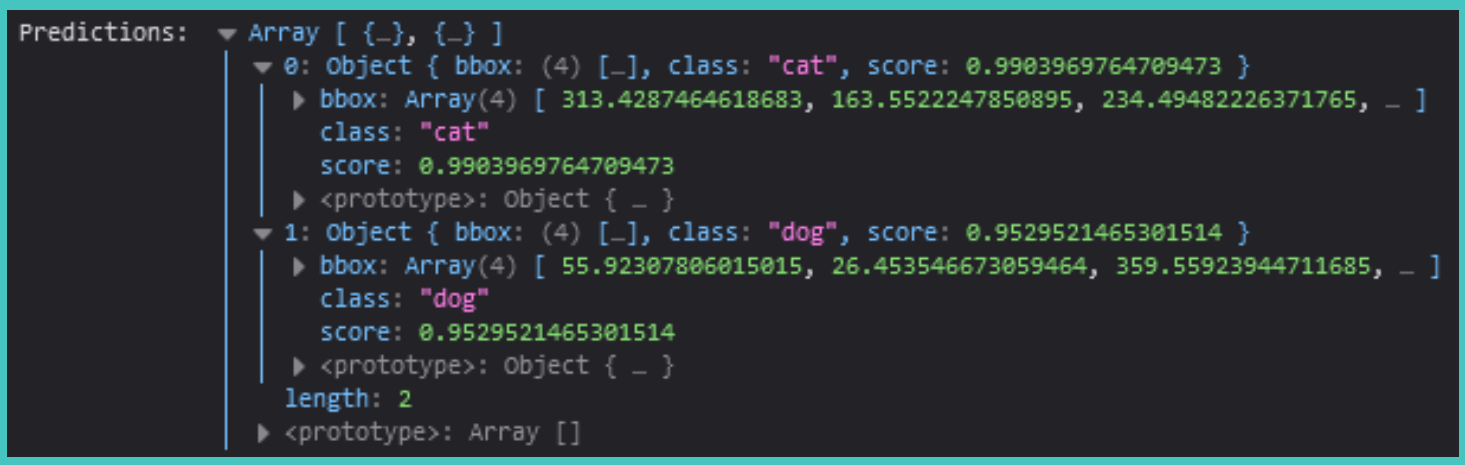
L'ordine gerarchico URI del sito web è stato progettato per offrire una struttura di navigazione intuitiva e facile secondo le specifiche di React



Quando viene utilizzato per elaborare un'immagine o un video, il modulo COCO-SSD restituisce una serie di risultati che descrivono gli oggetti individuati all'interno dei dati di input. Questi risultati possono includere informazioni come le coordinate del bounding box che circonda l'oggetto, la classe dell'oggetto (ad esempio, "automobile" o "persona") e la probabilità di appartenenza alla classe

array di oggetti JavaScript

Convolutional Orthogonal Computing with Structured Sparse Decomposition



Viene usato l'array delle predictions generato dal modulo per visualizzare i risultati della rilevazione degli oggetti in un'immagine. Estraendo le coordinate del bounding box contenute nell'array per disegnare un rettangolo intorno agli oggetti individuati e mostriamo la classe e la probabilità di appartenenza di ciascun oggetto

CODICE

Frammento di codice per tipologia

Frammenti del codice più significativo

01. HTML markup

Interessante la parte di codice HTML, perché grazie all'utilizzo di node.js abbiamo la possibilità di richiamare tutta la parte di html direttamente dentro un con id scelto da noi. In questo caso il div root conterrà tutto il sito web

```
<!-- Template -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="./logo.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="description" content="pwm image recognition ai"/>
    <meta name="author" content="Simone Tanzi">
    <title>React App</title>
  </head>
  <body>
    <div id="root"><!-- Logica --></div>
  </body>
</html>
```

02. CSS sheet style

per la parte di CSS viene usato anche la componente JSX, un'estensione di sintassi per JavaScript che consente di includere elementi simili a HTML nel codice. È comunemente usato in combinazione con React

```
1  import styled from "styled-components";
2  import { ObjectDetector } from "./components/objectDetector";
3
4  const AppContainer = styled.div`
5    width: 100%;
6    height: 100%;
7    background-color: #1c2127;
8    display: flex;
9    flex-direction: column;
10   align-items: center;
11   justify-content: center;
12   color: #fff;
13 `;
14
15 function App() {
16   return (
17     <AppContainer>
18       <ObjectDetector />
19     </AppContainer>
20   );
21 }
22
23 export default App;
24
```

importariamo solo il componente ObjectDetector da un file chiamato "./components/objectDetector" per essere utilizzato all'interno del codice corrente

La funzione App restituisce un componente React che include un altro componente chiamato AppContainer e ObjectDetector, il primo viene utilizzato per fornire un contenitore per l'ObjectDetector importato prima

03. API cocossd

La funzione carica il modello di object detection, quindi utilizza il modello per fare le predizioni sull'immagine fornita. Le predizioni vengono poi normalizzate in base alle dimensioni dell'immagine e salvate nello stato dell'applicazione. Infine, vengono stampate in console

```
108  const detectObjectsOnImage = async (imageElement, imgSize) => {
109    const model = await cocoSsd.load({});
110    const predictions = await model.detect(imageElement, 6);
111    const normalizedPredictions = normalizePredictions(predictions, imgSize);
112    setPredictions(normalizedPredictions);
113    console.log("Predictions: ", predictions);
114  };
```

Il modello viene caricato nella variabile model, però prima la funzione usa il **load** per caricare il modello in memoria, in modo che possa essere utilizzato per fare le predizioni. La funzione **load** è una funzione asincrona, quindi viene utilizzato l'operatore **await** per attendere che il modello sia caricato prima di assegnarlo alla variabile

CocoSSD analizza l'immagine e restituisce un array di predizioni come output. Le predizioni sono rappresentate come oggetti JavaScript con diversi campi, come l'ID dell'oggetto, la classe dell'oggetto e le coordinate del bounding box dell'oggetto nell'immagine

04. Node.js

Il codice principale di questo componente React si trova nell'istruzione di return, che consente il rendering della form. La form permette all'utente di caricare un'immagine e visualizza le predizioni del modello di object detection sull'immagine stessa, comprese le bounding box che vengono adattate alle dimensioni dell'ultima

```
146  return (
147    <ObjectDetectorContainer>
148      <DetectorContainer>
149        {imgData && <TargetImg src={imgData} ref={imageRef} />}
150        {!isEmptyPredictions &&
151          predictions.map((prediction, idx) => (
152            <TargetBox
153              key={idx}
154              x={prediction.bbox[0]}
155              y={prediction.bbox[1]}
156              width={prediction.bbox[2]}
157              height={prediction.bbox[3]}
158              classType={prediction.class}
159              score={prediction.score * 100}
160            />
161          ))}
162      </DetectorContainer>
163      <HiddenFileInput
164        type="file"
165        ref={fileInputRef}
166        onChange={onSelectImage}
167      />
168      <SelectButton onClick={openFilePicker}>
169        {isLoading ? "Recognizing..." : "Select Image"}
170      </SelectButton>
171    </ObjectDetectorContainer>
172  );
```

CONCLUSION

Grazie per l'attenzione

Il modello CocoSSD è stato addestrato su una grande quantità di dati di immagini etichettate e ha dimostrato di essere molto accurato nell'identificare oggetti in immagini nuove. Utilizzando Node.js, ho creato un'interfaccia utente che consente all'utente di caricare un'immagine e ottenere le predizioni del modello sull'immagine

In generale, è stata creata un'applicazione efficace di object detection utilizzando Node.js e il modello CocoSSD. Sono estremamente soddisfatto dei risultati ottenuti e tutti gli obiettivi sono stati raggiunti

NOTE BIBLIOGRAFICHE E SITOGRAFICHE

Le note bibliografiche del sito:
una guida alle fonti utilizzate per la creazione di contenuti accurati e attendibili

- **The Coding Train: ml5.js: Object Detection with COCO-SSD**
- **CocoSSD**
- **TensorFlow**
- **Github project**

Tutto questo ha reso possibile la creazione del progetto di PWM-Image-Recognition-AI

Contact

studente: Simone Tanzi
matricola: 978586

simone.tanzi@studenti.unimi.it