UNIVERSITÀ TELEMATICA
INTERNAZIONALE UNINETTUNO

Corso di Laurea in Ingegneria Informatica Magistrale – Big Data
A.A. 2023/2024

DIGITAL INNOVATION

ECOSENTINEL

SISTEMA CNN PER RILEVARE EMISSIONI IN ATMOSFERA INTEGRAZIONE DATABASE DI IMMAGINI

Professore:

Prof. Mauro Mazzei

Studente:
Emilio Verri
865HHHINGINFOR

INDICE

INTRODUZIONE

Lo scopo di questo elaborato è illustrare l'implementazione di una nuova funzionalità all'interno di EcoSentinel, un sistema (CNN) basato su reti neurali convoluzionali progettato per la rilevazione delle emissioni in atmosfera. Questa nuova funzionalità prevede la possibilità di catturare immagini delle emissioni rilevate e di salvarle in un database integrato nel sistema, accessibile tramite una sezione del sito dedicata. L'introduzione di questa capacità rappresenta un miglioramento nel monitoraggio ambientale, poiché consente una registrazione visiva continua e dettagliata delle emissioni.

La funzionalità di cattura delle immagini sarà attivata in tempo reale, utilizzando la sezione della pagina web dedicata all'analisi e all'identificazione delle emissioni. Le immagini ottenute verranno analizzate e successivamente archiviate nel database del sistema, dove saranno rese facilmente accessibili agli utenti. Questo database permetterà una consultazione rapida e sistematica delle immagini.

Il salvataggio delle immagini nel database garantisce numerosi vantaggi, fornirà una documentazione visiva e faciliterà la tracciabilità delle emissioni nel tempo.

L'obiettivo principale di questo elaborato è dimostrare come l'implementazione della capacità di catturare e salvare immagini delle emissioni all'interno di EcoSentinel rappresenti un avanzamento delle infrastrutture del sistema per il rilevamento ambientale.

CAPITOLO 1 – LE RETI CNN

1.1 Storia delle reti CNN

La storia delle reti neurali convoluzionali (CNN) ha inizio negli anni 1950 e 1960 con le scoperte di David Hubel e Torsten Wiesel, che identificarono neuroni specifici nella corteccia visiva di scimmie, in grado di elaborare caratteristiche visive. Queste scoperte costituirono la base per lo sviluppo e la ricerca delle CNN.
Nel 1980, Kunihiko Fukushima introdusse il neocognitron, una rete neurale convoluzionale innovativa che non richiedeva pesi addestrabili, questo porrà le basi per le attuali architetture delle reti neurali convoluzionali. Il neocognitron rappresentava un'importante scoperta, poiché dimostra come le reti neurali possono essere utilizzate per il riconoscimento di pattern complessi.

Nel 1998 ci fù un ulteriore passo avanti con la creazione della LeNet-5 da parte di Yann LeCun e dei suoi collaboratori. La LeNet-5 era una CNN progettata per il riconoscimento di cifre scritte a mano, e dimostrò l'efficacia delle reti CNN nel riconoscimento visivo. Questa rete era composta da più strati convoluzionali e strati di pooling, seguiti da strati completamente connessi.

Con l'arrivo delle GPU (Graphics Processing Units) rivoluzionò ulteriormente il campo delle CNN. Le GPU sono processori specializzati in grado di eseguire molte operazioni grafiche in parallelo, il che le rende ideali per l'addestramento delle reti neurali convoluzionali, che richiedono enormi capacità di calcolo per elaborare grandi volumi di dati.

Con l'utilizzo delle GPU, il tempo necessario per addestrare le CNN iniziò a ridursi drasticamente, permettendo di gestire dataset sempre più grandi e complessi.

Negli anni successivi, l'integrazione delle CNN con tecniche di deep learning e l'uso di grandi dataset hanno portato a miglioramenti significativi nelle loro prestazioni. Architetture avanzate come AlexNet, VGGNet, Inception e ResNet hanno dimostrato come le reti neurali possano essere utilizzate per affrontare problemi sempre più complessi.
Oggi, le CNN sono la parte software più importante di molte applicazioni di intelligenza artificiale.

Nel campo della visione artificiale, le CNN sono utilizzate per il riconoscimento facciale, la classificazione di immagini e la rilevazione di oggetti. Nella medicina, sono impiegate per l'analisi delle immagini mediche, aiutando nella diagnosi di malattie. Nella guida autonoma, le CNN sono essenziali per la percezione dell'ambiente circostante, consentendo ai veicoli di riconoscere pedoni, segnali stradali e altre vetture.

Le reti neurali trovano applicazione anche nella sorveglianza ambientale, come nel sistema EcoSentinel per la rilevazione delle emissioni atmosferiche, e nella sicurezza. La loro capacità di apprendere e generalizzare dai dati visivi le rende uno strumento versatile per affrontare diverse problematiche.

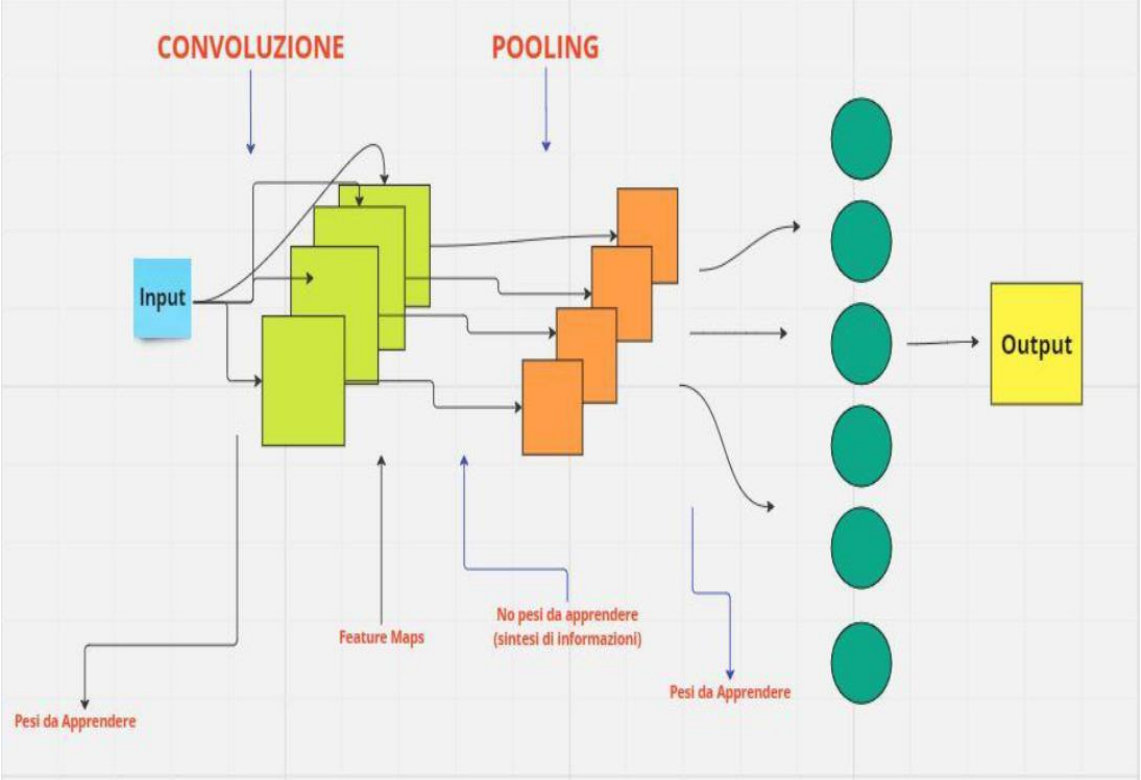
1.2 Cosa sono le reti CNN

Il processo delle reti neurali convoluzionali (CNN) inizia con un'immagine in input che viene sottoposta a convoluzione con dei filtri specifici. Questi filtri, piccole matrici di pesi, vengono trascinati su tutta l'immagine in modo da convolversi con essa e generare diverse mappe di caratteristiche (feature maps).
In questo processo, i filtri si traslano su tutta l'immagine e moltiplicano i loro valori con i pixel corrispondenti dell'immagine. Le feature maps risultanti catturano informazioni specifiche dell'immagine, come bordi, texture e altre caratteristiche visive.

Una volta generate le feature maps, queste subiscono un processo chiamato pooling, il cui scopo è ridurre il numero di pesi nell'architettura, sintetizzando le informazioni. Il pooling è un'operazione di sotto campionamento che non introduce nuovi pesi da apprendere, ma serve a ridurre il peso delle feature maps e a rendere il modello più robusto alle variazioni di posizione e scala nelle immagini di input.

Dopo il pooling, le feature maps vengono passate a una rete fully connected, tipicamente un perceptron multistrato (MLP). In una rete fully connected, ogni neurone del livello di pooling è connesso a ogni neurone del livello intermedio. Questo livello intermedio, spesso contrassegnato con sfere nei diagrammi, svolge un ruolo cruciale nella combinazione delle caratteristiche rilevate durante le fasi di convoluzione e pooling per produrre l'output finale. L'output, che varia in base alla problematica affrontata dalla CNN, è costituito da neuroni che rappresentano le classi da

riconoscere, come nel caso della classificazione dei fumi, dove l'output rappresenta le diverse categorie di fumi rilevati.



1.1 Funzionamento sistema CNN

1.3 Digital Innovation e Reti CNN

L'innovazione digitale ha trasformato radicalmente il modo in cui le tecnologie vengono sviluppate e applicate nei diversi settori. Le innovazioni più significative degli ultimi decenni, sono le reti neurali convoluzionali (CNN). Queste reti, ispirate alla struttura del sistema visivo umano, hanno rivoluzionato il campo della visione artificiale, consentendo progressi straordinari.

Le reti neurali convoluzionali sono al cuore di molte innovazioni digitali. La loro capacità di analizzare e interpretare immagini ha trasformato settori

come l'automazione industriale, la medicina, la guida autonoma e molto altro.

Le CNN sono fondamentali per lo sviluppo dei veicoli autonomi. Questi sistemi riescono a riconoscere segnali stradali, pedoni, altri veicoli e ostacoli in tempo reale per navigare in sicurezza. Le reti neurali convoluzionali elaborano costantemente le immagini catturate dai sensori del veicolo, permettendo al sistema di prendere decisioni istantanee.

Nell'automazione industriale, le CNN vengono utilizzate per il controllo qualità, il riconoscimento di difetti nei prodotti e l'automazione dei processi produttivi. Le reti neurali possono analizzare immagini di prodotti in tempo reale, identificando difetti che potrebbero sfuggire all'occhio umano. L'utilizzo delle reti CNN viene applicato anche alla sorveglianza alimentare.

Le CNN sono utilizzate anche nei sistemi di sorveglianza per il riconoscimento facciale, la rilevazione di attività sospette e la sicurezza pubblica. Questi sistemi possono analizzare grandi volumi di video in tempo reale, identificando persone e comportamenti potenzialmente pericolosi.

In campo medico, le CNN hanno rivoluzionato la diagnostica per immagini. I sistemi basati su CNN possono analizzare radiografie, risonanze magnetiche con una precisione sorprendente, assistendo i medici nella diagnosi di malattie come il cancro, le patologie cardiovascolari e altre condizioni.

Con l'avanzamento delle tecnologie e l'accesso a grandi quantità di dati e database, le reti neurali neurali continueranno a essere importanti nell'innovazione digitale. Le nuove architetture di rete stanno migliorando continuamente la velocità e la precisione delle CNN.

1.4 Digital Innovation e Python

Uno dei punti fondamentali della digital innovation è Python, un linguaggio di programmazione, che ha visto una crescita negli ultimi anni.
L'innovazione digitale implica l'adozione di tecnologie avanzate per migliorare processi, prodotti e servizi in vari settori La digital innovation è alimentata da tecnologie emergenti come l'intelligenza artificiale, l'Internet of Things, la blockchain, il cloud computing e la realtà aumentata.
Python è noto per la sua capacità di adattarsi a diversi ambiti, dallo sviluppo web all'analisi dei dati, dal machine learning all'automazione.
Questo linguaggio può essere utilizzato per sviluppare applicazioni web, analizzare dati, implementare algoritmi di machine learning, automatizzare compiti ecc…
Python è usato particolarmente nel campo dell'intelligenza artificiale e del machine learning. Librerie come Numpy forniscono strumenti per sviluppare modelli di apprendimento automatico.
Nel contesto dei big data, Python è emerso come uno strumento fondamentale per l'analisi dei dati. Librerie come Pandas, NumPy e Matplotlib offrono strumenti per la manipolazione, l'analisi e la visualizzazione dei dati.
L'analisi dei dati è diventata una componente essenziale dell'innovazione digitale, poiché permette alle aziende di comprendere meglio i propri clienti, ottimizzare i processi aziendali e identificare nuove opportunità di mercato.
Python è ampiamente utilizzato anche nello sviluppo web, grazie a framework come Django e Flask. Django, un framework che, consente di sviluppare rapidamente applicazioni web-app. Flask, un micro-framework, offre maggiore flessibilità e controllo, rendendolo ideale per applicazioni che

richiedono una configurazione personalizzata, come per esempio EcoSentinel.

Gli script Python possono essere utilizzati per automatizzare processi aziendali, gestire server, analizzare log, eseguire test automatizzati ecc…

L'automazione è una componente chiave dell'innovazione digitale, poiché consente alle aziende di ridurre i costi, minimizzare gli errori umani. Python, con le sue capacità di scripting, supporta le iniziative di automazione aziendale.

Python è utilizzato per automatizzare i processi aziendali in vari settori. Ad esempio, nella produzione, può essere impiegato per implementare sistemi di monitoraggio della qualità, della manutenzione e ottimizzazione delle linee di produzione.

La capacità di integrarsi con altre tecnologie emergenti, come l'AI, l'IoT e la blockchain, consente alle aziende di esplorare nuovi modelli di business e creare soluzioni innovative.

Sebbene Python offra numerosi vantaggi per la digital innovation, ci sono anche alcune sfide da considerare. Una delle principali sfide è la gestione delle prestazioni. Essendo un linguaggio interpretato, può essere più lento rispetto ad altri linguaggi compilati come C++ o Java. Tuttavia, questa limitazione può essere mitigata utilizzando librerie, come NumPy, e sfruttando le capacità delle GPU.

CAPITOLO 2 – IMPLEMENTAZIONE E SVILUPPO

2.1 Struttura del progetto

Nel backend, EcoSentinel gestisce il database e salva le immagini in una cartella dedicata del progetto. Questo sistema è stato progettato per catturare le immagini rilevate dalle reti neurali convoluzionali (CNN) e memorizzarle. Il backend di EcoSentinel include diverse componenti chiave che garantiscono il funzionamento del sistema:

Inizialmente, si è deciso di mantenere il database in formato testuale, utilizzando un file .txt per rappresentare i dati delle rilevazioni. Questo approccio semplice ha permesso una gestione iniziale rapida dei dati. Ogni rilevazione di fumo è registrata con dettagli come timestamp, tipo di fumo e altri dati rilevanti. Il file .txt è organizzato in modo da facilitare la ricerca.

Successivamente è stato deciso di integrare un DB di immagini.
Le immagini catturate dalle CNN vengono salvate in una cartella del progetto denominata "images". Questa cartella è strutturata in modo da contenere le immagini di ogni rilevazione, con immagini etichettate in base alla loro importanza (importante, grande, lieve fumata).

Il frontend di EcoSentinel è stato sviluppato per offrire agli utenti un'interfaccia intuitiva e facile da navigare, permettendo loro di interagire con il sistema in modo efficace. Le principali funzionalità del frontend includono:
Visualizzazione Cronologica: La pagina web del frontend permette agli utenti di visualizzare una cronologia delle rilevazioni di fumo. Le immagini

sono presentate in ordine cronologico, Questa funzione consente agli utenti di tracciare facilmente le emissioni nel tempo e identificare anomalie.

Gestione della Cartella "Images". Gli utenti possono accedere alla cartella "images" tramite il frontend per visualizzare, scaricare o eliminare le immagini salvate. Questa funzionalità è particolarmente utile per la gestione dello spazio di archiviazione e per mantenere solo le immagini rilevanti.

Pulizia della Cartella "Images": Il frontend offre anche la possibilità di svuotare la cartella "images" in modo sicuro. L'utente ha la possibilità tramite il "button" DELTE di cancellare tutte le immagini raccolte in una o più rilevazioni. Importante è da sottolineare il fatto che se viene premuto il pulsante le informazioni vengono cancellate.
È stato mantenuto nel progetto anche la sezione dedicata al database in formato txt, con la sua pagina dedicata.

1.2 Homepage EcoSentinel

2.2 Backend implementato

```
import cv2
import math
import datetime
import os
from ultralytics import YOLO
```

All'interno del backend importiamo 5 librerie:

1) CV2, libreria che viene utilizzata per la manipolazione di immagini e video;
2) MATH, per le operazioni matematiche;
3) DATETIME, per lavorare con date;
4) OS, per interagire con il sistema operativo;
5) ULTRALYTICS, per l'utilizzo del modello YOLO.

```
def video_detection(path_x):
    video_capture = path_x
```

Successivamente viene assegnato il valore del parametro "path_x" alla variabile "video_capture".

```
    cap = cv2.VideoCapture(video_capture)
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
```

Poi creerà un oggetto "VideoCapture" per leggere il video dal percorso inserito e andrà ad ottenere la larghezza e l'altezza dei frame del video.

```
model = YOLO("../Modelli-YOLO/fireSmoke.pt")
classNames = ['evidente_fumata', 'importante_fumata', 'lieve_fumata']
```

Alla variabile model viene assegnato il percorso del modello YOLO preaddestrato per rilevare le fumate e className definisce le classi di oggetti che il modello può andare a rilevare.

```
if not os.path.exists("static/images"):
    os.makedirs("static/images")
log_file_path = "log.txt"
```

Viene creata la cartella images se non esiste e crea una variabile con il nome del file di log in cui andrà a salvare le informazioni estratte.

```
while True:
    success, img = cap.read()
    results = model(img, stream=True)
```

Successivamente è stato implementato un ciclo while per leggere e processare i frame del video, success, img = cap.read() permette di andare a leggere un frame del video e andiamo a salvare dentro a results il rilevamento degli oggetti sul frame usando il modello YOLO.

```
    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0]
```

x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

Andando ad applicare un ciclo for su “results” itera sui risultati della rilevazione, salverà in boxes le coordinate dei delimitatori (quindi un rettangolo), itererà sui rettangoli delimitatori con un altro ciclo for su “boxes” e otterrà le coordinate del rettangolo. Successivamente è stato inserita int(“nome variabile”) per convertire le coordinate in valore interi.

cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 3)

Procede a disegnare un rettangolo blu intorno alla fumata che è stata rilevata. (x1, y1) come vertice superiore sinistro e (x2, y2) come vertice inferiore destro.

*conf = math.ceil((box.conf[0] * 100)) / 100*

Salva dentro in “conf” la confidenza dell’oggetto rilevato, box.conf[0]` è la confidenza associata al rilevamento, che viene moltiplicata per 100 e arrotondata per eccesso all'intero più vicino, quindi divisa di nuovo per 100 per ottenere un valore con due cifre decimali.

cls = int(box.cls[0])
class_name = classNames[cls]

Viene salvato dentro alla variabile “cls” l’indice della classe dell’oggetto rilevato e viene utilizzato l’indice della classe per ottenere il nome della classe dall’elenco dei “classNames” indicati precedentemente (“importante fumata”, evidente fumata”, lieve fumata”).

label = f"{class_name}_{conf}_{datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}"

t_size = cv2.getTextSize(label, 0, fontScale=1, thickness=2)[0]

c2 = x1 + t_size[0], y1 - t_size[1] - 3

Crea una label sopra al valore trovato in cui è presente il nome della classe, la confidenza e la data/ora come stringa.
Utilizzando la funzione "cv2.getTextSize" il programma procede a calcolare la dimensione del testo per la label. Ottiene la larghezza e l'altezza del testo. Dentro a c2 vengono calcolate le coordinate della parte inferiore destra del rettangolo che conterrà la label.

cv2.rectangle(img, (x1, y1), c2, [255, 0, 0], -1, cv2.LINE_AA)

cv2.putText(img, label, (x1, y1 - 2), 0, 1, [255, 255, 255], thickness=1, lineType=cv2.LINE_AA)

cve.rectangle(…) disegna un rettangolo di colore blu attorno alla label, il rettangolo inizia in (x1,y1) e finisce in (c2) che è l'angolo inferiore destro. Successivamente sull'immagine che verrà salvata scriverà il rettangolo della label e a fianco sono definirà le coordinate di partenza per posizionare il testo sopra il rettangolo

image_path = os.path.join("static/images", label + ".jpg")

cv2.imwrite(image_path, img)

Procede a salvare l'immagine rilevata nella cartella "images" mettendo un nome senza gli spazi (perché ci sono stati alcuni problemi con l'estrazione in javascript"). cv2.imwrite` salva l'immagine `img` nel percorso specificato.

Nel progetto va gestito anche il file di "log.txt":

```
with open(log_file_path, "a") as log_file:
    log_file.write(f"{class_name},
{datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}\n")
```

Nel progetto è stato aggiunta un file di log, infatti viene aperto il file di log con il comando ("a"), scrive la classe dell'oggetto e la data/ora. Ogni valore catturato viene scritto su una nuova riga del file così da poter individuare facilmente le fumate

```
    yield img
cv2.destroyAllWindows()
```

"yield img" restituisce l'immagine elaborata e ne restituisce una alla volta, mentre "cv2.destroyAllWindows()" distrugge tutte le finestre aperte di "OpenCV".

```
@app.route('/databaseimmagini')
def database_images():
  image_filenames = [f for f in listdir("static/images") if f.endswith((".jpg",
".png", ".jpeg"))]
 return                    render_template('databaseimmagini.html',
image_filenames=image_filenames)
```

Questa funzione recupera l'elenco di nomi delle immagini dalla cartella images e le indirizza verso il template "databaseimmagini.html".
Ottiene un elenco dei nomi dei file presenti dentro alla cartella "static/images" e successivamente applica un filtro per includere solo i file che terminano con le estensioni ".jpg", ".png" o ".jpeg". Nella return reindirizza il template verso "databaseimmagini.html", passandogli l'elenco dei nomi delle immagini.

Vengono effettuati successivamente 2 import "listdir e remove" per elencare e rimuovere file e "join" per unire i percorsi dei file.
Nel file della gestione dei template è stata introdotta anche la funzione per cancellare le immagini:

```
from os import listdir, remove
from os.path import join
@app.route('/delete_images', methods=['POST'])
def delete_images():
  try:
    images_folder = join(app.static_folder, 'images')
    files = listdir(images_folder)
    for filename in files:
      if filename.endswith('.jpg') or filename.endswith('.png') or filename.endswith('.jpeg'):
        file_path = join(images_folder, filename)
        remove(file_path)

    return '', 204
```

```
except FileNotFoundError:
  return '', 404

except Exception as e:
  print('Errore durante la cancellazione delle immagini:', e)
  return '', 500
```

È stata creata una nuova route “delete_images” in cui viene recuperato con “images_folder” il percorso della cartella “static/images” e viene salvato dentro a “files” l’elenco di tutti i file dentro alla cartella. Successivamente attraverso un ciclo for, si itera sui file e con “remove(file_path)” il ciclo procede ad eliminare i file dopo che ha fatto ovviamente un controllo sulle estensioni delle immagini. In fondo a questa funzione sono state inseriti dei return che vengono scaturiti in base al risultato della cancellazione delle immagini.

204=messaggio di successo, ma senza contenuto;

404=non trovato;

505=internal server error;

2.3 Frontend implementato

```
<button class="uk-button uk-button-primary"
onclick="deleteAllImages()">Cancella Tutte le Immagini</button>


<script>
 function deleteAllImages() {
  fetch('/delete_images', { method: 'POST' })
   .then(response => {
    if (response.ok) {
     window.location.reload();
    } else {
     alert('Errore durante la cancellazione delle immagini.');
    }
   })
   .catch(error => {
    console.error('Errore durante la cancellazione delle immagini:',
error);
    alert('Errore durante la cancellazione delle immagini.');
   });
 }
</script>
```

Per prima cosa è stato pensato di mettere all'inizio il <button> per la cancellazione immagini. Questo button va a richiamare la funzione "deleteAllImages()" quando viene cliccato, che è una funziona javascript.
La funzione permette di fare una chiamata post verso la route che si occupa della cancellazione delle immagini descritta nel backend "/delete_images".

Se la chiamata POST avviene correttamente allora viene ricaricata la pagina corrente in cui non saranno più presenti le immagini, invece se la chiamata POST verso la pagina non avviene correttamente stampa un “alert” che avvisa l’utente della presenza di qualche problematica. La funzione è contenuta dentro ai tag <script></script> presenti dentro al file html.

```
<tbody>
   {% for image in image_filenames %}
   <tr>
      <td><img src="{{ url_for('static', filename='images/' + image.replace(' ', '_')) }}" alt="{{ image }}"></td>
      <td>
         <p>Rilevazione:</p>
         <strong><span class="detection-label">{{ image.split('_')[0] }} {{ image.split('_')[1] }}</span></strong><p>Orario:</p>
         <strong><span class="timestamp">
            {{ image.split('_')[3] }} - {{ image.split('_')[4].split('.')[0] }}
         </span></strong>
      </td>
   </tr>
   {% endfor %}
</tbody>
```

All’interno del <tbody> della tabella viene inizializzato un loop su “image_filenames” che stampa dentro al tag <img> il percorso delle varie immagini. Dentro alla tabella vengono anche stampati il nome della classe rilevata, la confidenza, l’orario e la data della rilevazione. Il ciclo for si conclude con {% endfor %}. Ogni immagine viene stampante in un <td> che rappresenta la cella della tabella.

CAPITOLO 3 – SVILUPPI FUTURI

3.1 Miglioramenti futuri

Attualmente, il sistema utilizza un database locale per salvare le immagini rilevate, ed è implementato come servizio web. Tuttavia, per migliorare l'efficacia e l'efficienza di EcoSentinel, è necessario apportare alcune modifiche significative al backend e al frontend.

Uno dei principali miglioramenti necessari riguarda la gestione del database. Attualmente, utilizza un database locale nella cartella “images”. Questo approccio, sebbene semplice da implementare, presenta notevoli limitazioni dal punto di vista dell’affidabilità. Per superare queste limitazioni, bisogna utilizzare un database MySQL.
Un database MySQL offre diversi vantaggi rispetto a un database locale basato su file. Innanzitutto, MySQL è progettato per gestire grandi quantità di dati in modo efficiente, supportando operazioni di lettura e scrittura veloci. Questo è particolarmente importante per EcoSentinel, che deve elaborare un flusso continuo di immagini e dati in tempo reale.
Un altro vantaggio di MySQL è la sua capacità di garantire l'integrità dei dati anche in caso di errori o interruzioni del sistema. Questa caratteristica è essenziale per un sistema come EcoSentinel, che deve garantire l'accuratezza e la consistenza dei dati raccolti per fornire risultati affidabili. Inoltre, MySQL offre strumenti di backup, essenziali per proteggere i dati da perdite accidentali o danni.

Per implementare MySQL in EcoSentinel, è necessario apportare alcune modifiche al backend. Attualmente, il backend di EcoSentinel gestisce la

raccolta delle immagini e dei dati e li salva in una cartella. Questo processo deve essere modificato per utilizzare un database MySQL. Ciò implica la creazione di tabelle per memorizzare le informazioni sulle immagini rilevate, inclusi i metadati come la classe di emissione, la confidenza della rilevazione, la data e l'ora.

La migrazione a MySQL richiede anche l'aggiornamento delle query di accesso ai dati. Invece di leggere e scrivere dati da e verso file di testo, il backend dovrà eseguire operazioni di inserimento e aggiornamento su un database.

Oltre alla migrazione a MySQL, è necessario ottimizzare le prestazioni dello script di rilevamento delle immagini. Attualmente, lo script può risultare lento, soprattutto quando deve elaborare un grande volume di immagini. Questo rallentamento può compromettere la capacità del sistema di rilevare le emissioni, riducendo l'efficacia di EcoSentinel.

Una strategia per ottimizzare lo script è la gestione migliore del codice per ridurre il tempo di elaborazione delle immagini. Questo include la riduzione della complessità delle operazioni di elaborazione. Inoltre, è possibile sfruttare le capacità offerte dalle GPU (Graphics Processing Unit). Le GPU sono particolarmente adatte per eseguire operazioni su grandi quantità di dati, come le immagini, e possono accelerare significativamente le operazioni di convoluzione utilizzate dalle CNN.

Un'altra strategia per migliorare le prestazioni è l'uso di tecniche di caching. Il caching consente di memorizzare temporaneamente i risultati delle operazioni, in modo da poterli riutilizzare senza doverli ricalcolare ogni volta.

Oltre a questi miglioramenti del backend, è importante rivedere l'architettura complessiva di EcoSentinel. Attualmente, il sistema è implementato come servizio web, il che può limitare la sua portabilità e flessibilità. Un approccio

migliore potrebbe essere quello di implementarlo come applicativo da installare. Questo consentirebbe agli utenti di installare e configurare il sistema direttamente sul loro dispositivi, senza dipendere da un server web remoto.

Migliorerebbe la sicurezza e la privacy dei dati, poiché le immagini rilevate e i relativi dati verrebbero memorizzati localmente, anziché essere trasmessi a un server remoto.

Inoltre, un applicativo installabile sarebbe più flessibile e personalizzabile. Gli utenti potrebbero configurare il sistema in base alle loro esigenze specifiche.

Per implementare EcoSentinel come applicativo, è necessario sviluppare un'interfaccia utente grafica che consenta agli utenti di interagire facilmente con il sistema.

Un altro aspetto importante da considerare nell'implementazione, come applicativo è la gestione degli aggiornamenti. Poiché il software verrà installato sui dispositivi degli utenti, è necessario fornire un meccanismo per distribuire facilmente gli aggiornamenti. Questo può essere fatto utilizzando un sistema di aggiornamento automatico, che scarica e installa automaticamente le nuove versioni del software, o fornendo aggiornamenti manuali che gli utenti possono scaricare e installare quando necessario.

CONCLUSIONI

Il progetto di implementazione di una pagina web dedicata alla cronologia delle emissioni rilevate da EcoSentinel si è rivelato pienamente funzionante. La possibilità di catturare e archiviare immagini in tempo reale ha migliorato significativamente l'efficacia del monitoraggio ambientale. Gli utenti possono accedere a una cronologia visiva delle emissioni, facilitando l'analisi dei dati e la tracciabilità delle fonti di inquinamento atmosferico.

Questo progetto non solo ha potenziato le capacità del progetto EcoSentinel, ma ha anche rappresentato un esempio concreto di "DIGITAL INNOVATION". L'integrazione di tecnologie avanzate come le CNN e l'uso di database visivi accessibili via web evidenziano come l'innovazione digitale possa trasformare il modo in cui gestiamo e proteggiamo l'ambiente. La capacità di raccogliere, analizzare e condividere dati in tempo reale è fondamentale per affrontare le emergenze ambientali.

Progetti come EcoSentinel dimostrano come la tecnologia possa offrire soluzioni efficaci e innovative. L'uso delle CNN per il rilevamento delle emissioni, supportato da una robusta infrastruttura digitale, rappresenta un modello per future applicazioni in campo ambientale.

BIBLIOGRAFIA E SITOGRAFIA

[1] Michael Nielsen, A Beginner's Guide to Convolutional Neural Networks, arXiv, 2015.

[2] François Chollet, Deep Learning with Python, Manning Publications, 2017.

[3] Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton, ImageNet classification with deep

convolutional neural networks, Nature, 2012, 523 pagine.

[4] Zed Shaw, Learn Python the Hard Way, No Starch Press, 2015, 304 pagine.

[5] Mark Lutz, Python for Beginners, O'Reilly Media, 2019, 448 pagine.

[6] Hao Wang e Yu-Hua Chen, Convolutional Neural Networks for Smoke Detection,

Springer, 2020, 128 pagine.

[7] Jon Duckett, HTML and CSS: Design and Build Websites, Wiley, 2011.

[8] Dave Shreiner, GPU Computing for Beginners: Addison-Wesley Professional, 2012.

[9] Jason Brownlee, Python per principianti, Machine Learning Mastery, 2018, 12 pagine.

[10] John V. Guttang, Introduzione alla programmazione Python, MIT Press, 2017, 128

pagine.

[11] Akhil Garg, Una guida introduttiva alla programmazione di reti CNN con Python,

Towards Data Science, 2022, 3 pagne.

[12] Jason Brownlee, Come creare una rete CNN con Python, Machine Learning Mastery,

2022, pagine 10.
[13] Guido van Rossum, La storia di Python, Python Software Foundation, 2022, pagine 2
[14] Jason Brownlee, Ultralytics e OpenCV: una panoramica, Machine Learning Mastery,
2022, 5 pagine.
[15] David Beazley, Un confronto tra Ultralytics e OpenCV per la rilevazione di oggetti,
O'Reilly Media, 2022, 10 pagine.
[16] https://www.ibm.com/it-it/topics/convolutional-neural-networks, consultato in data
10/06/2023.
[17] https://github.com/abg3/Smoke-Detection-using-Tensorflow-2.2 , consultato in data
20/06/2023.
[18] https://www.html.it/articoli/google-colab-per-il-machine-learning-cose-e-come-siusa/,
consultato in data 14/07/2023.
[19] https://blog.roboflow.com/getting-started-with-roboflow/, consultato in data
18/07/2023.
[20] https://docs.ultralytics.com/ , consultato in data 25/07/2023