Politecnico di Torino

POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Gestionale – LM/31
Percorso ICT and data analytics
Tesi di Laurea Magistrale

Machine Vision per il miglioramento della qualità nei processi agroalimentari

Analisi dell'applicazione di sistemi di Machine Vision nei processi produttivi agroalimentari, e realizzazione di un prototipo sperimentale con eventuali ottimizzazioni dei modelli

Relatori:
Prof. Maurizio Galetto
Prof.ssa Elisa Verna

Candidato:
Lucia Ruffati

Anno Accademico 2024-2025
Sessione di Laurea – Novembre 2025

Ai miei genitori

Sommario

Introduzione

Contesto e motivazioni

Il settore alimentare è un mercato in espansione e in costante crescita, la domanda continua porta il sistema produttivo ad adattarsi alle esigenze globali, che richiedono standard di qualità elevati e che rispettino normative europee e mondiali sempre più esigenti.

Queste esigenze derivano sia dalle aspettative dei consumatori, sia dai requisiti normativi, che comportano controlli stringenti e sempre più normalizzati; a tali esigenze si aggiunge la necessità di ridurre il più possibile gli sprechi alimentari e gli sprechi produttivi, con una conseguente diminuzione dei costi derivanti dagli scarti, migliorandone l'efficienza produttiva.

Il Machine Vision (MV) rappresenta uno una delle soluzioni tecnologie più promettenti per affrontare le sfide odierne sul controllo qualità dei prodotti lavorati: i sistemi di visione artificiali comportano alcuni vantaggi come l'utilizzo di tecniche non invasive, analisi rapide, ripetibili e che possono accompagnare o sostituire il controllo manuale di produzione. Rispetto all'intervento umano di visione questi strumenti riescono a gestire meglio la velocità del nastro trasportatore, riuscendone a ridurre la variabilità e il rischio di errore.

Nonostante questi vantaggi e l'applicazione crescente dei sistemi di visione artificiale applicati all'industria, la letteratura scientifica appare carente di articoli riguardanti la specificità del settore, preferendone la materia prima rispetto alle linee di prodotti lavorati, mostrando questa differenza nella numerosità degli articoli emersi.

I prodotti lavorati risultano avere maggiore eterogeneità in termini di forma, colore e consistenza, ciò rende la creazione di un algoritmo robusto maggiormente complicato, e la relativa ricerca scientifica carente. È importante considerare inoltre la latenza delle tecnologie e quella degli studi analizzati, i sistemi automatici risultano essere relativamente nuovi e in via di sviluppo, che li rende maggiormente soggetti ai gap letterari.

Paper recenti dimostrano tuttavia il corretto funzionamento di alcune tecnologie di Machine Vision: Abdanan Mehduzadeh [1] ha sviluppato e pubblicato all'interno del suo studio un forno intelligente per la cottura dei cupcake, dimostrando come l'utilizzo di alcuni algoritmi di visione artificiale possano valutare in tempo reale la texture e la variazione di colore durante la cottura. A sostegno di questa tesi vi sono le conclusioni da loro raccolte, che dimostrano con un 98% di accuratezza come il MV possa ottimizzare sia la qualità del prodotto che il dispendio di energia.

L'applicazione del MV all'interno del controllo qualità risulta aver raccolto alcune applicazioni [2]: hanno dimostrato come tecniche di visione multispettrali migliorino il processo di trattamento igienico, conservazione e trasformazione del latte (codice ATECO 10.51.20), superando i limiti di un'ispezione manuale.

Un ostacolo nell'installazione di alcuni algoritmi di machine vision e del proprio utilizza risiede nella configurazione manuale dei sistemi: essi richiedono fasi di tuning basate sulla capacità umana e dall'esperienza, in modo tale da poterne trarre un ulteriore beneficio, questa soluzione risulta essere dispendiosa e poco scalabile in termini industriali.

Per ovviare questo problema Chetima e Payeur [3] hanno introdotto all'interno della loro ricerca l'utilizzo di un sistema integrato di taratura automatica dei sistemi di ispezione, applicato a prodotti identificati dal codice ATECO 10.71.10 (produzione di pane) quali tortillas e panini.

Gli autori grazie all'uso combinato di feature selection e modelli di apprendimento quali decision trees e Bayesian learning, sono riusciti a conseguire percentuali di accuratezza che sfiorano il 98%, riducendo drasticamente tempi di setup e la necessità di un intervento umano [3].

Attraverso l'analisi di questi studi emergono alcuni punti a favore della presente tesi: la necessità di indagare a livello bibliografico i lavori relativi lo stato dell'arte sull'applicazione del machine vision su prodotti lavorati e il bisogno di sviluppare sistemi più adattivi e meno dipendenti dall'errore umano.

A seguito di queste considerazioni lo sviluppo di algoritmi legati al machine vision risulta essere uno dei promettenti campi tecnologici ancora in via di sviluppo.

Obiettivo della tesi

Il presente lavoro di tesi si pone l'obiettivo di analizzare il panorama scientifico e tecnologico riguardante l'utilizzo dei sistemi di Machine Vision nel settore agroalimentare secondario, i prodotti trasformati, con l'obiettivo di valutarne l'applicabilità, i punti di forza e di debolezza, e in ultima istanza crearne un caso di studio simulato.

Le motivazioni di questa scelta vengono giustificate dalla sottorappresentazione letteraria del settore, tuttavia rilevante e di interesse dal punto di vista industriale, e dunque dalla volontà di donare maggiore consapevolezza scientifica e applicativa alla ricerca.

I sotto-obiettivi si propongono di analizzare lo stato dell'arte per crearne una visione generale e puntuale, analizzare le tecnologie utilizzate e trarne un confronto dei modelli trovati in situazioni reali.

Un secondo obiettivo rientra nella ricerca applicativa, inserendo un caso reale di applicazione algoritmica e valutarne le implicazioni tecniche e gestionali.

Le domande di ricerca che guidano il lavoro sono le seguenti:

- Qual è lo stato dell'arte di riferimento e in quale ambito specifico si colloca il presente lavoro di tesi?
- Qual è la relazione tra modelli sviluppati, le modalità di acquisizione delle immagini e i difetti rilevati sul prodotto analizzato?
- Come si comportano i diversi modelli a livello di accuratezza e metriche tecniche?
- Quali sono le implicazioni economiche, organizzative e ambientali derivanti dalle loro prestazioni?

In sintesi, il presente lavoro di tesi si pone l'obiettivo di analizzare e contribuire alla ricerca, per raggiungere tali scope il lavoro è stato organizzato in capitoli, ciascuno dedicato a una specifica fase della ricerca.

Struttura del lavoro

Il primo capitolo di tesi tratta la metodologia utilizzata durante il lavoro: la metodologia scientifica PRISMA applicata allo studio bibliografico, in cui sono presenti i criteri di applicazione, la strategia di ricerca e un paragrafo dedicato all'integrazione di un'analisi della letteratura grigia e dei brevetti e la successiva metodologia di ricerca che, d'altro lato, analizza i modelli identificati per l'applicazione, la scelta e spiegazione dei parametri, con annessa raccolta delle metriche tecniche, gestionali e dell'analisi della robustezza scelte.

Il capitolo Literature Review contiene una panoramica sugli output dello stato dell'arte, ponendo i risultati trovati in forme tabellari e commenti e osservazioni relative ad esse; mentre la sezione riguardante l'analisi degli output di ricerca tratta l'analisi dei risultati del caso sperimentale.

Le conclusioni, infine, riassumono il lavoro di tesi, donandone un punto di vista innovativo e spunti per applicazioni e miglioramenti futuri.

Literature Review

Al fine di indagare maturità tecnologica e potenziale manageriale delle soluzioni di Machine Vision Systems evidenziate per l'industria agroalimentare secondaria, si è compiuta un'analisi metodologica sia per la revisione bibliografica, che sull'applicazione reale di un caso di controllo qualità.
Per raggiungere tali obiettivi di ricerca è stata progettata una metodologia articolata in diverse fasi, come mostrato dalla pipeline in figura 1, la prima è composta da uno stream di Literature Review comprendente la ricerca di database scientifici, la selezione della query, due analisi dei testi emersi dalla ricerca e la suddivisione in cluster tematici.

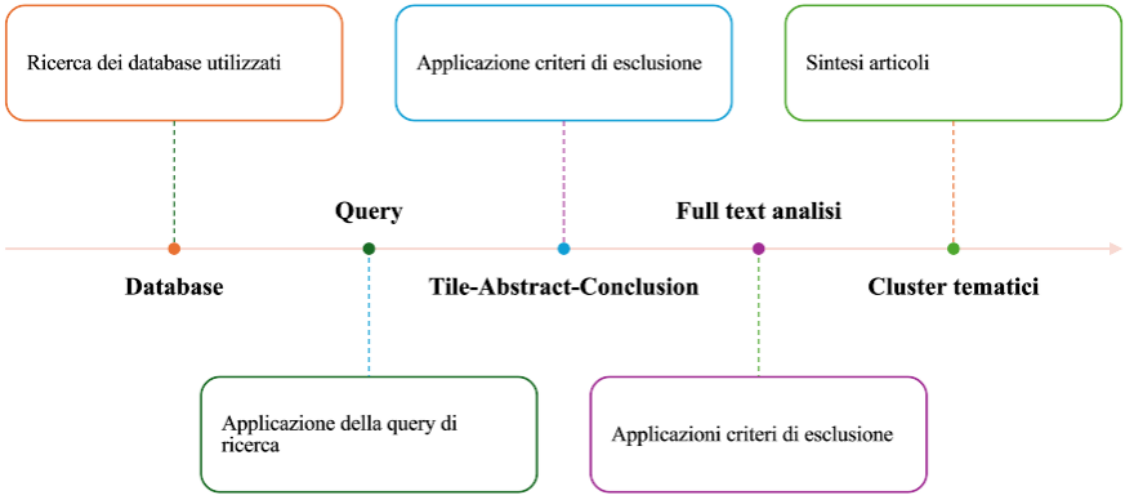


Figure 1- Flowchart rappresentante i punti chiave della metodologia utilizzata

La seconda fase metodologica segue il processo utilizzato per la fase applicativa di ricerca, spiegando l'approccio utilizzato per la selezione dei modelli, la composizione del dataset e la valutazione tecnica e gestionale dei risultati ottenuti.

Metodologia della revisione bibliografica

Per la presente tesi è stato svolto un lavoro di ricerca bibliografica attraverso una revisione sistematica della letteratura, volto ad analizzare in maniera ordinata e con rigore scientifico i paper trattanti Machine Vision e la sua applicazione nel settore alimentare durante la fase di controllo della qualità di prodotti lavorati.

La metodologia scelta (PRISMA) [4] permette di identificare gli articoli maggiormente rilevanti al fine della ricerca, minimizzandone in maniera significativa i bias cognitivi associati all'errore umano. Questa metodologia permette l'utilizzo di una logica sequenziale di ricerca, iniziando con l'applicazione di una query di ricerca all'interno di diversi database e identificandone attraverso parametri di esclusione e inclusione gli articoli scelti per una review più approfondita.

La scelta di compiere una ricerca sistematica permette di condurre con rigore, trasparenza e replicabilità il processo di selezione delle fonti; questa scelta si oppone alle revisioni narrative, che si basano sulla soggettività dell'autore, basandosi principalmente sulla loro opinione, ed eventuali esperienze personali [5].

L'adozione di questo protocollo standardizzato ha ulteriormente rafforzato la qualità metodologica del lavoro di tesi, permettendo la creazione di uno schema chiaro per la selezione e il reporting degli articoli, favorendone così la trasparenza del processo e assicurando che i risultati ottenuti siano basati su dati consolidati e verificabili.

Protocollo PRISMA

Il protocollo PRISMA consiste nel donare alcune regole e una visione logica del processo che dovrebbero costituire una ricerca scientifica affrontata con metodo e rigore. Originariamente pubblicato nel 2009, e aggiornato nel 2020 [4], fornisce una sequenza metodologica da seguire per allinearsi ai più recenti sviluppi metodologici e concettuali, offrendo uno strumento più flessibile ai ricercatori, ma ugualmente preciso e dettagliato.

Non solo la scelta del protocollo risulta essere fondamentale al fine di una ricerca scientifica adeguata, ma l'applicazione di questo protocollo nello specifico risulta essere appoggiata dallo studio " Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement" [4][6], il quale sostiene attraverso un confronto metodologico con altri strumenti di ricerca scientifica, l'applicazione del protocollo PRISMA come il più completo per l'analisi, confrontandola con altri standard precedenti come QUORUM.

Il processo della metodologia PRISMA segue tre fasi principali [4]:

1. 27 items da seguire per il raggiungimento di una revisione sistematica, i quali compongono una checklist di attività da compiere al fine di una revisione metodologica e scientifica. Questa

lista è stata seguita come linea guida nello svolgimento della tesi, e si sono inclusi i punti maggiormente rilevanti all'interno della scrittura analitica.

2. Una ricerca tramite query compiuta su diversi database elettronici, e la relativa selezione attraverso filtri applicati alla pagina e criteri di esclusione evidenziati dai ricercatori stessi.
3. La revisione in un primo momento di *title-abstract-conclusion* e in un secondo momento della totalità dei paper selezionati.

L'obiettivo principale della metodologia risulta essere la collezione di fonti chiare ed evidenti, al fine di una ricerca meno affetta da bias e offrire, inoltre, la possibilità di replicare il percorso di ricerca.

Strategia di ricerca

La ricerca bibliografica per il seguente lavoro di tesi è stata compiuta su due diversi database scientifici online: Web of Science e Scopus.

La scelta di questi database è motivata dalla ampia copertura interdisciplinare che offre il loro utilizzo combinato: sebbene studi recenti [7] abbiano dimostrato come Scopus tenda ad indicizzare un maggior numero di riviste rispetto a Web of Science, nel presente lavoro sono stati utilizzati entrambi, in modo tale da presentare una rappresentazione totale dei contenuti. Per permettere una ricerca oggettiva la stringa di ricerca utilizzata è stata la medesima per entrambi i database, cambiando esclusivamente la formattazione, punto necessario per evitare errori legati al format di scrittura.

La tecnica utilizzata al fine di comporre una query più completa possibile è stata collegare i tre elementi principali dello studio attraverso un operatore logico AND e, successivamente, aggiungere sinonimi o ulteriori categorie attraverso la funzione OR.

Il nucleo principale della query è composto da tre differenti aree ciascuna rappresentata da una parola chiave:

- Parola che denoti l'algoritmo o il punto focale della ricerca, *machine vision*
- Parola che denoti il contesto di produzione di cui la tesi vuole trattare, *food industry*
- Parola che identifichi il segmento di produzione che si vuole indagare, *quality control*

I tre punti core della ricerca hanno composto una prima bozza di stringa, che è stata arricchita con diversi sinonimi e campi di applicazione, come si può notare dal diagramma di Venn in figura 2 che riproduce in maniera visiva la query finale utilizzata, attraverso l'utilizzo grafico creato tramite Power Point.

Alcune parole specifiche, per le quali non si voleva solo il risultato singolo, ma anche eventuali collegamenti derivanti dalla stessa radice, sono state in parte troncate inserendo l'asterisco al posto della desinenza finale: "image process*", ad esempio, restituirà risultati aventi anche parole come processing o processed.

Punti core della ricerca	Query completa utilizzata
Machine Vision	"machine vision" OR "computer vision" OR cnn OR "convolutional neural network" OR "image process*" OR "deep learning" OR "machine learning"
	AND
Food Industry	"food industry" OR "food process*" OR "agri-food"
	AND
Quality control	"quality control" OR "safety control" OR "food defect detection" OR "food quality assessment" OR "food safety"

Tabella 1- Core della stringa di ricerca e query completa

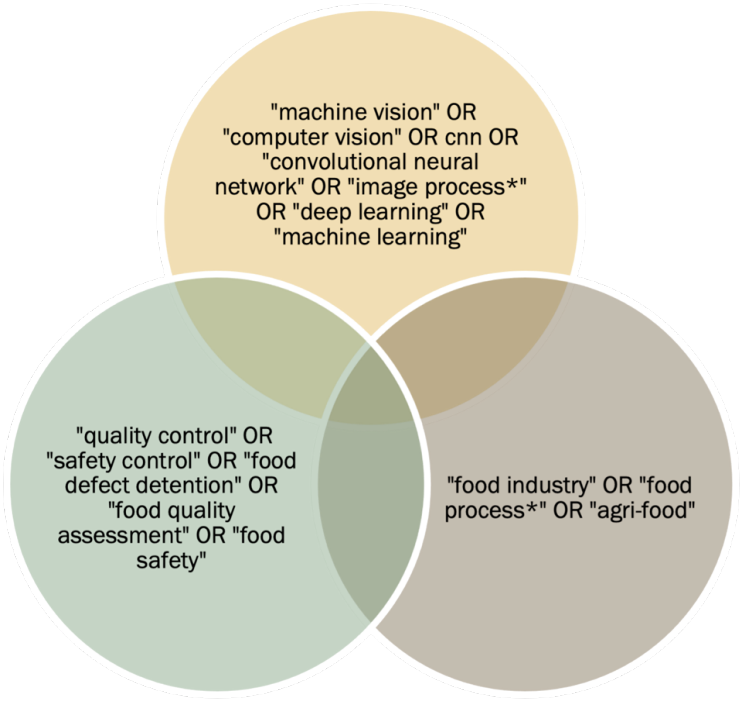


Figure 2 - Diagramma di Venn contenente la stringa di ricerca

La stringa nella sua versione completa (tabella 1) è stata cercata all'interno di entrambi i database attraverso la possibilità Advanced applicata ai campi Title, Abstract e Keywords.

Le ricerche sono state eseguite nell'aprile 2025, e i risultati emersi sono stati scaricati in formato .csv e successivamente elaborati, così facendo l'intera procedura è stata documentata al fine di poterne ricreare la replicabilità.

Il flowchart riportato in figura 3, creato attraverso il sito ufficiale della metodologia [8], indica il numero di articoli che la sola stringa di ricerca ha prodotto (n=581), il conteggio successivo all'eliminazione dei duplicati tra i due database (n=523); in seconda lettura riporta il numero di

articoli che hanno passato la fase di lettura di titolo – abstract – conclusione (n=115), passando così alla full text analisi che ha permesso l'identificazione dei 18 articoli utilizzati in questo lavoro di tesi per i prodotti lavorati.

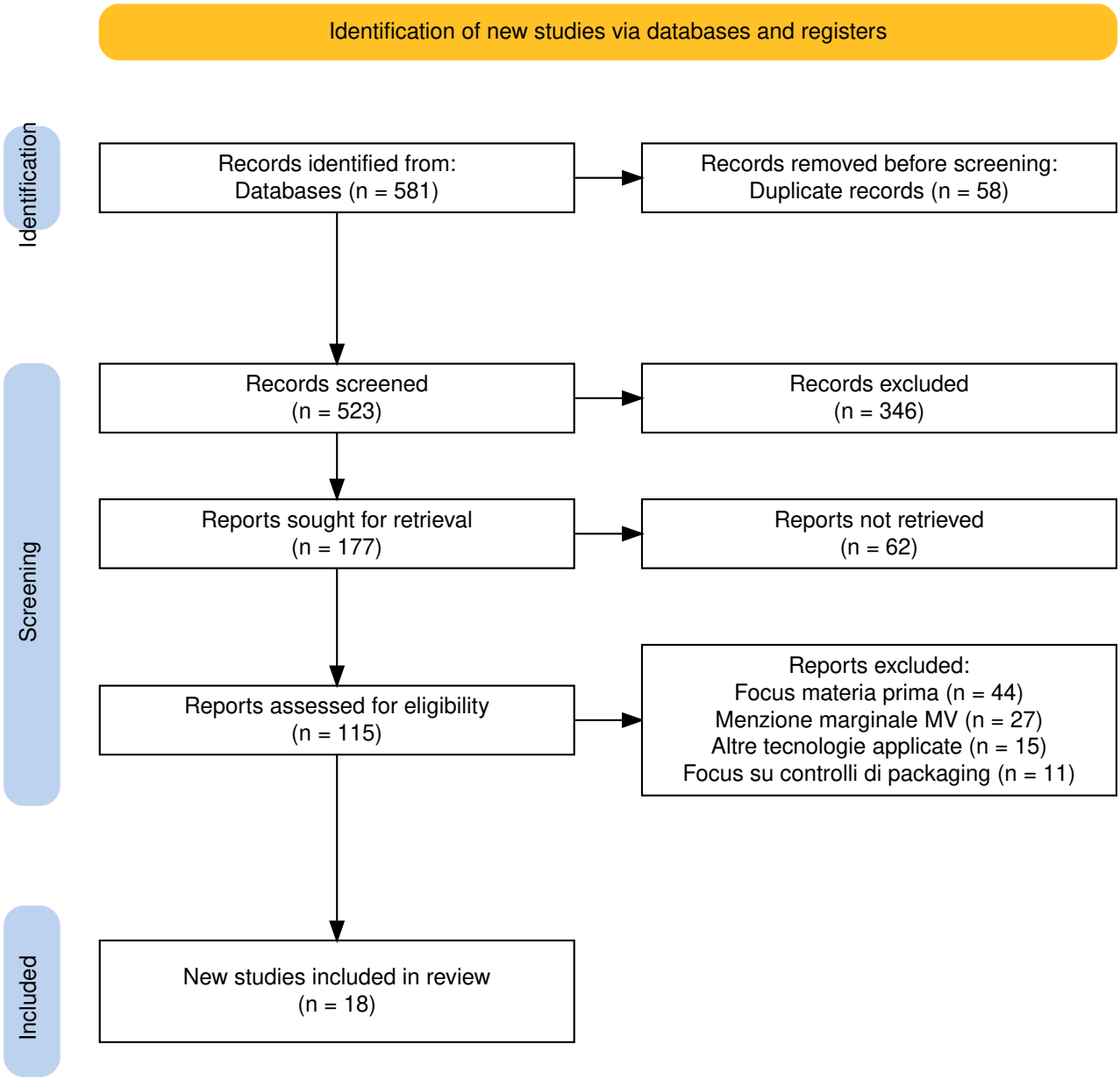


Figure 3- Diagramma PRISMA

Sebbene la ricerca sia stata strutturata in modo tale da non incorrere in eventuali ambiguità, essa presenta comunque alcuni limiti: la selezione di soli due database può risultare costrittiva, e sfociare nell'esclusione intrinseca di articoli rilevanti per la ricerca; inoltre la combinazione logica della stringa e l'utilizzo delle parole chiave da inserirvi sono state scelte in maniera soggettiva, e seppur condivise da più di una persona, essa rappresenta una fonte di variabilità e soggettività.

Nonostante la presenza di alcuni limiti, la ricerca è stata condotta attraverso alcune fasi di test preliminari delle query e una revisione manuale dei risultati, assicurando così un doppio livello di copertura rispetto agli obiettivi proposti.
Al fine di seguire la richiesta metodologica che PRISMA inserisce nella checklist, sono stati identificati diversi criteri di inclusione e conseguente esclusione, e successivamente applicati alle due fasi di analisi di lettura.

Seguendo la logica del workflow in figura 3, i criteri di esclusione nella revisione dello stato dell'arte sono stati decisi e applicati in fase preliminare, in modo da poterli utilizzare in maniera oggettiva e senza conseguente bias sociali e personali.
L'intero processo si è sviluppato in due fasi principali: una prima selezione basata su un'analisi tramite la lettura di titolo – abstract – conclusione e una successiva valutazione integrale degli articoli nella loro totalità.
All'interno della prima fase sono stati applicati i seguenti criteri di esclusione:

- Articoli non pubblicati in lingua inglese.
- Articoli non pubblicati negli ultimi 15 anni (anno pubblicazione >= 2010).
- Articoli aventi informazioni incomplete: mancanza di conclusioni, autori non noti, ricerca non conclusa o non pubblicata su riviste scientifiche.
- Materiali quali book chapter, o book citation, in quanto contenenti informazioni discorsive e meno tecniche.
- Articoli non inerenti all'ambito di ricerca della tesi: algoritmi di AI non contenenti tecniche di machine vision, algoritmi non applicati al settore alimentare e casi di studio con focus non appartenente alla metodologia ricercata.

Gli articoli trovati a seguito di questa analisi di esclusioni sono risultati essere 115, i quali sono stati successivamente letti nella loro totalità, e raggruppati seguendo ulteriori criteri di esclusione:

- Articoli contenenti focus sul controllo qualità della materia prima.
- Articoli non interamente incentrati su algoritmi di machine vision, ma con piccole digressioni a riguardo, non rilevanti per lo scopo del paper.
- Articoli contenenti principalmente altre tecnologie automatizzate e con sensori incorporati quali: e-nose, e-tongue, braccio robotico con sensori di peso, un controllo della qualità maggiormente interno, e sistemi incentrati sul sorting, dunque successivo alla fase di controllo dei difetti.
- Articoli con focus sulla qualità del packaging del prodotto, non direttamente del prodotto stesso: etichette non conformi alle normative, pacchi rigati e/o rotti, altri difetti superficiali.

I criteri di esclusione sono stati raccolti e motivati nelle tabelle 2 e 3, fornendo una visione schematica della logica applicata.

CRITERI DI ESCLUSIONE	MOTIVAZIONE
Articoli non pubblicati in lingua inglese	Standardizzazione linguistica per eliminare problemi di interpretazione e non dover ricorrere a tool di traduzione esterni
Articoli non pubblicati dal 2010 al 2025	Garantire che le fonti siano aggiornate e riflettano tecnologie moderne
Articoli con informazioni incomplete	Garantire una ricerca il più completa possibile, evitando fonti che non consentono una valutazione critica completa
Book chapter o book citations	Garantire articoli con rigore metodologico evidente che rispecchino dunque il lavoro di tesi
Articoli non pertinenti all'ambito di ricerca	Garantire la coerenza tematica con l'obiettivo della tesi

Tabella 2 - Criteri di esclusione title - abstract – conclusione

CRITERI DI ESCLUSIONE	MOTIVAZIONE
Articoli con focus sulla materia prima	La ricerca si concentra su prodotti lavorati e le relative tecnologie utilizzate
Algoritmi con menzione marginale al machine vision o correlati	Gli articoli selezionati devono rappresentare uno studio sostanziale riguardo l'applicazione e l'implementazione delle tecnologie selezionate
Articoli centrati su tecnologie non appartenenti alla categoria di visione artificiale	L'analisi di tesi si propone l'obiettivo di analizzare le tecnologie esclusive di visione artificiale
Articoli con focus su packaging	La ricerca indaga la qualità del prodotto alimentare direttamente, non dei materiali che lo contengono o simili

Tabella 3 - Criteri di esclusione degli articoli finali

Una volta selezionati gli articoli per la lettura totale si è proceduto alla suddivisione in cluster tematici e schematizzazione dei paper, in modo tale da raggruppare le informazioni secondo criteri fissi; gli articoli esclusi trattanti di materia prima sono stati tuttavia scaricati e utilizzati successivamente per un confronto con la letteratura dei prodotti lavorati.

I paper e i loro punti principali sono stati riassunti in tabelle suddivise per tipologia di tecniche di acquisizione di immagini per il componimento dei vari dataset, i diversi modelli applicati e la categoria di prodotto oggetto di studio.

Sebbene la ricerca abbia prodotto dei risultati interessanti, il corpus limitato di articoli trovati ha fatto sì che ci si ponessero alcune domande più specifiche sulla reale attuazione di queste tecnologie e se

potesse esserci una eventuale discrepanza tra letteratura scientifica e letteratura grigia, dando così vita a una parentesi di ricerca.

Analisi integrativa

La letteratura grigia è definita, come riportato da Paez [9], l'insieme di tutti quei documenti – ad esempio record governativi e atti di conferenza - non pubblicati da editori commerciali, ma da enti e organizzazioni secondari.

Al fine di ottenere una panoramica globale del contesto ricercato in questa tesi, si è deciso di introdurre una digressione rispetto alla metodologia bibliografica utilizzata e di integrare uno studio sulle possibili fonti non peer-review ed eventuali casi studio ad essi associati.

Come sostenuto da Paez [9] la letteratura grigia fornisce inoltre la possibilità di ridurre eventuali bias di pubblicazione e può contribuire alla ricerca fornendo una visione più equilibrata delle fonti a disposizione. Tuttavia, la ricerca della letteratura grigia appare spesso complicata e time-consuming, dunque il metodo con il quale la si affronta diventa fondamentale: risulta essenziale dichiarare un chiaro obiettivo e un ordine rigoroso di approccio.

I gap che la letteratura grigia ricercata per questa analisi vuole colmare corrispondono ai limiti della literature review scientifica trovati, come la discrepanza di numerosità di letteratura sui prodotti lavorati rispetto alla materia prima, e la possibilità di identificare meglio casi di studio applicati.

La ricerca parte, dunque, dall'identificazione delle aziende attive a livello globale che si occupano di Machine Vision per l'alimentare, identificandole per fatturato e importanza di clienti. Per compiere questo lavoro di selezione il punto di partenza è stato l'utilizzo di uno strumento di intelligenza artificiale (ChatGPT [10]) in grado di fornire una classifica per fatturato delle principali imprese produttrici di software per l'identificazione automatica di difetti nella qualità dei prodotti. L'utilizzo del tool di intelligenza artificiale è stato utilizzato esclusivamente in fase preliminare, per la raccolta delle fonti necessarie; la seconda fase di revisione dei siti associati ad ogni azienda è stata compiuta manualmente, e si sono selezionati – applicando gli stessi criteri di esclusione della metodologia PRISMA ove possibile - inserendone le informazioni in un output di ricerca evidenziato nella tabella presente all'interno del capitolo sull'analisi dell'output.

Al fine di implementare un'analisi completa, è stata compiuta una ricerca sulle licenze brevettate appartenente alla categoria di Machine Vision Systems.

Il sito utilizzato è lens.org [11], sito che mette a disposizione conoscenze integrate in ambito accademico e brevettuale come bene pubblico, al fine di supportare la risoluzione di problemi basata su scienza e tecnologia.

Così come i database scientifici il sito necessita di una query di ricerca, la stringa utilizzata segue il principio dei tre punti core dello studio e la sua forma completa è la seguente:
“"machine vision" OR "image analysis" OR "computer vision" AND "food quality" OR "food inspection" OR "food defect detection" OR "food surface analysis" OR "quality control" OR "visual inspection" AND "processed food" OR "semi-processed food"”.
Le correzioni della query rispetto alla sua versione precedente sono causate dalla necessità di adattare la ricerca al contesto dei brevetti e ai prodotti lavorati, così da poter confrontare i risultati anche da un punto di vista numerico.
Per restringere l’area di ricerca a situazioni paragonabili alla selezione PRISMA, si sono applicati i filtri di inclusione racchiusi nella tabella 4. All’interno si trovano i filtri riguardanti i codici CPC (Cooperative Patent Classification), i quali sono stati inseriti al fine di restringere la ricerca ai soli campi pertinenti alle tecnologie di visione artificiale, applicate al controllo qualità del settore alimentari dei prodotti lavorati.

Criteri di inclusione	Descrizione	Motivazione
Status: Active	Inclusi nella ricerca esclusivamente brevetti con stato attivo.	Garantisce l’attualità dei risultati negli anni di ricerca analizzati.
Anno compreso tra 2010 e 2025	Criterio di inclusione basato sull’anno di pubblicazione del brevetto.	Permette una comparazione oggettiva riguardo gli anni di ricerca effettuati.
CPC A23L 33/12 – Controllo qualità alimentare	Include brevetti classificati come metodi e dispositivi per il controllo della qualità alimentare.	Permette di focalizzare la ricerca ad ambiti di qualità e controllo di produzione.
CPC G06V 10/00 – Riconoscimento immagini	Include tecnologie di riconoscimento immagini anche mediante strumenti di machine learning e AI.	Consente di esplorare soluzioni basate su visione artificiale, rilevanti per il monitoraggio e la valutazione automatica dei prodotti.
CPC H04N 5/225 – Ispezione visiva automatica	Comprende sistemi e metodi per l’ispezione automatica di immagini o video.	Consente di esplorare soluzioni basate sull’ispezione visiva automatica di immagini o video.

Tabella 4 - Criteri di inclusione ricerca brevetti Lens.org

L'insieme di questi passaggi metodologici ha permesso un’analisi globale dello stato della ricerca sull’argomento di tesi, permettendone un’analisi quantitativa e qualitativa completa, comprendendo informazioni derivanti non solo da paper accademici pubblicati, ma integrando fonti della letteratura grigia e brevettuale.

Risultati della ricerca bibliografica

Il capitolo presentato volge a compiere un'analisi degli output di ricerca trovati attraverso lo studio metodologico della letteratura.

Attraverso la metodologia PRISMA, l'analisi della letteratura grigia e lo studio sui brevetti patentati, si sono ottenuti diversi output di ricerca qualitativi e qualitativi, che vengono analizzati in questo capitolo di tesi.

La struttura, in primo luogo, volge sull'analisi dei risultati relativi al Machine Vision applicato ai prodotti lavorati, fornendone uno studio sulle tecniche maggiormente in uso di acquisizione immagini e pre-processing, per proseguire successivamente al commento sui difetti tipici studiati in relazione alla tipologia di prodotto e algoritmo, e concludere con una review dello stato della ricerca e sulle eventuali lacune bibliografiche.

La seconda parte del capitolo è dedicata al confronto delle tecniche evidenziate sui prodotti lavorati con le materie prime, prese in analisi attraverso l'applicazione di uno dei criteri di esclusione della metodologia PRISMA.

Questo confronto risulta utile per una panoramica globale sullo stato dell'arte e sulle possibili implementazioni future.

MV applicata ai prodotti alimentari lavorati

La *Food processing technology* è una delle parti più rilevanti all'interno della vita moderna globale, e sta ricoprendo un ruolo fondamentale all'interno dell'industria alimentare: la qualità di produzione e la relativa sicurezza ad essa associata sono tra gli argomenti e le preoccupazioni principali della società attuale [12].

L'industria alimentare sta diventando maggiormente competitiva e dinamica, e la consapevolezza della clientela rispetto alla qualità dei prodotti sta aumentando con essa [12].

Con la crescente richiesta di domanda, l'applicazione di metodi efficaci ed efficienti per la detection e la separazione in linea produttiva di eventuali difetti e errori, risulta cruciale all'interno del mondo odierno [13][14].

Questa continua evoluzione si identifica principalmente nelle nuove tecniche di machine vision e associato deep learning, al fine di identificare eventuali difetti di produzione che vanno a inficiare sulla qualità del prodotto lavorato.

Questo capitolo offre una panoramica sullo stato dell'arte del machine learning applicato al settore alimentare, specialmente di prodotti lavorati, focalizzandosi sull'applicazione di tecniche di visione artificiale al fine di identificare difetti nella qualità dei prodotti, eventuali errori di fabbricazione, affinchè i costi di scarto ed eventuali costi di riproduzione possano essere individuati e bloccati senza bloccare la linea di produzione.

Analisi corpus di articoli evidenziati

Gli articoli derivanti dalla ricerca condotta attraverso la metodologia PRISMA sono stati analizzati non solo per i contenuti, ma anche per la distribuzione geografica e della rete di autori presente intrinsecamente tra le informazioni delle pubblicazioni; tutte le informazioni sono state prese dagli articoli e inserite in tabelle Excel con le informazioni di riferimento come autore, numero citazioni e paese di provenienza del relativo articolo.

Dalla ricerca emergono due dati fondamentali: il primo si occupa degli autori e coautori delle pubblicazioni, e dei possibili collegamenti tra di loro.

Per analizzare la co-autorità e possibili intersezioni, è stato applicato uno script Python (Allegato G) che producesse un grafo in cui i nodi fossero rappresentazione degli autori degli articoli e gli archi le relative co-authorships.

La dimensione dei nodi inoltre è stata programmata in maniera tale che riflettesse le citations degli articoli correlate al relativo autore: maggiore la dimensione dei nodi, maggiore il numero di citazioni ottenute.

Un filtro fondamentale applicato all'analisi è stato quello di permettere l'inserimento all'interno del grafo, esclusivamente ad autori aventi almeno due co-authorships tra gli studi analizzati, a questo filtro si è posta l'eccezione che l'autore fosse tra i maggiormente citati.

Dalla figura di output 4 è possibile evincere i cluster di co-autori e come la coppia di autori maggiormente citati sia tuttavia priva di altri scrittori correlati.

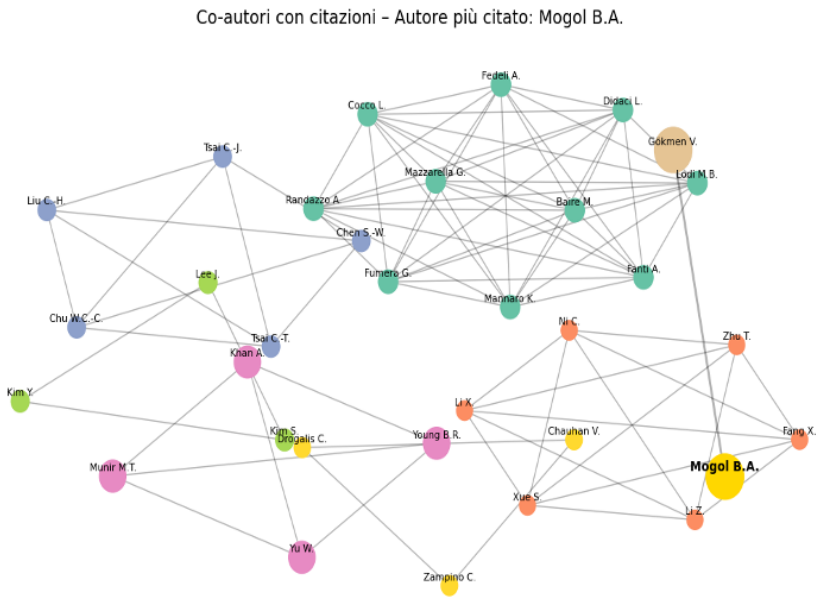


Figure 4 - Rete di coautori

I colori rappresentano cluster di autori che lavorano maggiormente tra loro e andando a cercare più del dettaglio le motivazioni si è potuto riscontrare che tendenzialmente:

- Vi sia un'appartenenza allo stesso dipartimento universitario
- Sia presente la stessa tipologia di prodotto come focus di ricerca

Questi due elementi rappresenterebbero così una possibile motivazione di creazione dei raggruppamenti e della rete narrativa.

A seguito della ricerca eseguita su chi ha pubblicato gli articoli evidenziati, la ricerca si è basata sul dove fossero pubblicati i suddetti paper, così da avere una panoramica globale sui paesi maggiormente attivi nella ricerca.

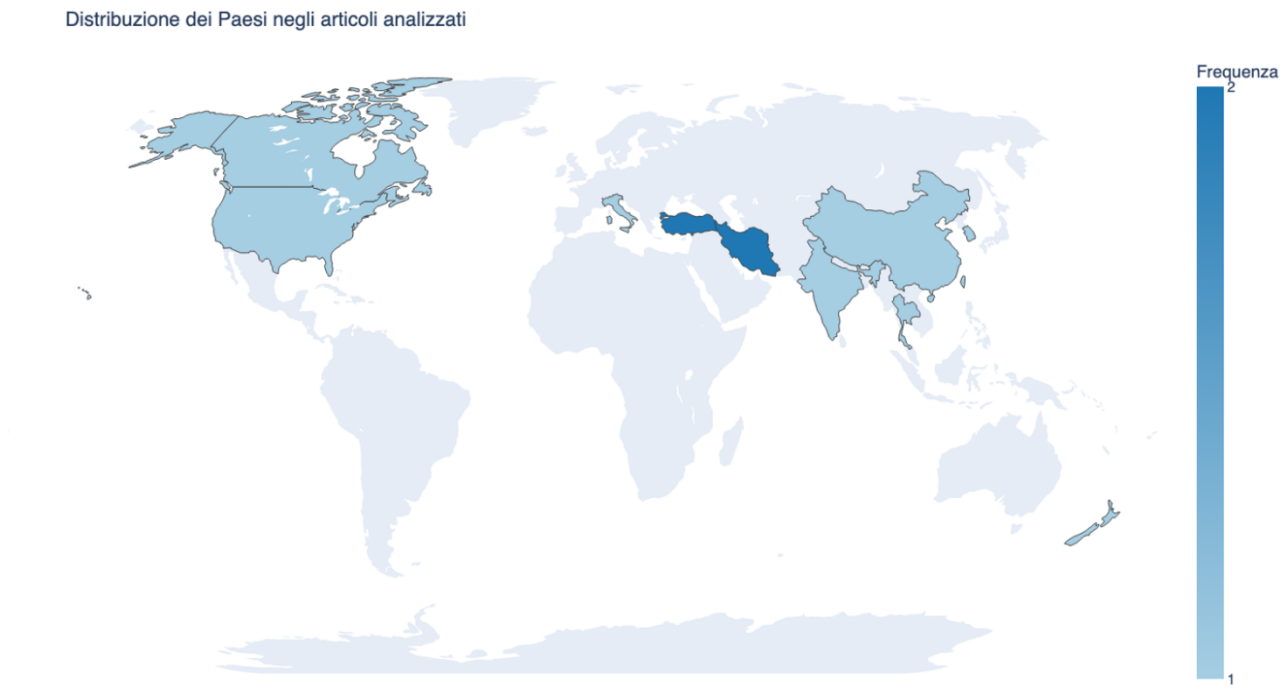


Figure 5 - *Distribuzione geografica articoli*

Sebbene sia importante considerare la numerosità ridotta del campione utilizzato come base strutturale dell'analisi, alcuni punti derivanti dall'immagine 5 risultano rilevanti per donare un contesto geografico alla ricerca.

Le aree con maggiori pubblicazioni risiedono in Medio Oriente e sono seguite da pubblicazioni presentate negli Stati Uniti, Canada, Italia ed Est Asiatico, con una menzione alla Nuova Zelanda fonte di un articolo pubblicato.

Queste considerazioni estraggono un interessante punto di ricerca: le pubblicazioni sono incentrate su zone per lo più in via di sviluppo, aventi un'elevata densità di popolazione ed eventualmente necessità di controlli più stringenti sulla qualità dei prodotti.

A seguito di questa breve analisi introduttiva sul contesto autoriale e geografico dei paper analizzati, è possibile passare alla fase della ricerca interna degli articoli, commentandone gli output rilevati e inserendoli in tabelle comparative adatte all'ambito di tesi.

Definizione e caratteristiche

Il machine vision nasce come uno sviluppo del computer vision, volto però al miglioramento delle tecniche di acquisizione e processamento dell'immagine, in modo da integrarle in una reale linea di produzione, ottenendo così una finalità maggiormente operativa e industriale.

I Machine Vision System (MVS) in ambiti industriali e di produzione; possono utilizzare da una a più telecamere al fine di catturare, elaborare e riconoscere oggetti statici o in movimento, tutto automaticamente [15].

Un sistema di MV generalmente è composto da delle *digital cameras*, un *image processing software* in funzione su hardware fisici collegati alla telecamera e può integrare *sistemi meccanici* per il sorting dei prodotti identificati, come mostrato in figura 6.

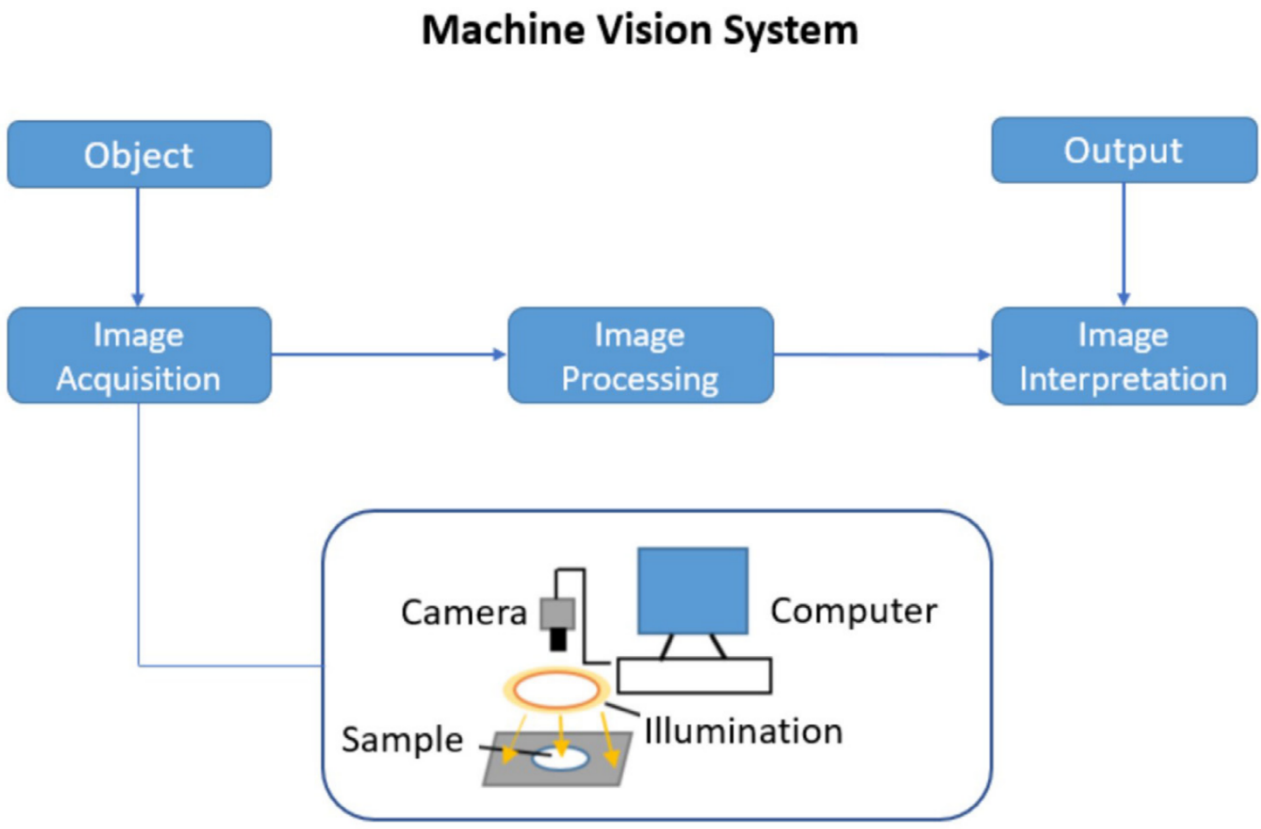


Figure 6 - Componenti principale di un MVS [12]

La struttura a moduli dei MVS presenta alcuni moduli principali e altri secondari, tutti collegati ad una specifica definizione [12]:

- Image acquisition: rappresenta la cattura dell'immagine mediante telecamere e illuminazione controllata (camere RGB, digital, HSI).
- Image Processing:
    - Pre-processing: Filtraggio, correzione dell'illuminazione e normalizzazione del colore.
    - Segmentation/Feature extraction: isolamento delle aree di interesse e successiva applicazione di calcolo delle proprietà geometriche e cromatiche.
- Classificazione/Interpretation: algoritmi di ML o DL che permettono di etichettare ed eventualmente quantificare difetti.
- Output/Automazioni: eventuali sistemi di scarto o smistamento automatico.

Un'ulteriori suddivisione di questa parte di sistema è data dal survey "Deep learning and machine vision for food processing" [15], che suddivide in tre fasi l'intero processo, identificandolo come low level processing (image acquisition e pre-processing), mid-level processing (image segmentation e description) e high level processing (recognition e interpretazione) [15].

Il processo di analisi delle immagini consiste dunque nell'identificare l'oggetto rispetto allo sfondo presente nel frame, e generare informazioni quantitative che verranno successivamente utilizzate in un controllo di sistema per favorire il decision making applicato.

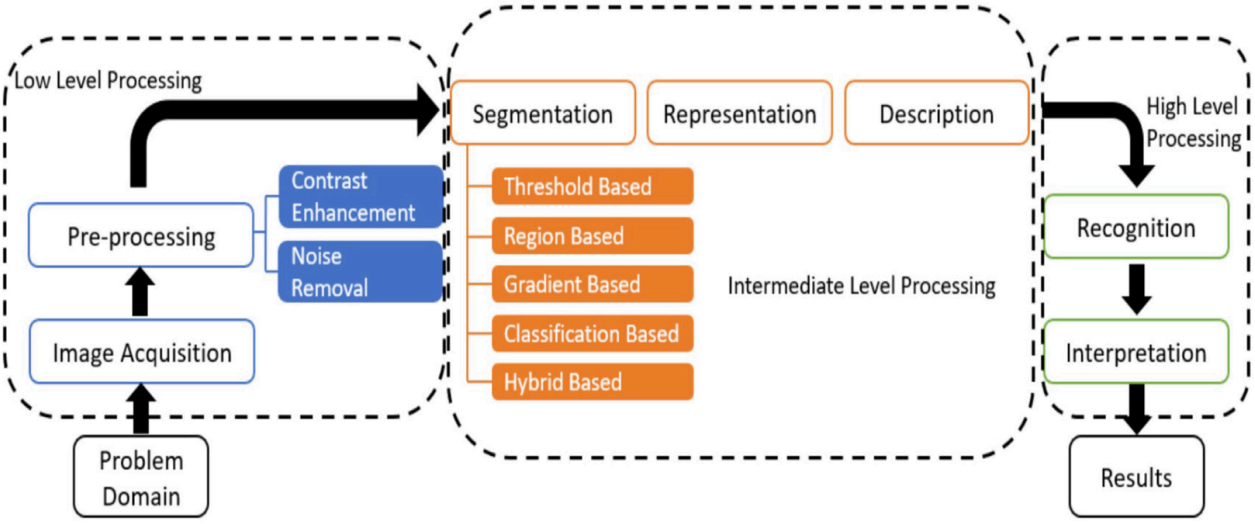


Figure 7 - Descrizione processo tecnico di Machine Vision [11], [15]

Il seguente capitolo segue la struttura identificata da [15] e integra l'analisi della ricerca bibliografica all'interno della suddivisione riportata, inserendo le ricerche su tecniche di acquisizione di immagini, fasi di pre-processing e gli algoritmi utilizzati per l'high level processing. In conclusione, i risultati di prodotti e difetti identificati come ouput verranno riportati in forme tabulari e descrittive.

Acquisizione immagini e tecniche di pre-processing

La parte di image acquisition emersa dalla ricerca dello stato dell'arte è mostrata in tabella 5.

Tipologia acquisizione immagini	**Formato [pixel x pixel]**	**Condizioni sperimentale**	**Note**	**Algoritmo**	**Autore**
Digital camera	256x256 150x150	Camera posizionata a 90° rispetto all'oggetto e fonti luminose a 45° (light source (D65). Camera posizionata verticalmente a una distanza di 24 cm, 2 flashlight posizionate a 45° rispetto l'oggetto e la camera. Due 50 w luci, a 45° di distanza dall'oggetto.	Utilizzato per acquisizione immagine su nastro trasportatore	Machine vision, CNN, machine learning	[1], [16], [17], [18], [19], [20], [21]
Optical camera	1024x768	Nessuna aggiunta di luce artificiale per catturare l'immagine.	Immagini singole da cui vengono estratte feature geometriche	SVM	[22]
Overhead camera	227x227x3 2000x2000 2376x584	Camera posizionata a 90° rispetto gli oggetti studiati.	Utilizzato per acquisizione immagine su nastro trasportare	CNN	[13], [23], [24]
Blackout chamber	/	Camera posizionata verticalmente rispetto il forno intelligente, con lampada LED posizionata a lato per illuminazione più uniforme.	Immagini singole	Machine learning	[25]

Tabella 5 - Tecnologie di acquisizione dati

Dalla tabella appare evidente come sia stata seguita negli articoli analizzati la struttura tipica dei MVS e come semplici digital camera risultino tra le tecnologie maggiormente utilizzate, dimostrando la semplicità e il controllo dei costi di implementazione della parte hardware di setup del sistema.
Solitamente la telecamera risulta posizionate verticalmente rispetto al nastro trasportatore, ed affiancata da due luci artificiali che ne consentono un utilizzo migliore e una cattura di immagine più definita. Le unità di misura dell'acquisizione dell'immagine non risultano eccessivamente rilevanti, in quanto verranno successivamente modificate in fase di pre-processing dell'immagine; è tuttavia importante considerare il contesto produttivo associato ad ogni esigenza, la blackout chamber risulta

essere macchinosa e costosa, catturando immagini molto dettagliate ma con latenza bassa e raramente applicabile in linee di produzione [25].
La soluzione maggiormente cost-effective risulta l'applicazione di una digital camera abbinata ad un sistema di luce inclinato di 45° rispetto al nastro trasportatore dove sono presenti gli oggetti di studio.
Affiancata alla fase di image acqusition è presenta la fase pre-processing, step fondamentale al fine di una corretta applicazione dell'algoritmo al dataset di studio.
Il pre-processing si occupa di generare nuove immagini partendo da quelle acquisite, con l'obiettivo di estrarre e implementare la caratterizzazione della zona di interesse (ROI) [15]: molto spesso le immagini acquisiste con semplici telecamere possono risultare imperfette riguardo a parametri come: luce, distanza, risoluzione o altre tipologie di interferenze.
All'interno della ricerca sullo stato dell'arte attuale un esempio di applicazione di algoritmi per il pre-processing è emerso nell'utilizzo di VGG16 [17], per attuare un transfer learning a una fase di fine tuning.
Sebbene a seguito del low level process dovrebbe subentrare l'intermediate level, lo stato dell'arte bibliografico a riguardo è carente di informazioni, in quanto incentrato maggiormente in algoritmi di real-time object detection da applicare in linea di produzione: questa distinzione è essenziale, siccome i nuovi algoritmi CNN-based, di deep o machine learning, puntano all'ottimizzazione computazionale e all'integrazione di alcuni layer che risultavano essenziali in algoritmi precedenti.
Dunque, seppur esistendo concettualmente, la fase intermedia tende ad essere inglobata dagli algoritmi moderni, che applicano direttamente tecniche di rilevamento e identificazione dei difetti [26].

Algoritmi utilizzati

La fase di high level processing rappresenta lo step finale nei sistemi di machine vision, integrando algoritmi che combinano una parte di recognition dell'immagine ad una di interpretazione, andando così a generare l'output finale atteso [15].
All'interno della ricerca bibliografica i risultati ottenuti riguardo le differenti tipologie di algoritmi sono stati raccolti all'interno della seguente tabella 9, che li identifica per categoria di base, tecnologia specifica applicata, task di visione, eventuale carico computazionale se presente all'interno dei paper analizzati e due colonne rappresentanti i vantaggi e gli svantaggi dell'implementazione dell'eventuale tecnologia analizzata all'interno di contesti di Piccole e Medie Imprese.

Categoria	Tecnologia	Task	Carico computazionale	Vantaggi	Svantaggi	Autori
CNN	VGG16	Transfer learning e fine-tuning con image augmentation. Ottimizzare performance dei mobile e embedded device.	Training time: 42.13 minuti per 30 epoche. Single GPU hardware.	- Architettura consolidata - Utile per applicazioni di classificazione e pre-processing	- Elevato consumo di memoria e tempo di addestramento - Impatto significativo sui costi computazionali e operativi	[17], [23]
	YOLOv3	Edge computing evironment, object detection.	GPU NVIDIA RTX 2080 super e 64 GB running memory.	- Rilevamento in tempo reale buona velocità - Adatto a embedded system e contesti produttivi a vincolo di latenza	- Ridotta accuratezza nella rilevazione di oggetti piccoli o sovrapposti - Possibile compromissione delle prestazioni in ambienti ad alta densità visiva	[27]
	YOLOv5	Object detection e identificazione corpi estranei.	NA	- Equilibrio tra accuratezza e velocità - Ampia documentazione - Supporto su hardware GPU mainstream	- Prestazioni subottimali su dataset complessi e realistici rispetto a versioni più recenti - Ridotta scalabilità in ambienti industriali ad alta variabilità visiva	[18]
	YOLOv7	Object detection e identificazione qualità estetica del prodotto.	10,9 ms e utilizzo di 73,1 Mb	- Elevata efficienza su GPU - Migliorata capacità di rilevamento rispetto alle	- Maggiore complessità architetturale - Richiede infrastruttura hardware dedicata e	[24]

				-	versioni precedenti Adatto a contesti di produzione automatizzata	personale tecnico con competenze specifiche	
	YOLOv8	Object detection e identificazione stato di cottura attraverso browning stages, e hold-out metodo	NA	-	Integrazione nativa di detection, segmentation e classification Flessibilità applicativa Interfaccia user-friendly	- Necessita di fase di tuning avanzato Richiede dataset etichettati di alta qualità e pipeline di validazione ben strutturate	[13]
Classificatore tradizione	SVM	Tecniche di segmentazione	Le performance e il relativo carico computazionale sono stati osservati da Gennaio 2020 a Marzo 2021.	-	Ottime prestazioni su piccoli dataset Efficace in problemi binari	- Scalabilità limitata Sensibilità alla scelta del kernel e dei parametri Complessità nel tuning manuale su dataset industriali ad alta dimensionalità	[22]
Machine Learning	Random Forest, KNN	Misurazione del volume del pane, del colore e analisi statistica.	Arduino Uno hardware	-	Integrazione diretta con macchine industriali Applicabilità elevata in ambiti di ispezione visiva automatica e controllo qualità	- Elevata sensibilità a condizioni ambientali Richiede calibrazione costante e ambienti controllati per garantire la ripetibilità dei risultati	[2], [25]

Tabella 6 - Modelli suddivisi per categoria, hardware e task

Le categorie evidenziate in tabella sono riportate in maniera generica, al fine di una rappresentazione lineare; tuttavia, è bene specificare come la categoria CNN si suddivida in due sottocategorie [28]:

1. Feature-based CNN: le quali si occupano di features e pre-processing; dunque, citate nel capitolo dedicato.

2. Detection-based CNN: le quali comprendono modelli quali YOLO che si occupano di object detection e di quality assessment in tempo reale, utilizzando tecnologie end-to-end.

Utilizzando questa panoramica in ottica scientifica, risulta evidente come l'utilizzo di un algoritmo dipenda da numerosi fattori e sia strettamente collegato alle esigenze aziendali del singolo: risorse hardware disponibili, eventuali costi associati e task di visione necessarie.

La letteratura mostra una evidente tendenza che volge verso architetture più complesse e di deep learning, le quali riducono il flusso di lavoro cercando di ottimizzare ed eliminare layer presenti all'interno delle reti neurali tradizionali. Questa applicazione comporta un trade-off di costo-qualità, che pone l'obiettivo di analisi sull'esigenze dell'azienda.

Gli algoritmi più avanzati come YOLOv7 e YOLOv8 offrono prestazione elevate e tempi di latenza minimi, ottimi in ottica di linea produttiva; tuttavia, essi comportano un utilizzo di GPU integrata: YOLOv7 richiede un sistema architetturale complesso, che lo pone in una situazione svantaggiosa in ottica di PMI, mentre YOLOv8 appare un giusto compromesso tra utilizzo di hardware robusti e architettura flessibile.

Sebbene i classificatori tradizionali come SVM e classificatori utilizzatori di Machine Learning come KNN e Random Forest, richiedano meno risorse in termini di applicazione computazionale, e riescano a runnare su sistemi CPU, essi non sono ottimali in linea produttiva, in quanto possessori di limiti in termini di scalabilità e accuratezza in ottica di real-time object detection.

I vantaggi e gli svantaggi operativi evidenziano i modelli deep learning come robusti in ambienti complessi, ma richiedenti di dati di alta qualità e fasi di tuning più accurata; i modelli tradizionali invece, più facili da integrare e più interpretabili, ma con una scarsa applicazione in ambito reale.

La tabella evidenzia inoltre possibili implicazioni per le PMI: per le piccole e medie imprese i modelli tradizionali potrebbero sembrare i favoriti, in quanto rappresentano una soluzione economica, tuttavia, non rappresentano una scelta da favorire in ambito di real-time detection.

I modelli appartenenti al deep learning invece, forniscono una valida alternativa fornendo maggiore automazione e affidabilità per linee produttive ad alta densità.

In conclusione, l'analisi del livello finale di processamento evidenzia come i modelli odierni tendano a virare verso un'opzione di deep learning attraverso l'impiego di reti neurali convoluzionali, capaci di integrare in un'unica pipeline la fase di riconoscimento, interpretazione e decisione.

Attraverso questa ottimizzazione dell'architettura la capacità operativa migliora, limitando i tempi di latenza e fornendo delle soluzioni ottimali applicabili in contesti produttivi.

Tuttavia, la scelta dell'algoritmo ricade anche dal prodotto che si vuole andare a valutare, e dai difetti qualitativi che si vogliono identificare: per fare questo il capitolo successivo si occupa

dell'identificazione e della mappatura dei diversi prodotti analizzati nello stato dell'arte, evidenziandone i difetti ad essi associati.

Difetti tipici nei prodotti lavorati

La fase finale dell'applicazione dei sistemi di Machine Vision applicata al contesto alimentare riguarda l'identificazione dei difetti di qualità ritrovati e i relativi prodotti a cui vengono maggiormente associati.

Questa identificazione dei difetti, e le relative misure di applicazione applicate, rappresentano il punto di incontro tra azienda produttrice e clienti, i quali richiedono costante qualità e affidabilità: la capacità di individuare eventuali difetti in linea produttiva, senza affidarsi all'errore umano, consente infatti di limitare i costi degli scarti ed elevare gli standard di qualità, eliminando il più possibile l'errore visivo.

Nella letteratura studiata appaiono quattro cluster di difetti, a cui è possibile ricondurre ciascun articolo presente nella selezione finale.

Tipo difetto	**Descrizione**	**Prodotti analizzati**	**Algoritmi usati**	**Task di visione**	**Metriche**	**Database**	**Autore**
Difetti di cottura	Stato di cottura (sottocotto/ottimale/sovracotto) con soglia termo/colore.	Cupcake, biscotti, pane, torta, patatine fritte	Machine vision, CNN (VGG16), machine learning	Image segmentation, Transfer learning, Misurazione del colore e del volume	Browning ratio, Accuracy, precision, recall, F1, P-value e volume	Privati e pubblico	[1], [16], [17], [19], [25]
Difetti igienico sanitari/conformità	Detection di presenza di glutine e di eventuali muffe	Torta, pane	Machine vision, CNN (YOLOv5),	Image processing di crumb e crust, Identificazione e classificazione delle immagini.	L*,a*,b* Precision, recall, F1-score	Privato	[18], [21]
Difetti geometrici	Imperfezioni di difetti superficiali come bolle/crepe/macchie, forma non corretta, distribuzione errata di ingredienti	Caramelle, biscotti, pane carasau	CNN (YOLOv3/v7/v8), SVM, machine learning	Segmentation e classificazione, identificazione difetti, feature extraction	Precision, recall, F1, mAP@50, mAP@50-95,	Privato, Privato, Pubblico	[13], [20], [22], [24], [27]

					confusion matrix		
Valutazione globale della qualità estetica	Idoneità alla vendita e quantità complessiva del prodotto	Latte in polvere, pane, taralli	Machine vision, machine learning	Classificazione classi di difetti,	Precision, recall, F1, mAP@50, mAP@50-95, confusion matrix	Privato	[23][2]

Tabella 7 - Tipologie di difetti analizzati suddivisi per task, prodotto e algoritmi

Le tipologie di difetti scelte per la suddivisione state identificate attraverso l'analisi degli articoli e raggruppate secondo cluster tematici:

- Difetti di cottura: legati al grado di cottura di prodotti, principalmente lievitati, i quali sono suddivisibili in under baked/normal/over baked e sono tendenzialmente valutati con una browning stage ratio, indice che ne identifica il livello di bruciatura.
- Difetti igienico sanitari/conformità: questa categoria è strettamente collegata agli standard igienici di qualità presenti nel campo alimentare. La presenza di eventuali principi di muffe, o eventualmente la detection di gluten in prodotti che dovrebbero esserne senza, può risultare fondamentale se identificata in fase precoce, senza necessariamente aspettare la contaminazione della linea.
- Difetti geometrici: alcuni prodotti alimentari risultano di qualità elevata grazie alla forma omogenea dell'oggetto, identificativa del brand o adatta per un certo tipo di trasporto. Questa categoria comprende tutti i difetti catturati dalla edge detection o dall'applicazione di algoritmi geometrici che ne identificano le linee e la forma, prima di identificare la classe di errore in sé.
- Valutazione globale della qualità estetica: questa categoria è complessiva di tutti i modelli rappresentante l'identificazione dell'idoneità alla vendita, e la qualità generale del prodotto. Ad esempio, rientra in questa categoria il caso studio del latte in polvere, che considera la qualità non solo della linea di produzione, ma del volume, dello spessore e di altri parametri dell'alimento.

Parallelamente all'identificazione e suddivisione in categoria dei difetti emersi, dalla letteratura emerge una forte correlazione tra tipologia di difetto, prodotti ad esso associati e modelli di algoritmi utilizzati.

Al fine di identificare la continuità nelle ipotesi e nella ricerca, in tabella vengono riportati i principali studi analizzati aventi questa tipologia di correlazione, suddivisi secondo la classificazione ATECO e associati ai relativi task di visione e dataset impiegati.

Codice ATECO	Categoria	Esempi	Task di visione	Algoritmi utilizzati	% totale	Immagini	Dataset	Dimensione dataset	Autore
10.71.10	Produzione di pane	Pane, pane carasau, panini al latte, tortilla	Classificazione difetti estetici, stato cottura, detection difetti igienico sanitari.	Machine learning, CNN, SVM	35%	Digital camera, overhead camera, blackout chamber	Privato Privato Pubblico Privato Privato	900 1881970 (pixels) / 4990 3342	[13], [17], [18], [22], [25]
10.72.00	Produzione di fette biscottate, biscotti, prodotti di pasticceria conservati	Biscotti, taralli	Edge detectiction per difetti estetici e geometrici, valutazione cottura, valutazione globale qualità estetica. (Brown ratio)	Machine vision, machine learning, CNN	23%	Digital camera, overhead camera	Privato Privato Privato Privato Privato	500 / 1225 4990 3342	[16], [20] [13], [23]
10.71.20	Pasticceria fresca	Cupcake, torta	Detection difetti di cottura e igienico sanitari.	Machine vision	12%	Logitech USB webcam, Digital camera	Privato Privato	6 /	[1], [21]
10.31.00	Lavorazione e conservazione delle patate	Caramelle gommose	Edge detection e difetti estetici e geometrici.	CNN	12%	Industrial camera	Privato Privato	/ 5000	[24], [27]
10.51.20	Trattamento igienico, conservazione e trasformazione del latte	Latte in polvere	Detection valutazione qualità globale.	Machine vision	6%	Immagini HSI	Privato	1225	[2], [23]

Tabella 8 - Prodotti analizzati per codice ATECO, task di visione associato e algoritmi

La ricerca bibliografica mostra una predominanza di prodotti alimentari identificati attraverso il codice ATECO 10.71.10 (Produzione di pane), i quali occupano circa il 35% della letteratura analizzata: questa percentuale è dovuta dalla grande variabilità dei prodotti da forno, i quali possono essere analizzati sia per difetti di cottura, che per edge detection, donando così problemi multiclasse. Oltre alla grande variabilità visiva che li identifica, i prodotti da forno hanno una forte rilevanza economica all'interno del mondo odierno, rendendoli così oggetti di studio perfetti, e un benchmark ideale per la comparazione di sistemi di visione automatica.

Si può notare inoltre come anche il settore della pasticceria fresca possa essere considerato come di possibile sviluppo, in quanto utilizza tecniche simili a quelli applicati alla produzione di pane, non fermandosi però alla sola fase di cottura, ma andandone a valutare anche eventuali difetti estetici composti da diverse parti dell'oggetto.

Dalla colonna identificativa della tipologia di dataset spicca immediatamente la quasi totalità di dataset privati, che valida la teoria secondo la quale il mercato appartenente a questa tipologia di algoritmi sia prevalentemente di ambito industriale o contesti sperimentali, confermando la quasi inesistenza di database pubblici contenenti queste specifiche tipologie di dati.

La numerosità dei dataset varia visibilmente, rendendo facilmente interpretabili le immagini catturate in ambito industriale e applicate successivamente a dei casi studio, e i modelli allenati e validati in ambiti di ricerca, senza l'utilizzo di fonti direttamente catturate all'interno della linea di produzione.

Dal punto di vista tecnico, inserito all'interno degli algoritmi utilizzati, prevalgono CNN con utilizzo di illuminazione controllata e telecamere verticalmente posizionate.

L'analisi affrontata sui difetti e i prodotti ad essi associati, dimostra come il Machine Vision in ambito alimentare sia in costante crescita, e si sviluppi progressivamente passando da semplici modelli di feature extraction a complessi sistemi di automazione, pronti per essere applicati in ambito industriale.

I settori più esplorati risultano colmi di interessanti spunti e processi consolidati, mentre l'applicazione su prodotti maggiormente deperibili, come carne o latticini, risulta ancora in via di sviluppo e in costante aggiornamento, bloccata anche dalla mancanza di dataset pubblici, disponibili per l'allenamento. La difficoltà delle reti neurali risiede inoltre nell'eterogeneità: prodotti industriali come pane in cassetta o cookies, i quali devono avere le stesse caratteristiche metriche, risultano oggetti più facilmente applicabili, a differenza di carne o prodotti freschi, che hanno come caratteristica principale la forte variabilità.

In prospettiva futura sicuramente l'accesso pubblico a dataset di tale numerosità favorirebbe la diffusione di studi applicativi, in quanto consentirebbe l'allenamento di modelli leggeri e scalabili,

senza dover necessariamente allenare algoritmi senza checkpoint consolidati o immagini realistiche di linee di produzione.

Risultati analisi integrativa: letteratura grigia e brevetti industriali

Un'ulteriore analisi integrativa aggiunta in fasi di confronto e valutazione dei risultati è quella degli output raccolta dalla letteratura grigia e lo studio sui brevetti industriali.

La tabella 9 include per ciascuna azienda informazioni quali: il paese della sede, la tipologia di prodotto venduto, il modello specificato e il focus applicativo; le ultime due colonne contengono il caso di studio evidenziato e la relativa sitografia.

Azienda	**Paese**	**Tipo prodotto**	**Modelli**	**Focus applicativi**	**Caso studio**	**Sitografia**
Ultralytics (YOLOv5/8/11)	UK	Framework + SDK	YOLO family (detection, segmentation, pose)	Rilevamento difetti visivi su pezzi singoli	Ultralytics YOLO powers Specialvideo's food inspection tool	[29]
Detectron2 (Meta AI)	USA	Framework PyTorch	Faster R-CNN, Mask R-CNN, REtinaNet	Segmentazione anomalie, difetti superficiali	Allergic Fruit Detection — notebook	[30]
MMDetection (OpenMMLab)	Cina	Toolbox PyTorch	YOLO, RetinaNet, Cascade R-CNN	Test multipli di architetture su dataset difetti	Phenotyping fruit morphology	[31]
MVTech HALCON/ deep learning tool	Germania	Software commerciale di visione	CNN per classificazione, detection di anomalie	Rilevamento corpi estranei, difetti di superficie, contaminazioni	Machine vision and deep learning detect defects in the food industry	[32]
Cognex	USA	HW+SW turnkey	CNN proprietare	Ispezione etichette, tappi, sigilli, contaminanti, difetti estetici	Cosmetic Defect Detection	[33]
Matrox Imaging (MIL, Design Assistant)	Canada	SDK + Vision Controller	CNN integrabili via moduli deep learning	Controllo qualità confezioni, bakery, carne	Machine vision inspects fruit using off-the-shelf software	[34]

Keyence (CV-X/XG con DL)	Giappone	Smart camera + AI	CNN embedded	Difetti nel packaging	High-Speed Inspection of Broken Biscuits	[35]
Viso.ai	Svizzera/Austria	Piattaforma enterprise no code	Supporto YOLO, SSD, EfficientDet	Implementazione end to end su linee food	Leading Global Food Retailer Enhances Safety with Viso AI	[36]
Robovision	Belgio	Piattaforma AI visione industriale	CNN custom, YOLO/Mask R-CNN based	Sorting alimenti, ispezione superfici, classificazione difetti	AI-Driven Food Sorting Robot for Peach Processing	[37]
Deepomatic	Francia	Piattaforma Computer vision Saas	CNN personalizzabili	QA packaging, verifiche conformità in filiera	Automated Meal Tray Detection in Corporate Dining	[38]

Tabella 9 - Tabella riassuntiva degli output della letteratura grigia

Dalla letteratura grigia evidenziata emergono alcuni dati importanti al fine della visione globale della ricerca di tesi: emerge infatti un forte interesse industriale e una presenza già avanzata di queste tecnologie volte al controllo della qualità alimentare; le aziende più attive risultano essere allocate in ambienti con elevata automazione industriale e con settori alimentari molto industrializzati quali Stati Uniti, Germania e Giappone.

A differenza della ricerca bibliografica, in cui il focus risulta essere applicativo a livello di miglioramento algoritmico e alla sperimentazione in ambienti di laboratorio, il focus applicativo della ricerca sulla letteratura grigia risulta focalizzarsi a livello industriale, analisi di costo di implementazione e sui risultati empirici dei sistemi implementati.

Da questa ricerca emerge dunque una complementarità tra le due differenti aree di ricerca: quest'ultima analisi permette di comprendere in maniera globale la complessità di una applicazione industriale e di come diversi fattori, non considerati nella letteratura scientifica, debbano invece essere presi in considerazione se l'obiettivo è l'applicazione aziendale.

L'ulteriore analisi integrativa è stata compiuta sui brevetti e se ne sono ricavati alcuni output interessanti presentati in figura 8 e 9.

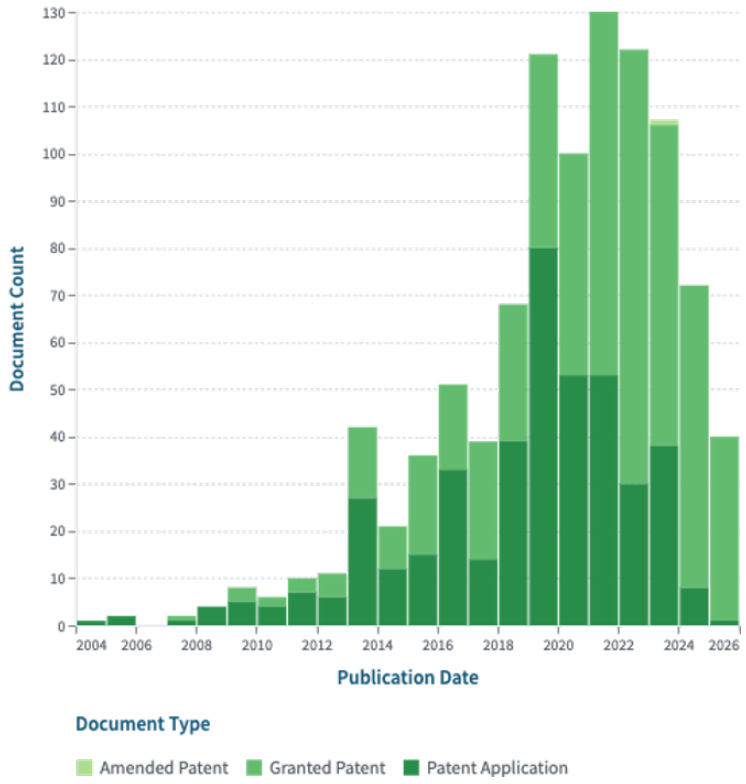


Figure 8 - Pubblicazioni brevetti anni 2004-2025 [11]

Il grafico a barre (figura 8) presenta l'andamento dei numeri di brevetti industriali pubblicati per anno, e suddivisi in Patent Application, Granted Patent e Amended Patent, pubblicati tra il 2004 e il 2025. L'analisi di questi dati mostra una crescita notevole a partire dal 2010, con incremento costante fino al raggiungimento del picco tra il 2019 e il 2022; in questo intervallo temporale è possibile notare il maggior numero di documenti, indice di un'intensa fase di ricerca e sviluppo, e di volontà nel volere proteggere il lavoro industriale svolto.

È importante sottolineare, tuttavia, come il tempo di vita di un brevetto non sia immediato, e come ciascuno abbia un tempo di latenza variabile tra i 18 e 36 mesi: questo dato sposta dunque il picco di studi intorno al 2017-2019, giustificando il calo subito nel 2024 e conseguente 2025.

Il grafico mostra inoltre un elevato numero di Patent Application, confermando quanto il settore sia emergente e come molti brevetti si trovino in fase di validazione e di eventuale pubblicazione.

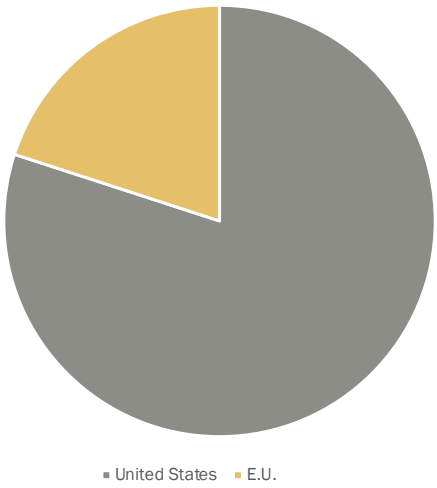


Figure 9 - Grafico rappresentante la suddivisione dei brevetti attivi pubblicati su Lens.org [11]

L'analisi brevettuale fornisce inoltre dei dati di appartenenza dei brevetti individuati (figura 9), che mostra un'elevata predominanza statunitense rispetto ai brevetti depositati in Unione Europea, evidenziando un'importante mancanza di rappresentazione di paesi Asiatici, emersi invece nelle ricerche precedenti.

La percentuale elevata dei brevetti depositati all'interno degli Stati Uniti, oltre ad essere giustificata da un maggiore interesse del Paese nel voler proteggere la proprietà intellettuale, può essere attribuita alla fonte che è stata utilizzata per la ricerca: Lens.org tende infatti a preferire una più grande copertura dei dati USPTO, il quale può fornire una lieve distorsione quantitativa.

Tuttavia, sebbene l'Europa rappresenti una quota nettamente minore, da una analisi dei soggetti depositari sul sito di Lens.org, è possibile osservare come essa comprenda brevetti principalmente depositati da grandi imprese manufatturiere e centri di ricerca associati.

Questa tendenza dimostra dunque la volontà di integrare questi meccanismi di visione in chiave produttiva, non attraverso lo studio esclusivo di framework algoritmici.

Attraverso l'analisi totale abbinata all'integrativa presente in questo capitolo, è possibile individuare uno stato preciso della ricerca bibliografica e le lacune evidenziate dagli output di ricerca.

Stato della ricerca e lacune bibliografiche

Lo stato dell'arte analizzato risulta essere completo di alcune caratteristiche e carente di alcuni fattori, entrambi gli aspetti evidenziati nella tabella 10 in maniera descrittiva comparativa.

Aspetto	Stato attuale	Lacune	Prospettive future
Algoritmi	CNN, SVM, Random Forest.	Mid-level processing spesso integrato e poco trattato in letteratura.	Integrazione DL end-to-end, ottimizzazione latenza e aggiunta di analisi di costi.
Prodotti	Pane e prodotti da forno (35%).	Prodotti deperibili come carne e latticini poco studiati.	Estensione a nuovi alimenti con dataset pubblici.
Difetti	Cottura, estetici/geometrici, igienico sanitario.	Difetti multiclasse complessi nulli o poco esplorati.	Approccio multi-task e valutazione di object detection real-time.
Dataset	Prevalentemente privati, dimensioni variabili e poco specifici su numerosità associata al data augmentation.	Scarso accesso a dataset pubblici di grande dimensione.	Creazione di dataset open source e standardizzati, al fine del training e dell'avere un benchmark oggettivo.
Applicazioni industriali	Linee di produzione standardizzate, spesso utilizzate con parametri esclusivi per lo studio di ricerca.	Poca applicazione su linee ad alta variabilità.	IoT e integrazione con sistemi avanzati di automazione di controllo della qualità.
Mid-level processing	Concettualmente presente, ma inglobato in algoritmi moderni e dunque poco studiato e citato.	Mancanza di studi dettagliati esclusivamente sui processi di segmentation.	Analisi più approfondita e metodologie standardizzate.

Tabella 10 - Confronto stato attuale della ricerca e lacune bibliografiche

In sintesi, l'applicazione del Machine Vision System all'interno del contesto di produzione alimentare è in continua crescita e in fase attuale di applicazione, con una progressiva transizione verso algoritmi end-to-end capaci di rilevare e classificare difetti in tempo reale.

Sebbene esistano tuttora alcune lacune in ambito scientifico, esse sono causate da una carenza di letteratura riguardo questo argomento, sia per cause di latenza di progetti scientifici, sia per implementazioni principalmente aziendali e industriali che non sempre richiedono uno studio metodologico di base.

Tuttavia, queste lacune identificano le possibili implementazioni future, e come rendere migliore la ricerca sull'argomento: pubblicando dataset open-source, o implementando maggiormente la varietà dei prodotti analizzati, incentrandosi anche su materiale più fragile e deperibile. Il capitolo successivo si concentrerà sul confronto dello stato della ricerca bibliografica dei prodotti lavorati con quello delle

materie prime, che appaiono maggiormente studiate in letteratura: questo paragone permette un'analisi più completa dei differenti approcci di studio e la possibilità di identificare eventuali soluzioni ibride.

Confronto con le materie prime alimentari

La qualità è un fattore chiave all'interno del product marketing per l'agricoltura e la food industry; numerosi ricercatori hanno consigliato e sviluppato l'utilizzo di sistemi di Machine Vision all'interno del campo delle materie prime, sfruttandola per l'ispezione di imperfezioni superficiali o difetti associati a frutta e verdure. [39]

La capacità di captare difetti come malattie superficiali, ammaccatura o grado di maturità del prodotto, appartiene a una vasta aerea di ricerca, ed è una delle principali e challenging task del settore agroalimentare, per questo motivo le tecnologie non invasive di cattura dell'immagine e rilevamento dei risultati, sono sempre in fase di sviluppo e ottimizzazione [40].

La produzione mondiale di frutta e verdura è stata di 1837 milioni di tonnellate prodotte in un anno, 694 di queste esclusivamente in Cina [40], 197 in India, secondo Paese per produzione. A discapito di questa produzione su larga scala, la disponibilità di frutta e verdura rimane comunque lontana dal consumo medio per essere umano, la cui richiesta risulta maggiore della produzione.

La causa di questa sottoproduzione è costituita da diversi fattori, tra cui la perdita dovuta ai controlli qualità e di sicurezza post raccolta, insieme alle perdite naturali dei prodotti dovute allo stato di maturazione [40].

Sebbene i controlli possano essere compiuti manualmente, l'automazione di essi rende i processi maggiormente rapidi, in modo tale che riescano dunque a seguire la linea di produzione, e maggiormente consistenti, cioè condizionati in maniera quasi inesistente dai bias umani di ispezione [41].

Sistemi MV per materie prime

I sistemi associati allo studio visivo delle materie prime rientrano nella categoria dei MVS Machine Vision System, ma comportano alcune differenze sostanziali rispetto alle tecniche citate per i prodotti lavorati.

Le tecnologie utilizzate analizzano principalmente le categorie di difetti e di particolari utili all'identificazione della qualità, che sono impercettibili all'occhio umano.

Gli studi analizzati mostrano come le architetture hardware siano tipicamente composte da telecamere RGB o multispettrali, sistemi di illuminazione controllata e piattaforme tipicamente statiche di raccolta delle immagini, non testate dunque in un ambito industriale in cui la latenza totale risulta una metrica essenziale.

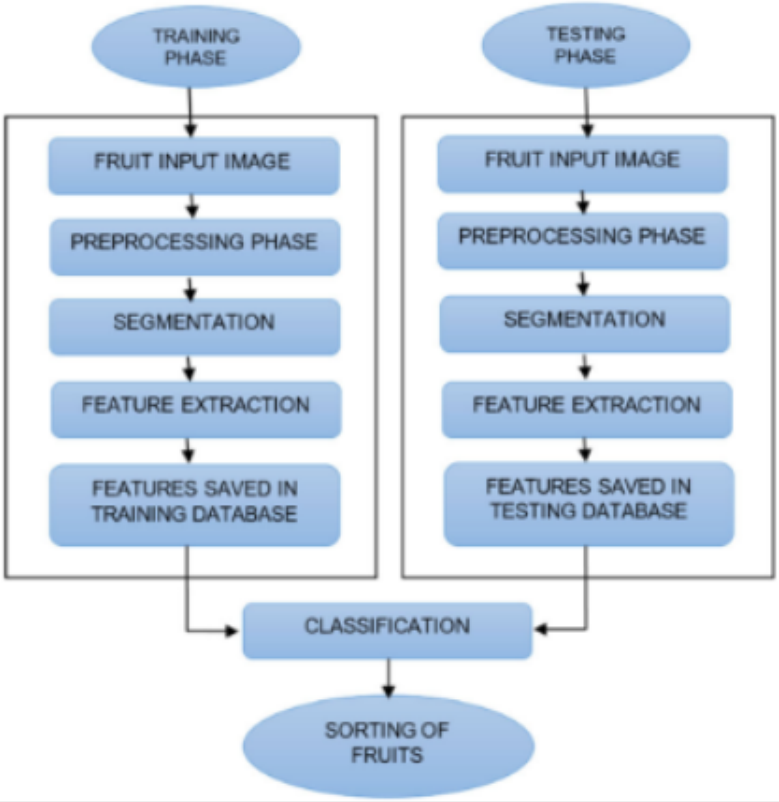


Figure 10 - Diagramma rappresentante il processo di MV per la "Fruit quality evaluation" [42]

Rappresentato in figura 6 il workflow appartenenti agli step base di allenamento e testing di algoritmi appartenenti alla categoria di MV: il training è composto da una prima fase di input dell'immagine, seguita da un pre-processamento della stessa, composta tendenzialmente da applicazione di filtri per la nitidezza dell'immagine, uno stretching di contrasto, una equalizzazione e una greyscale conversions, la quale può essere sostituita da RGB to binary conversion [41].

Durante la seconda fase di image segmentation, le immagini sono suddivise in piccole parti, raggruppate secondo vicinanza di immagine o aventi componenti simili: questa fase è compiuta applicando differenti tecniche composte da thresholding, clustering o region based [41], [42].

Gli ultimi due processi di feature extraction e features saved nei database sono legate tra loro e si occupano di ridurre l'immagine a un insieme di colori, grandezze, forme e materiali, e di salvarle all'interno di un database che sarà utilizzato per la classificazione.

L'ultimo step comprende dunque la classificazione delle diverse materie prime attraverso l'etichettatura delle immagini, evidenziandone classi difettose da classi prive di difetti, e dunque pronte per la fase successiva di produzione.

In questo ambito di classificazione gli algoritmi maggiormente utilizzati risultano essere: K-means, SVM, clustering e decision Trees [41].

Attraverso l'allenamento dell'algoritmo è dunque possibile successivamente testarlo su immagini mai caricate prima, prive di qualsiasi dato che l'algoritmo potrebbe aver già incontrato in fase di allenamento o di validation, meno presente su questi processi lineari.

I difetti maggiormente identificati all'interno dello studio sulla Materia Prima risultano essere i seguenti:

Difetti rilevati nelle materie prime

Codice ATECO	Prodotto	Difetti rilevati	Descrizione
01.24.00	Coltivazione di pomacee e frutta a nocciolo	Maturazione	Analisi del grado di maturazione (acerba, parzialmente matura, matura, decomposta) mediante parametri cromatici e di texture.
		Dimensione e forma	Misurazione degli edges e della simmetria per la vendita commerciale
		Tessuto spugnoso	Rilevamento qualità interna attraverso imaging multispettrale
		Malattie/anomalie biologiche	Rilevamento di muffe o infezioni.
01.25.00	Coltivazione di frutta a guscio	Crepe e rotture nel guscio	Difetti struttuali identificati mediante forma geometrica
		Conformità geometrica	Valutazione edges per vendita commerciale
		Stato del frutto	Classificazione binaria per separare i frutti integri da quelli compromessi.
01.25.01/01.24.01	Frutti di bosco e altri alberi da frutta	Colore e tonalità	Analisi del colore e della conformità cromatica
		Varietà e classificazione	Riconoscimento automatico della specie o della varietà
01.47.00	Allevamento di pollame	Crepe nel guscio	Rilevazione di difetti superficiali

Tabella 11 - Difetti rilevati con MVS riguardanti la materia prima

Dai risultati emerge un forte interesse sullo stato della qualità esterna della materia prima basato sul grado di maturazione ed eventuali difetti nella dimensione o nella forma.

Gli articoli inoltre presentano numerose tecniche di identificazione della qualità specifiche di un certo prodotto: risulta importante per il settore della materia prima, riconoscere le diverse varietà di alimenti, attraverso sistemi automatici di visione.

Un altro dato di interesse risulta il difetto principale identificato, il quale comprende difetti visivi e geometrici come variazione cromatica, dimensione, forma e simmetria: questi parametri infatti rientrano tra gli essenziali per la classificazione commerciale delle materie prime, e dunque di spiccato interesse.

Sebbene l'utilizzo di queste tecnologie appaia consolidato all'interno del contesto delle materie prime, la sua attualità nel settore dei prodotti alimentari lavorati risulta complesso a causa delle differenze nella texture, la non conformità della superficie e alla complessità dei processi produttivi di applicazione; ne consegue quindi la necessità di attuare un confronto tra le differenti casistiche.

Confronto

Al fine di compiere un'analisi approfondita di paragone, è stata organizzata la tabella 12 suddividendo le tipologie di difetti analizzate attraverso una comparazione tra materie prime e prodotti lavorati, fornendo gli algoritmi utilizzati per ciascuna categoria, le metriche di performance dove presenti e i riferimenti bibliografici utilizzati.

La tabella 12 dunque, riassume le differenze significative tra gli studi riguardanti la materia prima e i prodotti lavorati: l'analisi mette in evidenza come, sebbene la somiglianza tra i criteri di valutazione delle due parti, gli algoritmi e la complessità operativa siano differenti tra le due.

Come commentato nei capitoli precedenti le principali caratteristiche dei MVS nei prodotti lavorati sono la valutazione sulle caratteristiche post-processo, come lo stato di cottura, che vengono identificati da reti neurali spesso associate ad algoritmi di machine learning base per una migliore valutazione.

Le metriche di performance valutate mostrano prestazioni promettenti e affidabili, con gradi di robustezza elevati.

Per le materie prime, questa situazione cambia: i principali difetti identificati appartengono alla valutazione della qualità naturale del prodotto, come il grado di maturazione, la presenza di malattia o difetti strutturali, e alla classificazione basata sulla varietà del prodotto.

Gli algoritmi più utilizzati risultano appartenenti alla famiglia delle CNN e Faster R-CNN, capaci di gestire la variabilità dei prodotti agricoli; per questo motivo, causato dalla grande eterogeneità, le prestazioni dei modelli variano in funzione della caratteristica analizzata.

La valutazione globale della qualità estetica presenza performance elevate, dimostrando l'applicabilità trasversale dei modelli analizzati, sia in ambito pre produzione applicato dunque alle materie prime, sia in ambito di trasformazione, applicato ai prodotti lavorati.

In conclusione, i prodotti lavorati beneficiano di condizioni ambientali più stabili e forme geometriche più rigide e dunque giovano di un maggior controllo del processo produttivo; dall'altro lato le materie prime presentano maggiore variabilità naturale che rende dunque la classificazione più complessa, ma maggiormente rappresentata in letteratura.

Criterio	Tipo prodotto	Esempi	Descrizione	Algoritmo	Metriche di performance	Autore
Difetti di cottura	Prodotti lavorati	Cupcake, biscotti, pane, torta, patatine fritte	Valutazione stato cottura attraverso tecniche di imaging	Machine vision, CNN (VGG16), machine learning	Accuracy, precision, recall, F1, P-value e volume.	[1], [16], [17], [19], [25]
	Materie prime	/	/	/	/	/
Difetti igienico sanitari/conformità	Prodotti lavorati	Torta, pane	Detection di presenza di glutine e di eventuali mufffe	Machine vision, CNN (YOLOv5),	Precision, recall, F1, mAP@50, mAP@50-95, confusion matrix	[18], [21]
	Materie prime	Coltivazione di pomacee e frutta a nocciolo	Classificazione per maturazione, dimensione, tessuto e malattie	Faster RCNN, CNN, SVM	Accuracy: Maturazione 75%, dimensione 63,34%, combinata 59%. Tessuto spugnoso: CNN 99,20%, SVM 95,5%.	[43], [44], [45]
Difetti geometrici	Prodotti lavorati	Caramelle, biscotti, pane carasau	Imperfezioni di difetti superficiali come bolle/crepe/macchie, forma non corretta, distribuzione errata di ingredienti	CNN (YOLOv3/v7/v8), SVM, machine learning	Accuracy, precision, recall, F1, P-value e volume.	[13], [22], [24], [27]
	Materie prime	Coltivazione di frutta a guscio, Allevamento di pollame	Stato (Danneggiate, Non danneggiate), Crepe, Valutazione dei bordi e della conformità gemetrica	CNN, SVM	Classsification accuracy 99.78% . Average accuracy tra i vari tipi di noci: 98%. Crepe: Modelli FE accuratezza 91,61-94,46%	[46], [47], [48], [49], [50], [51], [52]
Valutazione globale della qualità estetica	Prodotti lavorati	Latte in polvere, pane, taralli	Idoneità alla vendita e quantità complessiva del prodotto	Machine vision, machine learning	Precision, recall, F1, mAP@50, mAP@50-95, confusion matrix.	[20], [23]
	Materie prime	Coltivazione di pomacee, Coltivazione di altri alberi	Maturazione (Crudo, Parzialmente maturo, Maturo,	CNN	91,26%, Distinzione/Grado:Accuratezza Test 100% (ACM), 99,89% (AGM);	[53], [54], [55]

	da frutta, frutti di bosco e in guscio	Decomposto), Distinzione Mela/Non-Mela, Grado. Varietà: (Gala, Cortland, Pink Lady, Honeycrisp, Fuji)	Accuratezza Validazione 94,47% (ACM), 99,79% (AGM)

Tabella 12 - Tabella confronto materie prime e prodotti lavorati

Caso applicativo

Il lavoro di tesi è stato svolto in ambito applicativo, scegliendo sulla base degli studi evidenziati differenti tecniche e algoritmi che sono state successivamente scelte e testate. Questo capitolo si occupa della descrizione della metodologia applicata allo studio sperimentale e ai risultati ottenuti.

Metodologia sperimentale

L'obiettivo di questo sottocapitolo è descrivere la metodologia con la quale è stata affrontata la parte sperimentale di tesi, la scelta degli algoritmi e dei criteri utilizzati per una successiva valutazione degli output.

La strategia di applicazione è stata progettata per costruire una valutazione obiettiva sul confronto dei tre algoritmi selezionati (YOLOv5, YOLOv8 e YOLOv11), attraverso l'utilizzo di metriche tecniche, analisi della robustezza e l'interpretazione di KPI gestionali.

Successivamente alla selezione degli algoritmi è stata effettuata la scelta riguardo il campione di analisi utilizzato per il training, la validation e la fase di testing, paragonandoli non solo attraverso una versione standard di un dataset base, ma anche attraverso una versione "stressata" che è stata ottenuta applicando modifiche al dataset inziale, attraverso azioni di *data augmentation* specifiche per il raggiungimento dell'obiettivo [56].

La metodologia sperimentale si articola dunque in quattro fasi principali:

1. Scelta degli algoritmi per il confronto: decisione basata sulla possibile applicazione in ambito PMI, considerando i risultati ottenuti durante l'assessment dello studio dell'arte e paragonandoli con l'ultima versione ufficiale del modello scelto.
2. Costruzione e preparazione del dataset: il processo di selezione di un dataset open source online e i criteri di scelta ad esso associati, nonché la preparazione in seconda fase delle immagini in esso contenute.
3. Definizione delle metriche di valutazione: questi criteri sono comprensivi di metriche operative come il mAP, la Precisione, Recall e la latenza totale, ma anche contenenti KPI gestionali, al fine di valutarne l'applicabilità e la realizzazione in ambito industriale.
4. Analisi della robustezza dei modelli evidenziati attraverso l'applicazione del metodo statistico Bootstrap.

L'applicazione di questa metodologia consente uno scenario complessivo sull'affidabilità e la robustezza dei modelli, permettendone un paragone scientifico e oggettivo.

Scelta dei modelli

Per la scelta degli algoritmi da inserire all'interno dello studio pratico si è fatto riferimento agli articoli scientifici considerati all'interno dello studio bibliografico e riportati nella tabella 13.

Categoria	Tecnologia	Task	Carico computazionale	Vantaggi	Svantaggi	Autori
CNN	VGG16	Transfer learning e fine-tuning con image augmentation. Ottimizzare performance dei mobile e embedded device.	Training time: 42.13 minuti per 30 epoche. Single GPU hardware.	- Architettura consolidata - Utile per applicazioni di classificazione e pre-processing	- Elevato consumo di memoria e tempo di addestramento - Impatto significativo sui costi computazionali e operativi	[17], [23]
	YOLOv3	Edge computing evironment, object detection.	GPU NVIDIA RTX 2080 super e 64 GB running memory.	- Rilevamento in tempo reale buona velocità - Adatto a embedded system e contesti produttivi a vincolo di latenza	- Ridotta accuratezza nella rilevazione di oggetti piccoli o sovrapposti - Possibile compromissione delle prestazioni in ambienti ad alta densità visiva	[27]
	YOLOv5	Object detection e identificazione corpi estranei.	NA	- Equilibrio tra accuratezza e velocità - Ampia documentazione - Supporto su hardware GPU mainstream	- Prestazioni subottimali su dataset complessi e realistici rispetto a versioni più recenti - Ridotta scalabilità in ambienti industriali ad alta variabilità visiva	[18]
	YOLOv7	Object detection e identificazione	10,9 ms e utilizzo di 73,1 Mb	- Elevata efficienza su GPU	- Maggiore complessità architetturale	[24]

		qualità estetica del prodotto.		- Migliorata capacità di rilevamento rispetto alle versioni precedenti - Adatto a contesti di produzione automatizzata	- Richiede infrastruttura hardware dedicata e personale tecnico con competenze specifiche	
	YOLOv8	Object detection e identificazione stato di cottura attraverso browning stages, e hold-out metodo	NA	- Integrazione nativa di detection, segmentation e classification - Flessibilità applicativa - Interfaccia user-friendly	- Necessita di fase di tuning avanzato - Richiede dataset etichettati di alta qualità e pipeline di validazione ben strutturate	[13]
Classificatore tradizione	SVM	Tecniche di segmentazione	Le performance e il relativo carico computazionale sono stati osservati da Gennaio 2020 a Marzo 2021.	- Ottime prestazioni su piccoli dataset - Efficace in problemi binari	- Scalabilità limitata - Sensibilità alla scelta del kernel e dei parametri - Complessità nel tuning manuale su dataset industriali ad alta dimensionalità	[22]
Machine Learning	Random Forest, KNN	Misurazione del volume del pane, del colore e analisi statistica.	Arduino Uno hardware	- Integrazione diretta con macchine industriali - Applicabilità elevata in ambiti di ispezione visiva automatica e controllo qualità	- Elevata sensibilità a condizioni ambientali - Richiede calibrazione costante e ambienti controllati per garantire la ripetibilità dei risultati	[2], [25]

Tabella 13 - Suddivisione ricerca bibliografica basata sulla classificazione degli algoritmi utilizzati

La scelta di utilizzare come framework principale YOLO è data dalla presenza sostanziale dell'algoritmo rispetto alle altre categorie analizzate, e dalla possibilità di poterlo implementare in un

contesto produttivo utilizzando un meccanismo di nastro trasportatore e di telecamera fissa che scorra i framework ad esso associati.

YOLO (You Only Look Once) è un algoritmo di *object detection* il cui nome si riferisce al fatto di essere in grado di compiere diverse *detection task* utilizzando esclusivamente un network, a differenza degli approcci precedenti che nella maggior parte dei casi utilizzano una *sliding window* e un *classificatore* [26].

Per questo motivo il framework YOLO risulta essere tra i più innovativi in ambito di *Real-time object detection*, punto importante nel contesto applicativo a cui questa tesi fa riferimento: il controllo qualità nei processi produttivi necessita di algoritmi con bassa latenza totale e architetture di rete adattabili in ambiti di produzione [57], [58], [59].

La scelta tra i diversi modelli di YOLO per il confronto è stata compiuta considerano il contesto aziendale a cui si voleva fare riferimento: Piccole o Medie Imprese (PMI) possedenti dunque infrastrutture hardware non eccessivamente potenti, né la possibilità di avere un gruppo tecnico dedicato al continuo adattamento del fine-tuning; si è inoltre considerata una linea mono produttiva, contente un singolo prodotto (alta omogeneità) e dataset non considerevolmente numerosi.

Criterio di selezione	**Descrizione e motivazione (orientata al contesto PMI)**	**Modelli esclusi/inclusi**
Supporto e stabilità del framework	Le PMI necessitano di soluzioni affidabili per mantenere un livello di manutenzione adeguato senza costi aggiuntivi.	Escluso: YOLOv7 non più mantenuto dal framework Ultralytics.
Compatibilità con risorse computazionali limitate	La selezione di modelli leggeri e addestrabili su GPU di potenza media permette la possibilità di integrazione anche ad aziende di Piccola o Media dimensione.	Esclusi: YOLOv9 e YOLOv10 risultano eccessivamente onerosi.
Rappresentazione dell'evoluzione architetturale	Campione ampio di evoluzione al fine di valutarne come essa incida sulle metriche di performance.	Escluso: YOLOv3 versione obsoleta Incluso: YOLOv5 e YOLOv11
Equilibrio tra accuratezza e latenza	Nei contesti di produzione reale a cui si vuole applicare lo studio, risulta cruciale testare un modello che abbia la latenza applicabile in linea produttiva e l'accuratezza necessaria al fine di non generare costi eccessivi.	Incluso: YOLOv8

Tabella 14 - Criteri di selezione delle versioni YOLO

La scelta, applicando i parametri di esclusione e inclusione rappresentati in tabella 14, è dunque ricaduta su YOLOv5, YOLOv8 e YOLOv11: i modelli risultano user-friendly e senza bisogno di elevate GPU su cui poter funzionare, con alta accuratezza e aventi un'avanzata possibilità applicativa [13], [18].

Come citato nella "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS" [26] YOLOv5 è stato sviluppato da *Ultralytics* nel 2020, ed è costituito da una struttura modulare scritta interamente in *PyTorch*, che lo rende adattabile anche all'essere runnato su tool collaborativi online come Google Colab. È inoltre un framework open-source, costantemente aggiornato e documentato, conforme dunque ad applicazioni industriali con risorse limitate e necessità di rapido deployment.

Scendendo nel dettaglio la versione 5 di YOLO presenta un backbone *CSPDarknet53* modificato con un layer *stem* per l'estrazione multi-scala efficiente e un neck *CSP-Pan* migliore rispetto alla versione dell'algoritmo precedente. [26]

Il secondo algoritmo selezionato per il confronto è YOLOv8 [13], [26], rilasciato nel 2023 dalla stessa *Ultralytics*, rappresenta un'evoluzione significativa della serie: presenta un'architettura di rete più flessibile, maggiore modularità e supporto ai task multipli quali *object detection, instance segmentation, pose estimation, tracking e classification.*

L'architettura mantiene il backbone dell'analoga versione cinque, sostituendo però i CSP-Pan con un nuovo modulo *C2f* (Cross-Stage Partial Bottleneck with two convolutions) il quale combina informazioni contestuali con caratteristiche di alto livello, migliorandone in tale modo la capacità di rilevazione dei difetti, soprattutto in condizioni luminose non eccellenti o parziali occlusioni [26].

Un'innovazione rilevante di questo aggiornamento è l'adozione di un modello anchor-free in cui le tre componenti principali, *objectness, classification e regression,* vengono gestite in maniera indipendente; questa separazione consente la divisione tra gli obiettivi, migliora l'accuratezza e la previsione collettiva. È importante considerare come l'architettura di YOLOv8 risulti più leggera e implementabile delle precedenti versioni, ma esso richiede maggiori risorse computazionali, specialmente nella versione "x" del modello, il che potrebbe essere un parametro da considerare per l'implementazione in ambienti altamente vincolati [29], [60].

Entrambe le versioni sono state sottoposte al paragone attraverso la pagina ufficiale del framework di pubblicazione, che evidenzia come l'area di interesse dei modelli ricada nell'ambito dell'automazione e controllo qualità nell'automazione [29].

Sebbene i modelli scelti siano stati pubblicati rispettivamente nel 2020 e 2023, al fine di raggiungere uno studio integrante i modelli più recenti presenti sul mercato, si è deciso di implementare il confronto utilizzando anche YOLOv11.

Essendo l'ultima versione rilasciata da Ultralytics il modello non è risultato tra le opzioni analizzate nei paper scientifici, ma è descritto in maniera minuziosa nella repository ufficiale [29].
Esso impiega un'architettura backbone e neck migliorata, introduce inoltre pipeline ottimizzate, le quali ottimizzano i tempi di latenza, ottenendo comunque un costante equilibrio tra precisione e prestazioni [29].

I tre modelli selezionati rispondono dunque alle esigenze delle PMI attraverso la loro leggerezza di architettura, rapidità di inferenza e disponibilità open-source: YOLOv5 rappresenta una soluzione consolidata, YOLOv8 offre una versione più recente, ma studiata e già integrata nella letteratura, mentre YOLOv11 dona un nuovo punto di vista e ricerca all'interno della metodologia sperimentale. I modelli sono stati successivamente testati e confrontati su due dataset differenti: il primo in versione base, il secondo modificato al fine di osservare gli algoritmi in condizioni di stress.

Scelta e costruzione del dataset

La qualità del dataset è essenziale per un lavoro di ricerca su algoritmi di visione artificiale, specialmente nella fase di training per l'*object detection*: i framework YOLO necessitano di immagini annotati con *bounding box* e di un dataset abbastanza consolidato da poterne permettere l'addestramento, la fase di validazione e il testing.
Per il seguente lavoro il dataset è stato estratto da *Roboflow* [61] [62] ed implementato manualmente attraverso la fase di etichettatura manuale delle classi di interesse, preprocessing e data augmentation. Successivamente si è compiuta la riorganizzazione in due versioni: *normale* e *stressata*, al fine di valutarne la robustezza e la generalizzazione dei modelli.

Il dataset di partenza è stato scelto attraverso una ricerca online tra le maggiori open-source repository - *Github, Kaggle e Roboflow* - utilizzando una stringa di ricerca composta da parole generiche quali: *"defect + cookies + dataset"*.
La selezione dei dataset ha seguito i seguenti parametri di esclusione:

- Dataset non contenti una licenza di tipologia C pubblica, dunque non replicabili.
- Dataset con numero non significativo di immagini totali (minori di 450).
- Dataset con immagini non frame di nastro trasportatori o repliche.
- Dataset che potessero contenere più di una classe difettosa.
- Dataset compatibili con l'esportazione in un formato non adatto all'utilizzo di YOLO.

Attraverso questa ricerca la fase di selezione si è conclusa con la scelta del dataset "Coookie Defect Detection Dataset" [62] su Roboflow [61].

Esso presenta una licenza di tipo CC BY 4.0 [63], che dunque è possibile condividere e adattare seguendo alcune specifiche come: condividere il link del dataset originale con la relativa licenza e la limitazione di non poter cambiare la tipologia di licenza rispetto l'originale.

Il dataset scelto presenta immagini di cracker sopra una replica di nastro trasportatore e con immagini composte da snapshot di un video, adatto dunque al contesto applicativo di ricerca.
Il dataset originale contiene 468 immagini suddivise in train (82%), validation (12%) e test (6%), parametri settati automaticamente da Roboflow, e due classi già etichettate composte da cracker *Normal* e cracker *Cracked*.
Successivamente alla decisione di adozione del dataset è stata effettuata una clonazione dell'originale e sono stati applicati due passaggi fondamentali: il controllo delle labels già inserite dal costruttore del set di dati, al fine di visionare che fossero correttamente disposte, e l'aggiunta dell'etichetta della classe *Over baked*; ciò ha permesso di completare le tre labels di interesse: *Normal, Over baked e Cracked,* così da poterne identificare e implementare uno studio multi-causale.

Al fine di creare una versione base di partenza dei dati, che fosse tuttavia consistente per ottenerne una versione robusta del modello, sono state applicate delle tecniche base di preprocessing e data augmentation, con l'unico intento di aumentare la numerosità del campione e cercare di raggiungere un maggiore bilanciamento delle classi [56].
Le operazioni effettuate includono un auto-orienting, applicato per correggere l'orientamento dell'immagine, resizing a 640x640 tramite stretching, e flip orizzontale progettato per generare tre output per ogni esempio di training [61].
A seguito di questa procedura si è ottenuto un dataset che viene definito al fine del lavoro di tesi "*normale*", cioè senza particolari modifiche che potrebbero causare stress in fase di addestramento, avente 658 immagini suddivise in train/validation/set con percentuali rispettivamente del: 82%, 12% e 6%; la percentuale elevata del modello di training permette un allenamento con minore rischio di underfitting, mentre il 12 % rappresenta la possibilità di ottimizzare gli iper-parametri e monitorare l'overfitting in fase di validazione. In ultima istanza il test set è utilizzato per valutare il modello su dati mai visti, senza però sottrarre immagini all'addestramento.
Per valutare la robustezza degli algoritmi e la capacità teorica in una linea produttiva si è inoltre creato un secondo dataset attraverso l'applicazione di tecniche di data augmentation più aggressive [56], generandone così una versione definibile "stressata".
Queste variazioni mirano a simulare possibili scenari realistici in sede di produzione e difetti di acquisizione di immagini, comuni in ambienti industriali; inoltre, la maggiore applicazione di data

augmentation fornisce un dataset di training più elevato in numerosità, così da poter effettivamente controllare la variazione tra i modelli, anche in funzione della differenza di campioni.
La versione stressata del dataset comprende 1045 immagini, generate dalle seguenti applicazioni [64]:

- Preprocessing
    - Auto-orient: correzione automatica dell'orientamento delle immagini al fine di creare coerenza con il modello nell versione base e non generare cambiamenti che potrebbero invalidare i dati.
    - Resize: ridimensionamento delle immagini a 640x640 pixel tramite stretching per uniformarne le caratteristiche
- Data Augmentation
    - Flip orizzontale: per aumentare la variabilità
    - Shear: deformazioni applicate orizzontalmente e verticalmente di dieci gradi al fine di simulare possibili frame non centrati, inclinati o distorti acquisiti in sede di produzione.
    - Blur: aggiunta di sfocatura di 4,3 pixel per simulare immagini sfocate o di bassa qualità, caso possibile nel contesto di PMI in cui la tesi si presuppone di operare

Queste tecniche sono state applicate per migliorare la capacità del modello nell'affrontare scenari non ideali, sia dal punto di vista di qualità delle immagini, che dalla latenza differente che potrebbero avere diverse linee di produzione.

Dati	Normale	Stressato	Motivazioni
Numero di immagini totale	658	1045	La versione stressata del modello contiene più immagini per aumentare la variabilità del dataset e testare la robustezza dei modelli in scenari più complessi.
Split in train/validation/test	537/81/40	924/81/40	Lo split cambia in percentuale per ottenere dei risultati comparabili tra validation e test, e vedere la differenza di qualità nell'allenamento attraverso una numerosità differente.
Preprocessing	Auto-Orient: Applied	Auto-Orient: Applied	Correzione dell'orientamento per garantire che tutte le immagini abbiano una base

			coerente, evitando problemi di allineamento per il momento.
	Resize: Stretch to 640x640	Resize: Stretch to 640x640	Uniforma la dimensione delle immagini per facilitarne il training su YOLO.
Data augmentation	Flip: Horizontal	Flip: Horizontal Shear: ±10° Horizontal, ±10° Vertical Blur: Up to 4.3px	La versione stressata applica shear e blur per simulare distorsioni e sfocature tipiche di una linea di produzione reale.

Tabella 15 - Caratteristiche delle versioni del dataset utilizzato

La creazione di una doppia versione di dataset – *normale e stressata* – permette di valutare e paragonare non solo la robustezza di un determinato modello, ma anche la capacità di generalizzazione in scenari più realistici e complessi.

Il dataset normale fornisce un termine di paragone ideale, in cui le immagini vengono catturate sotto condizioni standard, mentre il dataset stressato mostra una versione realistica con cambi di frame non ideali, inclinazione della telecamera ed eventuale malfunzionamento della cattura dell'immagine.

Questo approccio permette di valutare i tre modelli e di collegare le performance tecniche a implicazioni operative e gestionali, attraverso l'utilizzo di metriche distinte e KPI aggiornati.

Setup training

In questo sottocapitolo vengono presentate le caratteristiche tecniche delle impostazioni adottate per l'addestramento, con focus sull'ambiente di sviluppo, gli strumenti, la scelta e la configurazione degli iper-parametri e le tecniche di addestramento utilizzate.

L'intero processo di addestramento e delle successive fasi è stato svolto sull'ambiente virtuale Google Colab, piattaforma cloud che permette l'utilizzo di GPU per l'esecuzione di notebook Jupiter in Python; l'ambiente virtuale ha dunque utilizzato una GPU T4 NVIDIA 550.54.15 con 16 GB di memoria, avente una CPU Intel Xeon a 2,20 GHz e 12 GB di RAM.

La GPU è stata gestita da CUDA 12.4, garantendo così una velocità computazionale adatta a dataset medi e hardware disponibili.

Lato software il framework utilizzato è stato YOLO, implementato della versione 3.12.12 di Python, basato su PyTorch 2.8.0+cu126 come libreria principale per il calcolo tensoriale e la gestione del training su GPU.

Gli strumenti di supporto utilizzati sono stati librerie Roboflow per la gestione dei dati, NumPy versione 2.0.2 per l'elaborazione delle immagini, Matplotlib 3.10.0 per la visualizzazione dei risultati e TensorBoard per la registrazione e l'analisi delle loss del modello.

Nell'ambiente virtuale è inoltre stato montato il Google Drive personale, al fine di salvare i diversi checkpoint di addestramento, i pesi migliori di ogni epoca, così da poter fermare e far ripartire il processo in ogni momento, e gli output evidenziati.
Gli iper-parametri fissati per l'addestramento del modello sono stati scelti in funzione degli studi analizzati [13], [24], [26], [64] e sullo studio comparativo presente nella repository Ultralytics [60], e sono riportati in tabella 16.

Iper parametro	Valore	Descrizione	Fonti prese come riferimento
Learning Rate	0,01	Tasso di aggiornamento dei pesi fissato di default per framework YOLO.	[24]
Ottimizzatore	Adam	Ottimizzatore per aggiornamento pesi durante la fase di training.	[29]
Batch	8	Numero di campioni per l'aggiornamento, scelto attraverso la capacità della GPU disponibile e dalla stabilità delle stime dei gredienti.	[29]
Epoche	100	Scelta data per consentire al modello di convergere senza rischio di underfitting.	[20]
Weight Decay	0,0005	Riduce rischio di underfitting e consente la regolarizzazione dei pesi.	[24]
IoU threshold	0,5	Soglia per la valutazione delle bounding box.	[24]

Tabella 16 - Iper-parametri fase di addestramento dei modelli

Settati questi iper-parametri durante la fase di training il modello prosegue ad attuare alcuni step per completare il ciclo di addestramento: la prima fase è il *forward pass,* il modello analizza il batch e ne produce le proprie predizioni, per poi passare alla seconda fase di *backpropagation*, in cui viene calcolata la loss – differenza tra Ground Truth e predizioni - e il gradiente rispetto ai pesi valutati viene propagato all'indietro nella rete neurale.
L'ultimo step consiste nel salvare i pesi migliori, calcolati tramite l'ottimizzatore Adam e valutati attraversi il learning rate, che vengono immagazzinati nel Content del Drive, sotto il nome di *best.pt* e *last.pt*; il checkpoint con migliore mAP e metriche viene salvato come il principale e inserito come output del modello.

In conclusione, l’intero processo di addestramento ha seguito regole standard e di default del framework, affidandosi agli studi precedentemente analizzati, attribuendo modifiche minimali volte all’adattamento del modello alle risorse hardware disponibili. Questa scelta rende la ricerca riproducibile, trasparente e facilmente confrontabile tra le diverse versioni.

Metriche tecniche di valutazione

Un passaggio importante nella valutazione dei modelli di Machine Vision è l'inserimento delle giuste metriche di valutazione in modo tale che diano risalto alle caratteristiche dei modelli e facciano emergere eventuali punti critici o di miglioramento.
Per la valutazione dell'allenamento, validation e testing dei tre modelli utilizzati sono state applicate tre categorie di metriche: tecniche, di robustezza e gestionali.
La categoria di metriche tecniche permette di inserire i diversi modelli all'interno di un contesto chiaro e oggettivo rispetto a ciò che comunemente è la norma: utilizzando termini di accuratezza e di precisione è possibile collocare in maniera oggettiva i risultati ottenuti dall'allenamento degli algoritmi e dai loro risultati di test.
Dall'altro lato è importante, al fine del lavoro compiuto, che le valutazioni ottenute ed eventuali conclusione tratte, siano inserite in un contesto gestionale, che ne permetta l'effettiva valutazione in un contesto produttivo e realistico. Per fare ciò si sono scelti alcuni KPI che diano informazioni riguardo l'allocazione di questi modelli rispetto a tre impatti: *economico*, *organizzativo* e *ambientale*.
L'analisi della robustezza è stata inoltre inserita al fine di dare maggiore credibilità alla ricerca, donando dati statistici e un'analisi integrata.

Secondo Ultralytics [29] la scelta delle metriche è essenziale per la valutazione della *object detection* del modello, in quanto esse identificano la capacità di un modello nel predire la classe desiderata e come si occupa della gestione dei falsi positivi e negativi.
Per questa motivazione le metriche tecniche scelte e implementate per la valutazione dei modelli durante la fase di *training* sono le seguenti e sono state scelte analizzando le scelte compiute in letteratura ove applicati casi paragonabili [64]:

- **Precisione**: indica la percentuale di predizioni positive corrette, più il valore tenderà ad uno meno saranno i casi in cui il modello sbaglierà l'interpretazione. Viene calcolata come:

$$\frac{TP}{TP + FP}$$

- **Recall**: misura la percentuale di oggetti correttamente individuati sul totale reale; maggiore è il valore, minore sarà la presenza di falsi negativi predetti nel modello. Si calcola come:

$$\frac{TP}{TP + FN}$$

- **mAP@50**: Mean Average Precision avente soglia IoU dello 0,5. Sintetizza la precisione del modello su tutte le classi, considerando la soglia del 50%: è una misura di accuratezza del modello considerando esclusivamente le "easy" detections [29].

- **mAP@50-95**: Mean Average Precision avente soglia IoU che varia tra lo 0.5 e lo 0.95. Dona una visione completa della performance del modello su differenti livelli di difficoltà di detection [29].
- **Speed metrics**: queste misure sono critiche nel contesto di real-time object detection, questa sezione identifica tutti i diversi livelli che formano la latenza totale nel validation process, considerando sia il tempo di preprocessamento che quello di post.
  La latenza totale risulta dunque essere la somma di: *Preprocess* + *Inference* + *NMS.*
  Preprocess: tempo di preprocessamento dell'immagine, per prepararla alla predizione
  Inference: tempo effettivo che la rete neurale impiega per predire l'immagine
  NMS (Non-Max Suppression): tempo che la rete neurale impiega per eliminare le doppie predizioni e rifinire i bounding box.

Al fine della visualizzazione pratica dei risultati raggiunti in fase di *validation*, sono inoltre riportate come output di ricerca le matrici di confusione associate ai diversi modelli e le diverse versioni di dataset, in versione normalizzata per facilitarne l'impatto visivo.
La matrice di confusione appare diversa rispetto a un problema mono classe, siccome la presenza di tre classi fornisce una matrice di larghezza e lunghezza 3x3, questo comporta una precisazione importante: ogni classe può essere considerata a turno come classe positiva, mentre le altre risulteranno le negative [23].

Predicted \ True	Normal	Over baked	Cracked
Normal	TP	FP/FN	FP/FN
Over baked	FP/FN	TP	FP/FN
Cracked	FP/FN	FP/FN	TP

Tabella 17 - Descrizione matrice di confusione

La matrice rappresenta il confronto diretto tra le classi True, reali, e le classi predette (Predicted) dal modello, consentendo così di valutarne in maniera più precisa i risultati [13].
Ogni cella della diagonale principale rappresenta le classificazioni predette correttamente, identificate come TP (*True Positives*); fuori dalla diagonale sono presenti i FP (*False Positives*) e, in relazione inversa, i FN (*False Negatives*). I componenti identificati come TN (*True Negatives*) non vengono direttamente identificati all'interno di una matrice multiclasse, ma vengono calcolati indirettamente attraverso l'elaborazione delle metriche [23].
La matrice di confusione permette inoltre di valutare i due errori statistici presenti nel processo decisionale: gli errori identificati come Alpha e Beta. L'errore di tipo Alpha, anche detto FP, identifica

l'incapacità del modello nell'identificare un difetto, quando il prodotto risulta invece perfettamente conforme: questo comporta un aumento di sprechi non necessari in linea di produzione.
La seconda tipologia rientra nel Beta-errore, False Negative, che consiste nel caso in cui il sistema non identifica un errore, invece presente nella realtà, esso ha implicazioni industriali importanti, in quanto riduce la qualità dei prodotti considerati conformi.

Oltre la rappresentazione delle matrici di confusione, i modelli YOLO permettono l'utilizzo di output grafici rappresentanti le curve di Precision-Recall compiute in fase di validazione, in modo da osservare il comportamento del modello rispetto il trade-off tra precisione e sensibilità, attraverso una parte di dataset non precedentemente analizzata.
L'aerea sotto la curva identifica l'*Average Precision* (AP) del modello in fase di *validation*, identificandone anche le diverse classi di studio, donando così una visione totale dei differenti comportamenti [29]:

- AP = 1: classificazione quasi perfetta
- AP<0.5: scarsa capacità interpretativa
- mAP: media della precisione su tutte le classi considerate

In allegato inoltre vengono riportati gli output grafici salvati dai differenti modelli YOLO: i risultati visivi dell'andamento dell'addestramento, le curve Recall-Confidence e Precision-Confidence e in ultima istanza le curve F1-Confidence.
La fase di *testing*, successiva alla validation, viene valutata visivamente attraverso una tabella contenente le immagini di output predette dai diversi modelli, riportando tre casistiche e i relativi errori di predizione.

Analisi robustezza

Al fine di stimare l'incertezza dei modelli scelti per la valutazione, e valutarne complessivamente la robustezza, l'analisi delle metriche tecniche non risulta sufficiente se non accorpata ad un'analisi statistica completa, che fornisce parametri maggiormente puntuali sui modelli.

Il metodo scelto al fine di questa valutazione risulta essere Bootstrap, il quale emerge come tecnica potente per stimare gli intervalli di confidenza (IC), senza fare assunzioni restrittive sulla distribuzione dei dati [65].

Efron [65] sostiene come questa metodologia utilizzi tecniche di re-sampling al fine di generare nuovi campioni estraendo casualmente dal campione originale, andando poi a reinserirle all'interno senza manomissioni.

La procedura dei calcoli degli intervalli di confidenza secondo Bootstrap segue tre tecniche principali [65]:

1. Re-sampling: generazione di B campioni Bootstrap con annesso reinserimento
2. Stima del parametro: calcolo del parametro di interesse per ogni campion
3. Costruzione dell'intervallo di confidenza: derivazione degli intervalli attraverso la distribuzione empirica dei parametri stimati.

Prima di effettuare l'analisi, e per valutare le prestazioni dei differenti modelli di YOLO, il codice utilizzato (Allegato E) viene strutturato attraverso una procedura in più fasi; in primo luogo, il modello viene testato sul dataset di validazione, utilizzando differenti soglie di confidenza (0.25,0.5,0.75), in modo tale da poterne osservare i possibili cambiamenti rispetto alle diverse soglie di accettazione.

I risultati di questa prima analisi vengono successivamente salvati in un file formato JSON, che contiene informazioni su ogni predizione compiuta, da queste immagini viene dunque calcolato il numero dei rilevamenti e la media dei punteggi di confidenza, che vengono studiati dalla metodologia Bootstrap per valutarne la robustezza.

Attraverso l'analisi è dunque possibile avere distribuzioni empiriche, da cui vengono generati IC di 95%, permettendo di valutare anche la variabilità statistica delle stime effettuate.

I risultati raccolti per ogni modello e ciascuna sua versione vengono raggruppati e commentati in una tabella comparativa nell'analisi degli output sperimentali.

Metriche gestionali di valutazione

Nel contesto di questa tesi di ricerca l'analisi dei risultati tecnici del modello è rilevante tanto quanto l'interpretazione del progetto in ambito aziendale e produttivo; risulta dunque essenziale definire metriche gestionali, come KPI specifici, per introdurre un connubio tra i due aspetti.

Le metriche gestionali considerate vogliono introdurre una visione più ampia e applicativa dell'utilizzo del modello, inserendolo in tre diversi contesti e impatti: economico, organizzativo e ambientale.
Al fine di una analisi realistica i modelli considerati per questa parte di analisi sono esclusivamente le prove generate dal dataset in versione stressata, così da valutarne l'impatto gestionale all'interno di una eventuale linea di produzione, contenente problematiche simili a quelle ricreate. Ogni contesto individuato è associato a uno specifico KPI, il primo ad essere analizzato è l'impatto economico.
Per valutare l'impatto economico del modello la metrica utilizzata è la Cost Saving Ratio [66]:

$$CSR = \frac{C_{YOLOvbaseline\_stressato} - C_{YOLOvy\_stressato}}{C_{YOLOvbaseline\_stressato}} * 100\%$$

dove:

- $C_{YOLOvbaseline\_stressato}$ è il costo totale (€) di scarti nel modello utilizzato come termine di paragone nella versione stressata:

$$C_{YOLOvbaseline\_stressato} = \left(FP_{YOLOvbaseline\_stressato} + FN_{YOLOvbaseline\_stressato}\right) * \ C_{unitario}$$

- $C_{YOLOvy\_stressato}$ è il costo totale (€) di scarti del modello nella versione stressata che si vuole valutare:

$$C_{YOLOvy\_stressato} = \left(FP_{YOLOvystressato} + FN_{YOLOvy\_stressato}\right) * \ C_{unitario}$$

$C_{unitario}$, ovvero il costo unitario di produzione, viene ipotizzato a €0,30 in quanto non disponibile per il dataset scelto. Sebbene i valori economici siano simulati al solo scopo comparativo, la metodologia scelta segue quella adottata da Wuest et al. in "Machine learning in manufacturing" [67][59].

Al fine di compiere una analisi completa dei costi associati ai modelli si è inoltre implementata una logica di bundle, provata all'interno dei modelli per analizzarne la possibile capacità di risparmio sia in termini economici che di sprechi alimentari (Allegato F).

Il secondo impatto analizzato attraverso metriche gestionali è l'impatto organizzativo: esso si può misurare attraverso il throughput teorico (frame/s) il quale è calcolato come

$$TR = \frac{1}{L} \quad dove\ L = latenza$$

Questa metrica è importante per stabilire se la latenza del modello compie l'azione di un collo di bottiglia, e blocca il normale sviluppo della produzione: se una linea produttiva produce ad una velocità V, il modello dovrà essere $TR \geq V$, altrimenti risulterà essere collo di bottiglia. È documentato come modelli più rapidi abbiano troughput più alti, e come misurare la latenza sia essenziale per il contesto reale in cui si vuole porre il modello [68].

Il terzo impatto è l'ambientale, punto essenziale nel discorso di riduzione degli sprechi sia per una logica di rifiuti non necessari, che per una questione economica dal lato aziendale.

Il KPI scelto per affrontare questo impatto è il Waste Reduction Index (WRI)[69], che si occupa del calcolare la riduzione relativa degli sprechi di produzione derivanti dal miglioramento del modello, utilizzando i FP interpretandoli come scarti non necessari.

La formula utilizzata è la seguente:

$$WRI = \frac{FP_{YOLOvbaseline\_stressato} - FP_{YOLOvy\_stressato}}{FP_{YOLOvbaseline\_stressato}} * 100\%$$

dove:

- $FP_{YOLOvbaseline\_stressato}$ sono i falsi positivi nel modello versione stressata del primo algoritmo di paragone.
- $FP_{YOLOvy\_stressato}$ sono i falsi positivi nel modello versione stressata del secondo algoritmo paragonato.

La percentuale dimostra dunque l'aumento o la diminuzione relativa degli sprechi appartenente al miglioramento del modello. Questo tipo di indicatore ha inoltre un impatto importante su contesti manufatturieri e PMI, dove la riduzione degli scarti può avere un forte impatto sui costi di non qualità e sulla sostenibilità ambientale.

A sostegno del Waste reduction index e per analizzare in maniera più precisa gli scarti effettivi di produzione, si è inoltre introdotto e commentato l'Indice ambientale % [69] calcolato come:

$$Indice\ Ambientale\ \% \ = \ \frac{(FP + FN)}{TP + FN} \cdot 100\%$$

Esso permette di calcolare sia la sostenibilità produttiva, che l'efficienza economica, normalizzando l'impatto degli sprechi rispetto alla totalità dei prodotti, evidenziando quale modello è consigliato per la riduzione degli scarti, controllo dei difetti e sostenibilità operativa.
A sostegno del KPI per la categoria di impatto ambientale, all'interno della tesi si è voluta implementare una logica bundle precedentemente citata, che offre la capacità di testare l'algoritmo nell'identificare la possibilità di creare diverse tipologie di batch, aventi diverse caratteristiche, tutte volte alla diminuzione degli sprechi, e dei conseguenti costi.

Questi parametri teorici sono stati applicati alle diverse fasi del modello, evidenziandone i risultati di output durante il *training, la validation e il testing.*

Analisi output caso sperimentale

Questa parte di tesi si pone l'obiettivo di studiare e riportare gli output della ricerca sperimentale, confrontando le diverse versioni dei modelli e identificandone i miglioramenti e le possibili implementazioni future.

Il sottocapitolo si suddivide in una prima parte di analisi tecnica attraverso risultati numerici delle metriche, matrici di confusione e curve Precision-Recall, concludendo con un'analisi visiva degli output generati. La seconda parte si occupa di analizzare gli output generati dall'analisi della robustezza Bootstrap, mentre la terza e ultima parte presenta l'analisi delle metriche gestionali e un'ipotesi di possibili implementazioni in un contesto di Piccole Medie Imprese.

Analisi output metriche tecniche

Le metriche fornite dai modelli come output sono classificate paragonando sulla base del modello e della versione, è possibile inoltre notare in tabella 18 il modello maggiormente efficiente, evidenziato in grassetto.

Modello	**Scenario**	**Precision (P)**	**Recall (R)**	**mAP@50**	**mAP@50-95**	**Latenza Totale (ms)**
YOLOv5	Normale	0.903	0.968	0.975	0.958	24.4
	Stress	0.919	0.937	0.972	0.953	24.5
YOLOv8	Normale	0.881	0.942	0.974	0.953	24.2
	Stress	0.928	0.935	0.979	0.962	23.8
YOLOv11	Normale	0.910	0.954	0.971	0.959	19.7
	Stress	**0.945**	**0.934**	**0.978**	**0.961**	**15.7**

Tabella 18 - Confronto tra metriche tecniche principali, evidenziata in grassetto il miglior modello

Dalla tabella 18 emergono i risultati sperimentali dei modelli scelti: dai risultati riportati emerge in maniera evidente come tutti i modelli sostengano prestazioni elevate, sia in termini di Precisione e Recall, sia in termini di rapidità di elaborazione, riportando Speed Metrics adatte a un ambito produttivo industriale, dove emerge però la capacità migliore della versione 11 in condizioni di stress. Andando ad analizzare i modelli nelle loro versioni normali, YOLOv11 rispetto a YOLOv5, risulta avere sia Precision, mAP@50-95 che tempi di latenza lievemente migliori, il che indica una capacità maggiore nell'identificare correttamente tutti i cracker presenti nell'immagine; YOLOv8 risulta invece inferiore a entrambi sia nella Precision che per il Recall, dimostrando la minore capacità nell'inserimento corretto dei bounding box: i parametri numerici raggiunti dimostrano comunque buoni risultati, rimanendo in una fascia consona di livello numerico.

Il paragone sulla latenza totale dei tre algoritmi favorisce nettamente la versione di YOLO più moderna sia in versione normale che stressata: andando a verificare dove risiede questa differenza (tabella 19), si può notare come YOLOv5 abbia un tempo di Inference molto minore rispetto alla versione 8 del modello e come il gap con la versione 11 risieda specialmente nella fase di Inference rispetto al modello 8 e di postprocess rispetto alla versione 5.
I modelli YOLOv5 e YOLOv8 invece, risultano con un totale stimato che si discosta di pochi centesimi, anche se costruito in maniera molto differente nello specifico: la versione 5 ha un tempo di post-process che risulta essere più del doppio rispetto alla versione 8, che però impiega maggior tempo nella fase di Inference centrale.

Modello	Scenario	Preprocess	Inference	NMS / Postprocess	Totale stimato
YOLOv5	Normale	0.9 ms	7.7 ms	15.8 ms	24.4 ms
	Stress	0.9 ms	7.7 ms	15.9 ms	24.5 ms
YOLOv8	Normale	0.5 ms	17.6 ms	6.1 ms	24.2 ms
	Stress	4.8 ms	12.1 ms	6.9 ms	23.8 ms
YOLOv11	Normale	2.0 ms	8.0 ms	9.7 ms	19.7 ms
	Stress	1.2 ms	11.7ms	2.8 ms	15.7 ms

Tabella 19 - Speed metrics valutate suddivise tra i differenti modelli

In condizioni di stress il modello migliore risulta essere YOLOv11 che sembra avere una migliore capacità di adattamento, mostrando parametri migliori sia per Precision che mAP@50-95, che per la latenza totale utilizzata. YOLOv8 dimostra avere difficoltà nel training della versione normale, parametro in linea con le aspettative, il modello infatti dei requisiti di tuning degli iper-parametri elevati, probabilmente non rispettati nel dataset normale utilizzato. Questa combinazione di cause ha portato a un fenomeno di underfitting del modello, con una capacità maggiormente ridotta nell'identificare le classi, che si riflettte in una minore precisione. Questa considerazione è inutile in un contesto di decision-making sull'algoritmo da utilizzare: le aziende vorranno algoritmi che non richiedano tempo e costi maggiori per settare e raffinare.
Complessivamente YOLOv11 nella versione stressata risulta avere parametri migliori, dimostrando una solida robustezza del modello e una capacità adattiva a condizione di stress elevata, garantendo maggiore stabilità e resilienza in presenza di possibili perturbazioni; tuttavia, le performance non hanno un miglioramento netto nè esponenziale, se non nella latenza, dunque rimane importante considerare in maniera più attenta lo specifico utilizzo e l'ambito di applicazione, oltre a un'analisi di costo di implementazione.

Una volta identificate le metriche complessive dei diversi modelli e attribuite a ciascuna versione, è possibile analizzare più nello specifico le classi attribuite ai cracker presenti nello studio di ricerca:

- *Normal:* senza alcuna tipologia di difetti.
- *Cracked:* crepe sulla superficie del cracker.
- *Over baked:* presenza di zone di colore significativamente più scuro e non uniforme rispetto alla superficie normale del cracker.

Queste misure forniscono un'analisi più di dettaglio sui comportamenti dei modelli rispetto alle specifiche classi, evidenziandone i diversi atteggiamenti e alcune criticità.

Modello	Scenario	Classe	Precision	Recall	mAP@50	mAP50-95
YOLOv5	Normale	Normal	0.923	0.991	0.986	0.956
		Cracked	0.884	0.978	0.992	0.990
		Over baked	0.903	0.934	0.948	0.928
	Stressato	Normal	0.934	0.973	0.987	0.965
		Cracked	0.956	0.938	0.987	0.981
		Over baked	0.867	0.9	0.942	0.912
YOLOv8	Normale	Normal	0.908	0.969	0.985	0.969
		Cracked	0.894	0.978	0.990	0.982
		Over baked	0.841	0.880	0.936	0.909
	Stressato	Normal	0.960	0.943	0.988	0.970
		Cracked	0.939	0.862	0.966	0.940
		Over baked	0.885	1.000	0.992	0.976
YOLOv11	Normale	Normal	0.927	0.973	0.988	0.969
		Cracked	0.978	0.957	0.987	0.985
		Over baked	0.825	0.933	0.938	0.922
	Stressato	Normal	0.954	0.947	0.989	0.967
		Cracked	0.986	0.957	0.99	0.989
		Over baked	0.896	0.9	0.955	0.926

Tabella 20 - Metriche tecniche suddivise per classi, versioni e modelli

Dalla tabella 20 è possibile notare una classe con metriche inferiori alle altre: la *Over baked*.

Una possibile spiegazione a questa piccola differenza di parametri può essere attribuita al numero inferiore di immagini contenenti cracker bruciati rispetto a quelli rotti, il che va ad influire sull'allenamento del modello e dunque sulla predizione finale: la data augmentation è servita a irrobustire il modello complessivo, ma le classi denotavano uno sbilanciamento importante, che non è stato colmato a pieno.

Il criterio di bruciatura è, inoltre, un parametro molto complesso da descrivere e rilevare in maniera oggettiva, è possibile dunque che la scarsità di dati, unita alla poca oggettività di identificazione delle bounding box, abbia scaturito questo calo di efficienza nell'identificazione dei cracker *Over baked*.

Tuttavia, i parametri generali non risultano essere in una fascia critica per gli standard identificati tramite la ricerca bibliografica: gli scarti prodotti da questa classe sono maggiori rispetto alle altre studiate, ma non eccessivamente da causarne un calo economico importante.
In questo caso specifico il modello YOLOv5 nella versione normale ottiene il primato per robustezza e capacità di adattamento tra le diversi classi, mantenendo performance buone anche nella classe critica *Over baked.*
Considerando un livello di specificità maggiore è possibile analizzare i comportamenti dei modelli nell'identificazione dei TP (True Positives), FP (False Positives), FN (False Negatives) e TN (True Negatives), in modo tale da riuscire visivamente a stimare la correttezza del modello nell'identificazione degli oggetti, fornendo informazioni complementari rispetto alle metriche tecniche aggregate. Le matrici di confusione risultano normalizzate, essendo state prodotte direttamente dal framework Ultralytics [29].

Predicted \ True	Normal	Over baked	Cracked
Normal	0.97	0.20	0.02
Over baked	0.02	0.80	0
Cracked	0.01	0	0.98

Tabella 21 - Matrice di confusione normalizzata per il modello YOLOv5 versione normale

La versione normale del modello YOLOv5 presenta difficoltà nell'identificare la classe *Over baked* (tabella 21), questa tendenza è in linea con quanto identificato dalla tabella comprendente tutti i risultati delle metriche aggregate; è infatti possibile notare il gap tra l'identificazione dei cracker senza difetti o rotti, rispetto ai bruciati, con una differenza di circa del 17-18%. Il modello appare in evidente difficoltà nel riconoscere i cracker bruciati rispetto ai cracker cotti e senza tipo di difettosità.

Predicted \ True	Normal	Over baked	Cracked
Normal	0.98	0.27	0.02
Over baked	0.01	0.73	0
Cracked	0	0	0.93

Tabella 22 - Matrice di confusione normalizzata per il modello YOLOv5 versione stressata

Per la versione stressata del modello YOLOv5 il trend peggiora lievemente, subendo un calo del 7% nell'identificazione della classe *Over baked*, e un peggioramento del 5% nell'identificazione della classe *Cracked*, dimostrando così una difficolta del modello nell'affrontare in maniera positiva condizioni di stress, e non essendo dunque un modello consigliato all'applicazione in linee produttive.

Predicted \ True	Normal	Over baked	Cracked

Normal	0.98	0.2	0
Over baked	0.01	0.8	0
Cracked	0	0	1

Tabella 23 - Matrice di confusione normalizzata per il modello YOLOv8 normale

Passando al modello YOLOv8 la matrice di confusione normalizzata raggiunge valore perfetto (1 su 1 possibile) di riconoscimento della classe *Cracked*, e presenta errore costante nella valutazione della classe critica *Over baked*, mantenendo la percentuale di identificazione dell'80%.

Predicted \ True	Normal	Over baked	Cracked
Normal	0.96	0.23	0
Over baked	0.01	0.77	0
Cracked	0.02	0	1

Tabella 24 - Matrice di confusione per il modello YOLOv8 versione stressata

Sebbene il modello YOLOv8 versione stressata avesse alcune delle metriche generali migliori, la matrice di confusione evidenzia una continua difficoltà nella classificazione dei cracker bruciati, continuando a mantenere la linea vista nei dati precedenti.
La difficoltà nell'identificazione della classe *Over baked* mantiene una costanza anche all'interno del modello migliore YOLOv11, che risulta avere gli stessi problemi di classificazione sia nel modello normale, che quello stressato, non fornendo dunque paragoni utili alla ricerca.

Predicted \ True	Normal	Over baked	Cracked
Normal	0.98	0.23	0.04
Over baked	0.02	0.77	0
Cracked	0	0	0.96

Tabella 25 - Matrice di confusione normalizzata per il modello YOLOv11 versione normale

Predicted \ True	Normal	Over baked	Cracked
Normal	0.98	0.23	0.04
Over baked	0.02	0.77	0
Cracked	0	0	0.96

Tabella 26 - Matrice di confusione normalizzata per il modello YOLOv11 versione stressata

In conclusione, l'analisi delle matrici di confusioni normalizzate, dimostra l'effettiva difficoltà nell'identificazione di un difetto complicato da normalizzare, quale la bruciatura.
Al fine di dimostrare in maniera visiva le metriche aggregate del modello, vengono riportate i grafici di output generati da Ultralytics [29] rappresentanti la curva di Precision-Recall.

Le curve PR permettono di vedere non solo la media dei parametri delle metriche tecniche come l'analisi precedente, ma come l'andamento della precisione e del richiamo su ogni classe si influenzino a vicenda.

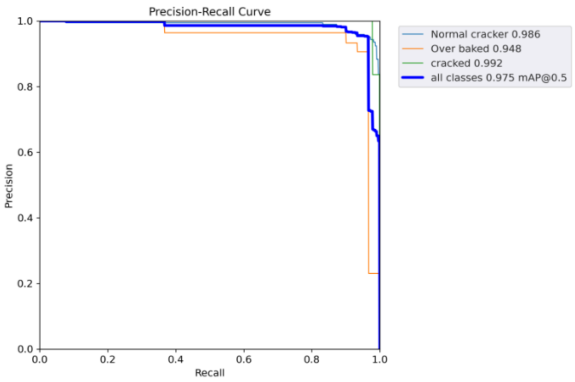
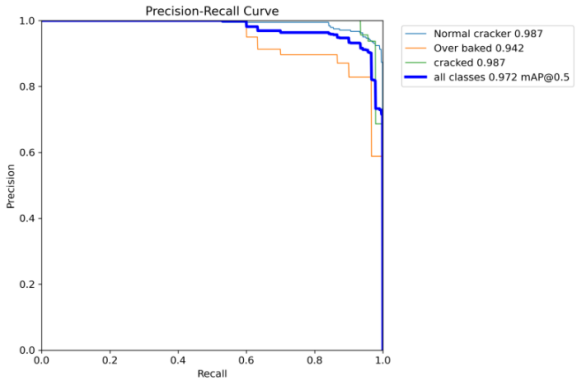
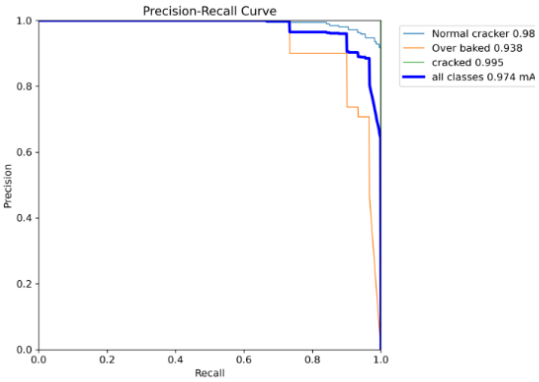
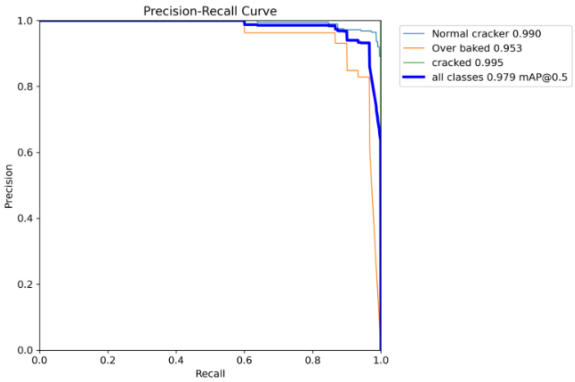
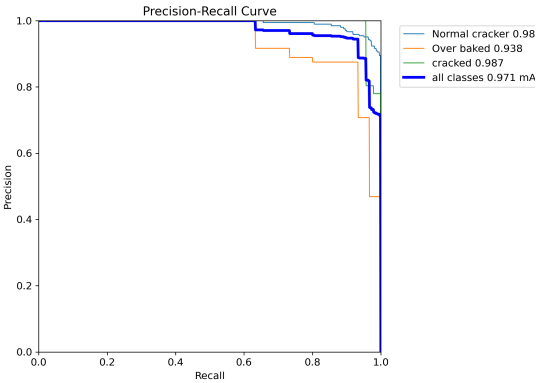
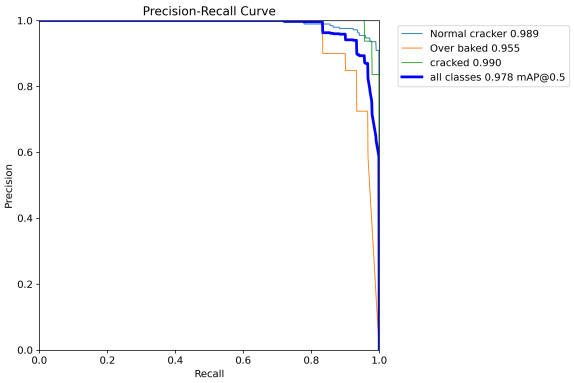
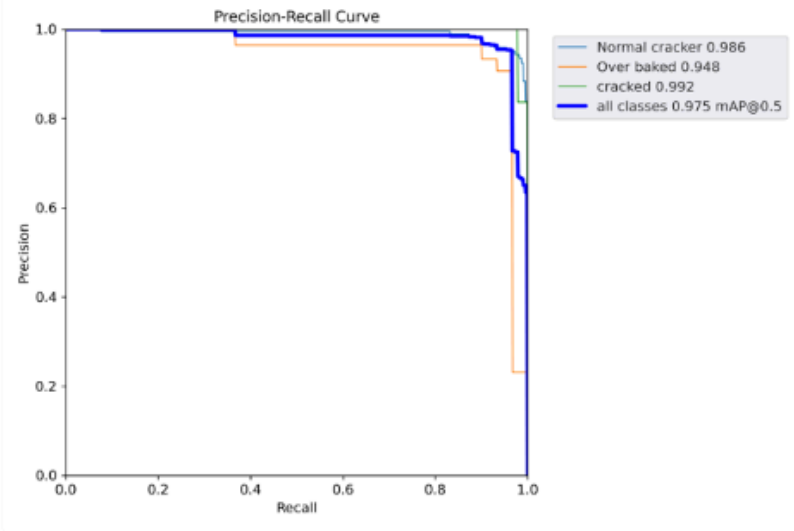
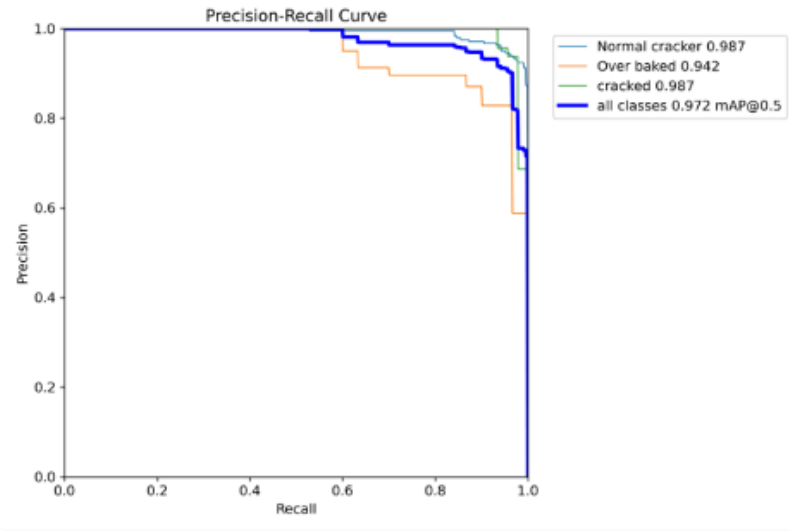
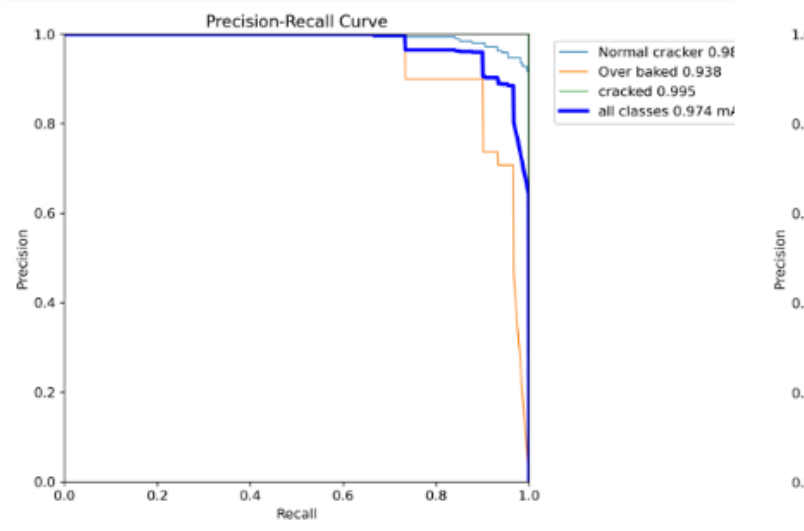
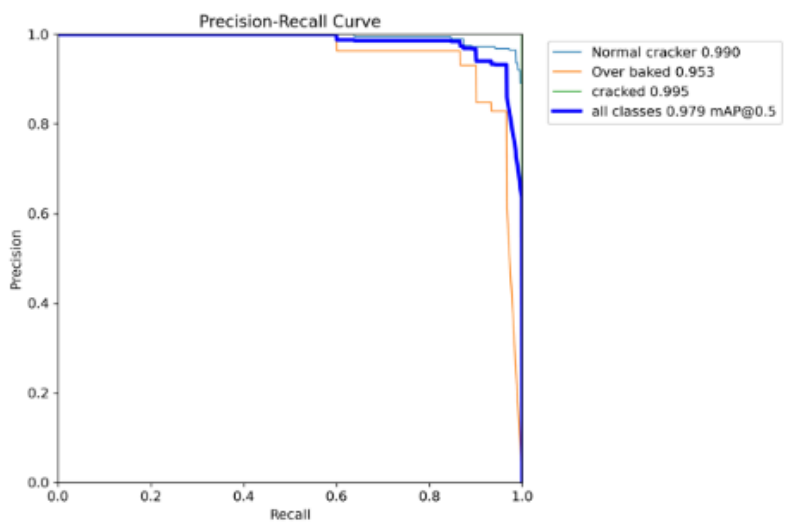
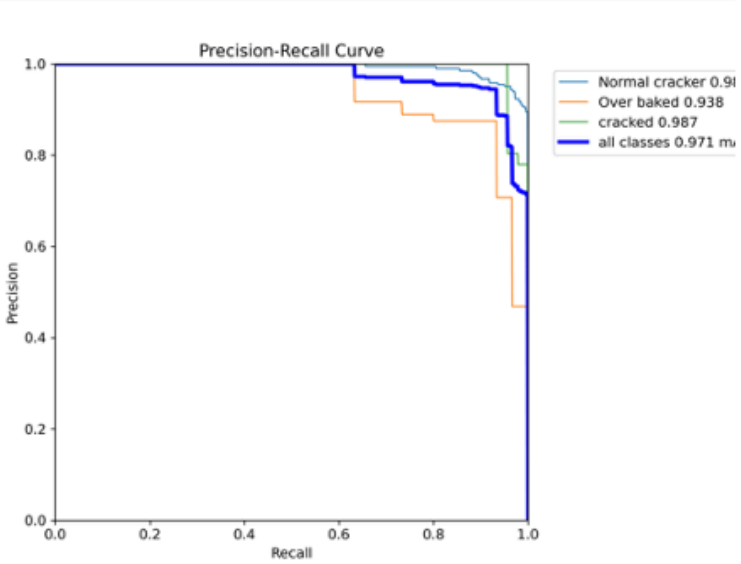
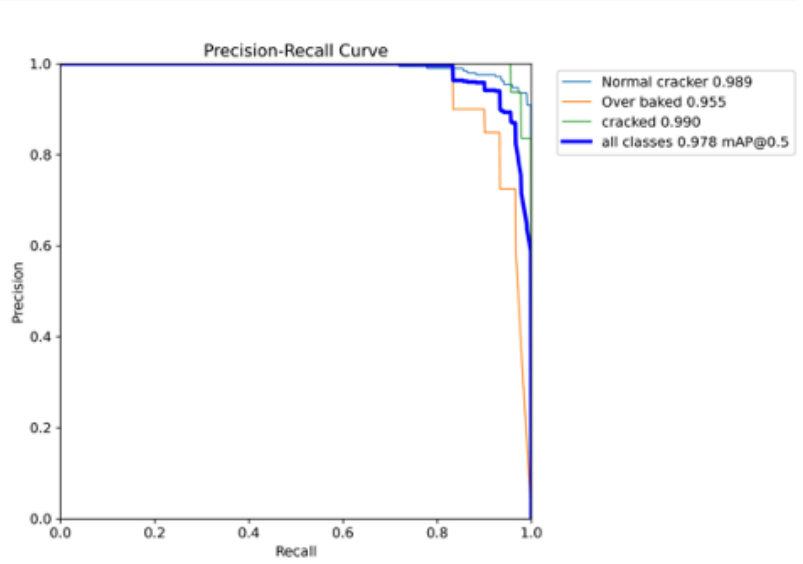
Modello	PR curve della versione normale	PR curve della versione stressata
YOLOv5		
YOLOv8		
YOLOv11		

Tabella 27 - Comparazione curve Precision-Recall suddivise per modelli e versioni analizzate

Modello	PR curve della versione normale	PR curve della versione stressata
YOLOv5		
YOLOv8		

YOLOv11


Precision-Recall Curve
Precision
Recall
Normal cracker 0.9
Over baked 0.938
cracked 0.987
all classes 0.971 m


Precision-Recall Curve
Precision
Recall
Normal cracker 0.989
Over baked 0.955
cracked 0.990
all classes 0.978 mAP@0.5

Le curve Precision-Recall riportate nella tabella di confronto mostrano a livello grafico il trade-off tra precisione e richiamo: generalmente una curva che rispetta requisiti soddisfacenti tende verso il punto più alto tra i due assi, nel quadrante in alto a destra.
A fronte di questa considerazione è possibile evidenziare come tutte le curve analizzate riportino dei risultati accettabili, notando in particolare come la tendenza delle versioni stressate risulti migliore per i modelli YOLOv8 e YOLOv11, e leggermente sotto la media per la versione cinque.
La classe con maggiore difficoltà si riconferma essere la *Over-baked,* la quale presenta risultati altalenanti e poco robusti, specialmente negli ultimi due modelli di algoritmi.
In conclusione, anche l'analisi grafica suggerisce l'utilizzo migliore delle ultime due versioni in condizioni reali, aventi condizioni reali che possono compromettere la precisione dell'object detection.

Completate le analisi quantitative delle metriche tecniche risultanti dal run dei modelli, è possibile analizzarne gli output qualitativi, considerando le immagini salvate dal modello e contenenti le predizioni individuate dagli algoritmi.
Per questo paragone si sono utilizzati gli output di testing dei modelli, entrambi contenuti nel batch 0 e allenati su dataset stressato.

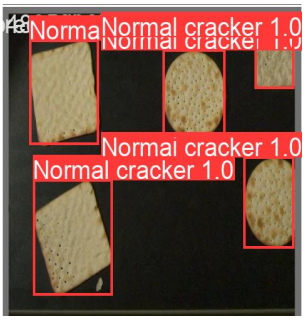
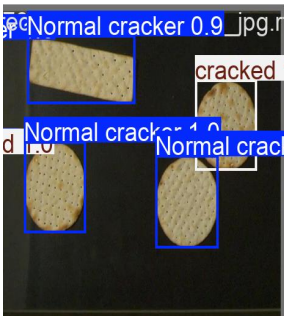
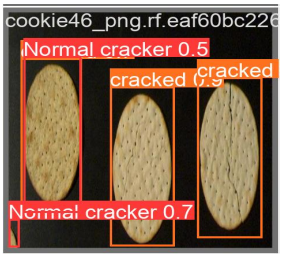
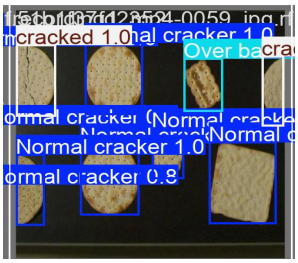
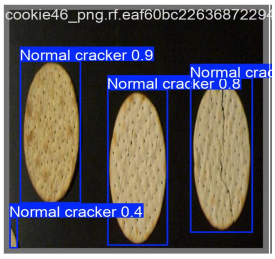
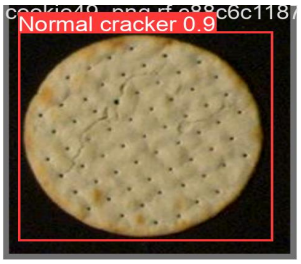
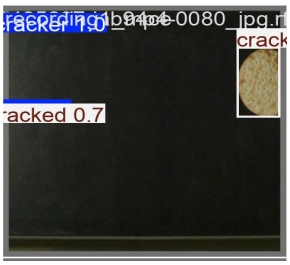
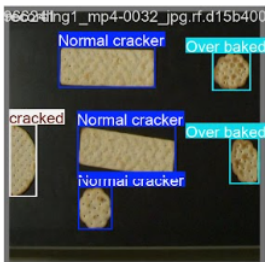
Caso	YOLOv5 stressato	YOLOv8 stressato	YOLOv11 stressato
Facile			
	TP: 4 FP:0 FN:0	TP: 3 FP:1 FN:0	TP:1 FP:0 FN:0
Difficile			
	TP: 3 FP:0 FN:0	TP: 7 FP: 0 FN:1	TP:2 FP:1 FN:0
Fallimento			
	TP:0 FP:1 FN:0	TP: 0 FP: 0 FN:1	TP: 5 FP:1 FN:1

Tabella 28 - Output visivi dei differenti modelli versione stressata, suddivisi per grado di difficoltà

Gli scenari considerati sono i tre più estremi tra gli esempi disponibili, composti da due immagini di casistica facile, composta da frame senza sovrapposizioni o eventuali distorsioni, una riga come componente difficile, identificata da immagini modificate o particolarmente over-crowded, e in un ultimo caso un fallimento.

Da questo output (tabella 28) si può notare come YOLOv5 nella versione stressata abbia problemi di identificazione dei cracker normali, fornendo un falso positivo.

In contrasto, è possibile notare come YOLOv8 tenda ad essere più “cauto” nell’etichettatura della classe Normal, utilizzando le label dei difettosi anche con cracker senza difetti, generando così molti falsi negativi, specialmente nei casi in cui l’immagine dell’oggetto appare troncata a metà, essendo un nastro trasportatore in movimento.

Nel complesso l’analisi visiva evidenzia gli atteggiamenti del modello, confermando la possibilità di aumentare la robustezza del modello, al fine di ottenere un equilibrio maggiormente saldo tra sensibilità e specificità nei modelli reali.

Analisi output robustezza

La tabella 29 riporta i risultati conseguiti dall'analisi Bootstrap applicata su ogni modello considerato e a ciascuna sua versione, comparando non solo la versione normale con quella stressata, ma anche differenze soglie di confidenza, al fine di un'analisi comparativa totale.
Le metriche considerate, inoltre, includono il mean score, cioè la media dei punteggi di confidenza, e l'intervallo di confidenza al 95% rappresentato dal lower bound e dall'upper bound.

Modello	**Versione**	**Conf_thre**	**Mean score**	**CI_lower**	**CI_upper**
YOLOv5	Normale	0,25	0.897167	0.867851	0.921029
		0,50	0.924508	0.908585	0.937550
		0,75	0.940702	0.933637	0.947010
	Stress	0,25	0.884637	0.857387	0.909987
		0,50	0.922580	0.910805	0.934444
		0,75	0.935772	0.927690	0.944020
YOLOv8	Normale	0,25	0.879325	0.850227	0.905915
		0,50	0.916756	0.900156	0.931031
		0,75	0.940107	0.930303	0.948085
	Stress	0,25	0.908610	0.886616	0.926940
		0,50	0.926116	0.913678	0.937444
		0,75	0.944233	0.937971	0.950528
YOLOv**11**	Normale	0,25	0.920025	0.900604	0.935332
		0,50	0.933551	0.923841	0.942285
		0,75	0.944603	0.936735	0.951132
	Stress	0,25	0.898818	0.877052	0.918724
		0,50	0.915076	0.898407	0.930673
		0,75	0.941952	0.934507	0.949113

Tabella 29 - Confronto diversi intervalli di confidenza e mean score analisi Bootstrap

Analizzando i risultati si può osservare un chiaro aumento nello mean score all'aumentare della confidenza per ogni modello e versione, questo accade in quanto soglie più alte indicano predizioni più affidabili, riducendo la presenza di falsi positivi.
I modelli YOLOv8 e YOLOv11 presentano prestazioni molto similari, e superiori a YOLOv5; YOLOv11 presenta il valore di *mean score* più elevato, con la confidence *threshold* a 0.75, evidenziando una chiara capacità di rilevamento dei biscotti privi di difetti.
Per la versione stressata, maggiormente complessa, YOLOv8 presenta valori leggermente migliori rispetto agli altri modelli analizzati, suggerendo così una maggiore capacità di identificazione dei difetti in condizioni più variabili.

Analizzando gli intervalli di confidenza appare evidente come il range sia particolarmente ristretto, il che suggerisce un'ottima capacità dei modelli nelle predizioni: un ristretto intervallo di confidenza indica affidabilità e stabilità delle stime.

In conclusione, l'analisi pone YOLOv8 come compromesso nella scelta tra i modelli, specialmente nella versione stressata, mentre conferma l'utilizzo di YOLOv11 come migliore nella versione normale.

Le evidenze raccolte confermano la scelta di soglie di confidenza elevate (³0.50) per applicazioni in cui è necessario ridurre in maniera significativa il numero di falsi positivi, pur mantenendo un elevato livello di recall.

Analisi output metriche gestionali

In seguito alle analisi delle metriche tecniche è importante commentare i modelli utilizzati sotto un punto di vista gestionale, introducendo il contesto aziendale al fine di valutarne le prestazioni in ambienti reali di PMI, aventi dunque possibilità di hardware limitate e linee produttive non eccessivamente avanzate.

Come anticipato precedentemente, le tre aree di impatto analizzate sono: economica, organizzativa e ambientale.

Per calcolare le percentuali di miglioramento ed eventuale peggioramento degli algoritmi, si sono considerate le versioni stressate, in quanto imitatrici di situazioni realistiche aziendali, e aventi maggiori immagini presenti nel dataset: così facendo i risultati ottenuti dai differenti KPI scelti, sono implementabili in ottica lavorativa e scalabili facilmente all'interno dell'organizzazione.

Per il calcolo del Cost Saving Ratio si è utilizzata la formula illustrata nell'analisi teorica delle metriche, calcolandone la percentuale e paragonando le versioni stressate dei tre modelli YOLO, così da poterne effettivamente verificare il miglioramento o il peggioramento.

Al fine dei calcoli si sono utilizzate le matrici di confusione della fase di validazione dei risultati riportate in tabella 30,31,32.

Predicted \ True	Normal	Over baked	Cracked
Normal	217	8	1
Over baked	2	22	0
Cracked	2	0	45

Tabella 30 - Matrice di confusione fase di validazione, YOLOv5 stressata

Predicted \ True	Normal	Over baked	Cracked
Normal	213	7	0
Over baked	3	23	0
Cracked	5	0	46

Tabella 31- Matrice di confusione fase di validazione, YOLOv8 stressata

Predicted \ True	Normal	Over baked	Cracked
Normal	216	7	2
Over baked	4	23	0
Cracked	1	0	44

Tabella 32 - Matrice di confusione fase di validazione, YOLOv11 stressata

Estraendo gli errori alpha e beta dalle seguenti matrici è possibile compiere i calcoli considerando gli errori totali commessi dagli algoritmi, estraendone così anche il costo di scarto eventualmente associato, ipotizzando un costo unitario di €0,30 al pezzo (Allegato A).

Modello	Versione	TP	Errori totali	Costo di scarto (€)	CSR con baseline YOLOv5 (%)	CSR con baseline YOLOv8 (%)	CSR con baseline YOLOv11 (%)
YOLOv5	Stressata	284	13	3,90	-	+13,33	+7,14
YOLOv8	Stressata	282	15	4,50	-15,38	-	-7,14
YOLOv11	Stressata	283	14	4,20	-7,69	+6,67	-

Tabella 33 - Rappresentazione Cost Saving Ratio per i diversi modelli esaminati

Calcolando il CSR, YOLOv8 risulta aumentare il costo totale di circa il 15,38%, corrispondenti a circa €0,60 per pezzo, rispetto alla baseline identificata con YOLOv5, dove anche YOLOv11 rispecchia un aumento del costo del 7,69%.

Questo risultato attraverso KPI gestionale, evidenza un problema di organizzazione e scelta rilevante: sebbene le metriche tecniche generali conducessero alla scelta del modello YOLOv11, il Cost Saving Ratio riporta il costo aggiuntivo di scarti che questa scelta comporterebbe.

Il risultato è causato dall'aumento dei falsi positivi della classe True Normal, relativo ai modelli YOLOv8 e YOLOv11, i quali tendono ad etichettare in maniera più aggressiva prodotti senza difetti, come aventi eventuali bruciature, causandone dunque un impatto economico immediato, specialmente nell'ottava versione del modello.

La classe Normal risulta inoltre in maniera dominante, come sarebbe realistico anche in condizioni aziendali, e dunque un lieve aumento della detection di FP si traduce in scarti maggiori rispetto all'aumento della Precisione delle altre classi minoritarie: basti considerare come YOLOv5 sbagli l'1% dei cracker normali (13 scarti totali), mentre YOLOv8 e YOLOv11 solamente il 2% dei cracker normali, generando però 14 e 15 scarti totali.

Ciò dimostra come le metriche globali spesso considerate per questi algoritmi cambino di poco, ma come gli effetti gestionali ed economici siano molto sensibili all'aggressività del modello: il CSR evidenzia dunque la problematica economica dell'impatto dei falsi scarti in linea di produzione.

Compiendo un'analisi più specifica dei costi dei modelli, è importante considerare da cosa siano dominati i costi presenti: YOLOv5 ha costi dominati dalla presenza di False Negatives, mancati difetti rilevati; YOLOv8 ha costi dipendenti dai False Positives, prodotti etichettati come scarti, ma che in realtà sono commercializzabili, in ultima lettura YOLOv11 invece presenta una situazione che si potrebbe definire intermedia, con FP e FN bilanciati, ma con costi superiori al baseline YOLOv5.

A seguito di questa analisi risulta importante lo scope e decisione aziendale: se l'obiettivo della PMI è limitare i costi a discapito della qualità, allora il modello di YOLOv5 appare migliore, se tuttavia si dovesse preferire la qualità del prodotto a discapito dei costi aggiuntivi allora risulta evidente l'utilizzo del modello YOLOv8, maggiormente robusto e indicato per questo tipo di analisi. Un compromesso intermedio potrebbe risultare nell'applicazione di YOLOv11 che diminuisce lievemente gli errori rispetto alla precedente versione, ma risulta comunque più costoso in termini di applicazione e implementazione.

Per cercare di limitare questo problema, e considerando che gli scarti prodotti da YOLOv8 e YOLOv11 hanno probabilità alta di essere in realtà prodotti commercializzabili senza difetti, si è implementata una logica di bundle che seguisse una logica basata sulla confidenza delle predizioni.
Analizzando lo script utilizzato (Allegato F) è possibile vedere come i cracker considerati lievemente rotti, cioè con una soglia cracked minore del 60%, sono stati considerati accettabili se ci fossero almeno 15 cracker senza alcun tipo di difetto e considerati appartenenti alla classe Normal.
Questa logica applicata al batch di produzione in fase di validation, ha prodotto un risparmio del 4,55% e dimostra come, seppur relativamente a questo singolo scenario, l'applicazione di queste strategie possa risultare in una diminuzione degli scarti e un relativo rientro economico, senza compromettere la qualità del prodotto.

Sebbene l'impatto economico preferisca il modello più standard, è rilevante considerare anche le necessità di linea, come l'ambito organizzativo suggerisce, e dunque un eventuale soglia di throughput ed eventuali colli di bottiglia.

Modello	**Versione**	**Speed metrics (ms)**	**Throughput (frame/s)**	**Aumento % TH Baseline YOLOv5**	**Aumento % TH Baseline YOLOv8**	**Aumento % TH Baseline YOLOv11**
YOLOv5	Stressata	24,5	40	-	-4,76	-36,5
YOLOv8	Stressata	23,8	42	+ 5	-	-33,3
YOLOv11	Stressata	15,7	63	+57,5	+50	-

Tabella 34 - Analisi di metriche gestionali organizzative

Sebbene YOLOv11 acquisisca e processi le immagini più velocemente, la scelta deve considerare il bilanciamento tra velocità e qualità delle predizioni, valutandone il trade-off aziendale.

Un altro parametro influente nel contesto produttivo e globale è l'impatto ambientale fornito dal WRI (Waste Reduction Index) calcolato con i valori FP presenti nelle matrici, i quali identificano tutti gli errori alpha: oggetti etichettati come difettosi, ma che in realtà sono privi di difetti.

Modello	FP totali	WRI con baseline YOLOv5 (%)	WRI con baseline YOLOv8 (%)	WRI con baseline YOLOv11 (%)
YOLOv5	4	-	+50	+20
YOLOv8	8	-100	-	-60
YOLOv11	5	-25	+37,5	-

Tabella 35 - Comparazione Waste Reduction Index

Commentando gli output tabellari appare evidente come l'algoritmo che produca meno scarti sia YOLOv5, questo concetto è però contestualizzabile: gli algoritmi come YOLO in versione 8 e 11 tendono ad assumere un atteggiamento maggiormente conservativo, che comporta la preferenza all'identificare come difettoso prodotti non necessariamente tali.

Sebbene la percentuale trovata possa sembrare molto elevata, è possibile scomporre maggiormente l'impatto ambientale realistico che effettivamente si crea, non basandosi esclusivamente sui FP, ma considerando il totale delle istanze analizzate: una delle alternative è dunque normalizzare l'impatto ambientale rispetto al totale dei prodotti utilizzati.

Svolti i calcoli i risultati ottenuti sono i seguenti:

Modello	FP	FN	TP+FN	Indice ambientale (%)
YOLOv5	4	9	293	4,44
YOLOv8	8	7	289	5,19
YOLOv11	5	9	292	4,79

Tabella 36 - Indice ambientale rapportato ai modelli YOLO

YOLOv11 risulta il compromesso ideale tra gli algoritmi utilizzati, evidenziando come, sebbene il modello generi alcuni scarti in più rispetto alla quinta versione, riesce comunque a contenere l'impatto complessivo sul totale della produzione.

È necessario, dunque, considerare il contesto totale di produzione, analizzando in maniera puntuale e specifica ogni indice gestionale relativo al contesto e i propri obiettivi aziendali.

In conclusione, il capitolo dimostra come la scelta migliore, in termini di parametri tecnici e gestionali, risiede nel trade-off che l'azienda è disposta a compiere rispetto alla qualità del prodotto e al costo ad esso associato.

Se l'obiettivo permane quello di una qualità di identificazione migliore, e dunque un atteggiamento conservativo in cui è meglio scartare piuttosto che ottenere in fase finale un prodotto difettoso, YOLOv11 risulta migliore; tuttavia, il modello YOLOv5 risulta avere un costo minore e prestazioni valide, seppure con FN presenti in quantità maggiore.

Una sintesi dei risultati emersi e delle motivazioni associate in un contesto di PMI è riportata nella tabella 37, in cui è possibile notare un mix di motivazioni date da metriche tecniche e gestionali.

Modello	Versione	Considerazioni
YOLOv5	Stressata	Maggior numero di Falsi Negativi, riduce i costi associati agli scarti. Scelta più conservativa economicamente in un contesto di PMI
YOLOv8	Stressata	Garantisce maggior rilevazione di difetti, a discapito dei FP, evidenziando come le metriche tecniche non sempre siano traducibili in risultati gestionali. Da valutare economicamente per contesto di PMI.
YOLOv11	Stressata	Situazione intermedia per costi e scarto, trade off costi di implementazione rispetto a miglioramento dei risultati non completamente favorevole.

Tabella 37 - Tabella riassuntiva delle considerazioni gestionali

In conclusione, l'analisi degli output non può essere compiuta solo attraverso le metriche globali, che danno una prima visione generica del comportamento del modello, ma deve comprendere un contesto e l'utilizzo di metriche gestionali per evidenziarne i difetti e i pregi in ottica aziendale, al fine di identificarne in maniera oggettiva l'implementazione all'interno della linea di produzione.
Tuttavia, dallo studio completato emerge come la versione 11 del modello YOLO possa essere un compromesso per le industrie alimentari.

Conclusioni

Sintesi risultati ottenuti

Il presente lavoro di tesi ha avuto come obiettivo l'analisi e la valutazione della ricerca scientifica in ambito Machine Vision applicato al settore agroalimentare secondario, focalizzandosi dunque su prodotti lavorati. Attraverso un approccio metodologico basato sul protocollo PRISMA, è stato possibile condurre una revisione sistematica e analizzare i risultati ottenuti, per poi applicare la knowledge raccolta in una fase sperimentale di caso studio, avente come modelli principali YOLOv5/8/11 valutati in condizioni differenti.

La revisione scientifica ha dimostrato come lo studio di queste tecniche si trovi ancora in una fase intermedia di implementazione, con una forte prevalenza di studi sperimentali condotti in laboratorio, rispetto a casi studio applicativi condotti su dati industriali. Le applicazioni esistenti mostrano tuttavia livelli di accuratezza elevati, e evidenziano la reale necessità di adottare modelli robusti se l'obiettivo è l'applicazione in ambiti reali, dove la variabilità delle fonti risulta essere un tema problematico.

La seconda parte della tesi volge la propria attenzione sul caso studio applicativo, una sezione comparativa di tre versioni del modello YOLO selezionate per la loro applicazione ipotetica in un contesto di PMI e valutate su tre dataset differenti, per compararne le caratteristiche in condizioni reali. La sperimentazione ha consentito un confronto oggettivo e puramente tecnico attraverso l'analisi di metriche di accuratezza, come precisione, recall, latenza, ed è stata arricchita da un'analisi della robustezza attraverso il metodo Bootstrap; una volta raccolti gli output tecnici si è compiuta un'analisi gestionale, analizzando KPI precisi per inserire la ricerca in un contesto economico, organizzativo e di impatto ambientale.

Complessivamente la ricerca ha restituito buoni risultati in termini di precisione e richiamo, dimostrandosi in linea con gli studi analizzati: YOLOv11 risulta essere un buon compromesso sia in termini di performance che di valutazioni economiche, succeduto da YOLOv8, che seppur con performance meno ottimali, mostra costi di implementazione e architetturali minori.

YOLOv5 rimane in linea con le supposizioni e si dimostra un modello particolarmente affidabile per applicazioni a bassa complessità.

Dal punto di vista gestionale i risultati evidenziati confermano la possibilità di diminuire i costi di scarto e applicare logiche diversificate per soluzioni ancora più ottimali, come l'applicazione di una logica bundle basata sul grado del difetto identificato.

Complessivamente la ricerca ha dimostrato la validità scientifica dell'utilizzo di sistemi di Machine Vision nel settore, e ne ha fornito una revisione metodologica integrata a una revisione sperimentale e valutazione gestionale.

Limiti dello studio

Sebbene la ricerca sia stata strutturata con rigore metodologico, sono presenti alcuni limiti all'interno del lavoro svolto.

In primis dal punto di vista della revisione bibliografica, un ambiente di ricerca così specifico ha ridotto la numerosità del campione di articoli analizzati, identificandone un numero limitato e dunque dovendo integrare tramite ricerche secondarie.

Dal punto di vista sperimentale i test sono stati condotti su dataset con limitate dimensioni e varietà dei difetti, scelto da repository online, e dunque simulato in ambiente di produzione, ma non direttamente derivante da dati industriali.

I test, infatti, sono stati compiuti in ambiente virtuale, e sebbene si sia compiuto un lavoro di adattamento del dataset a condizioni realistiche, la valutazione diretta di eventuali fattori ambientali risulta limitata.

Infine, l'analisi economica e gestionale, seppur teoricamente corretta e metodologicamente strutturata, è stata compiuta su valori simulati delle fasi di validazione, e dunque non direttamente conducibili all'ambito industriale.

Questi punti racchiudono i limiti della ricerca, ma rappresentano anche eventuali punti di partenza per possibili implementazioni e eventuali prospettive del lavoro compiuto.

Prospettive future e possibili implementazioni

Sulla base dei risultati ottenuti le prospettive future si orientano sul creare modelli di Machine Vision scalabili e facilmente adattabili a contesti di produzione industriale.

Una prima area di miglioramento risulta essere la creazione e la sperimentazione dei modelli su un dataset proprietario avente maggiore ampiezza e varietà, direttamente ottenuti da aziende produttrici, così da valutare in maniera più precisa l'applicazione operativa e l'eventuale miglioramento dei parametri del modello.

Parallelamente, l'adozione di tecniche avanzate di transfer learning e uno studio più specifico sul data augmentation dinamica potrebbero avvalorare i risultati e migliorarli maggiormente, raggiungendo potenzialmente risultati tendenti ai valori massimi di precisione e richiamo.

Dal punto di vista gestionale, una possibile implementazione potrebbe ricadere sull'integrazione di piattaforme IoT industriali, così da automatizzare il controllo della qualità in tempo reale e migliorare le linee produttive integrando strumenti automatici di sorting post rilevamento di difetti.
In prospettiva, lo studio si inserisce in un contesto di ricerca in via di sviluppo, contribuendo scientificamente e operativamente agli articoli presenti in letteratura, al fine di integrare una maggiore consapevolezza sui sistemi di visione artificiale nel settore agroalimentare secondario, promuovendo applicazioni efficienti e che inducono la riduzione economica e degli sprechi.

Riconoscimenti

Questa tesi è realizzata nell'ambito del progetto NODES, finanziato dal MUR sui fondi M4C2 - Investimento 1.5 Avviso "Ecosistemi dell'Innovazione", nell'ambito del PNRR finanziato dall'Unione europea – NextGenerationEU (Grant agreement Cod. n.ECS00000036).

A conclusione di questo elaborato di tesi, desidero ringraziare il mio relatore, il Professore Galetto, e la correlatrice Professoressa Verna, i quali hanno saputo guidarmi e consigliarmi lungo questo percorso, donandomi spunti in ogni step dell'elaborato.
Desidero ringraziare inoltre l'ingegnere Piovano, che ha seguito ad ogni passo il lavoro svolto, aiutandomi, guidandomi, e rispondendo alle mille mail scritte lungo tutto questo periodo. Grazie per la pazienza e la disponibilità che mi avete donato.
Grazie.

Allegati

Allegato A – Calcolo specifico dei costi di scarto

Modello	Versione	Costo di scarto (€) al pezzo
YOLOv5	Stressata	13*0,30 = 3,90
YOLOv8	Stressata	15*0,30 = 4,50
YOLOv11	Stressata	14*0,30 = 4,20

Allegato B – Script Python YOLOv5

```
-- coding: utf-8 --
"""Cracker_v5.ipynb
Automatically generated by Colab.
Original file is located at https://colab.research.google.com/drive/1r3vTTPdJkjX7niAa-36qgkd5ptlCxWEc """
Controllo della GPU
!nvidia-smi
Montaggio Drive per salvare i checkpoint e i risultati
from google.colab import drive drive.mount('/content/drive')
Download del dataset
!pip install roboflow
from roboflow import Roboflow rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb") version = project.version(1) dataset = version.download("yolov5")
!ls
Installazione di YOLOv5 e i relativi requirements
!git clone https://github.com/ultralytics/yolov5.git !pip install -r yolov5/requirements.txt
import os
paths = [ "/content/cookie-defect-detection-1/train/images", "/content/cookie-defect-detection-1/valid/images", "/content/cookie-defect-detection-1/test/images" ]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{p}: exists={exists}, immagini={nfiles}")
!cat /content/cookie-defect-detection-1/data.yaml
Commented out IPython magic to ensure Python compatibility.
%%bash
cat > data.yaml <<EOF
train: /content/cookie-defect-detection-1/train/images
val: /content/cookie-defect-detection-1/valid/images
test: /content/cookie-defect-detection-1/test/images
nc: 3
names:
- "Normal cracker"
```

```
- "Over baked"
- "cracked"
roboflow:
license: CC BY 4.0
project: cookie-defect-detection-6fojb
url: https://universe.roboflow.com/tesi-wltpv/cookie-defect-detection-6fojb/dataset/1
version: 1
workspace: tesi-wltpv
EOF
# mostra il file per verifica
echo "Contenuto di yolov5/data.yaml:"
cat data.yaml
verifica che le cartelle train/val/test esistano e contengano immagini
import os
paths = [ "/content/cookie-defect-detection-1/train/images", "/content/cookie-defect-detection-1/valid/images", "/content/cookie-defect-detection-1/test/images" ]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{p}: exists={exists}, immagini={nfiles}")
os.environ['WANDB_DISABLED'] = 'true'
Fase di training
!python                                                                 yolov5/train.py
--img                                                                              640
--batch                                                                              8
--epochs                                                                           100
--data                                                                /content/data.yaml
--weights                                                                   yolov5s.pt
--project                                                 /content/drive/MyDrive/yolov5_runs
--name                                                                 cookie_yolov5_v2
--cache
!ls /content/drive/MyDrive/yolov5_runs/cookie_yolov5/weights
import os os.environ['WANDB_DISABLED'] = 'true'
Linea di codice per ricominciare il training in caso di interruzione
!python train.py --resume /content/drive/MyDrive/yolov5_runs/cookie_yolov5/weights/last.pt
Commented out IPython magic to ensure Python compatibility.
%cd /content
!python                                                                   yolov5/val.py
--weights                  /content/drive/MyDrive/yolov5_runs/cookie_yolov5_v24/weights/best.pt
--data                                          /content/cookie-defect-detection-1/data.yaml
--img                                                                              640
--save-conf
--project                                                 /content/drive/MyDrive/yolov5_runs
--name cookie_yolov5_v24_val
!ls /content/drive/MyDrive/yolov5_runs/cookie_yolov5/weights
validation e testing utilizzando i pesi trovati in fase di allenamento
```

```
!python detect.py
--weights /content/drive/MyDrive/yolov5_runs/cookie_yolov5/weights/best.pt
 --img 640
--conf 0.25
--source /content/cookie-defect-detection-1/test/images
--project /content/drive/MyDrive/infer_yolov5
--name cookie_test
#metriche test !python val.py
--weights /content/drive/MyDrive/yolov5_runs/cookie_yolov5/weights/best.pt
--data data.yaml
--img 640
"""ANALISI BOOTSTRAP"""
from google.colab import drive drive.mount('/content/drive')
!ls /content/drive/MyDrive/yolov5_runs/cookie_yolov5_v24/weights/
"""Stimare intervalli di confidenza per le metriche"""
import os import json import pandas as pd import numpy as np from collections import defaultdict
Percorso modello e dataset
weights_path = "/content/drive/MyDrive/yolov5_runs/cookie_yolov5_v24/weights/best.pt" data_yaml =
f"{dataset.location}/data.yaml" img_size = 640 thresholds = [0.25, 0.5, 0.75]
results_summary = []
for conf in thresholds: print(f"\n=== Valutazione con conf-thres = {conf} ===")
exp_name = f"exp_conf{int(conf*100)}"

# Valutazione YOLO
!python val.py \
  --weights {weights_path} \
  --data {data_yaml} \
  --img {img_size} \
  --task test \
  --conf-thres {conf} \
  --save-json \
  --name {exp_name} \
  --exist-ok

# Individua il JSON generato
json_path = f"runs/val/{exp_name}/best_predictions.json"
if not os.path.exists(json_path):
   print(f"Attenzione: JSON non trovato in {json_path}, salto questa soglia")
   continue

# Genera CSV per immagine
with open(json_path) as f:
   preds = json.load(f)
```

```
agg = defaultdict(list)
for p in preds:
    agg[p["image_id"]].append(p)

rows = []
for image_id, boxes in agg.items():
    scores = [b["score"] for b in boxes]
    row = {
        "image_id": image_id,
        "num_detections": len(boxes),
        "mean_score": sum(scores)/len(scores) if scores else 0.0
    }
    rows.append(row)

df = pd.DataFrame(rows)
csv_path = f"metrics_per_image_conf{int(conf*100)}.csv"
df.to_csv(csv_path, index=False)
print(f"CSV generato: {csv_path}")

# Bootstrap
data = df["mean_score"].values
n_boot = 1000
boot_means = []

for _ in range(n_boot):
    sample = np.random.choice(data, size=len(data), replace=True)
    boot_means.append(np.mean(sample))

boot_means = np.array(boot_means)
ci_lower = np.percentile(boot_means, 2.5)
ci_upper = np.percentile(boot_means, 97.5)
mean_score = np.mean(data)

print(f"Media score: {mean_score:.4f}")
print(f"Intervallo di confidenza 95%: [{ci_lower:.4f}, {ci_upper:.4f}]")

results_summary.append({
    "conf_thres": conf,
    "mean_score": mean_score,
    "ci_lower": ci_lower,
    "ci_upper": ci_upper
})
```

```
Riepilogo confronto soglie
summary_df = pd.DataFrame(results_summary) print("\n=== Riepilogo confronto soglie ===")
print(summary_df)
```

La versione stressata presenta il medesimo codice, ma con l'API del database differente.

Allegato C – Script Python YOLOv8

```
-- coding: utf-8 --
"""YOLOv8_stress.ipynb
Automatically generated by Colab.
Original file is located at https://colab.research.google.com/drive/1zvVlMD9LEhEJ9wLA8AsyQr7_mu-Fz1SE
Il seguente notebook si occupa dell'allenamento, validation e testing attraverso YOLOv8s, utilizzando il dataset open source cookie-detection di Roboflow. Le classi allenate e rappresentate sono 3:
Cracked
Over-Baked
Normal """
controllare che la GPU sia collegata e venga utilizzata
!nvidia-smi
collegare il drive per salvare successivamente i checkpoint
from google.colab import drive drive.mount('/content/drive')
!pip install roboflow ultralytics
from roboflow import Roboflow from ultralytics import YOLO import os
!pip install roboflow
from roboflow import Roboflow rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb") version = project.version(2) dataset = version.download("yolov8")
dataset = version.download("yolov8")
import os
train_path = os.path.join(dataset.location, "train/images") val_path = os.path.join(dataset.location, "valid/images") test_path = os.path.join(dataset.location, "test/images")
paths = [train_path, val_path, test_path]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0
print(f"{p}: exists={exists}, immagini={nfiles}")
Controllare che data.yaml esista
!cat /content/cookie-defect-detection-2/data.yaml
model = YOLO("yolov8s.pt")
TRAINING
model.train( data="/content/cookie-defect-detection-2/data.yaml", epochs=100, imgsz=640, batch=8, project="/content/drive/MyDrive/yolov8_runs", name="cookie_yolov8_stress", cache=True )
VALIDATION
results_val = model.val( data="/content/cookie-defect-detection-2/data.yaml", imgsz=640 )
TEST/INFERENZA
model.predict( source="/content/cookie-defect-detection-2/test/images", conf=0.25, save=True, project="/content/drive/MyDrive/infer_yolov8", name="cookie_test_stress" )
Metriche su test
results_test = model.val( data="/content/cookie-defect-detection-2/data.yaml", imgsz=640, split="test" )
"""**LOGICA DEGLI SCARTI:**
I biscotti considerati lievemente rotti (confidence<=0.4) vengono flaggati come tali e le loro coordinate sul nastro trasportatore vengono salvate in un file .txt. Ogni 15 biscotti considerati "normali", quindi senza alcun tipo di difetto, l'algoritmo assegna al batch due biscotti lievemente rotti, così da limitarne gli sprechi e gli scarti di produzione. """
from ultralytics import YOLO
```

```
Costruire il modello basandosi sul file in cui sono salvati i pesi migliori del training
model = YOLO("/content/drive/MyDrive/yolov8_runs/cookie_yolov8_stress/weights/best.pt")
from roboflow import Roboflow
rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb")
version = project.version(2) dataset = version.download("yolov8")
test_path = dataset.location + "/test/images"
results_test = model.predict( source=test_path, conf=0.25, save=True, # salva immagini annotate project="/content/drive/MyDrive/infer_yolov8", name="cookie_test_stress", save_txt=True # genera immagini annotate così da poterle riutilizzare con eventuali braccia robotiche )
from pathlib import Path
results_dir = Path("/content/drive/MyDrive/infer_yolov8/cookie_test_stress2/labels")
Parametri
confidence_threshold = 0.6 # soglia per considerare "lievemente rotto" allowed_cracked_rule = 2 # quanti lievemente rotti si possono aggiungere se ci sono almeno 15 perfetti normal_threshold = 15 # soglia di biscotti senza difetti
Inizializzazione dei contatori
totals = {"Normal cracker": 0, "Over baked": 0, "Cracked": 0} lievemente_rotti_conf = 0
for txt_file in results_dir.glob("*.txt"): with open(txt_file) as f: for line in f: parts = line.strip().split() if len(parts) < 2: continue cls_id = int(parts[0]) conf = float(parts[1])
    if cls_id == 0:
        totals["Normal cracker"] += 1
    elif cls_id == 1:
        totals["Over baked"] += 1
    elif cls_id == 2:
        totals["Cracked"] += 1
        if conf <= confidence_threshold:
            lievemente_rotti_conf += 1

Applicazione della regola dei 2 lievemente rotti
normal_count = totals["Normal cracker"] allowed_cracked = allowed_cracked_rule if normal_count >= normal_threshold else 0 scarti_cracked = max(0, lievemente_rotti_conf - allowed_cracked)
Scarti totali
scarti_senza_logica = totals["Over baked"] + lievemente_rotti_conf scarti_con_logica = totals["Over baked"] + scarti_cracked
Risultati
print("Conteggio totale biscotti:", totals) print(f"Biscotti lievemente rotti considerati (conf <= {confidence_threshold}): {lievemente_rotti_conf}") print(f"Scarti senza logica: {scarti_senza_logica}") print(f"Scarti con logica: {scarti_con_logica}")
Risparmio percentuale
risparmio = scarti_senza_logica - scarti_con_logica percentuale = (risparmio / scarti_senza_logica) * 100 if scarti_senza_logica > 0 else 0 print(f"Risparmio di scarti applicando la logica: {risparmio} biscotti ({percentuale:.2f}%)")
"""Primo metodo di visualizzazione grafica"""
import matplotlib.pyplot as plt
labels = ['Scarti'] senza_logica = [scarti_senza_logica] con_logica = [scarti_con_logica]
x = range(len(labels)) plt.bar(x, senza_logica, width=0.4, label='Senza logica', align='center') plt.bar(x, con_logica, width=0.4, label='Con logica', align='edge') plt.ylabel('Numero biscotti') plt.title('Effetto della regola dei 2 lievemente rotti') plt.legend() plt.show()
"""Secondo metodo visualizzazione grafica"""
import matplotlib.pyplot as plt
```

```
labels = ['Scarti'] senza_logica = [scarti_senza_logica] con_logica = [scarti_con_logica]
x = range(len(labels)) width = 0.35
Crea figura
fig, ax = plt.subplots(figsize=(6,5))
Barre
bars1 = ax.bar([i - width/2 for i in x], senza_logica, width, label='Senza logica', color='#FF6F61') bars2 = ax.bar([i + width/2 for i in x], con_logica, width, label='Con logica', color='#6B5B95')
Annotazioni sopra le barre
for bar in bars1 + bars2: height = bar.get_height() ax.text(bar.get_x() + bar.get_width()/2, height + 0.2, f'{int(height)}', ha='center', va='bottom', fontsize=12)
Personalizzazioni
ax.set_ylabel('Numero biscotti', fontsize=12) ax.set_title('Effetto della regola dei 2 lievemente rotti', fontsize=14, fontweight='bold') ax.set_xticks(x) ax.set_xticklabels(labels, fontsize=12) ax.legend(fontsize=12) ax.set_ylim(0, max(scarti_senza_logica, scarti_con_logica)*1.3) # lascia spazio per le annotazioni ax.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout() plt.show()
"""ANALISI BOOTSTRAP"""
from google.colab import drive drive.mount('/content/drive')
Installa librerie
!pip install roboflow ultralytics
import os from roboflow import Roboflow
rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb") version = project.version(2) dataset = version.download("yolov8")
Verifica immagini
paths = [ dataset.location + "/train/images", dataset.location + "/valid/images", dataset.location + "/test/images" ]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{p}: exists={exists}, immagini={nfiles}")
Controlla il file data.yaml
!cat {dataset.location}/data.yaml
Controlla immagini validation
val_path = dataset.location + "/valid/images" exists = os.path.exists(val_path) nfiles = len([f for f in os.listdir(val_path) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{val_path}: exists={exists}, immagini={nfiles}")
Controlla il file data.yaml
!cat {dataset.location}/data.yaml
from ultralytics import YOLO
Carica checkpoint già addestrato
weights_path = "/content/drive/MyDrive/yolov8_runs/cookie_yolov8_stress/weights/best.pt" model = YOLO(weights_path)
print(dataset.location)
from ultralytics import YOLO import pandas as pd import numpy as np from collections import defaultdict import os
Config
thresholds = [0.25, 0.5, 0.75] results_summary = []
Percorsi
model_path = "/content/drive/MyDrive/yolov8_runs/cookie_yolov8_stress/weights/best.pt" val_images = "/content/cookie-defect-detection-2/valid/images" base_dir = "runs/detect"
Carica modello
model = YOLO(model_path)
```

```
for conf in thresholds: val_dir = f"val_conf{int(conf*100)}" print(f"\nElaborazione immagini con conf={conf}...")
# Predict e salva immagini/annotazioni
results = model.predict( source=val_images,
    conf=conf,
    save=True,
    project=base_dir,
    name=val_dir,
exist_ok=True
)

# Estrai predizioni da results
all_preds = []
for result in results:
    boxes = result.boxes
    image_id = os.path.basename(result.path) # prendi solo il nome del file
    for i in range(len(boxes)):
        score = float(boxes.conf[i])
        all_preds.append({"image_id": image_id, "score": score})

if not all_preds:
    print(f"Nessuna predizione trovata per conf={conf}")
    continue

print(f"Totale predizioni per conf={conf}: {len(all_preds)}")

# Aggrega per immagine
agg = defaultdict(list)
for p in all_preds:
    agg[p["image_id"]].append(p)

# Costruisci DataFrame per immagine
rows = []
for image_id, boxes in agg.items():
    scores = [b["score"] for b in boxes]
    row = {
        "image_id": image_id,
        "num_detections": len(boxes),
        "mean_score": sum(scores)/len(scores) if scores else 0.0
    }
    rows.append(row)

df = pd.DataFrame(rows)
csv_path = f"metrics_per_image_val_conf{int(conf*100)}.csv"
```

```
df.to_csv(csv_path, index=False)
print(f"CSV generato: {csv_path}")

# Bootstrap 95% intervallo di confidenza
data = df["mean_score"].values
n_boot = 1000
boot_means = []

for _ in range(n_boot):
    sample = np.random.choice(data, size=len(data), replace=True)
    boot_means.append(np.mean(sample))

boot_means = np.array(boot_means)
ci_lower = np.percentile(boot_means, 2.5)
ci_upper = np.percentile(boot_means, 97.5)
mean_score = np.mean(data)

print(f"Media score: {mean_score:.4f}")
print(f"Intervallo di confidenza 95%: [{ci_lower:.4f}, {ci_upper:.4f}]")

results_summary.append({
    "conf_thres": conf,
    "mean_score": mean_score,
    "ci_lower": ci_lower,
    "ci_upper": ci_upper
})

Riepilogo confronto soglie
summary_df = pd.DataFrame(results_summary) print("\n=== Riepilogo confronto soglie validation YOLOv8 ===")
print(summary_df)
```

Allegato D – Script Python YOLOv11

```
-- coding: utf-8 --
"""YOLOv11.ipynb
Automatically generated by Colab.
Original file is located at https://colab.research.google.com/drive/1LxzDV_8oDFO0VJbsw-hOhBH-J7sXDC0B """
!nvidia-smi
!pip install ultralytics==8.3.5 roboflow
from roboflow import Roboflow from ultralytics import YOLO
from google.colab import drive drive.mount('/content/drive')
import shutil import os
!pip install roboflow
from roboflow import Roboflow rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb") version = project.version(2) dataset = version.download("yolov11")
!wget https://huggingface.co/Ultralytics/YOLO11/resolve/main/yolo11n.pt
Carica il modello YOLOv11 scaricato
model = YOLO("yolo11n.pt")
drive_path = "/content/drive/MyDrive/yolov11_results" os.makedirs(drive_path, exist_ok=True)
model.train( data=f"{dataset.location}/data.yaml", epochs=100, imgsz=640, batch=8, device=0, project="yolov11_results", name="training_run", exist_ok=True )
print("\n VALIDATION in corso...") val_results = model.val( data=f"{dataset.location}/data.yaml", split="val", # Usa lo split di validazione imgsz=640, device=0 ) print("\n Validation completata!") print(val_results)
print("\n TEST su test set...") test_results = model.val( data=f"{dataset.location}/data.yaml", split="test", # Usa lo split di test (se esiste) imgsz=640, device=0 )
print("\n Test completato!") print(test_results)
import shutil
Esegui validazione locale con generazione grafici
val_results = model.val( data=f"{dataset.location}/data.yaml", split="val", imgsz=640, device=0, plots=True, # genera confusion_matrix.png, PR_curve.png, ecc. save_json=True )
Trova dove sono stati salvati i file
local_val_dir = val_results.save_dir # tipicamente 'runs/val/exp'
print(f"\n Grafici salvati localmente in: {local_val_dir}")
import os drive_path = "/content/drive/MyDrive/yolov11_results" val_drive_dir = os.path.join(drive_path, "validation_results") os.makedirs(val_drive_dir, exist_ok=True)
shutil.copytree(local_val_dir, val_drive_dir, dirs_exist_ok=True) print(f" Copiati su Drive in: {val_drive_dir}")
pred_dir = os.path.join(drive_path, "predictions") os.makedirs(pred_dir, exist_ok=True)
Esegui inferenza su un'immagine del dataset
example_img = os.path.join(dataset.location, "test/images") model.predict( source=example_img, # cartella o singola immagine imgsz=640, save=True, project=pred_dir, name="inference_results" )
print(f"\n Predizioni salvate in: {pred_dir}/inference_results")
"""ANALISI BOOTSTRAP"""
from google.colab import drive drive.mount('/content/drive')
Installa librerie
!pip install roboflow ultralytics
from roboflow import Roboflow import os
```

```
rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb")
version = project.version(2) dataset = version.download("yolov11")
Verifica immagini
paths = [ dataset.location + "/train/images", dataset.location + "/valid/images", dataset.location + "/test/images" ]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0
print(f"{p}: exists={exists}, immagini={nfiles}")
Controlla il file data.yaml
!cat {dataset.location}/data.yaml
Controlla immagini validation
val_path = dataset.location + "/valid/images" exists = os.path.exists(val_path) nfiles = len([f for f in os.listdir(val_path) if
f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{val_path}: exists={exists}, immagini={nfiles}")
from ultralytics import YOLO
Carica checkpoint già addestrato
weights_path = "/content/drive/MyDrive/yolov11_results/run_2025-10-09_15-19/validation_results/weights/best.pt" model =
YOLO(weights_path)
from ultralytics import YOLO import pandas as pd import numpy as np from collections import defaultdict import os
Config
thresholds = [0.25, 0.5, 0.75] results_summary = []
Percorsi
model_path = "/content/drive/MyDrive/yolov11_results/run_2025-10-09_15-19/validation_results/weights/best.pt" val_images =
"/content/cookie-defect-detection-2/valid/images" base_dir = "runs/detect"
Carica modello
model = YOLO(model_path)
for conf in thresholds: val_dir = f"val_conf{int(conf*100)}" print(f"\nElaborazione immagini con conf={conf}...")
# Predict e salva immagini/annotazioni
results = model.predict(source=val_images,
  conf=conf,
  save=True,
  project=base_dir,
  name=val_dir,
  exist_ok=True
)

# Estrai predizioni da results
all_preds = []
for result in results:
  boxes = result.boxes
  image_id = os.path.basename(result.path) # prendi solo il nome del file
  for i in range(len(boxes)):
    score = float(boxes.conf[i])
    all_preds.append({"image_id": image_id, "score": score})

if not all_preds:
  print(f"Nessuna predizione trovata per conf={conf}")
  continue
```

```
print(f"Totale predizioni per conf={conf}: {len(all_preds)}")

# Aggrega per immagine
agg = defaultdict(list)
for p in all_preds:
    agg[p["image_id"]].append(p)

# Costruisci DataFrame per immagine
rows = []
for image_id, boxes in agg.items():
    scores = [b["score"] for b in boxes]
    row = {
        "image_id": image_id,
        "num_detections": len(boxes),
        "mean_score": sum(scores)/len(scores) if scores else 0.0
    }
    rows.append(row)

df = pd.DataFrame(rows)
csv_path = f"metrics_per_image_val_conf{int(conf*100)}.csv"
df.to_csv(csv_path, index=False)
print(f"CSV generato: {csv_path}")

# Bootstrap 95% intervallo di confidenza
data = df["mean_score"].values
n_boot = 1000
boot_means = []

for _ in range(n_boot):
    sample = np.random.choice(data, size=len(data), replace=True)
    boot_means.append(np.mean(sample))

boot_means = np.array(boot_means)
ci_lower = np.percentile(boot_means, 2.5)
ci_upper = np.percentile(boot_means, 97.5)
mean_score = np.mean(data)

print(f"Media score: {mean_score:.4f}")
print(f"Intervallo di confidenza 95%: [{ci_lower:.4f}, {ci_upper:.4f}]")

results_summary.append({
```

```
    "conf_thres": conf,
    "mean_score": mean_score,
    "ci_lower": ci_lower,
    "ci_upper": ci_upper
})

Riepilogo confronto soglie
summary_df = pd.DataFrame(results_summary) print("\n=== Riepilogo confronto soglie validation YOLOv8 ===")
print(summary_df)
```

Allegato E – Script Python analisi Bootstrap

```
"""ANALISI BOOTSTRAP"""
from google.colab import drive drive.mount('/content/drive')
Installa librerie
!pip install roboflow ultralytics
from roboflow import Roboflow import os
rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb")
version = project.version(2) dataset = version.download("yolov11")
Verifica immagini
paths = [ dataset.location + "/train/images", dataset.location + "/valid/images", dataset.location + "/test/images" ]
for p in paths: exists = os.path.exists(p) nfiles = len([f for f in os.listdir(p) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0
print(f"{p}: exists={exists}, immagini={nfiles}")
Controlla il file data.yaml
!cat {dataset.location}/data.yaml
Controlla immagini validation
val_path = dataset.location + "/valid/images" exists = os.path.exists(val_path) nfiles = len([f for f in os.listdir(val_path) if f.lower().endswith(('.jpg','.jpeg','.png'))]) if exists else 0 print(f"{val_path}: exists={exists}, immagini={nfiles}")
from ultralytics import YOLO
Carica checkpoint già addestrato
weights_path = "/content/drive/MyDrive/yolov11_results/run_2025-10-09_15-19/validation_results/weights/best.pt" model = YOLO(weights_path)
from ultralytics import YOLO import pandas as pd import numpy as np from collections import defaultdict import os
Config
thresholds = [0.25, 0.5, 0.75] results_summary = []
Percorsi
model_path = "/content/drive/MyDrive/yolov11_results/run_2025-10-09_15-19/validation_results/weights/best.pt" val_images = "/content/cookie-defect-detection-2/valid/images" base_dir = "runs/detect"
Carica modello
model = YOLO(model_path)
for conf in thresholds: val_dir = f"val_conf{int(conf*100)}" print(f"\nElaborazione immagini con conf={conf}...")
# Predict e salva immagini/annotazioni
results = model.predict(source=val_images,
    conf=conf,
    save=True,
    project=base_dir,
    name=val_dir,
    exist_ok=True
)


# Estrai predizioni da results
all_preds = []
for result in results:
    boxes = result.boxes
    image_id = os.path.basename(result.path) # prendi solo il nome del file
```

```
    for i in range(len(boxes)):
        score = float(boxes.conf[i])
        all_preds.append({"image_id": image_id, "score": score})

if not all_preds:
    print(f"Nessuna predizione trovata per conf={conf}")
    continue

print(f"Totale predizioni per conf={conf}: {len(all_preds)}")

# Aggrega per immagine
agg = defaultdict(list)
for p in all_preds:
    agg[p["image_id"]].append(p)

# Costruisci DataFrame per immagine
rows = []
for image_id, boxes in agg.items():
    scores = [b["score"] for b in boxes]
    row = {
        "image_id": image_id,
        "num_detections": len(boxes),
        "mean_score": sum(scores)/len(scores) if scores else 0.0
    }
    rows.append(row)

df = pd.DataFrame(rows)
csv_path = f"metrics_per_image_val_conf{int(conf*100)}.csv"
df.to_csv(csv_path, index=False)
print(f"CSV generato: {csv_path}")

# Bootstrap 95% intervallo di confidenza
data = df["mean_score"].values
n_boot = 1000
boot_means = []

for _ in range(n_boot):
    sample = np.random.choice(data, size=len(data), replace=True)
    boot_means.append(np.mean(sample))

boot_means = np.array(boot_means)
ci_lower = np.percentile(boot_means, 2.5)
ci_upper = np.percentile(boot_means, 97.5)
```

```
mean_score = np.mean(data)

print(f"Media score: {mean_score:.4f}")
print(f"Intervallo di confidenza 95%: [{ci_lower:.4f}, {ci_upper:.4f}]")

results_summary.append({
    "conf_thres": conf,
    "mean_score": mean_score,
    "ci_lower": ci_lower,
    "ci_upper": ci_upper
})

Riepilogo confronto soglie
summary_df = pd.DataFrame(results_summary) print("\n=== Riepilogo confronto soglie validation YOLOv8 ===")
print(summary_df)
```

Allegato F – Script Python logica bundle

```
"""**LOGICA DEGLI SCARTI:**
I biscotti considerati lievemente rotti (confidence<=0.4) vengono flaggati come tali e le loro coordinate sul nastro trasportatore vengono salvate in un file .txt. Ogni 15 biscotti considerati "normali", quindi senza alcun tipo di difetto, l'algoritmo assegna al batch due biscotti lievemente rotti, così da limitarne gli sprechi e gli scarti di produzione. """
from ultralytics import YOLO
Costruire il modello basandosi sul file in cui sono salvati i pesi migliori del training
model = YOLO("/content/drive/MyDrive/yolov8_runs/cookie_yolov8_stress/weights/best.pt")
from roboflow import Roboflow
rf = Roboflow(api_key="j7ZS0CHQB9GpDI5UMFrS") project = rf.workspace("tesi-wltpv").project("cookie-defect-detection-6fojb")
version = project.version(2) dataset = version.download("yolov8")
test_path = dataset.location + "/test/images"
results_test = model.predict( source=test_path, conf=0.25, save=True, # salva immagini annotate project="/content/drive/MyDrive/infer_yolov8", name="cookie_test_stress", save_txt=True # genera immagini annotate così da poterle riutilizzare con eventuali braccia robotiche )
from pathlib import Path
results_dir = Path("/content/drive/MyDrive/infer_yolov8/cookie_test_stress2/labels")
Parametri
confidence_threshold = 0.6 # soglia per considerare "lievemente rotto" allowed_cracked_rule = 2 # quanti lievemente rotti si possono aggiungere se ci sono almeno 15 perfetti normal_threshold = 15 # soglia di biscotti senza difetti
Inizializzazione dei contatori
totals = {"Normal cracker": 0, "Over baked": 0, "Cracked": 0} lievemente_rotti_conf = 0
for txt_file in results_dir.glob("*.txt"): with open(txt_file) as f: for line in f: parts = line.strip().split() if len(parts) < 2: continue cls_id = int(parts[0]) conf = float(parts[1])
    if cls_id == 0:
        totals["Normal cracker"] += 1
    elif cls_id == 1:
        totals["Over baked"] += 1
    elif cls_id == 2:
        totals["Cracked"] += 1
        if conf <= confidence_threshold:
            lievemente_rotti_conf += 1

Applicazione della regola dei 2 lievemente rotti
normal_count = totals["Normal cracker"] allowed_cracked = allowed_cracked_rule if normal_count >= normal_threshold else 0 scarti_cracked = max(0, lievemente_rotti_conf - allowed_cracked)
Scarti totali
scarti_senza_logica = totals["Over baked"] + lievemente_rotti_conf scarti_con_logica = totals["Over baked"] + scarti_cracked
Risultati
print("Conteggio totale biscotti:", totals) print(f"Biscotti lievemente rotti considerati (conf <= {confidence_threshold}): {lievemente_rotti_conf}") print(f"Scarti senza logica: {scarti_senza_logica}") print(f"Scarti con logica: {scarti_con_logica}")
Risparmio percentuale
risparmio = scarti_senza_logica - scarti_con_logica percentuale = (risparmio / scarti_senza_logica) * 100 if scarti_senza_logica > 0 else 0 print(f"Risparmio di scarti applicando la logica: {risparmio} biscotti ({percentuale:.2f}%)")
```

```
"""Primo metodo di visualizzazione grafica"""
import matplotlib.pyplot as plt
labels = ['Scarti'] senza_logica = [scarti_senza_logica] con_logica = [scarti_con_logica]
x = range(len(labels)) plt.bar(x, senza_logica, width=0.4, label='Senza logica', align='center') plt.bar(x, con_logica, width=0.4, label='Con logica', align='edge') plt.ylabel('Numero biscotti') plt.title('Effetto della regola dei 2 lievemente rotti') plt.legend() plt.show()
"""Secondo metodo visualizzazione grafica"""
import matplotlib.pyplot as plt
labels = ['Scarti'] senza_logica = [scarti_senza_logica] con_logica = [scarti_con_logica]
x = range(len(labels)) width = 0.35
Crea figura
fig, ax = plt.subplots(figsize=(6,5))
Barre
bars1 = ax.bar([i - width/2 for i in x], senza_logica, width, label='Senza logica', color='#FF6F61') bars2 = ax.bar([i + width/2 for i in x], con_logica, width, label='Con logica', color='#6B5B95')
Annotazioni sopra le barre
for bar in bars1 + bars2: height = bar.get_height() ax.text(bar.get_x() + bar.get_width()/2, height + 0.2, f'{int(height)}', ha='center', va='bottom', fontsize=12)
Personalizzazioni
ax.set_ylabel('Numero biscotti', fontsize=12) ax.set_title('Effetto della regola dei 2 lievemente rotti', fontsize=14, fontweight='bold') ax.set_xticks(x) ax.set_xticklabels(labels, fontsize=12) ax.legend(fontsize=12) ax.set_ylim(0, max(scarti_senza_logica, scarti_con_logica)*1.3) # lascia spazio per le annotazioni ax.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout() plt.show()
```

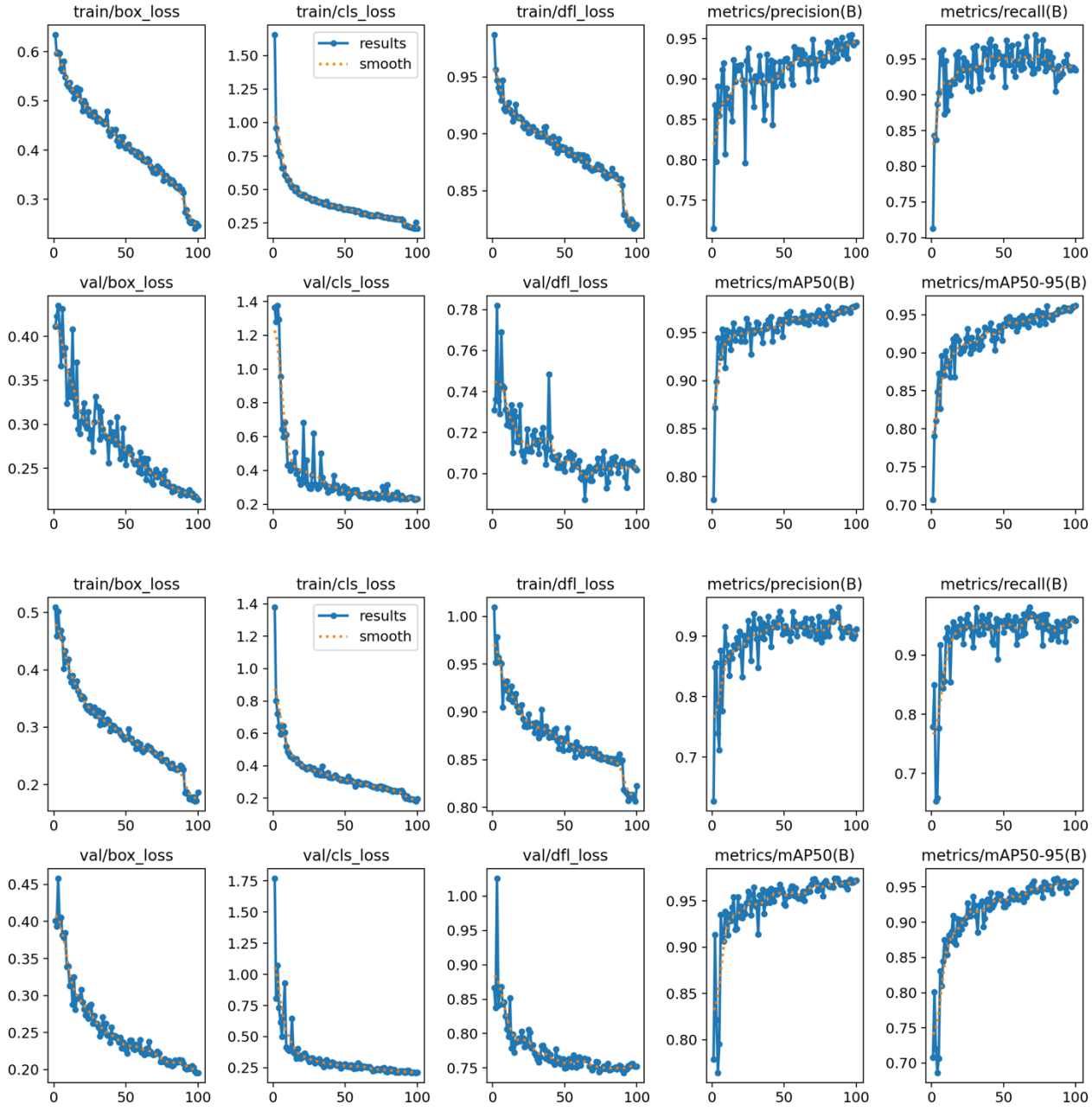
Allegato G – Script analisi autori articoli di ricerca

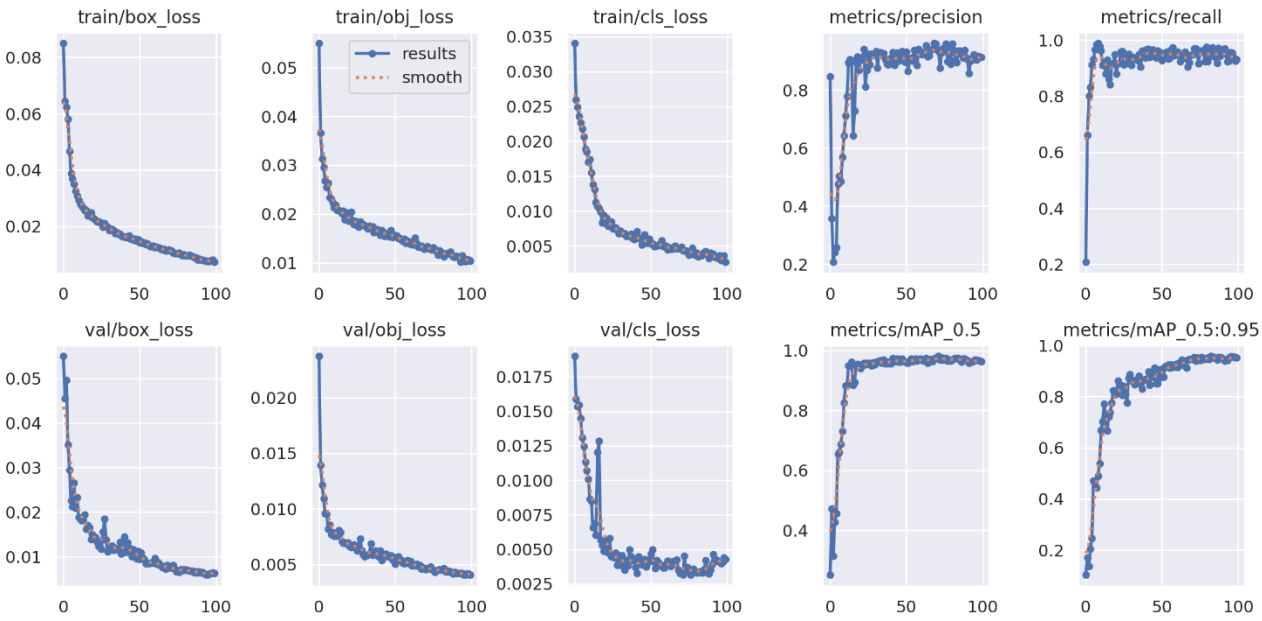
```
import pandas as pd import networkx as nx from pyvis.network import Network from community import community_louvain import os
file_path = "/Users/luciaruffati/Library/CloudStorage/OneDrive-PolitecnicodiTorino/Tesi Magistrale/PRISMA copia.xlsx" foglio = "Considerabili_Rilevanti" colonna_autori = "Authors" separatore_autori = ";"
Output
base_dir = os.path.dirname(file_path) output_dir = os.path.join(base_dir, "BNA output") os.makedirs(output_dir, exist_ok=True)
df = pd.read_excel(file_path, sheet_name=foglio, engine="openpyxl") df = df.dropna(subset=[colonna_autori]) df["Riga"] = df.index
# Per riferimenti incrociati
Grafo co-autori
G = nx.Graph()
for _, row in df.iterrows(): autori = [a.strip() for a in str(row[colonna_autori]).split(separatore_autori)] for i in range(len(autori)): for j in range(i + 1, len(autori)): G.add_edge(autori[i], autori[j])
print(f"Nodi (autori): {G.number_of_nodes()}") print(f"Collegamenti (co-autorship): {G.number_of_edges()}")
centralità e partition
degree_dict = dict(G.degree()) partition = community_louvain.best_partition(G) degree_df = pd.DataFrame.from_dict(degree_dict, orient="index", columns=["Degree"]) community_df = pd.DataFrame.from_dict(partition, orient="index", columns=["Community"])
results = degree_df.join(community_df) results.index.name = "Author" results.reset_index(inplace=True)
results_path = os.path.join(output_dir, "analisi_autori.csv") results.to_csv(results_path, index=False) print(f"Risultati salvati in '{results_path}'")
Autori isolati
isolati = results[results["Degree"] <= 1] isolati_path = os.path.join(output_dir, "autori_isolati.csv") isolati.to_csv(isolati_path, index=False) print(f"Autori isolati salvati in '{isolati_path}'")
net = Network(notebook=False, width="100%", height="800px", bgcolor="#ffffff", font_color="black") net.barnes_hut()
for node, deg in degree_dict.items(): net.add_node(node, label=node, title=f"Degree: {deg} | Community: {partition[node]}", value=deg, group=partition[node])
for edge in G.edges(): net.add_edge(edge[0], edge[1])
rete_path = os.path.join(output_dir, "rete_coautori.html") net.write_html(rete_path) print(f"Rete salvata in '{rete_path}'")
```

Allegato H – Script Python analisi geografica

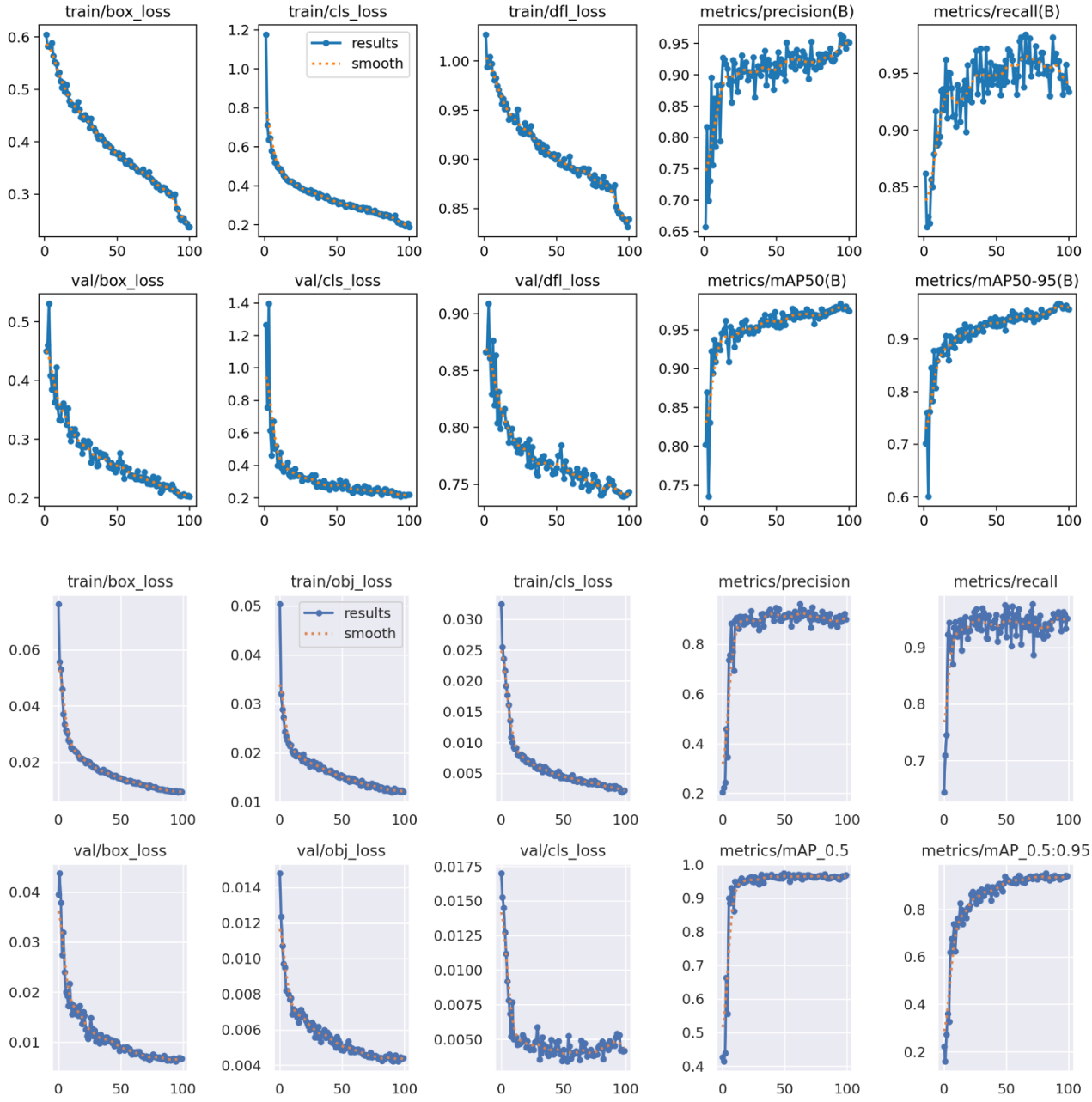
```
import pandas as pd
import plotly.express as px
# Dati dei Paesi
data = [#inserimento dataset letto da Excel]
df = pd.DataFrame(data, columns=["Paese"])
df_counts = df["Paese"].value_counts().reset_index()
df_counts.columns = ["Paese", "NumeroArticoli"]
# Mappa geografica interattiva
fig = px.choropleth(
    df_counts,
    locations="Paese",
    locationmode="country names",
    color="NumeroArticoli",
    hover_name="Paese",
    color_continuous_scale=px.colors.sequential.Tealgrn,
    title="Distribuzione Geografica degli Studi Analizzati",
)
# Personalizzazione layout
fig.update_layout(
    title_font=dict(size=26, family="Helvetica Neue, sans-serif", color="#222"),
    font=dict(size=16, family="Helvetica Neue, sans-serif", color="#333"),
    geo=dict(
        showframe=False,
        showcoastlines=True,
        coastlinecolor="LightGray",
        projection_type="natural earth"
    ),
    coloraxis_colorbar=dict(
        title="N° Articoli",
        titlefont=dict(size=16),
        tickfont=dict(size=14)),
    margin=dict(l=0, r=0, t=60, b=0)
)
# Mostra grafico interattivo
fig.show()
```

Allegato I – Grafici output Loss modelli YOLOv5/8/11 versione normale



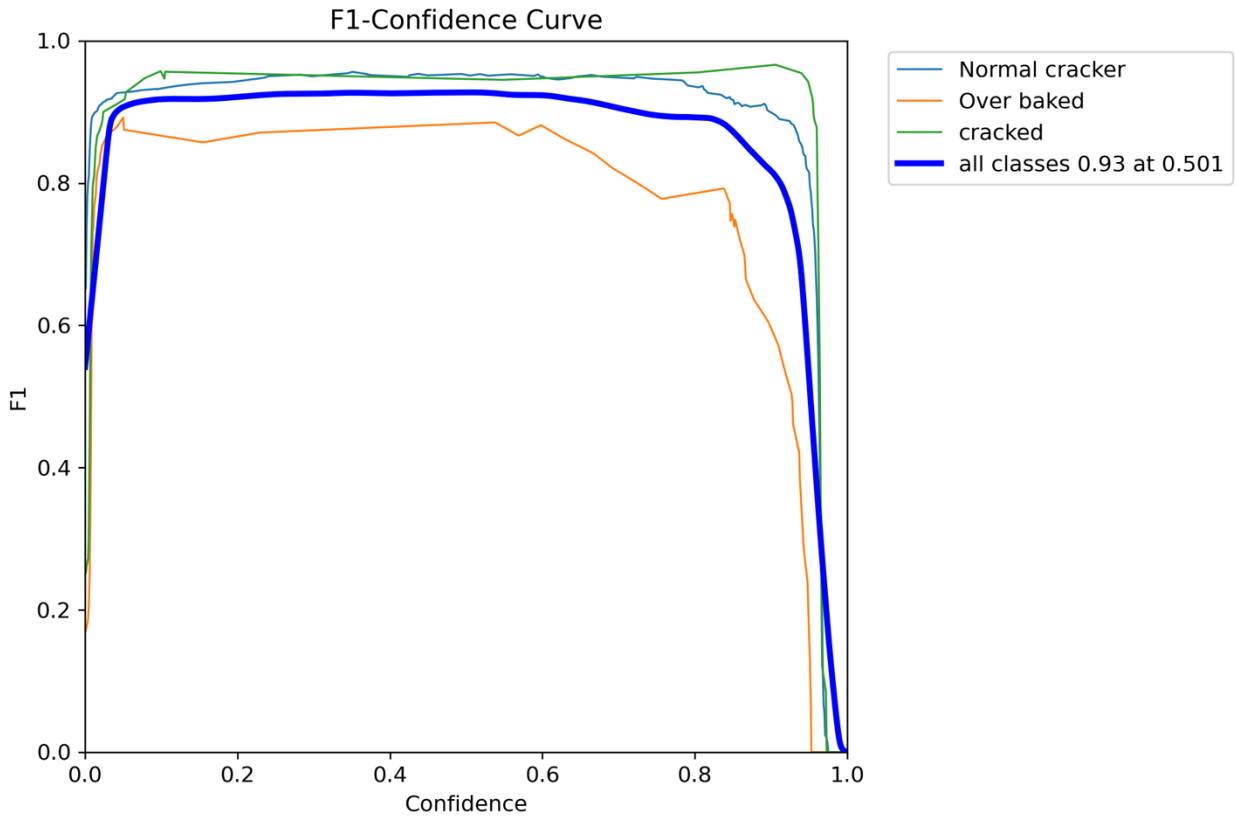

train/box_loss
train/obj_loss
results
smooth
train/cls_loss
metrics/precision
metrics/recall
val/box_loss
val/obj_loss
val/cls_loss
metrics/mAP_0.5
metrics/mAP_0.5:0.95

Allegato L – Grafici output Loss modelli YOLOv5/8/11 versione stressata

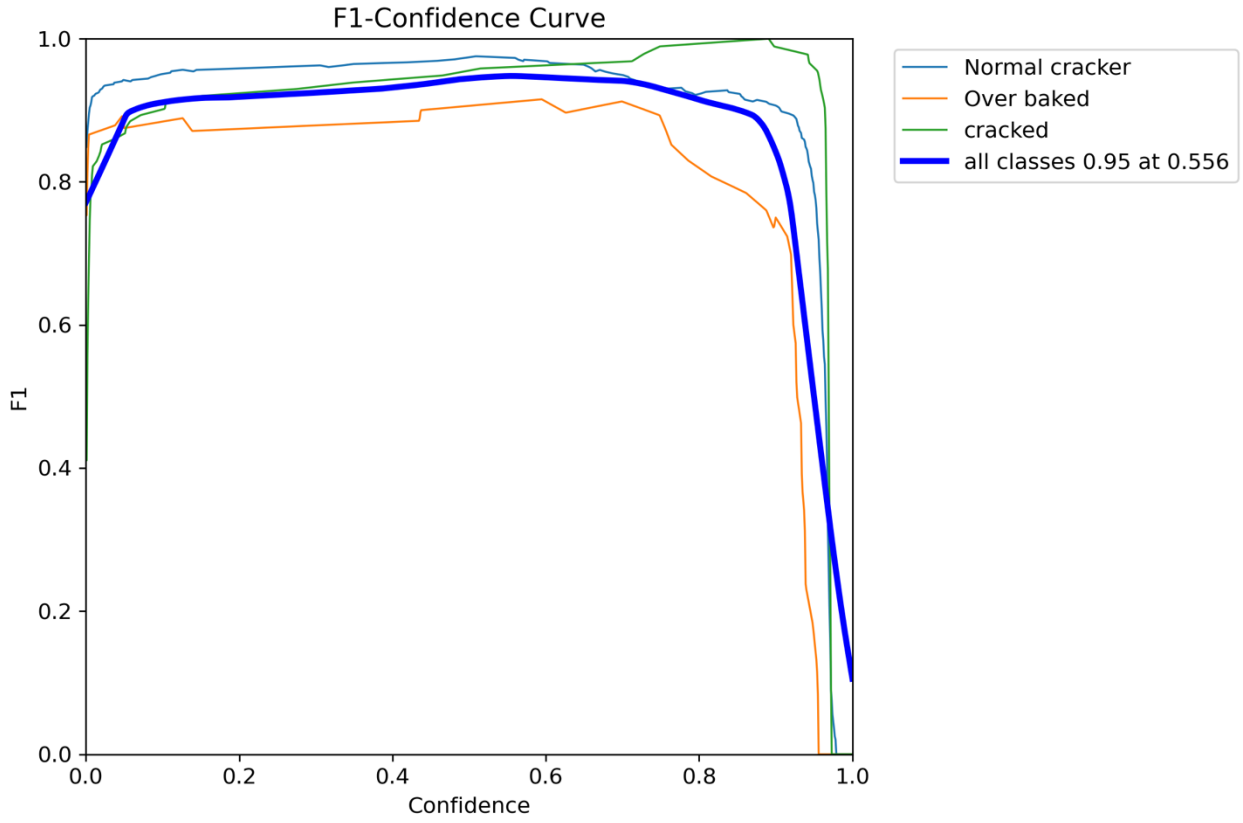


Allegato M – Grafici metriche valutazione modelli YOLO versione stressata

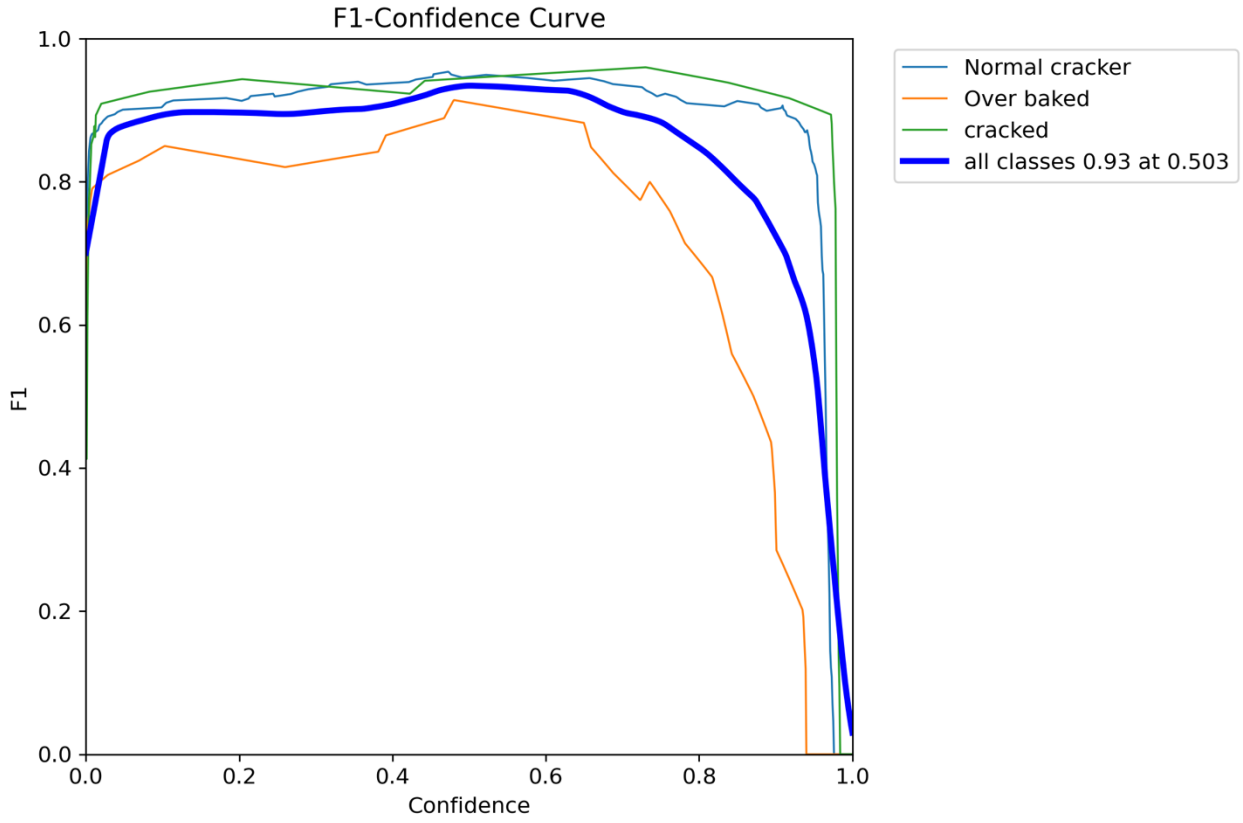
YOLOv5 F1



YOLOv8 F1


F1-Confidence Curve
Normal cracker
Over baked
cracked
all classes 0.95 at 0.556
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0

YOLOv11 F1


F1-Confidence Curve
Normal cracker
Over baked
cracked
all classes 0.93 at 0.503
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0

Bibliografia

[1] S. Abdanan Mehdizadeh, “Machine vision based intelligent oven for baking inspection of cupcake: Design and implementation,” *Mechatronics*, vol. 82, Apr. 2022, doi: 10.1016/j.mechatronics.2022.102746.

[2] A. Khan, M. T. Munir, W. Yu, and B. R. Young, “A Review Towards Hyperspectral Imaging for Real-Time Quality Control of Food Products with an Illustrative Case Study of Milk Powder Production,” May 01, 2020, *Springer*. doi: 10.1007/s11947-020-02433-w.

[3] *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. IEEE, 2012.

[4] M. J. Page *et al.*, “PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews,” Mar. 29, 2021, *BMJ Publishing Group*. doi: 10.1136/bmj.n160.

[5] B. Takahashi, I. A. dos Santos, and M. F. Salas, “Building Bridges: A Narrative Literature Review of Spanish and Portuguese-Language Climate Change Communication Scholarship from Latin America,” *Environ Commun*, vol. 19, no. 5, pp. 1016–1030, 2025, doi: 10.1080/17524032.2025.2458229.

[6] “19621072”.

[7] “https://link.springer.com/article/10.1007/s11192-015-1765-5.”

[8] “PRISMA flowchart,” https://www.prisma-statement.org/prisma-2020-flow-diagram.

[9] A. Paez, “Gray literature: An important resource in systematic reviews,” *J Evid Based Med*, vol. 10, no. 3, pp. 233–240, Aug. 2017, doi: 10.1111/jebm.12266.

[10] “ChatGPT,” https://chatgpt.com/.

[11] “The lens,” https://www.lens.org/.

[12] “pmc_9131190”.

[13] O. Kılcı and M. Koklu, “Automated Classification of Biscuit Quality Using YOLOv8 Models in Food Industry,” *Food Anal Methods*, vol. 18, no. 5, pp. 815–829, May 2025, doi: 10.1007/s12161-025-02755-5.

[14] Y. Taspinar, “CLASSIFICATION OF BISCUIT DEFECT STATES AND FOREIGN OBJECTS USING CNN-BASED FEATURES,” Oct. 2022.

[15] L. Zhu, P. Spachos, E. Pensini, and K. N. Plataniotis, “Deep learning and machine vision for food processing: A survey,” Jan. 01, 2021, *Elsevier B.V.* doi: 10.1016/j.crfs.2021.03.009.

[16] V. Gökmen and B. A. Mogol, “Computer vision-based image analysis for rapid detection of acrylamide in heated foods,” *Quality Assurance and Safety of Crops and Foods*, vol. 2, no. 4, pp. 203–207, 2010, doi: 10.1111/j.1757-837X.2010.00072.x.

[17] P. Tantiphanwadi and K. Malithong, “Bread Browning Stage Classification Model using VGG-16 Transfer Learning and Fine-tuning with Small Training Dataset,” *Engineering Journal*, vol. 26, no. 11, Nov. 2022, doi: 10.4186/EJ.2022.26.11.1.

[18] J. Rohan Joel and R. M. Benjamin, “Bread Quality Assessment Using Deep Learning with shape and Volume Metrics,” in *10th International Conference on Advanced Computing and Communication Systems, ICACCS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 2125–2130. doi: 10.1109/ICACCS60874.2024.10717194.

[19] B. A. Mogol and V. Gökmen, “Computer vision-based analysis of foods: A non-destructive colour measurement tool to monitor quality and safety,” *J Sci Food Agric*, vol. 94, no. 7, pp. 1259–1263, 2014, doi: 10.1002/jsfa.6500.

[20] C. Drogalis, C. Zampino, and V. Chauhan, “FOOD QUALITY INSPECTION AND SORTING USING MACHINE VISION, MACHINE LEARNING AND ROBOTICS,” 2023. [Online]. Available: http://asmedigitalcollection.asme.org/IMECE/proceedings-pdf/IMECE2023/87608/V003T03A066/7238663/v003t03a066-imece2023-113496.pdf

[21] F. Rezagholi and M. A. Hesarinejad, “Integration of fuzzy logic and computer vision in intelligent quality control of celiac-friendly products,” in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 325–332. doi: 10.1016/j.procs.2017.11.246.

[22] K. Mannaro *et al.*, “A Robust SVM Color-Based Food Segmentation Algorithm for the Production Process of a Traditional Carasau Bread,” *IEEE Access*, vol. 10, pp. 15359–15377, 2022, doi: 10.1109/ACCESS.2022.3147206.

[23] K. Tutuncu, E. T. Yasin, and M. Koklu, “Enhancing Quality Control: Defect State Classification of Taralli Biscuits with MobileNet-v2 and DenseNet-201,” in *Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 718–723. doi: 10.1109/IDAACS58523.2023.10348851.

[24] X. Li, S. Xue, Z. Li, X. Fang, T. Zhu, and C. Ni, “A Candy Defect Detection Method Based on StyleGAN2 and Improved YOLOv7 for Imbalanced Data,” *Foods*, vol. 13, no. 20, Oct. 2024, doi: 10.3390/foods13203343.

[25] J. Lee, Y. Kim, and S. Kim, “The Study of an Adaptive Bread Maker Using Machine Learning,” *Foods*, vol. 12, no. 22, Nov. 2023, doi: 10.3390/foods12224160.

[26] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” Dec. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/make5040083.

[27] S. W. Chen, C. J. Tsai, C. H. Liu, W. C. C. Chu, and C. T. Tsai, “Development of an Intelligent Defect Detection System for Gummy Candy under Edge Computing,” *Journal of Internet Technology*, vol. 23, no. 5, pp. 981–988, 2022, doi: 10.53106/160792642022092305006.

[28] N. O’ Mahony *et al.*, “Deep Learning vs. Traditional Computer Vision.”

[29] “Ultralytics.” Accessed: Oct. 06, 2025. [Online]. Available: https://docs.ultralytics.com/

[30] “Detectron AI,” https://ai.meta.com/tools/detectron2/.

[31] “MMDetection,” https://mmdetection.readthedocs.io/en/latest/.

[32] “MVTech,” https://www.mvtec.com/products/halcon.

[33] “Cognex,” https://www.cognex.com/it-it.

[34] “Matrox,” https://visionlink.it/matrox-imaging-library-sdk/.

[35] “Keyence,” https://www.keyence.it/.

[36] “Viso AI,” https://viso.ai/.

[37] “Robovision,” https://robovision.ai/.

[38] “Deepomatic,” https://deepomatic.com/.

[39] S. Makni, I. Fourati Kallel, and M. A. Triki, “Intelligent inspection and quality control of table olives,” 2024. [Online]. Available: www.jart.icat.unam.mx

[40] K. K. Patel and P. B. Pathare, “Principle and applications of near-infrared imaging for fruit quality assessment—An overview,” May 01, 2024, *John Wiley and Sons Inc*. doi: 10.1111/ijfs.16862.

[41] B. Dhiman, Y. Kumar, and M. Kumar, “Fruit quality evaluation using machine learning techniques: review, motivation and future perspectives,” *Multimed Tools Appl*, vol. 81, no. 12, pp. 16255–16277, May 2022, doi: 10.1007/s11042-022-12652-2.

[42] H. Suhendar, V. Efelina, and M. Ziveria, “Fruit Quality Classification using Convolutional Neural Network,” in *Journal of Physics: Conference Series*, Institute of Physics, 2022. doi: 10.1088/1742-6596/2377/1/012015.

[43] P. R. Kiran, P. Aradwad, T. V. Arunkumar, and R. A. Parray, “Enhancing mango quality control: A novel approach to spongy tissue inspection through image clustering and machine learning models via X-ray imaging,” *J Food Process Eng*, vol. 47, no. 6, Jun. 2024, doi: 10.1111/jfpe.14664.

[44] R. Maru, A. Kamat, M. Shah, P. Kute, S. C. Shrawne, and V. Sambhe, “Improved Faster RCNN for Ripeness and Size Estimation of Mangoes with multi-label output,” in *CINS 2024 - 2nd*

International Conference on Computational Intelligence and Network Systems, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/CINS63881.2024.10864434.

[45] V. Ashok, N. Sheela, and R. K. Bharathi, "An Automatic Non-Destructive External and Internal Quality Evaluation of Mango Fruits based on Color and X-ray Imaging using Computer Vision Techniques," *Inteligencia Artificial*, vol. 26, no. 72, pp. 223–243, Dec. 2023, doi: 10.4114/INTARTIF.VOL26ISS72PP223-243.

[46] J. Agrawal, K. Singh Gill, R. Chauhan, H. Singh Pokhariya, and K. R. Chythanya, "Revolutionizing Raisin Processing through Advanced Sorting and Visualization with a Deep Learning Based Support Vector Machine Model," in *2024 Asia Pacific Conference on Innovation in Technology, APCIT 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/APCIT62007.2024.10673463.

[47] S. N. Nobel *et al.*, "Categorization of Dehydrated Food through Hybrid Deep Transfer Learning Techniques," *Statistics, Optimization and Information Computing*, vol. 12, no. 4, pp. 1004–1018, 2024, doi: 10.19139/soic-2310-5070-1896.

[48] M. O. Ullah, M. I. Nazir, A. Akter, S. A. Raju, and M. S. R. Oion, "Dry Food Classification using Hybrid Deep Transfer Learning," in *2023 26th International Conference on Computer and Information Technology, ICCIT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCIT60459.2023.10441650.

[49] J. J. Pangaribuan, J. Thames, A. E. Widjaja, O. Putra Barus, and C. A. Haryani, "Classification of Dry Nut Types Using the Support Vector Machine Method," in *2024 3rd International Conference on Creative Communication and Innovative Technology, ICCIT 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICCIT62134.2024.10701219.

[50] R. Pillai, N. Sharma, and R. Gupta, "Classification of Pista Species using Fine-Tuned EfficientNetB3 Transfer Learning Model," in *2023 IEEE International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering, RMKMATE 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/RMKMATE59243.2023.10369347.

[51] T. A. Cengel, B. Gencturk, E. T. Yasin, M. B. Yildiz, I. Cinar, and M. Koklu, "Automating egg damage detection for improved quality control in the food industry using deep learning," *J Food Sci*, vol. 90, no. 1, Jan. 2025, doi: 10.1111/1750-3841.17553.

[52] B. Botta and A. K. Datta, "Deep transfer learning-based approach for detection of cracks on eggs," *J Food Process Eng*, vol. 46, no. 11, Nov. 2023, doi: 10.1111/jfpe.14425.

[53] J. O. Adigun, F. M. Okikiola, E. E. Aigbokhan, and M. M. Rufai, "Automated system for grading apples using convolutional neural network," *International Journal of Innovative*

Technology and Exploring Engineering, vol. 9, no. 1, pp. 1458–1464, Nov. 2019, doi: 10.35940/ijitee.A4246.119119.

[54] Yashu, V. Kukreja, T. P. S. Brar, and V. Jain, "Apple Variety Classification: Leveraging CNN and Random Forest for Precision," in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking, SMART GENCON 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/SMARTGENCON60755.2023.10441902.

[55] B. J. Bipin Nair, D. Rushab, and S. Kruthik, "A Comparison of Deep Learning Based Methods for Grading Papaya Fruit by Maturity Level," in *Proceedings of the 2024 3rd Edition of IEEE Delhi Section Flagship Conference, DELCON 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/DELCON64804.2024.10867117.

[56] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," Dec. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.array.2022.100258.

[57] S. B. Dhal and D. Kar, "Leveraging artificial intelligence and advanced food processing techniques for enhanced food safety, quality, and security: a comprehensive review," Jan. 01, 2025, *Springer Nature*. doi: 10.1007/s42452-025-06472-w.

[58] M. Subramanian, S. Sree Varagi, T. S. Kumar, K. Sowmiya, and N. Prashithaa, "IoT-Enhanced Quality Bread Assurance System(IQBAS)," in *2023 Intelligent Computing and Control for Engineering and Business Systems, ICCEBS 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCEBS58601.2023.10448554.

[59] T. Reichenstein, T. Raffin, C. Sand, and J. Franke, "Implementation of Machine Vision based Quality Inspection in Production: An Approach for the Accelerated Execution of Case Studies," *Procedia CIRP*, vol. 112, pp. 596–601, 2022, doi: https://doi.org/10.1016/j.procir.2022.09.058.

[60] "Ultralytics (YOLOv5/8/11)," https://docs.ultralytics.com/it/.

[61] B. , N. J. , H. T. , et al. Dwyer, "Roboflow (Version 1.0) [Software]," Available from https://roboflow.com.

[62] Tesi, "Cookie Defect Detection Dataset," https://universe.roboflow.com/tesi-wltpv/cookie-defect-detection-6fojb.

[63] "Licenza Dataset," https://creativecommons.org/licenses/by/4.0/.

[64] S. Sudharson, P. Kokil, R. Annamalai, and N. V. Sai Manoj, "Enhanced Food Classification System Using YOLO Models For Object Detection Algorithm," in *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCCNT56998.2023.10307943.

[65] S. Fairus Mokhtar, Z. M. Yusof, and H. Sapiri, “Confidence Intervals by Bootstrapping Approach: A Significance Review,” 2023.

[66] S. Moggi, S. Bonomi, and F. Ricciardi, “Against food waste: CSR for the social and environmental impact through a network-based organizational model,” *Sustainability (Switzerland)*, vol. 10, no. 10, Sep. 2018, doi: 10.3390/su10103515.

[67] T. Wuest, D. Weimer, C. Irgens, and K. D. Thoben, “Machine learning in manufacturing: Advantages, challenges, and applications,” *Prod Manuf Res*, vol. 4, no. 1, pp. 23–45, Jun. 2016, doi: 10.1080/21693277.2016.1192517.

[68] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski, “Latency and Throughput Characterization of Convolutional Neural Networks for Mobile Computer Vision,” *arXiv e-prints*, p. arXiv:1803.09492, Mar. 2018, doi: 10.48550/arXiv.1803.09492.

[69] F. Lehn and T. Schmidt, “Sustainability Assessment of Food-Waste-Reduction Measures by Converting Surplus Food into Processed Food Products for Human Consumption,” *Sustainability (Switzerland)*, vol. 15, no. 1, Jan. 2023, doi: 10.3390/su15010635.

Indice delle tabelle