



POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea in Ingegneria del Software e Fondamenti Web

Implementazione di Tecniche di Machine Learning e Computer Vision a supporto del Settore Oleario

Relatore

Prof. Ing. Marina Mongiello

Laureando
Bartolo Morgigno

Anno Accademico 2023 - 2024

Abstract

L'industria olearia è uno dei pilastri della cultura mediterranea e dell'economia globale. La produzione olearia si basa ancora in gran parte su processi tradizionali che possono risultare inefficaci e dispendiosi; pertanto, la necessità di migliorare l'efficienza operativa e ridurre l'errore umano spinge verso l'adozione di tecnologie innovative che possano automatizzare il monitoraggio e la valutazione delle colture. Questo tipo di innovazione non solo aumenterebbe la precisione delle valutazioni ma ridurrebbe significativamente i tempi e i costi legati alla raccolta manuale dei dati. Questo lavoro di tesi propone lo sviluppo di una web app che integra tecniche avanzate di machine learning e computer vision per automatizzare il processo di rilevazione e classificazione delle olive. L'applicazione consente agli utenti di caricare immagini di olive provenienti da diverse fonti, come fotocamere o droni. Il sistema, utilizzando uno specifico script di elaborazione, riesce ad identificare automaticamente le olive nelle immagini e a stimarne il numero, il peso e il grado di maturità. La web app è progettata come una Single Page Application (SPA), con un frontend che garantisce un'interfaccia utente reattiva e intuitiva sviluppato in React e con un backend, realizzato con Flask, che gestisce l'interazione fra le diverse componenti dell'applicazione e l'elaborazione delle immagini tramite uno script Python che esegue la predizione tramite un modello addestrato di machine learning e applica tecniche di computer vision. Il sistema è pensato per essere altamente scalabile, flessibile e utilizzabile da utenti senza particolari competenze tecniche. La proposta mira a migliorare la gestione delle operazioni di monitoraggio e raccolta nel settore oleario, fornendo una soluzione tecnologica innovativa e accessibile che ottimizza l'efficienza e la qualità del processo produttivo.

Indice

Introduzione	1
1 Importanza del settore oleario	3
1.1 Background storico e culturale	3
1.2 Tipologie di olive e prodotti derivati	4
1.3 Dimensione del mercato oleario e importanza socio-economica . . .	5
1.4 Ostacoli ed opportunità	5
2 Panoramica sulle tecnologie introdotte	7
2.1 Introduzione al Machine Learning	7
2.1.1 Reti neurali	8
2.1.2 Reti neurali convoluzionali	8
2.1.3 Utilizzo della GPU per il Machine Learning	10
2.2 Introduzione alla Computer Vision	10
3 Stato dell'arte del settore oleario	13
3.1 Principali sfide del settore	13
3.2 Panoramica delle tecnologie disponibili	14
3.2.1 Monitoraggio delle colture e rilevamento delle malattie in tempo reale	14
3.2.2 Classificazione e smistamento delle olive	15
3.2.3 Analisi della qualità delle olive	17
3.2.4 Analisi della qualità dell'olio d'oliva	18
4 Progettazione e sviluppo	19
4.1 Creazione di un virtual environment	19
4.2 Progettazione dello script di esecuzione	20
4.2.1 Scelta del linguaggio Python e della libreria Pytorch	20
4.2.2 Scelta del modello di rete neurale YOLOv8	21
4.3 Progettazione del Frontend	22
4.3.1 Scelta del linguaggio Javascript e del framework React	22
4.4 Progettazione del Backend	23

4.4.1	Scelta del micro-framework Flask basato su Python	23
5	Modello e architettura	25
5.1	Analisi dei Requisiti	25
5.1.1	Requisiti Funzionali	25
5.1.2	Requisiti Non Funzionali	26
5.2	Modello progettuale	26
5.3	Struttura e funzionamento dell'architettura	27
6	Implementazione e validazione	31
6.1	Preparazione dei dati	31
6.2	Addestramento del modello Yolov8	32
6.3	Creazione dello script di elaborazione	33
6.4	Creazione del server Backend	36
6.5	Realizzazione dell'interfaccia utente Frontend	38
6.5.1	Schermata iniziale	38
6.5.2	Schermata di anteprima delle immagini	39
6.5.3	Schermata di elaborazione	41
6.5.4	Schermata di visualizzazione dei risultati	42
	Conclusioni e scopi futuri	45
	Bibliografia	57

Introduzione

Il settore oleario è un pilastro dell'economia e della cultura mediterranea, con una tradizione secolare legata alla coltivazione dell'olivo e alla produzione di olio d'oliva. Le olive e l'olio rappresentano prodotti versatili, utilizzati sia nell'industria alimentare che cosmetica, e la loro importanza si riflette nell'impatto economico che hanno a livello globale soprattutto in Paesi come l'Italia, la Spagna e la Grecia. Tuttavia, nonostante il loro valore, gran parte dei processi di gestione delle colture si basa ancora su metodi manuali. Questi metodi, oltre a richiedere tempo e competenze specialistiche, possono essere soggetti a errori legati alla variabilità del giudizio umano. Con l'aumento delle esigenze in termini di efficienza e precisione, soprattutto nelle aziende che operano su larga scala, si fa sempre più evidente la necessità di introdurre soluzioni tecnologiche che possano automatizzare e ottimizzare i processi. L'adozione di tecnologie avanzate, basate su algoritmi di intelligenza e visione artificiale, offre l'opportunità di trasformare il modo in cui vengono gestite le operazioni di monitoraggio e classificazione delle olive. L'automazione non solo migliora l'accuratezza e la rapidità delle valutazioni ma riduce significativamente i costi operativi e il rischio di errore, contribuendo a rendere il settore oleario più competitivo e sostenibile.

Questa tesi si propone di sviluppare una web app che utilizza tecnologie basate su machine learning e computer vision per automatizzare la rilevazione e la classificazione delle olive. L'applicazione permette agli utenti di caricare immagini delle olive acquisite da fotocamere o droni e di ottenere un'analisi automatica. La rete neurale convoluzionale YOLOv8 viene utilizzata per identificare le olive, stimarne il numero e il peso, e classificarle in base al grado di maturità. Questo sistema di analisi automatizzata riduce la dipendenza dai metodi manuali, accelerando i processi e rendendoli più precisi e affidabili. La web app è stata progettata come una Single Page Application (SPA) per garantire un'interfaccia utente fluida e intuitiva. Il frontend, sviluppato in React, consente un'interazione semplice e immediata con il sistema, mentre il backend, realizzato in Flask, gestisce la logica dell'interazione tra le componenti e l'elaborazione delle immagini tramite uno script di esecuzione. L'intero sistema è concepito per essere altamente scalabile e versatile, permettendo di elaborare immagini provenienti da diverse fonti e con qualità variabili, mantenendo alti standard di efficienza. L'approccio proposto offre soluzioni sia per le

piccole aziende agricole, che necessitano di strumenti agili per monitorare le coltivazioni, sia per le grandi aziende, che richiedono analisi su larga scala. L’obiettivo è rendere accessibili tecnologie avanzate anche a operatori con limitate competenze tecniche, migliorando così la produttività e riducendo i margini di errore nel processo decisionale legato alla raccolta delle olive. Questo studio si articola in sei capitoli:

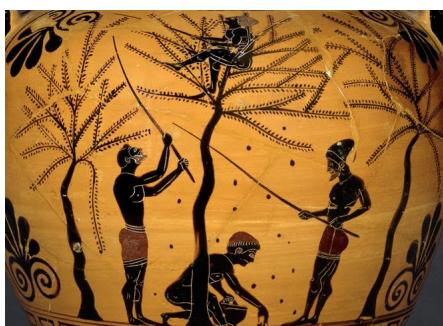
- Il Capitolo 1 introduce il contesto generale e discute l’importanza del settore oleario, analizzando il background storico e culturale, la dimensione economica e le principali sfide e opportunità legate al settore;
- Nel Capitolo 2 viene fornita una panoramica delle tecnologie introdotte, con un approfondimento su concetti chiave come il machine learning, le reti neurali convoluzionali e le tecniche di computer vision impiegate per lo sviluppo dell’applicazione;
- Il Capitolo 3 esplora lo stato dell’arte del settore oleario, descrivendo le principali tecnologie disponibili per il monitoraggio delle coltivazioni e l’analisi della qualità delle olive;
- Il Capitolo 4 si concentra sullo sviluppo e sulle scelte progettuali adottate, delineando il processo di realizzazione dell’ambiente di sviluppo, dello script di elaborazione, del frontend e del backend;
- Nel Capitolo 5 vengono analizzati i requisiti funzionali e non funzionali dell’applicazione e viene descritta l’architettura del sistema, evidenziando come sia stata progettata per soddisfare le esigenze del progetto;
- Il Capitolo 6, infine, presenta l’implementazione e la validazione del sistema, discutendo le prestazioni dell’applicazione e i risultati ottenuti, concludendo con proposte per sviluppi futuri.

Capitolo 1

Importanza del settore oleario

1.1 Background storico e culturale

Il settore oleario ha radici profonde nella storia, nella cultura e nella mitologia dei Paesi mediterranei. Quella dell'ulivo è stata una delle prime forme di coltivazione agricola nella storia dell'evoluzione umana: secondo testimonianze storiche, infatti, già nel 3.000 a.C. gli antichi popoli del Mediterraneo utilizzavano l'olio d'oliva per l'alimentazione e per scopi medici e rituali [20]. L'olivo è divenuto col tempo simbolo di identità culturale, di pace e di prosperità e, analogamente, l'olio d'oliva è diventato un pilastro della dieta mediterranea, riconosciuto dall'UNESCO come patrimonio culturale immateriale dell'umanità per i suoi benefici sulla salute [28]. In Paesi come Spagna, Italia, Grecia e Portogallo la coltivazione dell'olivo è una vera e propria tradizione secolare; la raccolta delle olive è un momento di condivisione e di celebrazione che alleggerisce il carico del lavoro e le tecniche agricole vengono tramandate di generazione in generazione come forma di eredità culturale. Tutto ciò ha contribuito, col passare del tempo, a modellare la cultura e le abitudini alimentari delle comunità locali.



(a) Anfora greca raffigurante il momento della raccolta [1]



(b) Affresco etrusco raffigurante un momento di celebrazione rituale [3]

Figura 1.1

1.2 Tipologie di olive e prodotti derivati

Le olive possono essere suddivise in due principali categorie: olive da olio e olive da tavola. Le olive da olio sono principalmente coltivate per estrarre dal loro interno l'olio d'oliva, uno dei prodotti più versatili e apprezzati dalla cucina mondiale. Gli oli d'oliva presentano molteplici varietà di gusti e colori delle quali l'Olio Extra Vergine di Oliva (EVO) è sicuramente la più utilizzata e apprezzata. Questo tipo di olio è ottenuto tramite processi meccanici a freddo, senza l'utilizzo di sostanze chimiche, e si contraddistingue per la sua bassa acidità (inferiore allo 0,8%). Insieme all'olio EVO troviamo anche l'Olio Vergine di Oliva, anch'esso estratto per via meccanica ma con un'acidità leggermente superiore, e l'Olio di Oliva semplice, prodotto tramite miscelazione di oli vergini e raffinati [4].

Le olive non sono solo una fonte di olio, ma sono anche consumate direttamente come olive da tavola. Circa il 20% della produzione globale di olive è, infatti, destinato a questo scopo; queste ultime possono essere lavorate in molteplici modi che comprendono la conservazione in salamoia o sott'olio e la trasformazione in paté [2]. Ogni metodo di preparazione offre un prodotto finale unico nel suo genere, che si distingue dagli altri per gusto, consistenza e caratteristiche organolettiche.



(a) Olive da olio [27]



(b) Olive da tavola [26]

Figura 1.2

Questi prodotti sono molto apprezzati a livello globale e sono presenti in numerose ricette, dalla cucina mediterranea a quella internazionale. La dieta mediterranea, caratterizzata dal consumo di alimenti freschi, frutta, verdura e cereali integrali e dall'uso regolare dell'olio d'oliva come principale fonte di grassi, è stata associata a numerosi benefici per la salute, tra cui la prevenzione delle malattie cardiovascolari [28]. Oltre al consumo alimentare, le olive e i loro derivati trovano applicazione in molteplici settori. Nel campo della cosmetica vengono sfruttate le proprietà antiossidanti dell'olio d'oliva per la produzione di creme, saponi e prodotti per la cura della pelle; nel campo farmaceutico, invece, l'olio d'oliva è apprezzato per i suoi benefici sulla salute grazie al suo alto contenuto di grassi monoinsaturi e composti bioattivi come i polifenoli.

1.3 Dimensione del mercato oleario e importanza socio-economica

Il mercato oleario globale è in costante crescita, alimentato dalla crescente domanda di prodotti biologici e salutari. Il valore del mercato dell'olio d'oliva nel mondo ha superato i 13 miliardi di dollari nel 2020, e le previsioni indicano una crescita continua nei prossimi anni. Questa espansione non è solo un segno della popolarità e dell'effettivo utilizzo dell'olio d'oliva, ma riflette anche un cambiamento nelle abitudini alimentari di tutti e un maggiore interesse per il benessere e per la salute. La Spagna, in particolare, domina la scena contribuendo a circa il 50% della produzione globale di olio d'oliva, grazie a un'industria oleicola altamente sviluppata che offre una vasta gamma di oli con diverse caratteristiche organolettiche ed che è in grado di soddisfare sia i mercati di massa che quelli di nicchia. L'Italia invece, pur producendo un volume inferiore, si contraddistingue per la qualità e la varietà dei suoi oli, spesso legati a denominazioni di origine protetta (DOP) che ne certificano l'eccellenza. L'Unione Europea è, non a caso, il principale produttore e consumatore di olio d'oliva al mondo ed è per questo motivo che la produzione non è più limitata ai tradizionali Paesi mediterranei; negli ultimi anni, infatti, anche Paesi come gli Stati Uniti, l'Australia e il Cile hanno iniziato a investire nell'olivicoltura, contribuendo allo sviluppo e alla diversificazione del mercato [2].

Il settore oleario rappresenta, infatti, una fonte di reddito significativa per molti agricoltori e produttori, ma la sua importanza va ben oltre il semplice aspetto economico. La coltivazione dell'olivo svolge un ruolo chiave nella conservazione del territorio e nella promozione della biodiversità, specialmente nelle regioni rurali dove rappresenta un fattore di stabilità socio-economica. Le pratiche agricole legate all'olivicoltura, infatti, sono spesso orientate alla sostenibilità e alla tutela dell'ambiente, contribuendo al corretto mantenimento del paesaggio agrario e a prevenire il degrado del suolo.

1.4 Ostacoli ed opportunità

Nonostante la sua importanza culturale ed economica il settore oleario fronteggia costantemente numerose difficoltà. La produzione e la qualità dell'olio d'oliva sono strettamente legate alle condizioni climatiche e ambientali e, con l'aumento delle temperature e la variabilità delle precipitazioni, il surriscaldamento globale sta diventando sempre di più una minaccia concreta per la coltivazione dell'olivo. Eventi climatici estremi come siccità prolungate o gelate improvvise possono avere un impatto fortemente negativo sul raccolto, compromettendo sia la quantità che la qualità delle olive [27]. Oltre alle difficoltà legate a cause ambientali esistono anche quelle legate ai processi di produzione e valutazione. La raccolta delle olive è un'operazione delicata che richiede spesso analisi visive e sensoriali nonché l'intervento

di manodopera specializzata per garantire che il frutto venga raccolto al momento giusto e nel modo corretto. Infine, le pratiche di coltivazione intensiva degli olivi possono portare a lungo andare, se non gestite correttamente, a problemi di sostenibilità ambientale come l'erosione del suolo e la perdita di biodiversità. Tutto ciò può portare a inefficienze nel processo produttivo, influenzando negativamente la qualità del prodotto finito e la sua commerciabilità. In questo contesto l'innovazione tecnologica rappresenta un'opportunità unica per rivoluzionare il settore, rendendo l'intero processo produttivo più efficiente e affidabile grazie all'automazione. Con l'utilizzo di tecnologie all'avanguardia si possono superare ostacoli significativi andando ad ottimizzare le operazioni, ridurre al minimo gli errori umani e ad aprire la strada a nuove possibilità di crescita e sviluppo.

Capitolo 2

Panoramica sulle tecnologie introdotte

In questo capitolo verranno approfonditi i concetti fondamentali di machine learning e computer vision. Queste tecnologie permettono l'elaborazione automatica di dati visivi complessi e, pertanto, sono state scelte per automatizzare l'analisi delle immagini nell'ambito del progetto corrente e di molti altri progetti che verranno esposti nel capitolo successivo riguardante lo stato dell'arte. Il machine learning, in particolare le reti neurali, consente ai sistemi di apprendere dai dati, migliorando le prestazioni analitiche attraverso l'esperienza. La computer vision, d'altra parte, offre strumenti per l'elaborazione e l'interpretazione delle immagini, rilevando automaticamente elementi come forma, dimensioni e colore. Questo capitolo fornirà una panoramica di queste tecnologie, spiegando come la loro integrazione sia stata determinante per un sistema efficace di analisi automatica delle immagini.

2.1 Introduzione al Machine Learning

Il machine learning è una branca dell'intelligenza artificiale che si focalizza sulla progettazione di algoritmi in grado di apprendere dai dati. La definizione di machine learning può essere formulata come "la capacità di un sistema di migliorare le proprie prestazioni nel tempo tramite l'analisi di dati, senza essere programmato esplicitamente per ogni operazione". In pratica, si fa in modo che un modello possa adattarsi e riconoscere autonomamente determinati pattern nei dati, facendo previsioni o prendendo decisioni basate su quanto appreso. Questa tecnologia risulta particolarmente utile in contesti dove i dati sono complessi o difficili da modellare manualmente, come nel riconoscimento delle immagini, nel linguaggio naturale o nella previsione di tendenze. I modelli di machine learning, attraverso un processo di addestramento continuo, migliorano nel tempo, rendendoli ideali per applicazioni in cui è necessario analizzare grandi quantità di dati in modo efficiente e preciso [11].

2.1.1 Reti neurali

Una rete neurale è un modello matematico di machine learning ispirato alla struttura del cervello umano. Ogni neurone presente all'interno della rete neurale prende il nome di **nodo** ed ognuno di essi è caratterizzato da parametri chiave come il **peso** e il **bias**. A differenza dei metodi tradizionali di programmazione, dove gli sviluppatori devono specificare regole e parametri estremamente dettagliati per gestire gli scenari più disparati, le reti neurali sono progettate per apprendere pattern complessi in maniera automatica dai dati in ingresso così da poter effettuare predizioni probabilistiche su nuovi dati sconosciuti alla rete.

Tutto ciò avviene attraverso un processo di addestramento della rete neurale. Durante l'addestramento supervisionato il modello riceve in ingresso un set di dati etichettati e ogni dato che attraversa la rete, passando attraverso i vari strati che la compongono, genera una predizione; la rete confronta questa predizione con l'etichetta reale tramite una **funzione di perdita**, che misura l'errore della predizione. Inizialmente i pesi dei vari nodi vengono impostati casualmente ma, una volta terminato il confronto, l'algoritmo di **backpropagation** calcola quanto ciascun peso ha contribuito a generare l'errore complessivo e aggiorna i pesi dei nodi in modo tale da minimizzare la perdita. Ogni nodo, dopo aver combinato gli input con i pesi e il bias, applica la **funzione di attivazione** e per introdurre non linearità nel sistema. Questo è importante perché, senza di essa, la rete effettuerebbe solo calcoli lineari e non riuscirebbe a captare relazioni più complesse nei dati. Questo processo si ripete iterativamente per tutti i dati nel set di addestramento per più iterazioni (dette epoch) e, man mano che la rete è esposta a un numero crescente di esempi, i suoi pesi vengono raffinati e adattati per migliorare la capacità del modello di fare previsioni accurate anche su nuovi dati mai visti prima. In altre parole, la rete generalizza le informazioni apprese dai dati di addestramento diventando più precisa nel rilevare e classificare nuovi dati che essa non conosce. L'obiettivo finale è ridurre progressivamente l'errore di previsione, in modo che il modello diventi altamente accurato e in grado di riconoscere schemi complessi con un elevato grado di precisione [11].

2.1.2 Reti neurali convoluzionali

Le reti neurali convoluzionali (CNN, che sta per *Convolutional Neural Network*) sono un tipo di rete neurale progettata specificamente per l'elaborazione di dati con una struttura a griglia, come le immagini. A differenza delle reti neurali tradizionali, che trattano ogni pixel in modo indipendente, le CNN sfruttano la struttura spaziale delle immagini per riconoscere pattern locali come bordi, texture e forme. La struttura delle CNN si articola in una sequenza di strati il cui numero, dimensione e posizione varia a seconda delle necessità di elaborazione [21].

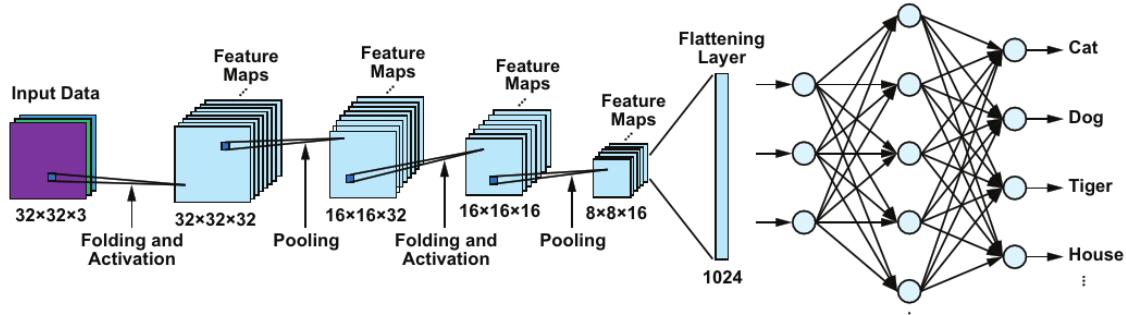


Figura 2.1: Esempio di rete neurale convoluzionale multiclass [21]

Lo **strato convoluzionale** applica un’operazione matematica detta *convoluzione* per la quale un piccolo filtro (o kernel) scorre sull’immagine esaminando porzioni ridotte (ad esempio, 3x3 o 5x5 pixel). In ogni posizione, il filtro calcola una mappa di attivazione che evidenzia la presenza di particolari caratteristiche nell’immagine come bordi o angoli. Poiché lo stesso filtro viene applicato su tutta l’immagine, non modificando le relazioni spaziali tra i pixel totali, viene ridotto il numero di parametri che la rete deve apprendere; ciò rende le CNN molto efficienti nel rilevare pattern comuni in diverse parti dell’immagine, migliorando la capacità di generalizzare su nuovi dati.

Lo **strato di pooling** viene utilizzato successivamente a quello convoluzionale per ridurre la dimensione delle mappe di attivazione. Esistono versioni come *max-pooling*, *min-pooling* o *average-pooling* che permettono di estrapolare il valore desiderato da piccole regioni della mappa: se, ad esempio, una regione 2x2 pixel viene processata da uno strato di max-pooling in uscita vi sarà un unico pixel avente valore pari a quello del massimo tra i 4 pixel precedenti. Questo processo non solo riduce la dimensionalità e, di conseguenza, la quantità di calcoli necessari ma aumenta anche la robustezza del modello rispetto a variazioni minori nell’immagine come traslazioni o rotazioni.

Una volta attraversati diversi strati convoluzionali e di pooling, i dati vengono appiattiti tramite **strati di flattening** e inviati agli **strati fully connected**; questi ultimi presentano una struttura per la quale ogni nodo è connesso a tutti i nodi dello strato precedente, e il loro scopo è quello di combinare tutte le caratteristiche estratte per produrre una classificazione finale.

In sintesi, le CNN sono particolarmente efficaci per il riconoscimento e la classificazione delle immagini per la loro capacità di apprendere come rilevare pattern locali complessi in maniera autonoma e scalabile e ciò rende le CNN lo strumento ideale per la rilevazione delle olive all’interno delle immagini in questo progetto.

2.1.3 Utilizzo della GPU per il Machine Learning

Nel contesto del calcolo ad alte prestazioni e dell'intelligenza artificiale, l'uso della GPU (*Graphics Processing Unit*) è diventato sempre più diffuso per i suoi vantaggi rispetto alla tradizionale CPU (*Central Processing Unit*). Sebbene entrambe svolgano funzioni simili vi sono differenze fondamentali nel modo in cui eseguono le operazioni di elaborazione dei dati, che rendono le GPU preferibili in molti contesti. Le CPU sono progettate per gestire una vasta gamma di operazioni generiche e sono ottimizzate per l'elaborazione di compiti complessi in modo sequenziale. Ciò significa che la CPU può gestire più attività contemporaneamente ma generalmente esegue le operazioni una alla volta, con un numero limitato di core dedicati a ciascun processo. Questo è ideale per applicazioni che richiedono logica complessa, come il controllo del sistema operativo o l'esecuzione di applicazioni desktop.

Al contrario, le GPU sono progettate per eseguire un gran numero di operazioni identiche in parallelo e questo le rende particolarmente adatte per task che coinvolgono grandi quantità di dati o operazioni matematiche complesse che devono essere suddivise e gestite in parallelo. Inizialmente le GPU erano state sviluppate per gestire il rendering delle immagini nei videogiochi ma sono state rapidamente adottate in campi come il machine learning e il deep learning per via della loro capacità di eseguire calcoli paralleli su larga scala. Per esempio, l'addestramento di una rete neurale implica la manipolazione di migliaia, se non milioni, di parametri in ogni ciclo di calcolo. Le GPU, grazie alla loro struttura, possono eseguire queste operazioni simultaneamente, riducendo significativamente il tempo richiesto per completare l'addestramento rispetto a una CPU. Le GPU possono, inoltre, gestire carichi di lavoro di dimensioni crescenti senza un significativo rallentamento delle prestazioni, il che le rende ideali per progetti come questo che necessitano di scalabilità elevata [15].

La scelta della GPU rispetto alla CPU è quindi consigliata in situazioni che richiedono una potenza di calcolo non indifferente, come nel caso del machine learning, in quanto offrono un'enorme capacità di elaborazione simultanea e permettono di ridurre i tempi di esecuzione con un'efficienza notevolmente superiore rispetto alle CPU tradizionali.

2.2 Introduzione alla Computer Vision

Gli algoritmi di computer vision svolgono un ruolo centrale nell'elaborazione e analisi delle immagini. Essi consentono di trasformare dati visivi grezzi in informazioni strutturate e utili per vari tipi di applicazioni: permettono di identificare e interpretare oggetti all'interno delle immagini, rilevare caratteristiche specifiche, segmentare aree rilevanti e molto altro ancora, rendendo le immagini più adatte per ulteriori analisi automatizzate o per processi decisionali. La computer vision ha trovato applicazione in vari settori, dall'automotive con le auto a guida autonoma,

alla medicina per l’analisi di immagini radiografiche, fino ai sistemi di sorveglianza e sicurezza [10]. Grazie a tecniche come il deep learning e le reti neurali convoluzionali, i sistemi di computer vision possono oggi analizzare grandi quantità di dati visivi con precisione e affidabilità.

Tra le principali operazioni che possono essere effettuate utilizzando algoritmi di computer vision troviamo sicuramente:

- **Ridimensionamento delle immagini:** tecnica che permette di adattare le dimensioni delle immagini in modo da renderle compatibili con i requisiti specifici degli strumenti di analisi. Ad esempio, in molte applicazioni di machine learning o reti neurali convoluzionali, le immagini devono avere dimensioni uniformi per essere processate in modo corretto. Il ridimensionamento garantisce che ogni immagine segua uno standard preciso, facilitando il lavoro degli algoritmi di analisi successivi.
- **Conversione delle immagini tra diversi spazi colore:** in molti casi, lo spazio colore RGB (Red, Green, Blue) in cui le immagini vengono acquisite non è sempre il più adatto per rilevare determinate caratteristiche; pertanto, convertire le immagini in spazi colore alternativi come lo spazio HSV (Hue, Saturation, Value) può aiutare a evidenziare dettagli cromatici che risultano difficili da distinguere nel formato originale. Questa trasformazione risulta particolarmente utile in applicazioni dove il colore gioca un ruolo chiave nel riconoscimento o classificazione degli oggetti.
- **Normalizzazione:** questa operazione comporta l’uniformazione dei valori dei pixel all’interno dell’immagine, riducendo l’effetto di variazioni come il luminazione irregolare o differenze nei dispositivi di acquisizione. Questo processo è cruciale perché garantisce che tutte le immagini presentino caratteristiche uniformi, facilitando il processo di analisi automatica che risulterebbe più complesso in presenza di variazioni imprevedibili nei valori di luminosità o contrasto.
- **Riduzione del rumore:** tecnica che migliora significativamente la qualità delle immagini. Il rumore può derivare da varie fonti come interferenze durante l’acquisizione o condizioni ambientali non ottimali. Gli algoritmi che riducono il rumore, come i filtri gaussiani o mediani, permettono di eliminare questi disturbi producendo immagini più pulite e nitide e rendendo il processo di analisi più preciso ed efficace.

In definitiva, gli algoritmi di computer vision hanno l’obiettivo di migliorare la qualità visiva delle immagini e di prepararle per le successive fasi di elaborazione. Questi algoritmi contribuiscono in modo determinante alla precisione e all’affidabilità dei processi di riconoscimento automatico, soprattutto nei processi industriali, garantendo che le immagini siano pulite, uniformi e ottimizzate per l’analisi.

Capitolo 3

Stato dell'arte del settore oleario

Negli ultimi anni il settore oleario sta abbracciando sempre di più il concetto di innovazione, subendo una trasformazione significativa. L'introduzione di nuove tecnologie digitali e l'automazione stanno progressivamente cambiando il modo in cui le operazioni di coltivazione, raccolta e trasformazione delle olive vengono gestite. L'integrazione di strumenti come sensori avanzati, intelligenza artificiale (AI), visione artificiale e sistemi di mappatura tramite droni ha permesso un miglioramento importante dell'efficienza e della precisione in tutte le fasi del processo produttivo. Queste tecnologie non solo ottimizzano la gestione delle risorse e migliorano la qualità del prodotto finale, ma riducono anche i tempi e i costi operativi, garantendo risultati più affidabili. La transizione verso processi più automatizzati e digitalizzati, tuttavia, non è sempre semplice, soprattutto per produttori che operano su piccola scala o che non dispongono delle competenze tecniche necessarie per adottare questi sistemi. Un gran numero di aziende del settore continua a scontrarsi con ostacoli significativi che vanno inevitabilmente ad incidere sui tempi, sui costi e sulla qualità del prodotto finito; pertanto, risulta fondamentale sviscerare e comprendere le principali sfide che caratterizzano il settore e valutare come le tecnologie emergenti possano risolverle.

3.1 Principali sfide del settore

Le operazioni svolte nei vari processi di lavorazione delle olive rimangono onerose e complesse e vengono, pertanto, affidate a personale esperto con anni di esperienza alle spalle. La valutazione accurata di parametri come quantità, grandezza, colore e grado di maturità richiede una conoscenza approfondita del prodotto e delle sue varietà, il che risulta particolarmente complicato quando l'analisi va effettuata su larga scala, con centinaia o migliaia di ulivo da analizzare. Molti produttori, inoltre, non dispongono dei fondi o delle risorse necessarie per adottare strumenti avanzati o non dispongono di personale competente che sappia utilizzarli; non è

sempre possibile disporre di esperti del settore e, quando più operatori sono coinvolti, possono sorgere divergenze di opinione che portano a risultati soggettivi e non sempre coerenti. Un'altra sfida riguarda la natura dispendiosa e imprecisa delle attuali pratiche di rilevazione e analisi delle informazioni. La raccolta di dati accurati richiede tempo e i metodi tradizionali, che vengono perlopiù effettuati manualmente, sono soggetti all'errore umano e limitati nella loro capacità di fornire risposte immediate. Le condizioni ambientali, come la scarsa visibilità o l'ombra degli alberi nelle prime ore del giorno, possono ulteriormente complicare il lavoro degli operatori, riducendo la capacità di identificare con precisione le olive presenti. Questi ostacoli evidenziano la necessità di uno strumento che, utilizzando dati oggettivi, sia in grado di fornire una stima preliminare attendibile sia sulla quantità che sulla qualità delle olive analizzate. I risultati ottenuti devono essere esposti in maniera chiara e intuitiva per far sì che sia coloro che non hanno competenze tecniche avanzate sia gli operatori possano interpretarle facilmente. Pertanto, è necessaria una soluzione semplice da usare che non richieda l'intervento di esperti per fornire informazioni utili sulle caratteristiche delle olive, aiutando i produttori a prendere decisioni operative, in particolare per la raccolta e la trasformazione delle olive.

3.2 Panoramica delle tecnologie disponibili

In risposta alle sfide che caratterizzano il settore oleario ad oggi sono state sviluppate diverse soluzioni all'avanguardia atte ad ottimizzare i vari processi produttivi. Queste ultime stanno progressivamente cambiando la gestione dell'intera filiera produttiva, rendendo le operazioni e meno dispendiose in termini di tempo e risorse e meno dipendenti dalle competenze manuali. Alcune delle tecnologie più innovative attualmente disponibili saranno esposte nelle sottosezioni seguenti, ognuna delle quali affronta una fase specifica del processo produttivo delle olive e dell'olio d'oliva:

3.2.1 Monitoraggio delle colture e rilevamento delle malattie in tempo reale

Tradizionalmente, il monitoraggio delle colture e il rilevamento delle malattie degli olivi sono compiti svolti da operatori umani esperti che si recano fisicamente sul campo per eseguire una valutazione qualitativa e quantitativa delle olive presenti, oltre a verificare la presenza di eventuali malattie. Questo approccio, sebbene efficace in termini di esperienza e conoscenza diretta del terreno, risulta spesso lento, soggettivo e spesso limitato dalla capacità di coprire grandi estensioni di colture in tempi brevi. Con l'avvento di tecnologie avanzate è ora possibile monitorare le colture su larga scala in modo efficiente e in tempo reale.

Esistono, infatti, strumenti che consentono di stimare la densità dei fiori di olivo presenti sugli alberi di un intero terreno, indicatore fondamentale per prevedere la resa del raccolto [8]. Sono stati sviluppati sistemi avanzati che combinano l'uso di droni per l'acquisizione di immagini ad alta risoluzione e tecniche di deep learning per l'analisi delle infezioni sulle foglie degli olivi [25] o altri studi che hanno fatto ricorso alla stessa combinazione tecnologica per classificare in maniera automatica varie malattie delle olive [6]. Ulteriori progressi sono stati compiuti con l'introduzione di algoritmi avanzati come la rete neurale YOLOv4 per il rilevamento e il conteggio dei principali parassiti delle olive come la mosca olearia [12]. In aggiunta, l'algoritmo YOLOv8 è stato utilizzato per il rilevamento di trappole appiccicose impiegate nel monitoraggio dei parassiti, consentendo un monitoraggio più approfondito dell'ambiente di coltivazione e un intervento più preciso [13]. Tecnologie innovative come l'utilizzo combinato di droni UAV e di algoritmi di machine learning consentono un monitoraggio accurato e tempestivo delle colture, fornendo informazioni essenziali per una gestione proattiva dei parassiti e delle malattie. Tali informazioni permettono ai produttori di intervenire in modo mirato riducendone la diffusione, minimizzando l'uso di trattamenti chimici e quindi massimizzando la produzione.



Figura 3.1: Rilevazione delle infezioni sulle foglie d'ulivo [25]

3.2.2 Classificazione e smistamento delle olive

La visione artificiale ha introdotto un nuovo standard di efficienza nella classificazione e nello smistamento delle olive, permettendo un'analisi rapida e precisa delle loro caratteristiche attraverso l'elaborazione delle immagini. Grazie a questi avanzamenti tecnologici è ora possibile distinguere automaticamente le olive in base alla forma, al colore e ad altri parametri chiave. Alcuni sistemi recenti si basano su algoritmi di **segmentazione** delle immagini che isolano le olive dallo sfondo o le separano cromaticamente in base alla loro colorazione. Questi algoritmi sono in grado di distinguere in modo accurato tra olive verdi, gialle e nere, analizzando il

contorno e la tonalità di ciascun frutto consentendo una classificazione automatica molto precisa [19]. Un esempio di applicazione di queste tecniche di segmentazione è visualizzabile in Figura 2.2. Un altro approccio avanzato prevede l'utilizzo dell'algoritmo *k-nearest neighbors* (KNN) per determinare il grado di maturazione delle olive. Questo metodo si basa sulla conversione delle immagini in spazio colore RGB e sull'analisi della distribuzione cromatica, un parametro cruciale per valutare se le olive siano più idonee per la produzione di olio o per il consumo diretto [16].

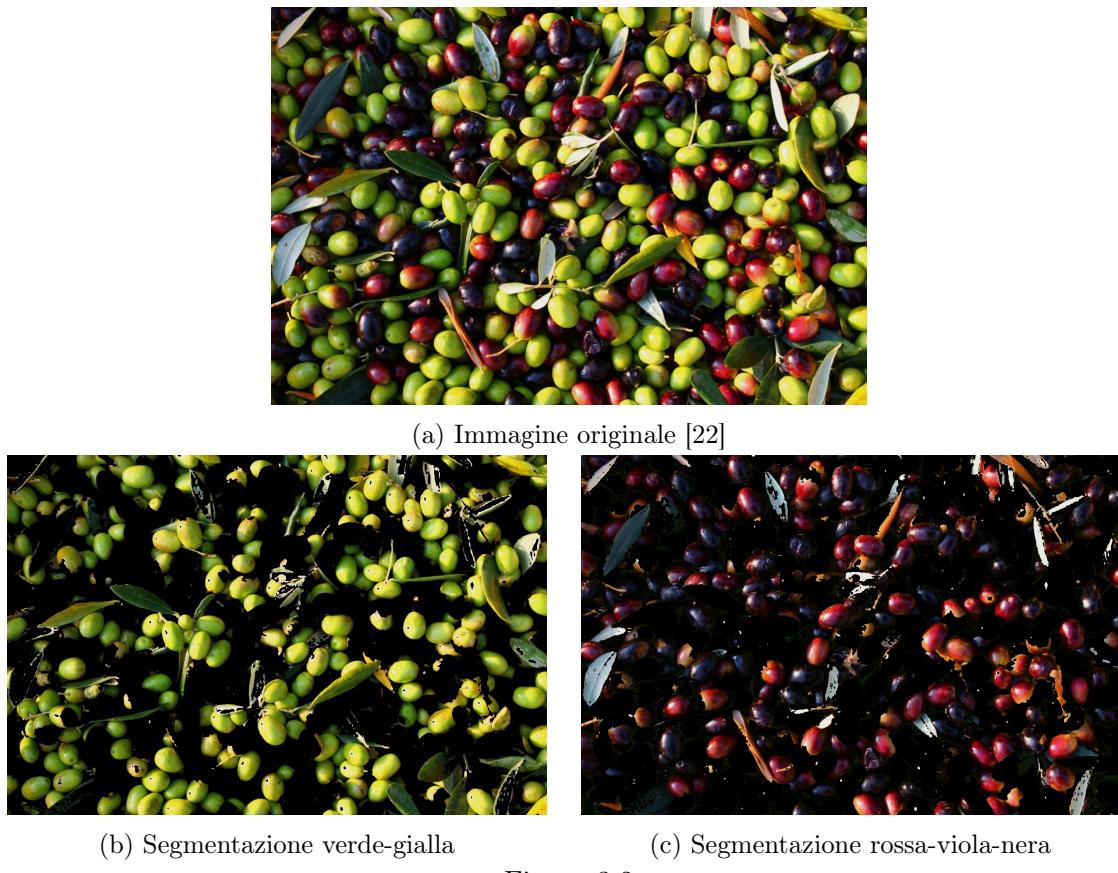


Figura 3.2

Altri approcci più sofisticati hanno fatto uso di reti neurali convoluzionali per la classificazione delle varietà di olive, ottimizzando così i processi di produzione e la selezione dei frutti migliori per l'estrazione dell'olio [18]. Queste reti neurali sono in grado di riconoscere pattern complessi nelle immagini delle olive, permettendo una classificazione più dettagliata e accurata rispetto ai metodi tradizionali. Grazie a queste tecniche il processo di classificazione e smistamento diventa automatizzato, più affidabile e più veloce rispetto ai metodi tradizionali, migliorando la qualità complessiva del prodotto finale.

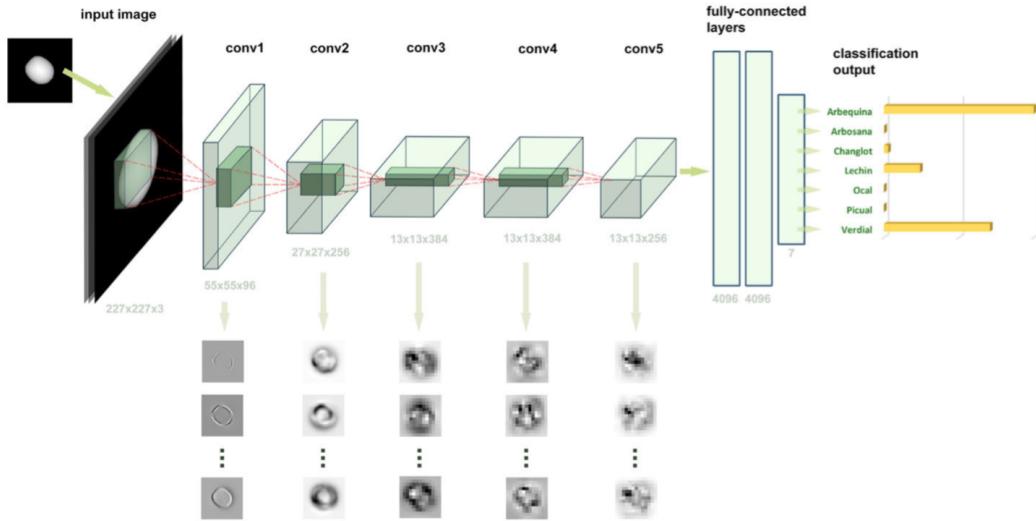


Figura 3.3: Smistamento delle varietà delle olive tramite CNN [18]

3.2.3 Analisi della qualità delle olive

In passato la valutazione e la selezione delle olive erano affidate interamente all’analisi manuale da parte di operatori esperti, che esaminavano le principali caratteristiche fisiche dell’oliva come dimensioni, colore e forma. Questi fattori possono, tuttavia, variare notevolmente in base al grado di maturazione e alla varietà delle olive rendendo la selezione manuale un compito complesso e incline a errori. L’eterogeneità delle olive e la natura soggettiva dell’analisi manuale rendevano difficile garantire una selezione costante e di alta qualità [16]. Con i recenti progressi nella visione artificiale, è ora possibile implementare sistemi automatizzati sulle linee di produzione per identificare e tracciare le olive in tempo reale mentre si muovono sui nastri trasportatori [7]. Questi sistemi sono progettati per funzionare in condizioni difficili, rilevando le olive anche in presenza di rumore e interferenze visive, e offrono soluzioni di automazione a costi relativamente contenuti. Inoltre, sono stati sviluppati algoritmi specifici che analizzano la texture e l’omogeneità delle olive per rilevare con precisione eventuali difetti esterni, distinguendo in modo efficace i frutti sani da quelli difettosi [9]. Studi recenti hanno inoltre evidenziato una correlazione significativa tra le caratteristiche visive delle olive e la qualità dell’olio prodotto. Sulla base di questa correlazione, sono stati sviluppati modelli che analizzano fattori come il colore e la turgidità delle olive per prevedere la qualità dell’olio ottenuto a fine processo di lavorazione [14]. Questi modelli consentono ai produttori di prevedere la qualità dell’olio in modo più affidabile, ottimizzando il processo produttivo e garantendo standard elevati per il prodotto finale.

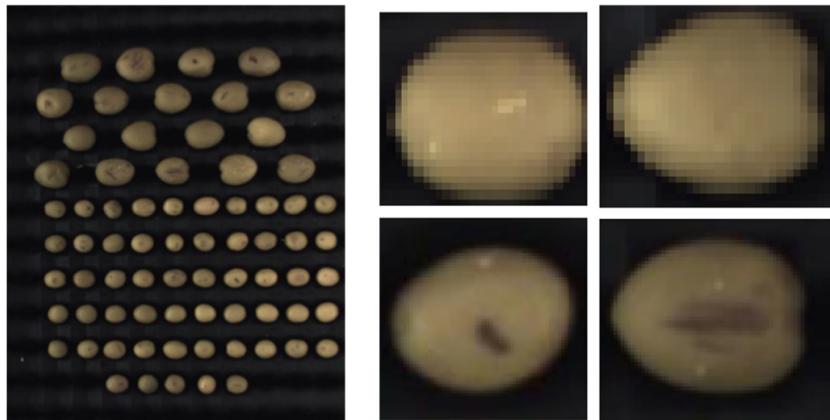


Figura 3.4: Rilevazione dei difetti sulla superficie delle olive [9]

3.2.4 Analisi della qualità dell'olio d'oliva

Oltre alla rilevazione e all'analisi della qualità delle olive, la valutazione della qualità dell'olio prodotto rappresenta un elemento di fondamentale importanza. La qualità dell'olio d'oliva è influenzata da vari fattori, tra cui il grado di maturazione delle olive, la presenza di fogliame indesiderato e il metodo di estrazione utilizzato. Ogni fase, dalla selezione delle olive al processo di estrazione, può avere un impatto significativo sulle caratteristiche organolettiche e nutrizionali dell'olio, determinando la sua classificazione come olio extra vergine, vergine o di categoria inferiore. Gli sviluppi tecnologici recenti hanno introdotto l'uso di algoritmi di machine learning avanzati per analizzare dati ottenuti da tecniche come la *gascromatografia*, al fine di fornire una classificazione più precisa dell'olio d'oliva [30]. La gascromatografia è una tecnica analitica utilizzata per separare e identificare i composti volatili che consente di analizzare la composizione chimica dell'olio. Integrando i risultati di questa tecnica con algoritmi di machine learning, è possibile classificare in modo accurato l'olio in base ai suoi profili chimici, garantendo così un controllo di qualità più rigoroso. Inoltre, alcuni studi hanno sviluppato metodi che combinano la spettroscopia dielettrica e la visione artificiale per prevedere la stabilità ossidativa dell'olio d'oliva [24]. La *spettroscopia dielettrica* misura la risposta dell'olio a campi elettrici variabili e fornisce informazioni sulle sue proprietà fisico-chimiche. Quando combinata con la visione artificiale, questa tecnica permette di eseguire un'analisi non distruttiva e rapida della qualità dell'olio. Tali metodi sono stati progettati per essere integrati nelle linee di produzione, offrendo un monitoraggio in tempo reale della qualità dell'olio e fornendo un importante strumento di controllo durante tutto il processo produttivo.

Capitolo 4

Progettazione e sviluppo

In questo capitolo verranno descritte le principali scelte progettuali che hanno guidato il processo di sviluppo e di realizzazione dell'applicazione. Saranno approfondite le motivazioni alla base delle decisioni tecniche adottate nella selezione di strumenti e tecnologie chiave per la creazione delle principali componenti dell'applicazione tra cui: lo script di esecuzione che integra una rete neurale convoluzionale e tecniche di computer vision per l'analisi delle immagini, il frontend per la visualizzazione da parte dell'utente e il backend per la gestione delle API. Tutto ciò ha contribuito a garantire che il sistema creato sia efficiente, scalabile e accessibile.

4.1 Creazione di un virtual environment

Il primo passo nello sviluppo del progetto è stato creare un ambiente di lavoro isolato e controllato tramite la configurazione di un ambiente virtuale o *virtual environment*. La decisione di utilizzare un ambiente virtuale per lo sviluppo del progetto piuttosto che operare direttamente nell'ambiente globale è stata presa per diverse ragioni. In primis un virtual environment permette di mantenere il progetto completamente isolato da altri progetti o applicazioni che potrebbero essere in esecuzione sulla stessa macchina. Questo isolamento garantisce che tutte le librerie e dipendenze utilizzate nel progetto non interferiscano con le librerie di sistema o con altri progetti che potrebbero richiedere versioni diverse delle stesse librerie. Utilizzando un'ambiente virtuale è quindi possibile evitare problemi di compatibilità e conflitti tra versioni garantendo che il progetto funzioni in maniera prevedibile e stabile. Il suo impiego consente, inoltre, di testare e aggiornare facilmente le dipendenze nell'ambiente globale senza compromettere il funzionamento sistema. Ad esempio, se in futuro fosse necessario aggiornare una particolare libreria o un framework ad una versione successiva, farlo all'interno di un ambiente virtuale permetterebbe di testare il nuovo aggiornamento senza influenzare le altre applicazioni che utilizzano versioni precedenti. Questo garantisce maggiore flessibilità

nel mantenere e sviluppare ulteriormente il progetto. Ulteriore vantaggio sta nella portabilità poiché, essendo tutte le dipendenze del progetto installate all'interno del virtual environment, è più semplice replicare lo stesso ambiente di sviluppo su un'altra macchina o condividerlo con altri sviluppatori. Chiunque voglia eseguire o contribuire al progetto può creare rapidamente lo stesso ambiente semplicemente attivandolo e installando le stesse dipendenze, il che riduce notevolmente i problemi di configurazione.

4.2 Progettazione dello script di esecuzione

4.2.1 Scelta del linguaggio Python e della libreria Pytorch

La scelta di Python [29] come linguaggio di programmazione per lo sviluppo dello script di elaborazione è stata guidata da una serie di vantaggi che lo rendono particolarmente adatto per progetti di machine learning e computer vision. Python si contraddistingue, innanzitutto, per la sua semplicità e chiarezza sintattica, aspetto cruciale nelle fasi di implementazione e sperimentazione poiché facilita la scrittura, il debugging e l'ottimizzazione del codice riducendo i tempi di sviluppo. Altro elemento centrale di Python è la disponibilità di un ecosistema molto ampio di librerie dedicate all'elaborazione dei dati e delle immagini. Strumenti come *NumPy* e *Pandas* permettono una gestione efficiente dei dati, mentre *OpenCV* offre una vasta gamma di funzionalità per la manipolazione delle immagini e l'implementazione di tecniche di computer vision. La versatilità di Python, unita a queste librerie specializzate, consente di affrontare con efficacia attività complesse legate all'analisi delle immagini, come la rilevazione e classificazione degli oggetti. Un altro fattore decisivo è la capacità di Python di integrare facilmente il calcolo su GPU, necessario per l'addestramento e l'inferenza delle reti neurali convoluzionali. Per questo progetto, è stata scelta la libreria PyTorch [17] che offre un'interfaccia semplice e potente per sviluppare modelli di deep learning, supportando nativamente l'uso della GPU tramite CUDA. Questa integrazione permette di sfruttare pienamente la potenza delle GPU per accelerare le operazioni di calcolo intensivo come l'addestramento di modelli complessi su grandi volumi di immagini.

Lo script Python è il cuore dell'intero progetto in quanto gestisce tutto il flusso di lavoro dall'addestramento del modello all' elaborazione dei dati e delle immagini. La scelta combinata di Python e PyTorch si è dimostrata vincente per sviluppare una pipeline completa e performante, garantendo sia la potenza computazionale necessaria per le reti neurali, sia la flessibilità per gestire l'intero ciclo di elaborazione delle immagini.

4.2.2 Scelta del modello di rete neurale YOLOv8

YOLOv8 (You Only Look Once) è una rete neurale convoluzionale avanzata progettata per il rilevamento e la classificazione in tempo reale di oggetti all'interno delle immagini. Quest'architettura si contraddistingue per la sua capacità di eseguire l'intero processo di rilevamento in un'unica fase, risultando rapida ed efficiente e differenziandosi da altre reti convoluzionali che suddividono il rilevamento in più fasi. Uno studio si è occupato di confrontare vari modelli di CNN nella rilevazione delle olive all'interno delle immagini per valutarne le prestazioni: la rete YOLOv8, come mostrato nelle tabelle in Figura 4.1, ha mostrato risultati più promettenti rispetto agli altri modelli impiegati in termini di precisione e tempo di addestramento, pur utilizzando un'architettura più semplice e un minore numero di parametri impiegati [8].

Models	No. of Layers	No. of Parameters	Training Time (in h)	Inference Time (in s)	Models	Precision	Recall	F1-score	Dice Index
YOLOv8	261	35 842 476	2.2	0.7	YOLOv8	0.8611	0.8312	0.8459	0.8670
Mask R-CNN	325	39 163 213	3.5	2.04	Mask R-CNN	0.8219	0.8158	0.8188	0.7606
U-Net	395	45 714 688	4.8	3.29	U-Net	0.7852	0.7546	0.7696	0.7139
FCN	420	46 490 104	5.2	3.95	FCN	0.7391	0.7223	0.7306	0.6212

(a) Confronto fra le architetture

(b) Confronto fra le prestazioni

Figura 4.1

Uno dei principali vantaggi di YOLOv8 è il suo equilibrio ottimale tra velocità e precisione. Il progetto richiede un'elaborazione veloce ed efficace di numerose immagini, anche ottenute da diverse fonti tra cui fotocamere o droni. La rete riesce inoltre a mantenere un elevato livello di accuratezza anche in condizioni difficili, come per immagini caratterizzate da illuminazione variabile o riprese da angolazioni inconsuete. YOLOv8 è in grado di identificare oggetti anche in immagini caratterizzate da rumore o da ostruzioni parziali, riducendo così il rischio di errori di rilevamento o falsi positivi. Questa robustezza è cruciale nel caso di condizioni di visibilità non favorevoli perché compromesse, ad esempio, dalla presenza di fogliaame o da un'illuminazione irregolare ed è particolarmente rilevante per il progetto poiché la qualità delle immagini può essere influenzata da molteplici fattori ambientali o da variazioni nella configurazione del dispositivo di acquisizione. Infine, uno degli aspetti più rilevanti che rende YOLOv8 particolarmente adatta per il progetto è la sua capacità di rilevare, con un alto grado di precisione, oggetti di piccole dimensioni all'interno di immagini più grandi e complesse, essenziale per garantire un'accurata rilevazione delle olive e una corretta analisi visiva delle coltivazioni.

Questo particolare modello di rete neurale convoluzionale è, quindi, stato scelto per il suo equilibrio tra velocità e accuratezza, per la sua capacità di adattamento a scenari complessi e per la sua compatibilità con le tecnologie di accelerazione su

GPU; queste caratteristiche chiave la rendono lo strumento ideale per la rilevazione e classificazione delle olive e, di conseguenza, per la corretta realizzazione del progetto.

4.3 Progettazione del Frontend

4.3.1 Scelta del linguaggio Javascript e del framework React

JavaScript [5] è il linguaggio più utilizzato per la creazione di applicazioni web interattive. Quest'ultimo consente di creare interfacce dinamiche, gestire l'interazione con l'utente e comunicare con il backend in modo efficiente grazie alla vasta gamma di librerie e framework disponibili che accelerano lo sviluppo, garantendo flessibilità e un'ampia possibilità di personalizzazione. Javascript è supportato da tutti i moderni browser e, inoltre, la sua capacità di gestire sia la logica di presentazione che le chiamate asincrone lo rende una scelta ideale per applicazioni web che necessitano di un aggiornamento dei dati in tempo reale.

Per sfruttare appieno le potenzialità di JavaScript è stato scelto il framework React [31] per lo sviluppo del frontend. React facilita lo sviluppo di interfacce modulari grazie alla sua architettura basata su componenti; l'interfaccia viene, infatti, suddivisa in blocchi indipendenti che risultano facilmente riutilizzabili e manutenibili. Il *Virtual DOM* di React ottimizza gli aggiornamenti dell'interfaccia, migliorando le prestazioni soprattutto in applicazioni dinamiche. Ciò avviene perché quando si interagisce con la pagina vengono aggiornate dinamicamente solo le porzioni di pagina o le componenti direttamente interessate, non la pagina nella sua interezza. Tutto questo si traduce in una navigazione più fluida e in un'esperienza utente migliorata, anche quando l'applicazione è chiamata a gestire grandi quantità di dati o richiede aggiornamenti frequenti. Per la gestione degli stili è stata adottata la libreria *Styled-components*, che consente di scrivere CSS all'interno dei file JavaScript legando gli stili direttamente alla logica del componente. Questo approccio riduce i conflitti di stile e mantiene il codice pulito e facile da gestire, favorendo la coerenza visiva dell'applicazione. Infine, per la gestione delle richieste HTTP e la comunicazione con il backend è stata utilizzata la libreria *Axios*, che semplifica l'invio di dati (come le immagini da analizzare) e il recupero dei risultati. Axios gestisce in modo efficiente le chiamate API, facilitando l'interazione tra frontend e backend.

In sintesi, la combinazione di JavaScript e React, insieme a strumenti come styled-components e axios, ha permesso di realizzare un frontend moderno, reattivo e facilmente scalabile, garantendo un'elevata usabilità e prestazioni ottimizzate per l'utente finale.

4.4 Progettazione del Backend

4.4.1 Scelta del micro-framework Flask basato su Python

Python, già ampiamente introdotto ed utilizzato per lo script di esecuzione, è stato scelto principalmente per la sua semplicità d'uso e perché garantisce coerenza tra le varie componenti dell'applicazione, evitando la frammentazione tecnologica e semplificando il processo di sviluppo. Questo approccio unificato ha ridotto la complessità complessiva del progetto, migliorando l'integrazione e rendendo il sistema più robusto ed efficiente.

Python integra, come già esposto, una vasta gamma di librerie e framework, molte delle quali facilitano la gestione del backend e delle richieste asincrone. In particolare per lo sviluppo del server backend è stato scelto Flask [23], un micro-framework minimale per lo sviluppo di web app. Uno dei motivi principali per cui è stato scelto Flask è la sua architettura semplice e la capacità di offrire agli sviluppatori un controllo completo su come strutturare il backend. A differenza di framework più completi come *Django*, che impongono un'architettura rigida e includono molte funzionalità predefinite, Flask consente di personalizzare completamente il flusso di lavoro. Questa flessibilità è stata essenziale per l'integrazione con lo script di elaborazione delle immagini e per la gestione delle richieste inviate dal frontend. Questo micro-framework supporta nativamente il protocollo WSGI (Web Server Gateway Interface) consentendo di scalare facilmente l'applicazione tramite l'integrazione con server web come *Gunicorn*. Questo rende il framework adatto anche per applicazioni che, pur mantenendo una struttura semplice, necessitano di gestire un elevato numero di richieste in parallelo. Flask si è inoltre rivelato ottimale per il supporto alle estensioni. Questa caratteristica ha reso facile l'integrazione di strumenti come *Flask-CORS* per gestire le richieste cross-origin tra il frontend e il backend garantendo la sicurezza e l'efficienza del flusso di dati. L'estensione *Werkzeug* è stata utilizzata per garantire la sicurezza dei file caricati, prevenendo vulnerabilità legate ai nomi dei file e assicurando che ogni file sia trattato in modo sicuro e affidabile.

In definitiva, Flask ha permesso di mantenere il sistema snello, consentendo di costruire solo le funzionalità strettamente necessarie per garantire una gestione efficace dei file caricati, delle API e della comunicazione con il frontend. La sua architettura minimalista ha permesso di adattare l'applicazione alle specifiche esigenze del progetto senza dover gestire la complessità aggiuntiva di framework più pesanti. La combinazione di Flask con Python ha fornito una soluzione semplice e potente, consentendo uno sviluppo rapido e una facile integrazione con le altre componenti del sistema, garantendo prestazioni elevate e facilità di manutenzione.

Capitolo 5

Modello e architettura

5.1 Analisi dei Requisiti

L'applicazione deve essere progettata per rispettare una serie di requisiti funzionali e non funzionali che permettono di soddisfare le aspettative degli utenti, offrire un'esperienza utente ottimale e operare in modo efficiente in diversi contesti.

5.1.1 Requisiti Funzionali

I requisiti funzionali descrivono le operazioni principali che l'app deve eseguire per rispondere alle necessità degli utenti. Il primo requisito da rispettare è la **funzione di caricamento** delle immagini: gli utenti devono poter caricare immagini dei rami d'ulivo utilizzando dispositivi comuni in modo intuitivo e veloce. Questo processo deve essere immediato e non richiedere competenze tecniche avanzate, così da permettere un utilizzo ampio da parte di utenti con diversi livelli di esperienza.

Un'altro requisito da rispettare sta nell'**analisi automatica** delle immagini: una volta caricate, le immagini devono essere processate sul momento utilizzando algoritmi di machine learning e visione artificiale. Questo comporta la capacità dell'applicazione di estrarre dalle immagini, in maniera rapida e precisa, parametri chiave come la quantità di olive presenti, la loro dimensione, il colore e il grado di maturità.

L'applicazione deve inoltre assicurare la **rielaborazione e aggregazione dei dati prodotti**. Dopo l'analisi delle singole immagini, l'app deve raggruppare le informazioni ottenute per fornire stime totali e rappresentazioni statistiche della distribuzione dei parametri osservati. Per esempio, l'utente dovrebbe poter vedere non solo la quantità totale di olive analizzate, ma anche la distribuzione della loro grandezza o il grado medio di maturità delle olive su un intero campo o lotto.

Infine, l'applicazione deve garantire un'**esposizione dei risultati ottenuti**, utilizzando formati di output di semplice interpretazione. Questo significa che l'app

deve presentare le informazioni attraverso grafici, tabelle o testi semplici, facilmente interpretabili sia per produttori esperti che per hobbisti o piccoli coltivatori.

5.1.2 Requisiti Non Funzionali

Oltre ai requisiti funzionali, l'applicazione deve soddisfare anche una serie di requisiti non funzionali, che riguardano la qualità del sistema e l'esperienza utente complessiva. Il primo requisito è l'**accessibilità**. L'app deve infatti essere accessibile da qualsiasi dispositivo dotato di browser senza necessità di installare software specifici. Ciò implica che sia compatibile con diverse piattaforme, dai computer desktop agli smartphone, garantendo una fruibilità universale e adattandosi alle esigenze di ogni produttore.

Un altro requisito da soddisfare è che l'interfaccia utente sia **intuitiva** e **user-friendly**. L'applicazione deve essere progettata in modo tale da ridurre al minimo la curva di apprendimento, consentendo a utenti con un qualsiasi livello di competenza tecnologica di utilizzarla senza difficoltà. Le operazioni più comuni, come il caricamento delle immagini e la visualizzazione dei risultati, devono essere eseguibili con pochi clic e presentate in un layout semplice e ordinato.

Anche la **scalabilità** è un requisito cruciale, soprattutto per i grandi produttori che gestiscono oliveti di grandi dimensioni. L'applicazione deve essere in grado di gestire contemporaneamente un elevato numero di immagini e utenti senza comprometterne le prestazioni. Ciò significa che il sistema deve essere capace di elaborare grandi volumi di dati, garantendo al contempo una risposta rapida e mantenendo alta l'efficienza anche in situazioni di carico pesante.

Infine, l'app deve essere progettata per gestire la **variabilità** nelle condizioni delle immagini. Poiché le foto possono essere scattate in condizioni di luce e visibilità diverse, o con dispositivi di qualità variabile, l'applicazione deve implementare algoritmi di pre-elaborazione delle immagini per correggere eventuali difetti dati da scarsa luminosità, ombre o distorsioni. Così facendo viene garantito uno standard di qualità minima delle immagini prima che esse vengano sottoposte all'analisi.

5.2 Modello progettuale

Il progetto ha come obiettivo principale la realizzazione di una web app che utilizza tecnologie avanzate di intelligenza artificiale e computer vision per automatizzare i processi di rilevazione e classificazione delle olive. La web app deve essere strutturata come una **Single Page Application** il cui sistema è concepito per facilitare l'analisi automatizzata delle immagini caricate dagli utenti, sfruttando tecniche di preprocessamento delle immagini per migliorare la qualità visiva e una rete neurale convoluzionale per identificare e classificare automaticamente le olive all'interno delle immagini. Essendo una SPA, l'intera applicazione opera su una singola pagina

web che dinamicamente aggiorna i contenuti senza richiedere il caricamento di nuove pagine. Questo approccio migliora notevolmente l’esperienza utente, rendendo l’applicazione più veloce e fluida. L’utente può caricare le immagini, visualizzare i risultati e interagire con l’interfaccia in modo continuo e reattivo, senza interruzioni o ridondanze. L’architettura della web app si articola in tre componenti principali:

- Uno **script di esecuzione** che integra sia una rete neurale convoluzionale YOLOv8 che algoritmi di computer vision e li utilizza per l’elaborazione delle immagini, la rilevazione e classificazione automatica delle olive e per fornire informazioni dettagliate come la dimensione e la maturità delle olive.
- Un **frontend**, sviluppato con tecnologie moderne come React, che gestisce l’interfaccia utente in modo dinamico e reattivo permettendo agli utenti di caricare facilmente immagini e di visualizzare i risultati dell’analisi.
- Un **backend** basato su Flask, che si occupa della gestione delle richieste server-side, gestendo l’esecuzione dello script Python per l’elaborazione delle immagini e restituendo i risultati al frontend.

Grazie all’approccio Single Page Application, l’applicazione garantisce velocità, reattività e facilità d’uso, generando risultati accurati e affidabili attraverso l’uso di tecniche avanzate di machine learning e computer vision.

5.3 Struttura e funzionamento dell’architettura

L’applicazione sviluppata adotta un’**architettura client-server**. Vi è una chiara separazione dei ruoli tra frontend e backend i quali comunicano attraverso **richieste HTTP** standard: il frontend si occupa dell’interazione con l’utente e della presentazione dei dati, mentre il backend gestisce l’elaborazione e coordina l’esecuzione dello script di elaborazione. La stretta collaborazione tra le componenti assicura che ogni parte del sistema svolga il proprio in maniera coordinata, contribuendo all’efficienza complessiva dell’applicazione.

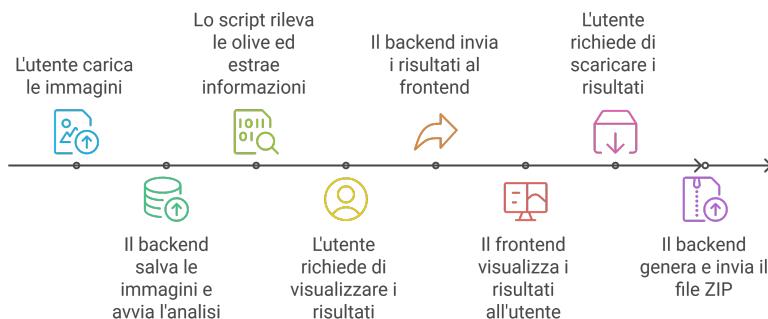


Figura 5.1: Flusso operativo di funzionamento della web app

Per comprendere il flusso operativo dell'applicazione (rappresentato graficamente in Figura 5.1) si può fare riferimento ad una rappresentazione più completa e dettagliata fornita dal diagramma di sequenza presente in Figura 5.2. Basandoci sul diagramma, possiamo descrivere in dettaglio le fasi operative dell'applicazione, evidenziando come le diverse componenti collaborino tra loro nei vari step del funzionamento:

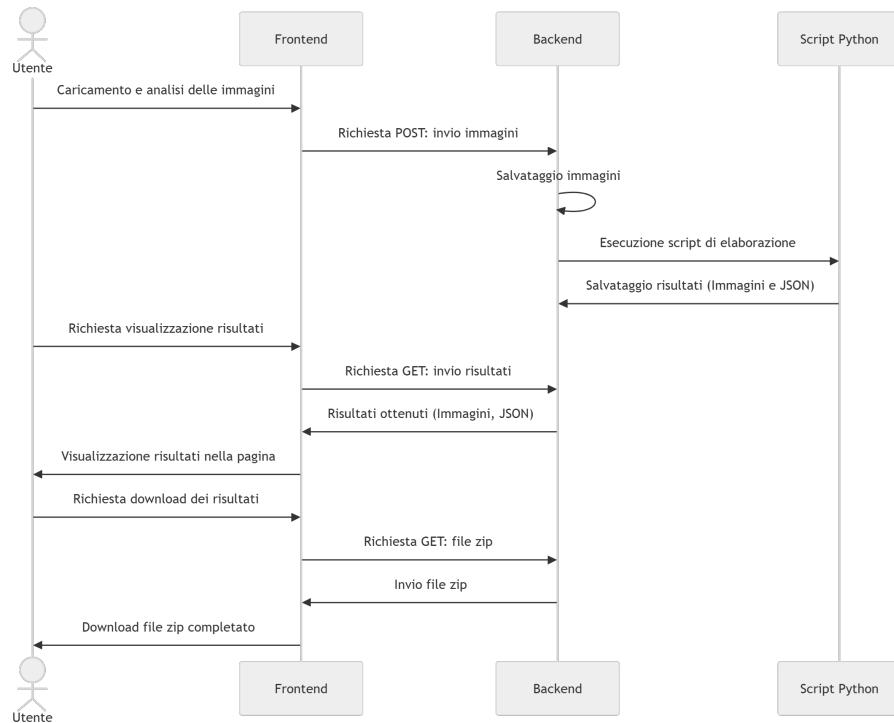


Figura 5.2: Diagramma di sequenza

1. Caricamento e analisi delle immagini

Il processo ha inizio quando l'utente interagisce con il frontend dell'applicazione per caricare le immagini delle olive da analizzare. L'interfaccia permette di selezionare facilmente una o più immagini dal dispositivo dell'utente e, una volta effettuata la selezione, il frontend invia le immagini al backend tramite una richiesta HTTP POST. Il backend riceve le immagini e le salva temporaneamente in una cartella dedicata sul server. Questa scelta semplifica la gestione dei file durante l'elaborazione ed evita la complessità di utilizzare un database per l'archiviazione delle immagini. Dopo aver salvato le immagini, il backend avvia l'esecuzione dello script Python responsabile dell'analisi. Il backend coordina l'esecuzione dello script, passando le immagini da elaborare e gestendo eventuali eccezioni o errori che potrebbero verificarsi. Lo script identifica le olive tramite la predizione da parte della rete neurale e disegna bounding box attorno a ciascuna di esse. Inoltre, attraverso tecniche di

computer vision, vengono estratte informazioni aggiuntive come le dimensioni delle olive, il peso stimato e il grado di maturità. La comunicazione client-server avviene attraverso un protocollo standardizzato, garantendo una trasmissione efficiente e sicura dei dati. I risultati dell’elaborazione, comprensivi delle immagini annotate, dei grafici e dei file JSON contenenti i dati strutturati dell’analisi, vengono salvati dal backend in una nuova cartella diversa da quella del caricamento. Questa organizzazione facilita l’accesso ai dati e semplifica le operazioni di recupero da parte del backend quando il frontend richiede i risultati. La coordinazione tra frontend e backend in questa fase è fondamentale per assicurare che le immagini vengano ricevute correttamente per essere poi elaborate tramite lo script e per garantire un’analisi accurata e tempestiva.

2. Richiesta di Visualizzazione dei Risultati

Una volta completata l’elaborazione, l’utente può richiedere di visualizzare i risultati tramite l’interfaccia del frontend. L’utente interagisce con l’applicazione per avviare questa fase e il frontend invia una richiesta HTTP GET al backend per ottenere i dati elaborati. Il frontend deve essere in grado di presentare i risultati non appena disponibili e il backend deve rispondere prontamente con i dati richiesti fornendo al frontend i file JSON, i grafici e le immagini annotate contenuti all’interno della stessa cartella. Il frontend elabora questi dati e li presenta all’utente in modo chiaro e accessibile sotto forma di dashboard in maniera tale che l’utente possa interpretare facilmente i risultati dell’analisi. La coordinazione tra le componenti assicura che i dati siano correttamente trasmessi e visualizzati, offrendo all’utente un’esperienza soddisfacente.

3. Richiesta e Download dei Risultati

Se l’utente desidera scaricare i risultati dell’analisi per salvarli sul suo dispositivo può farlo attraverso un’apposita funzionalità del frontend inviando una richiesta HTTP GET al backend; quest’ultimo genera un file ZIP contenente tutti i risultati elaborati e lo invia al frontend permettendo all’utente di scaricarlo direttamente sul proprio dispositivo. La coordinazione tra frontend e backend in questa fase garantisce che il file venga generato correttamente e che il download avvenga senza intoppi.

Ogni componente può essere aggiornata o migliorata in maniera indipendente, senza influire negativamente sulle altre parti dell’applicazione. Questa flessibilità è essenziale per garantire che l’applicazione possa evolvere nel tempo, adattandosi a nuove esigenze o integrando tecnologie emergenti. Nel complesso, la separazione di questi componenti permette di realizzare un’applicazione potente, versatile e facilmente adattabile a scenari operativi diversi.

Capitolo 6

Implementazione e validazione

In questo capitolo viene illustrato l'intero processo di implementazione e validazione della web app per l'analisi e la classificazione delle olive il cui processo di sviluppo e realizzazione è stato ampiamente esposto nei capitoli precedenti. Il processo si è articolato in 5 fasi principali che riflettono la sequenza temporale delle attività svolte: la preparazione dei dati, l'addestramento del modello YOLOv8, lo sviluppo dello script Python per l'elaborazione delle immagini, l'implementazione del backend per la gestione delle API e, infine, la realizzazione del frontend per l'interfaccia utente.

6.1 Preparazione dei dati

La prima fase del progetto è stata dedicata alla raccolta e alla preparazione dei dati necessari per l'addestramento della rete neurale convoluzionale YOLOv8. Durante il processo di ricerca dei dati è stato individuato un dataset contenente immagini di rami di olivo, ciascuna di esse corredata con una specifica etichetta YOLO. Questo particolare tipo di etichetta consiste in un file testuale contenente informazioni riguardo la posizione e le dimensioni di ognuno dei riquadri detti **bounding box** che racchiudono all'interno una singola oliva; l'insieme delle etichette associate alle immagini consente, pertanto, di mappare la presenza di tutte le olive presenti all'interno della totalità delle immagini. Il dataset iniziale conteneva nello specifico 972 istanze definite dalla coppia immagine-etichetta tuttavia, per addestrare un modello preciso e capace di generalizzare efficacemente su nuove immagini mai scansionate, sono state utilizzate particolari tecniche di **Data Augmentation**. Effettuate mediante l'utilizzo della libreria *Albumentations*, queste tecniche consentono di aumentare la variabilità, la rappresentatività e le dimensioni del dataset iniziale attraverso un aumento artificiale del numero di immagini disponibili attraverso trasformazioni come rotazioni, traslazioni, cambiamenti di luminosità e contrasto. Insieme alle immagini, per garantire l'integrità e la coerenza del data set aumentato, vengono anche create le nuove label associate alle immagini aumentate che

contengono le informazioni aggiornate sulla posizione e dimensione delle bounding box una volta effettuate le trasformazioni. Ciò ha permesso di generare, per ogni istanza di partenza, 5 nuove istanze con caratteristiche modificate, portando il numero totale di elementi nel dataset da 972 a 5832. Il dataset complessivo (raw data + augmented data) è stato quindi suddiviso in tre set distinti per l’addestramento del modello: il training set, che rappresenta l’80% del dataset (4665 istanze) ed è stato utilizzato per l’addestramento vero e proprio, il validation set, che rappresenta il 10% (583 istanze) ed è stato impiegato per monitorare le prestazioni del modello durante l’addestramento, e il test set, che rappresenta il restante 10% (584 istanze) ed è riservato alla valutazione finale delle prestazioni su dati non visti in precedenza.

6.2 Addestramento del modello Yolov8

Dopo aver completato la fase di preparazione dei dati l’attenzione si è focalizzata sull’addestramento del modello di rete neurale convoluzionale YOLOv8. L’intero processo di addestramento è stato condotto su una piattaforma hardware dotata di una GPU NVIDIA RTX 4060 per laptop che ha permesso di mantenere elevate prestazioni computazionali, riducendo significativamente i tempi di addestramento anche grazie ai suoi 8 GB di VRAM dedicata. Sul fronte software è stata utilizzata la libreria PyTorch, che ha consentito di sfruttare appieno le potenzialità della GPU grazie all’integrazione con pacchetti ottimizzati come *CUDA* v11.8 e *cuDNN* v9.0. Questi strumenti sono risultati essenziali per accelerare i complessi calcoli tipici delle reti neurali profonde.

Sono state considerate tre varianti del modello YOLOv8: NANO, SMALL e MEDIUM. Questi modelli, già pre-addestrati, possiedono parametri e pesi ottimizzati per il riconoscimento di una vasta gamma di oggetti all’interno delle immagini. La principale differenza tra le varie versioni risiede nel numero di parametri e nella complessità dell’architettura, elementi che influenzano sia le prestazioni sia i requisiti computazionali. Attraverso un processo di riaddestramento detto **fine-tuning**, questi modelli sono stati specializzati per il compito specifico di riconoscere le olive a partire dal dataset fornito. Questo approccio ha permesso di sfruttare le conoscenze preesistenti del modello, adattandole al nuovo contesto applicativo. La scelta degli iperparametri ha rivestito un ruolo cruciale nel garantire un equilibrio efficace tra il tempo di addestramento e le prestazioni finali del modello. Le immagini del dataset sono state ridimensionate a una risoluzione standard di 640×640 pixel, processo che ha uniformato l’input al modello facilitando le operazioni di convoluzione e ottimizzando l’utilizzo delle risorse computazionali. Per quanto riguarda il numero di epoche sono state previste 200 iterazioni per consentire al modello un apprendimento approfondito ma è stata implementata una strategia di **early**

stopping con una pazienza di 10 epoch che interrompe l’addestramento in assenza di miglioramenti significativi nelle prestazioni. Il batch size è stato fissato a 16, un compromesso tra l’efficienza computazionale e l’utilizzo ottimale della memoria GPU mentre, per evitare un sovraccarico della GPU mantenendo un’efficiente parallelizzazione, il numero di worker attivi è stato impostato a 2. Altri iperparametri come il learning rate e il momentum sono stati decisi dinamicamente dall’algoritmo stesso durante la fase di addestramento.

Durante l’addestramento di ogni versione del modello, al termine di ogni epoca, è stata effettuata una fase di validazione. Questo passaggio ha permesso di monitorare in tempo reale le metriche chiave del modello, quali precisione, recall e mean Average Precision (mAP). L’analisi di questi indicatori ha fornito una valutazione dettagliata dell’efficacia di ciascuna versione del modello, consentendo confronti diretti tra le diverse architetture. Dai risultati ottenuti il modello YOLOv8 MEDIUM ha dimostrato le migliori prestazioni nel riconoscimento delle olive, raggiungendo un’accuratezza dell’88%. Questo risultato è in linea con la maggiore complessità e il numero di parametri della sua architettura rispetto alle versioni SMALL e NANO, che hanno comunque ottenuto risultati promettenti rispettivamente con un’accuratezza dell’85% e del 83% (come visibile nella Figura 6.1). Alla luce di questi risultati, il modello YOLOv8 MEDIUM addestrato è stato selezionato per le successive fasi operative. Il modello è stato salvato e integrato nello script Python per le operazioni di inferenza, garantendo un elevato livello di accuratezza nel riconoscimento delle olive.

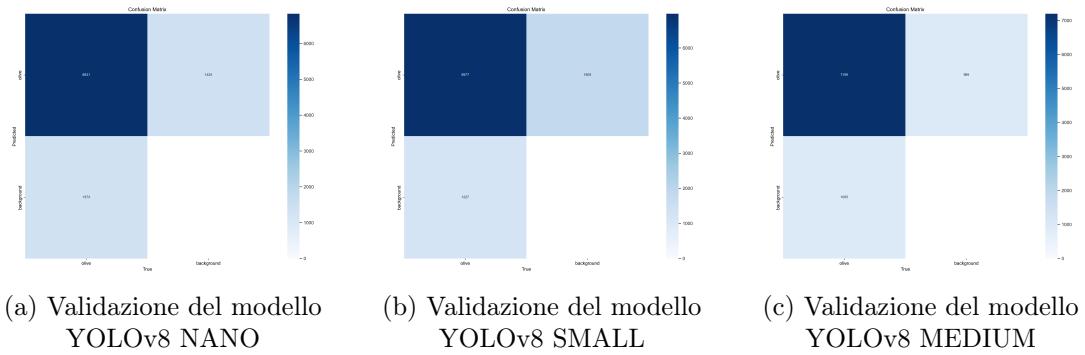


Figura 6.1: Confronto fra i risultati delle validazioni

6.3 Creazione dello script di elaborazione

La terza fase del progetto ha riguardato la creazione dello script Python che implementa l’intera pipeline di elaborazione delle immagini, partendo dalla rilevazione delle olive tramite il modello YOLOv8 addestrato fino all’estrazione e alla visualizzazione delle informazioni chiave come dimensioni, peso, colore e grado di maturità

delle olive. Il processo di sviluppo è iniziato in un ambiente Jupyter Notebook, che ha facilitato notevolmente la fase di prototipazione poiché ha permesso di testare rapidamente le varie componenti dello script e di visualizzare immediatamente i risultati ottenuti, consentendo di iterare efficacemente sulla soluzione. Una volta verificato il corretto funzionamento e ottimizzato il codice, lo script è stato convertito in un file Python eseguibile, pronto per essere integrato nell'architettura della web app. Per l'elaborazione delle immagini e l'implementazione delle funzioni di computer vision sono state utilizzate le librerie *OpenCV* e *NumPy* mentre la libreria *Matplotlib* è stata impiegata per la generazione di grafici e visualizzazioni dei risultati. Inoltre, il modello YOLOv8 è stato gestito tramite la libreria *Ultralytics* che offre un'interfaccia intuitiva per l'utilizzo dei modelli YOLO, facilitando l'inferenza e l'elaborazione dei risultati.

Lo script inizia con il caricamento delle immagini caricate dall'utente nella cartella di upload. Per ogni immagine, viene effettuato un potenziamento tramite una funzione che aumenta la saturazione e la luminosità utilizzando lo spazio colore HSV. Questo passaggio è fondamentale per accentuare le differenze cromatiche tra le olive, facilitando le successive fasi di analisi del colore e classificazione del grado di maturità. Il modello YOLOv8 precedentemente addestrato viene caricato e utilizzato per eseguire l'inferenza su ogni immagine. Il modello identifica le olive presenti e restituisce le bounding box i relativi indici di confidenza, che indicano la probabilità che l'oggetto rilevato sia effettivamente un'oliva. Per ogni bounding box individuata, lo script procede a tracciare un'ellisse inscritta che rappresenta un'approssimazione abbastanza accurata della forma dell'oliva, consentendo una stima più precisa delle sue dimensioni reali. Viene quindi calcolata l'area dell'ellisse in pixel e, associando un valore medio di 0,003 grammi per pixel quadrato, si riesce a stimare il peso della singola oliva. Considerando che le dimensioni apparenti delle olive possono variare significativamente a seconda della distanza e dell'angolazione della foto, è stato implementato un algoritmo di controllo per gestire eventuali sovrastime del peso: se il peso calcolato di un'oliva supera gli 11 grammi, viene impostato a un valore predefinito di 6 grammi ma quel valore viene contrassegnato come *adjusted*. Questo meccanismo garantisce che le stime siano realistiche e coerenti, evitando che olive fotografate da vicino vengano considerate anormalmente pesanti. Successivamente avviene la determinazione del grado di maturità delle olive, basato sull'analisi del colore. Utilizzando una maschera che evidenzia i colori predominanti all'interno delle ellissi tracciate l'immagine viene convertita in scala di grigi e per ogni oliva viene calcolato il valore medio di intensità di grigio interno all'ellisse. In base a questo valore, le olive vengono classificate in tre categorie:

- "Matura" se il valore di grigio è inferiore o uguale a 140, che rappresenta una colorazione dell'oliva nello spettro del rosso/viola/nero;
- "Poco matura" se compreso tra 140 e 180, che rappresenta una oliva in stato di maturazione con colorazione tendente al giallo;

6.3 – Creazione dello script di elaborazione

- "Non matura" se superiore a 180, che identifica un'oliva ancora prematura di colorazione verde.

Questa classificazione permette di distinguere le olive in base al loro stadio di maturazione, sfruttando le differenze cromatiche naturali che si manifestano durante il processo di maturazione. A seguito della classificazione all'interno delle immagini le ellissi generate vengono riempite di colore arancione, giallo o verde a seconda che esse vengano classificate rispettivamente come non mature, poco mature o mature; ciò permette un'interpretazione grafica più semplice ed intuitiva da parte dell'utente in fase finale. Per ogni immagine processata lo script genera, quindi, una serie di immagini annotate in diversi subplot che identificano i diversi stadi della pipeline di elaborazione, tra cui: l'immagine contenente il numero di olive rilevate contrassegnate dalle rispettive bounding box di colore blu e le etichette sull'indice di confidenza, l'immagine contenente informazioni sul peso totale delle olive rilevate e le ellissi di colore rosso per la ricostruzione della loro forma e l'immagine potenziata in fase iniziale con la classificazione sul grado di maturità. Un esempio di immagine generata al termine della pipeline è visualizzabile all'interno della Figura 6.2.

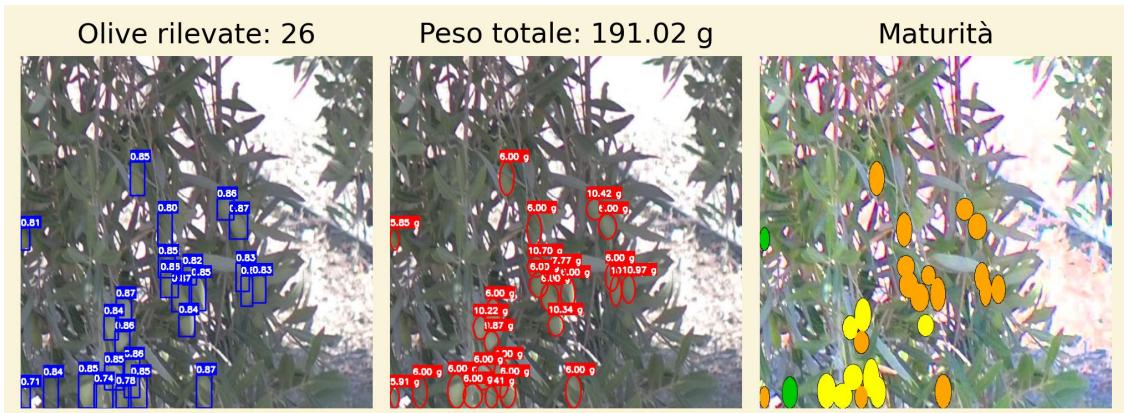


Figura 6.2: Risultato grafico generato dalla pipeline di elaborazione

Vengono inoltre calcolate statistiche numeriche come il numero totale di olive rilevate, il peso totale stimato, la distribuzione delle olive per grado di maturità e altri indicatori chiave. Questi dati vengono salvati in un file JSON permettendo una facile integrazione con il frontend dell'applicazione web e facilitando la visualizzazione delle informazioni all'utente. Per rappresentare graficamente i risultati lo script utilizza *Matplotlib* per generare grafici che mostrano la distribuzione del peso delle olive e la distribuzione per grado di maturità, visualizzabili in Figura 6.4. Particolare attenzione è stata dedicata all'estetica dei grafici, utilizzando colori e stili che facilitano la comprensione e l'interpretazione dei dati da parte dell'utente.

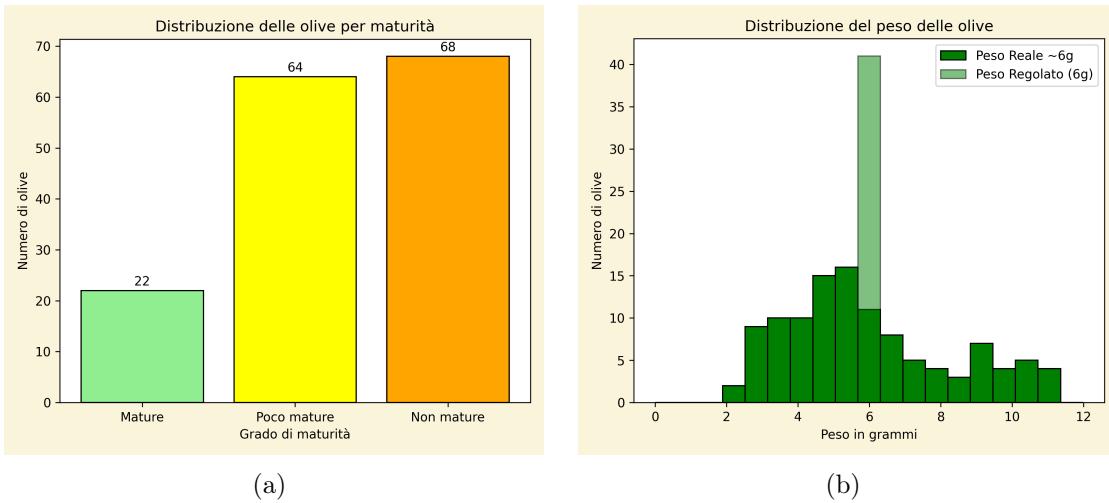


Figura 6.3: Grafici delle distribuzioni delle olive per grado di maturità (a) e per peso (b)

Per migliorare l’esperienza utente, lo script include una funzione che aggiorna un file JSON con lo stato di avanzamento dell’elaborazione. Questo permette al frontend di mostrare all’utente una barra di progresso, fornendo feedback in tempo reale sullo stato dell’operazione e migliorando l’interattività dell’applicazione. Dal punto di vista delle prestazioni l’uso di funzioni modulari all’interno dello script e la gestione attenta delle risorse computazionali consentono di elaborare rapidamente un numero elevato di immagini senza sovraccaricare il sistema. La struttura modulare del codice facilita inoltre la manutenzione e l’estensione futura delle funzionalità, rendendo lo script flessibile e adattabile a nuove esigenze o miglioramenti.

6.4 Creazione del server Backend

Dopo aver completato lo script di elaborazione delle immagini, lo step successivo è stato concentrarsi sulla realizzazione del server backend, fondamentale per gestire l’interazione tra le varie componenti dell’applicazione. Il backend ha, infatti, il compito di gestire le richieste provenienti dal frontend, coordinare l’esecuzione dello script di elaborazione e fornire i risultati all’utente in modo efficiente e sicuro. Per gestire il flusso di dati tra il frontend e il backend, sono stati definiti diversi endpoint API, ciascuno con un ruolo specifico all’interno del flusso operativo dell’applicazione. Questi endpoint permettono di orchestrare le varie fasi del funzionamento della web app, dall’inizializzazione dell’ambiente all’ottenimento dei risultati finali.

All’avvio dell’applicazione, o prima di iniziare una nuova sessione di analisi, il frontend invia una richiesta HTTP, utilizzando il metodo POST, all’endpoint `/clean_folders` il quale ha il compito di pulire le cartelle `upload` e `results`, eliminando eventuali dati residui da precedenti elaborazioni, e di reimpostare il file `progress.json` che tiene traccia dello stato di avanzamento dell’elaborazione.

Questa operazione garantisce che ogni nuova sessione parta da uno stato pulito, evitando conflitti o sovrapposizioni di dati. Una volta preparato l’ambiente, l’utente può caricare le immagini da analizzare attraverso l’interfaccia web. Il frontend gestisce l’upload dei file e, una volta selezionate le immagini, invia una richiesta POST all’endpoint `/upload`. Il backend riceve i file, verifica che siano presenti nella richiesta e controlla che ciascuno abbia un’estensione consentita (come `.jpg`, `.png` o `.jpeg`); inoltre, i nomi dei file vengono sanificati utilizzando la funzione `secure_filename`. Queste verifiche sono fondamentali per prevenire l’upload di file non supportati o potenzialmente dannosi e per evitare possibili vulnerabilità legate a percorsi o nomi malevoli.

Dopo aver salvato le immagini nella cartella `upload`, il backend avvia l’esecuzione dello script di elaborazione in un thread separato, operazione gestita attraverso la libreria *threading* di Python, che consente al server di continuare a rispondere ad altre richieste mentre l’elaborazione è in corso, migliorando la reattività dell’applicazione. Durante l’elaborazione delle immagini, il backend aggiorna periodicamente un file chiamato `progress.json` che contiene informazioni sullo stato di avanzamento, come la percentuale di completamento calcolata sulla base del numero di immagini elaborate. Il frontend, per mantenere l’utente informato sui progressi, interroga periodicamente l’endpoint `/progress` tramite richieste GET, il quale legge i dati dal file `progress.json` e restituisce al frontend le informazioni necessarie per aggiornare una barra di avanzamento visibile nell’interfaccia utente.

Al termine dell’elaborazione, i risultati generati sono disponibili nella cartella `results` e il frontend può ottenerne la lista inviando una richiesta GET all’endpoint `/get_results`. Questo endpoint scansiona la cartella dei risultati e restituisce un elenco dei file disponibili come le immagini annotate e i grafici prodotti dallo script di elaborazione. Queste informazioni permettono al frontend di presentare all’utente i risultati ottenuti, consentendo la visualizzazione diretta delle immagini elaborate. Per gestire la visualizzazione individuale dei file dei risultati, il backend include l’endpoint `/results/<path:filepath>`, accessibile tramite una richiesta GET, che permette al frontend di richiedere singoli file dalla cartella `results`, specificando il percorso relativo del file desiderato. In questo modo, il frontend può caricare dinamicamente le immagini e i grafici necessari per l’interfaccia utente, senza esporre direttamente l’intera struttura del filesystem del server. Per fornire all’utente un riepilogo dettagliato dell’analisi, il frontend può richiedere i dati riassuntivi tramite l’endpoint `/get_summary_results`. Questo endpoint, anch’esso accessibile tramite una richiesta GET, legge il file `summary_results.json` generato durante l’elaborazione e restituisce al frontend informazioni come il numero totale di olive rilevate, il peso stimato, la distribuzione delle olive per grado di maturità e altre metriche significative. Questi dati sono fondamentali per fornire all’utente una visione d’insieme dei risultati ottenuti dall’analisi.

Nel caso in cui l’utente desideri scaricare tutti i risultati per un’analisi più approfondita o per conservarli, il backend offre l’endpoint `/download_results`. Il

backend, infatti, inviando una richiesta GET a questo endpoint, comprime tutti i file presenti nella cartella `results` in un unico file ZIP e lo restituisce al frontend cosicché l’utente possa scaricare l’archivio e accedere a tutti i risultati su macchina locale. Questo meccanismo è implementato utilizzando la libreria `zipfile` di Python, che permette di creare archivi ZIP in memoria senza la necessità di scrivere file temporanei su disco.

Durante l’implementazione del backend, è stata posta particolare attenzione alla sicurezza e alla robustezza dell’applicazione. Oltre alla sanificazione dei nomi dei file e al controllo delle estensioni, sono stati gestiti adeguatamente gli errori e le eccezioni, fornendo risposte informative al frontend senza esporre dettagli sensibili del sistema. L’uso dell’estensione `Flask-CORS` ha permesso di gestire correttamente le richieste cross-origin, necessarie quando frontend e backend risiedono su domini o porte diverse. L’integrazione tra il backend e lo script di elaborazione è stata curata per garantire una comunicazione efficiente e una sincronizzazione ottimale. Lo script di elaborazione viene avviato con i percorsi corretti delle cartelle `upload` e `results` e utilizza funzioni condivise per aggiornare il progresso e salvare i risultati. Con la realizzazione del server backend, abbiamo completato con successo l’integrazione tra le varie parti dell’applicazione. Il sistema è ora in grado di gestire l’intero flusso operativo, dall’upload delle immagini da parte dell’utente all’elaborazione avanzata tramite il modello YOLOv8 fino alla presentazione dei risultati in un formato chiaro e accessibile.

6.5 Realizzazione dell’interfaccia utente Frontend

Dopo aver completato lo sviluppo del backend e aver garantito un corretto funzionamento della logica di interazione è stata avviata la fase finale del progetto: la creazione dell’interfaccia utente (frontend), fondamentale per garantire un’interazione efficace e intuitiva con l’applicazione. L’obiettivo principale di questa fase è stato quello di progettare un’interfaccia semplice e funzionale che permettesse agli utenti di caricare le immagini, monitorare lo stato di elaborazione e visualizzare i risultati. L’interfaccia si articola in quattro schermate principali, ciascuna corrispondente a una fase specifica del processo di rilevazione e analisi delle immagini contenenti olive:

6.5.1 Schermata iniziale

La schermata iniziale è il punto di ingresso dell’utente nell’applicazione. Si presenta con un design essenziale, caratterizzato da un titolo centrale "OLIVE DETECTION" e da un unico pulsante verde etichettato **Carica Immagini**. Questo pulsante, una volta cliccato dall’utente, apre una finestra del file manager che consente all’utente di selezionare dal proprio dispositivo le immagini sulle quali vuole

effettuare l’analisi. Una volta confermata la selezione delle immagini sul file manager queste ultime verranno immesse all’interno di un vettore che ne salva i nomi e si passa direttamente alla schermata successiva. Durante questa fase iniziale non vengono effettuate richieste HTTP poiché l’utente, pur avendo scelto le immagini da processare, potrebbe volerle modificare in corso d’opera per aggiungerne o rimuoverne altre; pertanto, la schermata iniziale funge da semplice punto di partenza e di introduzione alla web app da cui l’utente può immediatamente passare alla fase successiva contenente la visualizzazione delle anteprime delle immagini caricate.



Figura 6.4: Visualizzazione della schermata iniziale

6.5.2 Schermata di anteprima delle immagini

Una volta caricate, le anteprime delle immagini selezionate dall’utente vengono visualizzate all’interno di una griglia organizzata. La griglia può raggiungere una dimensione massima di 3 righe x 3 colonne (per un totale di 9 immagini contemporaneamente) per motivi di bellezza estetica nella visualizzazione; tuttavia, la dimensione della griglia varia a seconda del numero totale delle immagini selezionate dall’utente:

- se il numero di immagini è minore o uguale a 3 la griglia si estenderà su una sola riga occupando le colonne necessarie da sinistra a destra
- se il numero di immagini risulta, invece, compreso tra 4 e 6 la griglia si estenderà su 2 righe con la stessa logica di riempimento
- per un numero di immagini maggiore di 6 la griglia occuperà tutte le 3 righe a sua disposizione



Figura 6.5: Visualizzazione delle anteprime delle immagini

Chiaramente per gestire un numero di immagini selezionate dall'utente maggiore di 9 sono state implementate delle icone a forma di freccia direzionale verso sinistra e verso destra che permettono la navigazione fra le immagini salvate nel vettore attraverso un aggiornamento in tempo reale delle anteprime contenute all'interno della griglia. Per garantire che l'utente possa rimuovere eventuali immagini indesiderate tra quelle selezionate sono state inserite delle pratiche icone a forma di X che, una volta cliccate, permettono la rimozione dell'immagine dal vettore di salvataggio e della griglia delle anteprime. Se, al contrario, l'utente decidere di aggiungere delle nuove immagini da analizzare può farlo attraverso il

bottone in basso a destra denominato **Carica altre immagini** che ha esattamente la stessa funzione del pulsante **Carica immagini** nella schermata iniziale. Quando l’utente ha terminato di apportare eventuali modifiche può procedere con l’elaborazione cliccando sul pulsante **Inizia Analisi**. Quest’ultimo, una volta cliccato, genera l’invio di una richiesta http POST all’endpoint `/upload`, che permette il caricamento delle immagini selezionate su una specifica cartella `upload` presente sul server backend all’endpoint, e di una successiva richiesta POST all’endpoint `/start_processing` attraverso la quale si avvia lo script di elaborazione delle immagini caricate sul server. L’esecuzione dello script può infatti avere luogo, il server comincia a processare le immagini inviate presenti in `/upload` e a rilevare e classificare le olive presenti e, una volta iniziata l’elaborazione, si passa direttamente alla schermata successiva nella quale si può avere un feedback visivo sull’avanzamento del processo in atto.

6.5.3 Schermata di elaborazione

Dopo aver avviato l’analisi l’utente accede alla schermata di elaborazione, dove può monitorare lo stato di avanzamento del processo grazie a una barra di caricamento. La comunicazione tra il frontend e il backend in questa fase avviene attraverso richieste GET periodiche all’endpoint `/progress` il quale restituisce informazioni sullo stato attuale dell’analisi. Questo meccanismo di polling consente di aggiornare in tempo reale lo stato di avanzamento della barra, mostrando all’utente che le operazioni di elaborazione stanno avvenendo in maniera sequenziale su tutte le immagini contenute all’interno della cartella `upload`.



Figura 6.6: Visualizzazione della barra dell’analisi in corso

Quando il processo di elaborazione è completato, la barra raggiunge il 100% e un messaggio di conferma appare sullo schermo informando l’utente che può procedere

alla visualizzazione dei risultati cliccando sul pulsante **Visualizza Risultati**. Questa fase è cruciale per mantenere l'utente informato sullo stato delle operazioni, fornendo un feedback continuo e riducendo l'incertezza sui tempi di attesa.



Figura 6.7: Visualizzazione dell'analisi delle immagini completata

6.5.4 Schermata di visualizzazione dei risultati

La schermata dei risultati rappresenta il punto finale dell'interazione dell'utente con l'applicazione. Qui l'utente può visualizzare sia i grafici statistici relativi all'analisi complessiva di tutte le immagini immesse sia le immagini annotate. La schermata è divisa in due sezioni principali: la colonna sinistra è dedicata ai grafici delle distribuzioni delle olive per peso e per grado di maturità (come mostrato nella Figura 6.3) mentre la colonna destra mostra le immagini con i subplot generati ai vari stadi della pipeline e le annotazioni di numero di olive rilevate nell'immagine, peso totale delle olive rilevate e grado di maturità (come mostrato nella Figura 6.2). I grafici e le immagini annotate vengono generati dallo script di elaborazione e, tramite gli endpoint `/get_summary_results` e `/get_results` vengono restituiti al frontend insieme ai risultati sul numero e il peso totale delle olive salvati nel file `summary_results.json`. In maniera analoga a quanto avveniva nella seconda schermata, l'utente ha la possibilità di navigare tra i risultati presenti nella colonna a destra grazie a un sistema di paginazione che permette di navigare verso destra o sinistra tramite opportune frecce direzionali. Nella parte inferiore della schermata è possibile inoltre visualizzare i risultati complessivi contenuti all'interno del file `.json` che indicano il numero e il peso totale (medio e approssimativo) delle olive rilevate nonché il numero di olive considerate `adjusted` poiché difficile stimarne il peso. Se l'utente desidera scaricare una copia locale dei risultati ottenuti, può cliccare

sul pulsante **Scarica Risultati** il quale attiva una richiesta HTTP GET all’endpoint `/download_results` che genera file ZIP comprimendo tutti i file prodotti nei risultati e lo restituisce all’utente per il download.

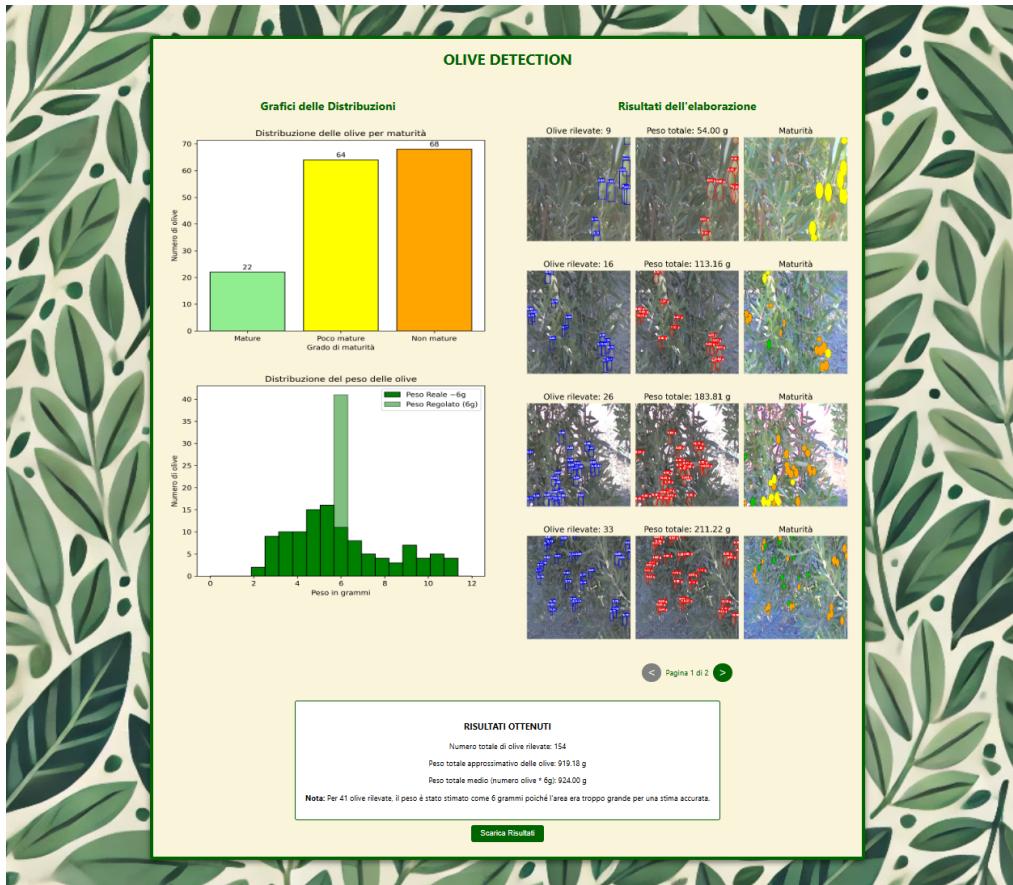


Figura 6.8: Visualizzazione dei risultati generati dalla pipeline di elaborazione

La progettazione dell’interfaccia utente ha seguito un principio di semplicità e usabilità, garantendo che ogni fase del processo fosse chiara e accessibile. La comunicazione continua con il backend tramite chiamate API ha permesso di mantenere l’interfaccia aggiornata in tempo reale, fornendo un’esperienza interattiva fluida e senza interruzioni. Grazie a questo approccio, l’utente può caricare immagini, monitorare l’elaborazione e visualizzare i risultati con un’interazione minima, senza mai sentirsi disconnesso dal processo. Il sistema, nel suo insieme, risulta così intuitivo, veloce e affidabile, con un’interazione tra frontend e backend che garantisce un’esperienza ottimale.

Conclusioni e scopi futuri

L’obiettivo di questa tesi è stato quello di esplorare e applicare tecnologie avanzate di deep learning e computer vision nel settore oleario, un ambito profondamente radicato nella tradizione, ma che oggi si confronta con le sfide dell’innovazione. Attraverso l’utilizzo della rete neurale convoluzionale YOLOv8 e di tecniche di visione artificiale è stato creato un sistema automatizzato per la rilevazione e classificazione delle olive, capace di sostituire efficacemente metodi tradizionali manuali spesso lenti e soggetti a errori. Il sistema implementato ha dimostrato come l’intelligenza artificiale possa trasformare un settore come quello oleario, migliorando significativamente la precisione e l’efficienza operativa, specialmente nella fase di raccolta e analisi della qualità delle olive. La scelta di utilizzare YOLOv8 si è rivelata vincente, grazie alla sua capacità di rilevare le olive con un’accuratezza molto elevata. La versione MEDIUM del modello ha raggiunto un’accuratezza eccellente confermando come, attraverso tecniche di fine-tuning, sia possibile ottenere prestazioni di alto livello in scenari complessi e variabili. Il sistema ha integrato una web app con un’interfaccia semplice e intuitiva, permettendo anche agli utenti meno esperti di caricare immagini, avviare l’analisi e visualizzare facilmente i risultati. Questo ha reso l’applicazione versatile e accessibile, aprendo la strada a un utilizzo diffuso tra produttori di diverse dimensioni.

Nonostante i risultati promettenti, il progetto offre ancora ampi margini di miglioramento e spunti per sviluppi futuri. Uno degli obiettivi principali sarà l’espansione del dataset per includere immagini di uliveti provenienti da diverse aree geografiche e riprese in differenti condizioni ambientali. Questo permetterebbe di rendere il modello più adattabile a situazioni reali, dove le variabili come la luce, la prospettiva o lo stato di maturazione delle olive possono cambiare drasticamente. Parallelamente, sarà cruciale continuare a ottimizzare il modello di rete neurale per migliorare ulteriormente l’accuratezza, riducendo al contempo i tempi di elaborazione. Un’altra direzione interessante per i futuri sviluppi riguarda il potenziale di rilevamento delle malattie e dei parassiti attraverso la visione artificiale. Con l’aggiunta di nuove funzionalità, il sistema potrebbe non solo identificare le olive, ma anche riconoscere segni di infezioni o la presenza di parassiti sulle foglie e sui frutti. In particolare, l’applicazione di YOLOv8 e di altri modelli di machine learning

potrebbe permettere di individuare precocemente problematiche come la mosca olearia, contribuendo a un monitoraggio più accurato e tempestivo delle colture, riducendo la necessità di interventi chimici. Un passo importante verso un’ulteriore automazione del settore oleario potrebbe essere rappresentato dall’integrazione di droni equipaggiati con telecamere ad alta risoluzione e sistemi di intelligenza artificiale. I droni, grazie alla loro capacità di coprire vaste aree in tempi ridotti e di fornire riprese a 360° degli alberi, offrirebbero la possibilità di effettuare una mappatura completa degli uliveti. Con un sistema in grado di elaborare in tempo reale le immagini catturate dai droni si potrebbe ottenere una visione dettagliata della distribuzione delle olive su ogni albero e della loro maturazione, consentendo una pianificazione più efficiente della raccolta. L’integrazione di droni UAV (*Unmanned Aerial Vehicles*) potrebbe rivoluzionare la gestione delle colture, permettendo di monitorare ampi terreni e individuare problemi locali o aree in cui la produzione è più concentrata, riducendo i costi operativi e migliorando la qualità del prodotto finale. Oltre alla rilevazione delle olive, i droni potrebbero anche raccogliere dati multispettrali o termografici, utili per identificare eventuali stress idrici o altre problematiche legate alla salute delle piante. Questi dati, combinati con l’elaborazione AI in tempo reale, potrebbero essere integrati in mappe interattive che forniscono ai produttori una visione complessiva dello stato di ogni singolo albero. Una tale integrazione potrebbe portare a una gestione completamente automatizzata e ottimizzata degli uliveti, con enormi benefici in termini di efficienza e sostenibilità. Infine, un ulteriore passo avanti potrebbe essere l’implementazione di dashboard interattive e dinamiche, che consentano di visualizzare i dati raccolti in tempo reale in modo più approfondito e personalizzato. Tali dashboard potrebbero includere analisi comparative, previsioni di raccolta e suggerimenti per l’ottimizzazione delle risorse, offrendo così agli agricoltori uno strumento potente per la gestione delle loro attività. Questo tipo di visualizzazione faciliterebbe anche l’adozione della tecnologia da parte di produttori meno abituati all’uso di strumenti digitali, aumentando così la portata e l’impatto del sistema sviluppato.

In conclusione, l’applicazione di tecniche innovative basate su intelligenza artificiale e computer vision nel settore oleario ha già dimostrato di poter rivoluzionare processi chiave, come la raccolta e l’analisi delle olive, migliorando la precisione e riducendo i tempi e i costi operativi. L’integrazione di tecnologie emergenti, come i droni e i sistemi di monitoraggio in tempo reale, rappresenta una sfida interessante per il futuro e una straordinaria opportunità di trasformazione per un settore che, seppur ancorato alla tradizione, è pronto a cogliere i benefici della modernizzazione.

Bibliografia

- [1] Orsomarso Blues. La raccolta delle olive nell'antica grecia, 2023. Accessed: 2024-10-08.
- [2] International Olive Council. Olive oil production and market growth. *Olive Market Report*, 4:13–17, 2020. Accessed: 2024-10-08.
- [3] Villa Dama. L'olio nell'antichità: La produzione dell'olio nell'antica grecia, 2021. Accessed: 2024-10-08.
- [4] Frantoi d'Italia. Tipi di olio d'oliva: l'elenco completo, 2024. Accessed: 2024-10-08.
- [5] Brendan Eich and the JavaScript Community. *JavaScript Documentation*. Mozilla Developer Network (MDN), 2024.
- [6] Mohamed Lachgar El Mehdi Raouhi, Hamid Hrimech, and Ali Kartit. Optimizing olive disease classification through transfer learning with unmanned aerial vehicle imagery. *International Journal of Electrical and Computer Engineering (IJECE)*, 14(1):891–903, 2024.
- [7] George Georgiou, Petros Karvelis, and Christos Gogos. Computer Vision-based Detection and Tracking in the Olive Sorting Pipeline. In Dionysis D. Bochtis, Dimitrios E. Moshou, Giorgos Vasileiadis, Athanasios Balafoutis, and Panos M. Pardalos, editors, *Information and Communication Technologies for Agriculture—Theme II: Data*, volume 183, pages 131–160. Springer International Publishing, Cham, 2022. Series Title: Springer Optimization and Its Applications.
- [8] Youness Hnida, Mohamed Adnane Mahraz, Jamal Riffi, Hamid Tairi, and Ali Achebour. Deep Learning-Based Estimation of Olive Flower Density from UAV Imagery. In *2024 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–6. IEEE, 2024.
- [9] Nashat M. Hussain Hassan and Ahmed A. Nashat. New effective techniques for automatic detection and classification of external olive fruits defects based on image processing techniques. *Multidimensional Systems and Signal Processing*, 30(2):571–589, April 2019.
- [10] IBM. What is computer vision?, 2024. Accessed: 2024-10-08.
- [11] IBM. What is machine learning?, 2024. Accessed: 2024-10-08.

- [12] Nariman Mamdouh and Ahmed Khattab. YOLO-based deep learning framework for olive fruit fly detection and counting. *IEEE Access*, 9:84252–84262, 2021. Publisher: IEEE.
- [13] João Mendes, Guido S. Berger, José Lima, Lino Costa, and Ana I. Pereira. Pest Management in Olive Cultivation Through Computer Vision: A Comparative Study of Detection Methods for Yellow Sticky Traps. In Lino Marques, Cristina Santos, José Luís Lima, Danilo Tardioli, and Manuel Ferre, editors, *Robot 2023: Sixth Iberian Robotics Conference*, volume 978, pages 373–385. Springer Nature Switzerland, Cham, 2024. Series Title: Lecture Notes in Networks and Systems.
- [14] Javiera Navarro Soto, Silvia Satorres Martínez, Diego Martínez Gila, Juan Gómez Ortega, and Javier Gámez García. Fast and reliable determination of virgin olive oil quality by fruit inspection using computer vision. *Sensors*, 18(11):3826, 2018. Publisher: MDPI.
- [15] NVIDIA. What's the difference between a cpu and a gpu?, 2024. Accessed: 2024-10-08.
- [16] Luciano Ortenzi, Simone Figorilli, Corrado Costa, Federico Pallottino, Simona Violino, Mauro Pagano, Giancarlo Imperi, Rossella Manganiello, Barbara Lanza, and Francesca Antonucci. A machine vision rapid method to determine the ripeness degree of olive lots. *Sensors*, 21(9):2940, 2021. Publisher: MDPI.
- [17] Adam Paszke, Sam Gross, and the PyTorch Team. *PyTorch Documentation*. Meta AI Research, 2024.
- [18] Juan M. Ponce, Arturo Aquino, and Jose M. Andujar. Olive-fruit variety classification by means of image processing and convolutional neural networks. *IEEE Access*, 7:147629–147641, 2019. Publisher: IEEE.
- [19] Juan Manuel Ponce, Arturo Aquino, Borja Millan, and Jose M. Andujar. Automatic counting and individual size and mass estimation of olive-fruits through computer vision techniques. *IEEE Access*, 7:59451–59465, 2019. Publisher: IEEE.
- [20] Catherine E. Pratt. *Oil, Wine, and the Cultural Economy of Ancient Greece: From the Bronze Age to the Archaic Era*. Cambridge University Press, 2021.
- [21] Radiolocman.com. Development of olive picking robot for small farms, 2024. Accessed: 2024-10-08.
- [22] Redakcja. 55 najbardziej interesujących faktów o oliwkach, mar 2017.
- [23] Armin Ronacher and the Flask Team. *Flask Documentation*. Pallets Projects, 2024.
- [24] Alireza Sanaeifar and Abdolabbas Jafari. Determination of the oxidative stability of olive oil using an integrated system based on dielectric spectroscopy and computer vision. *Information processing in agriculture*, 6(1):20–25, 2019. Publisher: Elsevier.
- [25] Themistoklis Sarantakos, Daniel Mauricio Jimenez Gutierrez, and Dimitrios Amaxilatis. Olive Leaf Infection Detection Using the Cloud-Edge Continuum.

- In Ioannis Chatzigiannakis and Ioannis Karydis, editors, *Algorithmic Aspects of Cloud Computing*, volume 14053, pages 25–37. Springer Nature Switzerland, Cham, 2024. Series Title: Lecture Notes in Computer Science.
- [26] Fernandez Sonia. La alcacarra: un ingrediente mediterráneo aromático y de intenso gusto, November 2020.
 - [27] ISDA Staff. Crippling frost, climate change strip \$1 billion from olive oil harvest, March 2019.
 - [28] UNESCO. The mediterranean diet - inscribed in 2010 on the representative list of the intangible cultural heritage of humanity, 2024. Accessed: 2024-10-08.
 - [29] Guido van Rossum and the Python Software Foundation. *Python Documentation*. Python Software Foundation, 2024.
 - [30] Belén Vega-Márquez, Isabel Nepomuceno-Chamorro, Natividad Jurado-Campos, and Cristina Rubio-Escudero. Deep learning techniques to improve the performance of olive oil classification. *Frontiers in chemistry*, 7:929, 2020. Publisher: Frontiers Media SA.
 - [31] Jordan Walke and the React Team. *React Documentation*. Meta Platforms, Inc., 2024.