Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di laurea

Progettazione e sviluppo di una Metodologia Data-Driven di Quality Inspection nel contesto dell'Industria 4.0

Autore

Vincenzo Topazio

Relatore

Prof. Tania Cerquitelli

Anno Accademico 2019/2020

Abstract

Oggi molte aziende manifatturiere effettuano i controlli di qualità sui prodotti in modo manuale o basandosi su tecniche approssimative. Grandi realtà industriali hanno investito molto su tecnologie innovative per assicurare la qualità dei loro prodotti, ma altre aziende stentano a innovarsi per via dello scoglio dei costi di sviluppo.

L'obiettivo di questa tesi è quello di proporre la Proof of Concept (PoC) di un sistema progettato per l'ispezione di qualità e la rilevazione di difetti nei processi produttivi delle industrie di oggi. L'approccio scelto è quello di sfruttare il modello di Cloud Computing applicato al machine learning per offrire alle aziende manifatturiere un servizio di controllo della qualità flessibile ed economicamente accessibile, senza pregiudicarne le performance.

Per far ciò sono discusse le tecniche allo stato dell'arte per la quality inspection in generale e con il supporto di algoritmi di machine learning. A partire da quest'analisi si è studiato come i processi produttivi possano essere trasformati e migliorati, disegnando un architettura completa che includesse un'applicazione web intermediaria tra un servizio di classificazione remoto e il software di acquisizione immagini e segnalazione difetti installato su di un computer edge nella linea di produzione.

Sono state infine misurate le prestazioni della metodologia proposta. Sono stati considerati e analizzati vari casi di test attraverso l'uso di metriche di valutazione note nell'ambito della classificazione automatica. I risultati ottenuti sono promettenti, in quanto allineati alle aspettative iniziali. Su questa base, è possibile raffinare il funzionamento di quanto proposto per applicazioni reali.

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno aiutato nella realizzazione di questa tesi. Ringrazio la professoressa Tania Cerquitelli, relatore, per aver accettato di sostenere questa tesi e per il supporto offerto alla sua stesura.

Ringrazio i dipendenti della manufacturing busiess unit di Blue Reply per il continuo supporto ricevuto allo sviluppo delle applicazioni di questa tesi. In particolare Andrea Mastroberardino per l'organizzazione del piano e la disponibilità offerta alla risoluzione di dubbi, Fabio Raimondi per avermi offerto la proposta di questa tesi e Federico Paradiso per il restyling grafico dell'applicazione web.

Un ringraziamento va ai colleghi che mi hanno supportato durante i mesi spesi a sviluppare e scrivere questa tesi. Infine vorrei ringraziare la mia famiglia e in particolare i miei genitori per avermi permesso di portare a termine questo percorso.

Indice

Capitolo 1

Introduzione

Nell'industria manifatturiera, il **controllo di qualità** è oggi considerato parte integrante dei processi produttivi da cui non si può prescindere. Esso è stato inizialmente assente nelle fabbriche dei primi del '900. Con l'incrementare dei volumi di produzione, le aziende hanno dovuto sostenere costi sempre più grandi nella sostituzione di prodotti difettosi. Frederick Winslow Taylor [1] ha cambiato l'approccio alla definizione dei processi produttivi, introducendo il metodo scientifico nella gestione dei lavoratori. Questo cambiamento di paradigma ha portato a sua volta, oltre che all'evoluzione dei processi industriali strettamente detti, all'introduzione della gestione di qualità. Sono comparsi dei nuovi ruoli dedicati al controllo della qualità: un controllore ha la mansione di ispezionare i prodotti. Ciò riduce complessivamente i costi della gestione di unità difettose, andando a scartarle o ripararle prima che finiscano al cliente.

L'altro fattore che ha cambiato in meglio i processi industriali è l'**automazione**. Il continuo sviluppo delle tecnologie meccaniche e informatiche durante il ventesimo secolo ha reso l'automazione industriale un fattore determinante per la competitività sul mercato. A livello software, i sistemi informativi aziendali hanno permesso di ridurre drasticamente gli errori umani nelle varie funzioni aziendali: ad esempio i software di tipo Enterprise Resource Planning (ERP) integrano in modo congiunto i principali processi aziendali come la gestione dei fornitori, del magazzino, della contabilità, distribuzione, vendita e supporto post-vendita dei prodotti. L'uso di macchinari e robot, guidati da algoritmi, ha contribuito anch'esso all'abbattimento dei costi produttivi su larga scala. La branca che si occupa di elaborare automaticamente informazioni visive, detta Machine Vision, ha permesso di utilizzare algoritmi sempre più efficienti e indipendenti dall'intervento umano. Negli ultimi anni il campo dell'intelligenza artificiale (IA), specialmente quella basata sui dati raccolti (machine learning), ha aperto nuove frontiere nell'automazione dei processi produttivi.

Una delle sfide di oggi è quella di automatizzare l'ispezione di qualità, per ridurre ulteriormente sia i costi per la sostituzione di prodotti difettosi che il personale addetto alla supervisione, in modo da distribuirlo in compiti meno ripetitivi. Molte aziende manifatturiere di grandi dimensioni hanno già investito e sviluppato soluzioni interessanti nel campo dell'ispezione automatizzata di qualità ma, di contro, aziende di più piccole dimensioni sono riluttanti ad investire su queste tecnologie per via dei costi di sviluppo e della relativa acerbità delle soluzioni già presenti sul mercato.

L'obiettivo di questa tesi è quello di proporre la Proof of Concept (PoC) di un sistema progettato per l'ispezione di qualità e la rilevazione di difetti nei processi produttivi delle industrie di oggi. L'approccio scelto è quello di sfruttare il modello di Cloud Computing applicato al machine learning per offrire alle aziende manifatturiere un servizio di controllo della qualità flessibile ed economicamente accessibile, senza pregiudicarne le performance. Utilizzando l'accuratezza come parametro di valutazione della classificazione, è necessario raggiungere un valore di 0,95 per considerare affidabile la soluzione proposta.

Prima di iniziare la fase di progettazione, nel *Capitolo 2* sono ripresi i concetti teorici necessari e le tecniche allo stato dell'arte per la quality inspection, in generale e con il supporto di algoritmi di machine learning. Ci si è focalizzati sulle recenti tecnologie di deep learning e in particolare sulle reti neurali convoluzionali, che ad oggi sono le più accurate in letteratura per la classificazione di immagini. È inoltre discusso il paradigma di Deep Learning As A Service, che offre la possibilità alle aziende di esternalizzare la gestione dell'hardware e dei framework relativi alla classificazione di immagini mediante reti neurali convoluzionali.

La progettazione del sistema proposto nel *Capitolo 3* prende in considerazione i tre attori del servizio, che sono la business function manifatturiera del cliente, il fornitore del servizio di quality inspection e del servizio di classificazione. Definiti i soggetti, si è studiato successivamente come i processi produttivi possano essere trasformati e migliorati, andando a progettare un architettura completa. La struttura del sistema proposto comprende un'applicazione web intermediaria tra un servizio di classificazione remoto e il software di acquisizione immagini installato su di un computer edge nella linea di produzione.

Nel *Capitolo 4* si è scesi nei dettagli implementativi della soluzione. Vengono mostrate le principali funzionalità delle due applicazioni sviluppate e sono spiegati gli algoritmi che stanno dietro alle operazioni disponibili. Il *Capitolo 5*, dopo una breve introduzione dei principali metodi di valutazione dei classificatori, mostra quali sono i risultati della valutazione del sistema. Nel *Capitolo 6* sono riassunti e discussi i risultati ottenuti in questo studio, ne sono mostrate le limitazioni e i possibili sviluppi futuri.

Capitolo 2

Quality Inspection, Machine Vision e Deep Learning

In questo capitolo verranno offerti gli spunti teorici su cui basare la progettazione e lo sviluppo del sistema di Quality Inspection oggetto di questa tesi, a partire dal processo di Quality Inspection fino alle moderne tecniche di elaborazione delle immagini.

2.1 Quality Inspection

La **Quality Inspection** è il processo manifatturiero che ha il compito di valutare la conformità di un prodotto. La qualità di un prodotto è determinata sia da come esso è stato progettato che da quanto esso sia vicino alle specifiche di progetto: l'obiettivo del processo di quality inspection è quello di soddisfare la seconda. La valutazione può avvenire durante o in una fase successiva alla produzione. Questo è il compito di supervisori specializzati, che ad occhio o mediante misurazioni danno un giudizio sui prodotti analizzati e decidono se bisogna scartarli o meno.

La rilevazione dei difetti nei prodotti ha l'ulteriore scopo di ridurre i problemi di qualità futuri. I dati relativi alle rilevazioni sono inviati ai manager, che analizzano l'ambiente di produzione per cercare di ridurre o rimuovere del tutto la causa del difetto. Il processo più generale che include il meccanismo di feedback appena esposto è chiamato **Quality control**.

2.1.1 Metodologie di controllo

Allo stato dell'arte esistono principalmente i seguenti approcci per la valutazione dei prodotti da analizzare:

- Ispezione completa: vengono analizzati tutti i prodotti. Questa soluzione solitamente non è praticabile, ma lo è per produzioni di poche unità.
- Ispezione statistica: i prodotti sono scelti casualmente. I parametri di questo metodo sono la dimensione e frequenza dei campioni scelti.
- **Statistical Process Control** (SPC) [2]: È un passo della Quality Control e si basa sulle statistiche dei difetti rilevati. Vengono misurate e assegnate delle probabilità di errore alle varie fasi produttive, così da poterne costruirne

un modello. I control chart e i diagrammi Causa-Effetto sono gli strumenti utilizzati per cercare di prevenire gli errori di produzione.

2.1.2 Modello e costi dell'ispezione industriale

La scelta della corretta strategia per il controllo di qualità è dettata principalmente dai numeri della produzione. Processi produttivi di grandi dimensioni sono solitamente modellati applicando la Statistical Process Control [2].

La produzione di unità limitate, dette *short-runs*, richiedono un approccio differente. La soluzione proposta da Franceschini et al. [3] è quella di ridurre i costi complessivi, andando a effettuare controlli in fasi non monitorate.

Il modello è rappresentato da un grafo di passi produttivi, dall'inizio all'ottenimento del prodotto finito.

I due errori possibili sono falsi positivi (detti di tipo I) e falsi negativi (detti di tipo II). I primi appaiono quando un prodotto non difettoso è scartato, mentre i secondi quando un prodotto difettoso è erroneamente considerato conforme.

L'ipotesi del modello è che i tipi di difetti sono separati e non correlati tra loro.

I parametri del modello sono:

- p_i: probabilità dell'apparizione di un errore al passo i.
- pFP_i: probabilità dell'apparizione di tipo I.
- pFN_i: probabilità dell'apparizione di tipo II.

Il loro calcolo può essere impegnativo. p_i dipende dalla qualità della produzione, pFP_i e pFN_i dipendono dalla qualità della verifica.

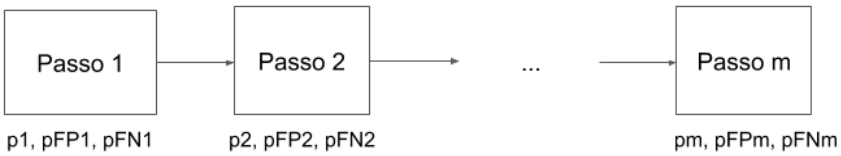


Figura 2.1: Modello di processo produttivo da ispezionare

In figura 2.1 è mostrato un esempio di processo produttivo lineare con fasi in sequenza e accanto ad ogni fase sono mostrati i parametri relativi.

La probabilità di rilevare il difetto è

$$P_i(rileva\ difetto) = p_i \cdot (1 - pFN) + (1 - p_i) \cdot pFP \tag{2.1}$$

mentre quella di non rilevarlo è

$$P_i(non\ rileva\ difetto) = p_i \cdot pFN + (1 - p_i) \cdot (1 - pFP) \tag{2.2}$$

I due addendi della probabilità 2.1 mostrano rispettivamente la probabilità che il difetto rilevato sia autentico o un falso positivo. Analogamente nella probabilità 2.2 il primo addendo mostra la probabilità che non venga rilevato un difetto reale mentre il secondo mostra la probabilità che la non rilevazione del difetto sia corretta.

Una misura dell'efficacia del sistema di ispezione è data dalla media percentuale dei difetti non rilevati lungo tutte le fasi del processo di produzione:

$$EFF = \sum_{1}^{m} p_i \cdot pFN \tag{2.3}$$

I costi che sono affrontati durante il processo sono:

- costi di valutazione (c_i)
- costi di prevenzione (cP_i)
- costi di prevenzione non necessari (cFP_i)
- costi di mancata prevenzione (cFN_i)

I primi tre costi sono facilmente valutabili, mentre i costi di mancata prevenzione sono generalmente complessi da stimare perché difficili da individuare. Il costo per ogni passo produttivo è:

$$C_i = c_i + cP_i \cdot p_i \cdot (1 - pFN) + cFP_i \cdot (1 - p_i) \cdot pFP + cFN_i \cdot p_i \cdot pFN \tag{2.4}$$

mentre il costo totale della quality inspection è la somma per ogni i di C_i.

Il modello esposto mostra come prevedere i costi e le prestazioni di un sistema produttivo; per lo sviluppo di questa tesi è importante dimostrare come ottimizzare i parametri del sistema. I costi di valutazione e prevenzione sono come già detto dati e indispensabili, ma i costi di prevenzione non necessari e di mancata prevenzione sono da ridurre il più possibile. Le spese da sostenere per un falso negativo sono generalmente superiori a quelle di un falso positivo: pertanto bisogna tenere conto di ciò e pensare come bilanciare questi errori per rendere il sistema di quality inspection valido.

2.1.3 Stato dell'arte

Ad oggi i controlli di qualità sono parte integrante degli ambienti industriali che fanno della qualità un punto di forza.

Nelle sfide della moderna quality inspection elencate da Stanisław Borkowski e Krzysztof Knop [4] figura, tra le altre, la ridefinizione dei processi produttivi in cui si aggiungono la segnalazione dei difetti ai comparti responsabili. Uno strumento utile per ridurre i problemi di produzione agendo alla fonte è quello della **dashboard**. Una dashboard facilmente accessibile e consultabile dai manager semplifica il lavoro di analisi e correzione dei problemi nei processi produttivi. L'altra grande sfida è quella di **automatizzare** i sistemi di controllo per ridurre l'incidenza degli errori umani e incrementare la competitività. Mentre in molte aziende è usato da tempo il controllo manuale, con la nascita di nuove tecnologie di classificazione automatica stanno cominciando a farsi strada nel mercato nuove tecniche di quality inspection che fanno un uso limitato dell'intervento umano.

Esistono diversi lavori presenti in letteratura che mirano all'evoluzione della quality inspection. *Xiaomian Li e Yufeng Shu* [5] hanno pubblicato un articolo che analizza la situazione dell'industria del vetro e ne propongono un miglioramento. La richiesta di grandi quantità di prodotti con standard di alto livello è in costante aumento. Gli autori hanno sviluppato un sistema basato sulle analisi delle immagini di superfici di vetro. Il principio usato per la rilevazione del difetto è quello di sottoporre la superficie ad una luce con un certo angolo, che in caso di presenza

di imperfezioni li evidenzia ad una camera di tipo CCD (Charge Coupled Device). L'immagine restituita da questo tipo di camera è costituita da pixel bianchi per porzioni di superfici conformi e ombre per quelle soggette a difetti come graffi o crepe. Un microcontrollore PLC (Programmable Logic Controller) esterno collegato ad un sensore è attivato ogni volta che un unità di vetro passa sotto la camera. L'immagine è così catturata, inviata ad un computer e inoltrata ad un algoritmo di analisi dell'immagine. Le immagini subiscono del preprocessing, come la trasformazione in scala di grigi, per far risaltare i difetti.

Il lavoro di *Simon-Frédéric Désage et al.* [6] va ad analizzare la produzione di pellicole con spessori a livello atomico. Il processo di fabbricazione fa uso di un sistema di raccolta delle particelle costituenti la futura pellicola. La configurazione del sistema è data da un liquido che scorre su di una pendenza fino ad una zona di raccolta, che fa da base alla pellicola. Il prodotto finito è quindi sottoposto a luci da diverse angolazioni. Le caratteristiche dei colori riflessi e dell'effetto arcobaleno risultante permette di acquisire informazioni sullo stato dell'ordine delle particelle. In particolare questi stati sono: particelle disordinate, ordinate irregolarmente e non compatte oppure ordinate regolarmente e in modo compatto. L'analisi è effettuata a livello locale e globale. Il controllo globale è solitamente assegnato ad un controllore umano poiché è sufficiente l'osservazione ad occhio nudo, mentre per l'analisi locale è richiesta l'osservazione al microscopio. Quest'ultima è più complessa da effettuare manualmente e quindi risulta un ottima candidata per l'uso di metodologie automatizzate.

Xinman Zhang et al. [7] hanno analizzato la produzione di barre d'acciaio, proponendo anch'essi metodologie basate su tecnologie di Machine Vision. Sono valutate le immagini di barre d'acciaio disposte a griglia. Esse subiscono un "pretrattamento" che rileva i bordi delle barre: sono utilizzati operatori noti in letteratura, come la segmentazione con il metodo Otsu e il Canny edge detector [7]. Successivamente è usato il metodo di localizzazione dei limiti sub-pixel (SPBLM) per estrarre i diametri minimi e massimi delle barre d'acciaio filettate. I risultati di questo sistema sono promettenti: gli errori di calcolo del diametro delle barre è di 0,04mm in assoluto e di 0,002% in relativo, mentre per il calcolo dello spazio tra barre l'errore è di 2,8mm (3,00%).

Chollada Laofor, Vachara Peansupap [8] affrontano il problema della valutazione estetica dei lavori architetturali. Sono diffuse svariate metodologie oggettive per l'ispezione del materiale necessario alla costruzione, come per le barre d'acciaio che formano il cemento armato. Di contro, la successiva analisi di qualità a costruzione terminata è ad oggi prevalentemente effettuata da supervisori, che danno un giudizio soggettivo e solitamente non affidabile. L'obiettivo degli autori è quello di offrire metodologie per localizzare e quantificare i difetti soggettivi. Per far ciò sono utilizzate tecniche di trattamento automatico delle immagini. Il caso studio trattato dall'articolo è l'ispezione di lavori di piastrellatura. Gli algoritmi mostrati valutano l'uniformità della distanza tra le piastrelle e che gli angoli delle linee tra le piastrelle siano corretti. Anche in questo lavoro è dimostrato che un sistema automatizzato possa avere prestazioni pari o superiori a quelle di un operatore umano.

L'analisi di qualità non è solo prerogativa delle industrie manifatturiere, infatti nell'industria alimentare sono presenti soggetti che controllano le caratteristiche del cibo. Un esempio è dato dal lavoro di *S. A. Araújo et al.* [9] che propone tecniche di Computer Vision per la classificazione di fagioli brasiliani in base al colore. L'incarico, che solitamente è dato ad un operatore umano, viene assegnato ad un sistema automatizzato per ridurre il tasso di errori. In particolare il sistema preprocessa le immagini, trasformandole da RGB in scala di grigi per evidenziare i fagioli che hanno un colore previsto dallo sfondo. Successivamente sono segmentati i singoli grani usando la tecnica Watershed e infine ogni fagiolo nell'immagine è classificato usando una tecnica basata su algoritmi di Machine Learning k-means e k-Neural Network. Questo sistema ha raggiunto la precisione di 98.87%.

Queste diverse soluzioni vanno a considerare solitamente settori industriali specifici, mentre il lavoro di questa tesi si pone l'obiettivo di offrire un meccanismo più generale da applicare nei contesti in cui si fa ricorso al controllo visivo. È comunque possibile estrarre principi comuni da questi lavori, come l'architettura e l'uso di algoritmi specifici per l'elaborazione delle immagini.

2.2 Machine Vision

La **Computer Vision**[10] (*visione artificiale*) è definita come quel campo scientifico che consiste nell'acquisizione, processamento e analisi automatizzata di immagini digitali.

> "Ad un livello astratto, l'obiettivo della visione artificale è usare i dati dell'immagine osservata per dedurre qualcosa riguardante il mondo."[10]

Per capire il contenuto di un'immagine si fa riferimento a tecniche che simulino la visione umana. Queste spaziano vari ambiti, tra cui metodi statistici e Machine Learning mediante apprendimento supervisionato.

Per quanto la Computer Vision risulti semplice nelle intenzioni, ad oggi non esiste un unico strumento che permetta di simulare la visione umana in modo completo. Ciò è dovuto anche alla complessità sia del sistema visivo umano che dei problemi intrinseci della visione. Luminosità, rotazione, traslazione sono solo alcune delle variabili da considerare nel riconoscimento di oggetti in un immagine acquisita in un contesto reale.

Le applicazioni principali in questo campo vanno dalla semplice classificazione di oggetti al riconscimento della loro posizione. In particolare l'applicazione di nostro interesse è quello del riconoscimento automatizzato di oggetti difettosi. Quest'applicazione è presente principalmente nell'industria manifatturiera ed è anche detta **Machine Vision** per via della presenza diffusa di macchinari automatici (robot) in questo contesto.

2.2.1 Acquisizione delle immagini

La prima fase del processo di riconoscimento degli oggetti è l'**acquisizione delle immagini** sul campo. La scelta del dispositivo di acquisizione delle immagini è parte integrante della progettazione di un sistema di Machine Vision, a differenza

di ambiti come il riconoscimento generico di oggetti nell'ambiente; in quest'ultimo caso le immagini possono provenire da qualsiasi dispositivo.

L'esigenza di scegliere il giusto dispositivo è dovuta al fatto che in un ambiente di produzione industriale gli oggetti, solitamente identici tra loro, devono essere analizzati nel modo più efficiente ed omogeneo possibile.

In alcuni contesti produttivi le condizioni di visibilità di oggetti da analizzare devono essere mantenute il più possibile uniformi per ridurre fenomeni noti, come l'*overfitting* che verrà approfondito nella sezione dedicata alla classificazione. L'uso di apparecchiature quali fonti di luci fisse e bracci meccanici permettono di raggiungere quest'obiettivo.

Machine Vision Camera

Nel caso specifico delle linee di produzione, le **Machine Vision Camera** possono essere presenti in più fasi della produzione e sono collegate a **computer edge** mediante interfacce quali USB o Ethernet.

Le camere sono caratterizzate principalmente dal tipo di sensore montato. Il **sensore d'immagine** è un componente elettronico di forma rettangolare che ha il compito di catturare i fotoni e in base all'intensità convertire la luce in determinati segnali elettrici. [11]

Si distinguono in due categorie: **Charge Coupled Device** (CCD) e **Complementary MOS** (CMOS).

Il **CCD** è la prima tecnica di cattura immagine nata. Il sensore è costituito da una matrice di *photosites*, da cui vengono estratte le informazioni per ogni riga. I dati sono passati prima in un amplificatore e poi ad un convertitore analogico digitale (ADC). La maturità acquisita da questa tecnologia negli anni ha permesso di offrire un'alta qualità delle immagini insieme ad alte risoluzioni.

Il **CMOS** è una tecnologia meno costosa che integra all'interno del sensore elementi che nel CCD sono separati dal sensore, come il convertitore ADC. Questo riduce il *fill factor* (fattore di riempimento), ovvero quanti fotoni che colpiscono il sensore sono acquisiti, rendendo necessario tempi di esposizione maggiori.

In questi sensori si presenta anche del rumore dovuto a imperfezioni dei circuiti. È possibile ridurre questo fenomeno grazie a filtri, anche se ciò causa dei ritardi nell'acquisizione dell'immagine.

Oggi i sensori CMOS, anche grazie al loro basso costo di produzione e i loro bassi consumi, sono sempre più diffusi. I progressi tecnologici inoltre riducono i problemi, rendendo conveniente la scelta di questi sensori.

In contesti particolari è necessario l'uso di camere sensibili a spettri elettromagnetici differenti da quello visivo: ad esempio l'analisi della temperatura di un oggetto può essere effettuata con l'uso di camere operanti nello spettro Near-Infrared (infrarosso).

La scelta della giusta camera passa anche dall'analisi di differenti parametri come:

- Sensibilità ai colori

  In alcuni contesti produttivi la distinzione dei colori non è necessaria per la rilevazione di difetti. Un sensore monocromatico è sufficiente a riconoscere

colori nella scala di grigi, mentre un sensore a colori permette di acquisire lo spettro elettromagnetico visibile. Un sensore a colori è più complicato da produrre perché è composto da 25% di sensori sensibili al rosso, 25% al blu e 50% al verde. Questa distribuzione è dovuta alla natura dell'occhio umano più sensibile al verde.

- Risoluzione

  Allo stato dell'arte le reti neurali convoluzionali che costituiscono i classificatori accettano in ingresso immagini con una risoluzione relativamente bassa, nell'ordine dei 200pixel per lato. Pertanto la scelta di dispositivi ad alta risoluzione può essere spinta da esigenze come l'analisi di dettagli di un immagine.

- Frame Rate

  I nastri trasportatori in una linea di produzione si muovono ad una certa velocità. Di conseguenza il tasso di immagini acquisite in un secondo può diventare un parametro importante poiché possono presentarsi fenomeni di blurring dell'immagine.

- Modalità di esposizione (Global Shutter o Rolling Shutter)

  Nel caso dei sensori CCD, l'unica modalità di esposizione possibile è Global Shutter, mentre per i sensori CMOS è possibile scegliere tra Global e Rolling Shutter. Il Global Shutter è la modalità in cui tutti i photosites acquisiscono la luce per un determinato tempo e trasferiscono le informazioni acquisite. Di contro nel Rolling Shutter vengono esposte diverse parti del sensore nel tempo. Lo svantaggio principale sta nell'uso in ambienti con fonti di luce artificiale: la corrente alternata di linea ha una frequenza tale che tempi troppo lunghi di scorrimento dell'intero sensore causano zone dell'immagine illuminate in modo disomogeneo.

2.2.2 Elaborazione delle immagini

L'immagine acquisita è elaborata da o dai computer su cui vengono eseguiti gli algoritmi di Computer Vision scelti. Tra gli algoritmi ricordiamo sicuramente quelli di Machine Learning, ma anche *Edge Detection*, lettura di codici a barre e riconoscimento di testo. [10] Per ogni oggetto analizzato, l'output dell'algoritmo causa un determinato comportamento deciso in fase di progettazione. Nell'ambito di studio di questa tesi la reazione del sistema comporta la segnalazione di un prodotto difettoso attraverso segnalazione sonora (allarme), interruzione del nastro trasportatore o scarto dell'oggetto difettoso.

I dati raccolti per ogni immagine possono essere memorizzati a lungo termine per analisi statistiche di produttività.

2.3 Machine Learning

Il **Machine Learning** [12] è quella branca dell'Intelligenza Artificiale che delega ai sistemi informatici lo svolgimento di processi volti al raggiungimento di una decisione o di una predizione.

Tra le tante applicazioni di queste tecniche, oltre alla sopracitata Machine Vision, ricordiamo i sistemi di filtraggio di spam. Questi sistemi, basandosi su mail contrassegnate da utenti come spam, garantiscono con un certo grado di accuratezza il riconoscimento e smistamento di posta indesiderata.

Il funzionamento di questi algoritmi è legato ad una fase di apprendimento (*learning*) che la letteratura suddivide in due categorie: **apprendimento supervisionato** e **non supervisionato**.

Tra i principali algoritmi con apprendimento supervisionato ricordiamo la **classificazione** e la **regressione**, mentre tra gli algoritmi con apprendimento non supervisionato abbiamo il **clustering**.

2.3.1 Algoritmi con Apprendimento Supervisionato

Negli algoritmi con apprendimento supervisionato la fase di training (*addestramento*) è effettuata sulla base di un *training set*, che consiste in un insieme di oggetti etichettati con una certa classe. La fase di apprendimento del modello avverrà sulla base di questi dati.

Dopo l'apprendimento, la fase di **valutazione** del modello è basata a sua volta su di un *test set*, costituito da oggetti già etichettati usati come input da classificare: il confronto tra l'etichetta vera (*ground truth*) e la predizione in output permette di assegnare al modello valori basati su metriche di valutazione come accuratezza, richiamo e precisione.

La **classificazione** è il ramo più diffuso nell'ambito dell'apprendimento supervisionato. Mentre la classificazione ha il compito di effettuare predizioni su etichette di classe discrete, la regressione predice valori di tipo continuo.

La **regressione** è utile in ambiti come la previsione di valori azionari, che sono definiti in un insieme di valori reali. Nel caso specifico della regressione lineare è possibile tracciare su di un grafico la relazione tra i valori di input e di output mediante una retta. Il calcolo di questa retta può avvenire con algoritmi di minimizzazione della funzione costo tramite tecniche matematiche quali la discesa del gradiente.

In questa tesi si approfondisce il funzionamento della classificazione per via della natura discreta delle classi da riconoscere (oggetto difettoso o non difettoso).

2.3.2 Algoritmi con Apprendimento Non Supervisionato

Contrariamente agli algoritmi con apprendimento supervisionato, quelli con apprendimento non supervisionato non necessitano della interazione umana ma danno dei risultati a partire da dati in ingresso senza nessun genere di etichettatura.

Il **clustering**[13] ha l'obiettivo di estrarre gruppi di dati con caratteristiche simili a partire da dati non catalogati. La regola da seguire consiste nel **minimizzare** la distanza dei dati all'interno di un cluster e **massimizzare** la distanza dei dati tra i cluster. La sfida da affrontare nella definizione degli algoritmi di clustering è individuare cosa determina la vicinanza o la lontananza tra i dati.

2.4 Caratteristiche dei dati

La scelta dell'algoritmo giusto è guidata anche da come i dati sono rappresentati. Tipologie di dati più complesse, come le immagini e i suoni, sono elaborate a partire

dai valori numerici che assumono per ogni componente unitaria, che siano i pixel o il valore dell'onda in un certo intervallo di tempo. Sono detti **attributi**, **feature** o **dimensioni** quelle caratteristiche dei dati che sono considerate dagli algoritmi di machine learning per determinare il risultato. In strutture dati semplici come quelle tabulari, le feature sono le colonne della tabella; mentre per le immagini le feature possono essere la forma dei bordi di un oggetto o il suo colore.

2.4.1 Qualità di dati

I dati in forma "grezza" sono soggetti a problemi di diverso tipo, risolti con tecniche note di preprocessing.

- **Rumore**

  gli attributi dei dati risultano distorti.

- **Outliers**

  sono i dati che hanno proprietà molto differenti rispetto alla maggior parte dei dati che condividono caratteristiche simili tra loro.

- **Valori mancanti**

  dovuti a rilevazioni non effettuate o attributi non valutabili. Si può procedere in diversi modi, ad esempio con la cancellazione dei dati con attributi mancanti o la stima dei valori non presenti.

L'analisi di una grande quantità di dati può risultare difficoltosa per vari motivi, a partire dall'impossibilità della loro raccolta o dai limiti imposti dall'algoritmo, come nel caso delle reti neurali che analizzano immagini con risoluzione non superiore ad un certo valore prefissato. Per questo esistono diverse tecniche per l'aggregazione di dati, come nel nostro caso la riduzione della risoluzione.

2.4.2 Campionamento

Nel caso di grandi quantità di dati è possibile procedere con la scelta di **campioni rappresentativi**, ovvero un insieme di elementi dotati di caratteristiche simili all'insieme di dati originario.

I tipi di campionamento sono semplice casuale (ogni elemento ha la stessa probabilità di essere estratto), con o senza rimpiazzamento (che modifica la probabilità di selezione degli elementi ma che elimina la presenza di duplicati) e stratificato (i dati originari sono suddivisi in base ai loro attributi e campionati a partire da questi sottoinsiemi).

2.4.3 Riduzione dimensionalità

Il concetto di **Maledizione della Dimensionalità** definito da Bellman [14] afferma che un eccessivo numero di dimensioni causa effetti indesiderati. Alcuni sono noti, come un chiaro aumento dei tempi di esecuzione degli algoritmi, altri invece sono meno intuitivi, come l'eccessiva omogeneizzazione dei dati che rende la definizione di misure di distanza meno efficaci e di conseguenza complica il lavoro di distinzione dei dati a partire dai loro attributi.

Nei classificatori questo porta ad *overfitting*, cioè alla scarsa accuratezza dei modelli di classificazione eseguiti su dati diversi da quelli di training.

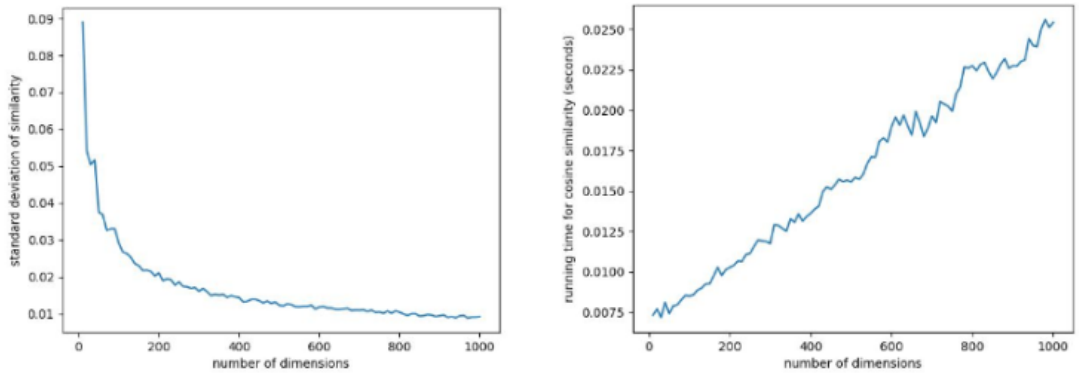


Figura 2.2: Deviazione standard di similarità e tempi di esecuzione rispetto al numero di dimensioni [15]

In figura 2.2 è mostrata la tendenza della deviazione standard della similarità e dei tempi di esecuzione al variare del numero delle dimensioni. È chiaro come un eccessivo numero di dimensioni rende indistinguibili i dati.

Esistono diverse tecniche per ridurre questo fenomeno, tra cui ricordiamo la **Principal Component Analysis** (**PCA**), che consiste nella proiezione di più attributi in un unica dimensione, e la **Feature Subset Selection**, che rimuove gli attributi ridondanti.

2.5 Classificazione

La **classificazione**[13] è un processo di apprendimento automatico supervisionato volto alla definizione di un modello matematico (*classificatore*) che assegna ad ogni oggetto non etichettato una delle classi definite in fase di training, come mostrato in figura 2.3.

In quanto categoria di apprendimento supervisionato, la definizione di un classificatore richiede la specifica di training set e test set.

Esistono svariati algoritmi di classificazione, ognuno con i propri vantaggi e svantaggi. La scelta del giusto algoritmo dipende dal problema specifico: certi algoritmi sono predisposti all'elaborazione di dati in formato più tradizionale (come dati tabulari), mentre strutture dati più complesse come grafi o immagini necessitano di algoritmi più raffinati.

Le proprietà che contraddistinguono i diversi algoritmi di classificatore sono:

- **Accuratezza**:

  misura della qualità di un classificatore basato sul tasso di errore.

- **Efficienza**:

  per essere applicati nei casi reali, bisogna tener conto dei tempi di creazione del modello e classificazione

- **Scalabilità**:

  comportamento del processo di classificazione all'aumentare delle dimensioni del training set o del numero di attributi.

- **Robustezza**:

  resistenza della classificazione a problemi di dati mancanti o di rumore.

- **Interpretabilità**:

  l'analisi dell'esecuzione di alcuni algoritmi più semplici (come gli alberi di decisione) permette di trarre delle conclusioni sulla logica delle scelte di classificazione, mentre altri algoritmi (come le reti neurali) non risultano comprensibili ad un operatore umano.

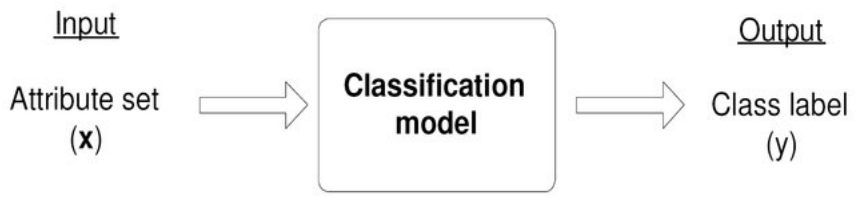


Figura 2.3: Modello ad alto livello di un classificatore [13]

2.5.1 Underfitting e Overfitting

Studiando il tasso di errore del classificatore di tipo "albero di decisione" in figura 2.4, salta immediatamente all'occhio un fenomeno noto come **overfitting**: all'aumentare del numero di nodi presenti nell'albero, l'accuratezza del modello sui dati di training aumenta ma sui dati di test crolla.

In generale l'**overfitting** è legato all'eccessivo adattamento del modello sui dati di training. Difatti, le caratteristiche dei dati (*feature*) non hanno tutte la stessa importanza. Caratteristiche particolari che risultano su alcuni dati di una classe e non in altri della stessa classe (rumore), se considerati, possono andare a minare l'accuratezza del classificatore. L'addestramento corretto di un classificatore deve tener conto di questo fatto.

L'**underfitting** si manifesta invece quando il modello non è sufficientemente elaborato, caso che avviene quando il training set è troppo piccolo.

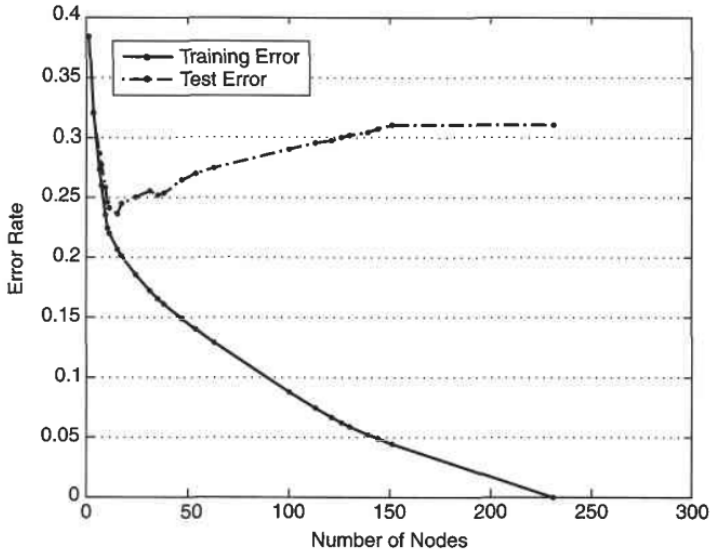


Figura 2.4: "Tassi di errore per training set e test set al variare del numero di nodi dell'albero di decisione" [13]

2.6 Principali algoritmi di classificazione

Gli algoritmi che funzionano meglio per l'elaborazione di immagini sono K-Nearest neighbor, Support Vector Machines, Classificatore Bayesiano e Reti Neurali. [16]

2.6.1 K-Nearest neighbor

Nell'algoritmo K-Nearest neighbor i dati vengono proiettati in uno spazio n-dimensionale e si definisce una metrica di distanza tra i punti. Un punto in input è classificato con l'etichetta di classe relativa alla classe più frequente tra i k punti più vicini.

Non è presente una vera e propria fase di training, visto che il modello corrisponde esattamente ai record del training set. Questo garantisce il vantaggio in fase di addestramento, ma di contro la classificazione è più lenta perché bisogna calcolare la distanza tra un punto e tutti gli altri.

2.6.2 Support Vector Machine

Questo algoritmo (**SVM**) è basato sul concetto di decision boundary.

Si definisce **Decision Boundary** l'iperpiano di separazione tra due regioni contenenti punti di classi differenti. In uno spazio di dati con due dimensioni (dati con due attributi) l'iperpiano si riduce ad una retta.

L'obiettivo dell'algoritmo è quello di trovare il decision boundary ottimale. Per far ciò si definisce come funzione obiettivo massimizzare il margine tra le due regioni. Il margine è calcolato come la distanza tra i support vectors (punti più vicini al decision boundary) di due classi.

Nel caso in cui non esista un decision boundary lineare si procede con la trasformazione dello spazio dei dati mediante Kernel, aggiungendo dimensioni in modo tale da trovare un iperpiano di separazione.

Gli algoritmi SVM non sono interpretabili e la fase di training è onerosa computazionalmente.

2.6.3 Classificatore Bayesiano

I classificatori bayesiani sono incentrati sul teorema della statistica di Bayes che ricordiamo essere:

$$P(C \mid X) = \frac{P(X \mid C)\, P(C)}{P(X)} \tag{2.5}$$

dove C e $X =< x_1, x_2, ..., x_k >$ sono le variabili casuali rappresentanti rispettivamente l'etichetta di classe e il dato da classificare.

$P(X)$ è costante per tutti i C quindi non risulta utile per la previsione della classe.

$P(C)$ è la frequenza della classe C nel training set (N_C/N).

$P(C \mid X)$ è la probabilità che il dato X appartenga alla classe C. Bisogna trovare questo valore applicando 2.5.

$P(X \mid C) = P(x_1, x_2, ..., x_k \mid C)$ è calcolabile in modo diverso in base alle assunzioni fatte sull'**indipendenza statistica** tra gli attributi di un dato. L'ipotesi naive dell'indipendenza statistica

$$P(X \mid C) = P(x_1 \mid C)P(x_2 \mid C)...P(x_k \mid C) \tag{2.6}$$

semplifica il calcolo di $P(X \mid C)$: per ogni attributo k,

$$P(x_k \mid C) = |x_{kC}|/N_C \tag{2.7}$$

dove $|x_{kC}|$ è il numero di dati che assumono il valore $|x_k|$ e appartengono alla classe C.

Il vantaggio di questo algoritmo sta nel suo essere incrementale in fase di training: $P(X \mid C)$ è aggiornabile incrementando denominatore e numeratore (per gli attributi coinvolti) di uno. L'accuratezza risente dell'ipotesi di indipendenza statistica. Il calcolo della classe risulta interpretabile poiché è possibile estrarre gli attributi k che influiscono con il valore più alto di $P(x_k \mid C)$.

2.6.4 Reti Neurali

Gli elementi base dell'algoritmo basato su Reti Neurali (**NN**) sono i neuroni, detti anche **perceptron**. Essi sono strutturati in diversi livelli e collegati tra loro similmente a quanto avviene nel cervello umano.

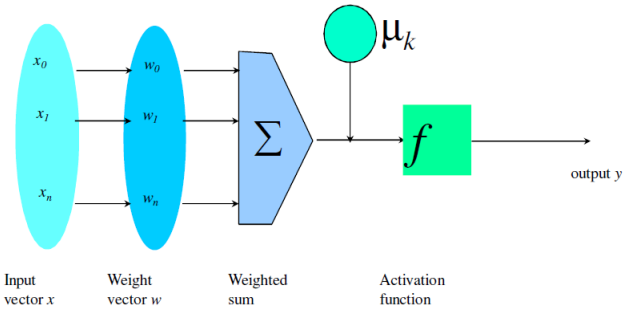


Figura 2.5: Struttura di un perceptron [17]

Come mostrato in figura 2.5, ad ogni **perceptron** è dato in input un vettore di input $X =< x_0, x_1, ..., x_n >$ e di questo ne è fatto il prodotto scalare con il vettore di pesi $W =< w_0, w_1, ..., w_n >$. Come risultato ne esce la somma pesata degli attributi del dato in input e ad esso si somma un offset passato come parametro. Infine si applica una **funzione di attivazione** a quanto calcolato, che costituirà l'output del perceptron.

La **funzione di attivazione** ha il compito di rimappare non linearmente i valori in ingresso da un intervallo infinito ad un intervallo ristretto, ad esempio $[0, 1]$. Le funzioni di attivazioni più diffuse sono la sigmoide e la funzione segno.

La funzione sigmoide è definita come:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

La non linearità dell'output è necessaria per offrire una maggiore profondità di apprendimento del modello. L'assenza di non linearità ridurrebbe la rete neurale ad un modello di regressione lineare, rendendolo inutilizzabile per applicazioni più complesse come il riconoscimento di immagini.

I neuroni, così definiti, sono posizionati in strutture parallele e suddivise in livelli (figura 2.6). La distribuzione parallela di essi aumenta il numero di pesi e conseguentemente il dettaglio nell'analisi degli attributi dei dati; la presenza di più livelli incrementa la non linearità del modello.

Il funzionamento della rete è il seguente: il vettore di input entra nei perceptron e gli output dei perceptron di primo livello si propagano come input dei livelli successivi, fino a raggiungere il livello di output che restituisce un vettore contenente i punteggi di confidenza relativi ad ogni classe.

Costruzione di una Rete Neurale

All'inizio della fase di definizione del modello di NN si definiscono casualmente l'insieme dei pesi e valori di offset per ogni nodo. Mediante un approccio iterativo basato sui dati di training si aggiornano i valori inizialmente specificati.

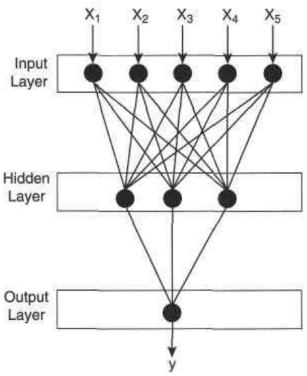


Figura 2.6: Struttura di una rete neurale multilivello [13]

Si elabora un record del training set alla volta nella fase detta di **Forward Propagation**: si calcolano gli output per ogni perceptron fino ad ottenere il vettore di confidenza delle classi. Si compara l'output uscente con quello previsto e si calcola l'errore, che è usato nella fase di **Backpropagation**. In quest'ultima fase si aggiornano i pesi e gli offset per ogni perceptron.

Si reitera il processo fino a che non viene raggiunta la convergenza, determinata dalle condizioni scelte. Queste comprendono il superamento di una predeterminata soglia di accuratezza oppure la riduzione della percentuale di errore sotto una certa soglia.

Inoltre solitamente si pone un limite al numero di epoche massime. Con **epoca** si intende l'elaborazione di tutte le istanze del training set.

Discesa del Gradiente

La fase di **Backpropagation** è implementata mediante l'algoritmo di **discesa del gradiente**.

Per ogni iterazione dell'algoritmo si calcola l'errore, definito come differenza tra valore ottenuto e valore atteso. La minimizzazione dell'errore è effettuata mediante il calcolo del *gradiente* dei pesi, da cui si estrae la direzione e si procede a muovere i pesi in senso opposto.

La regola di aggiornamento dei pesi è la seguente:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i} \tag{2.9}$$

$\frac{\partial L}{\partial w_i}$ è la derivata della **Funzione Loss** (di perdita o costo) rispetto al peso da aggiornare. η è il parametro che gestisce la rapidità di discesa del gradiente.

La funzione loss,

$$L(\mathbf{x}, \mathbf{y}, \theta) = -\sum_j y_j \log p(c_j|\mathbf{x}) \tag{2.10}$$

$$y = [\overbrace{0}^{1}, \overbrace{0}^{2}, ..., \overbrace{1}^{k}, ..., \overbrace{0}^{n}]$$

$p(c_j|\mathbf{x}) = e^{o_k} / \sum_j e^{o_j}$ = probabilità della classe k dato l'input x (softmax)

θ = parametri da correggere

è detta "negative log-likelihood" perché è più alta quando i valori di output sono distribuiti omogeneamente, cioè una classificazione poco comprensibile.

La derivata della funzione Loss rispetto l'output si riduce ad essere

$$\frac{\partial L}{\partial \mathbf{o}} = p(c|\mathbf{x}) - \mathbf{y} \tag{2.11}$$

e applicando la regola della catena di derivate si può calcolare la derivata della funzione loss rispetto ai pesi del livello precedente, così da poter aggiornare i pesi:

$$\frac{\partial L}{\partial W^{n-1}} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial W^{n-1}} \tag{2.12}$$

n = numero di livelli.

Si prosegue a ritroso nella rete fino ad aggiornare i pesi del primo livello.

Vantaggi e Svantaggi

Il vantaggio principale delle NN è dato dall'alta accuratezza raggiunta alla fine della fase di training, unita all'alta velocità di classificazione. Tra gli altri vantaggi è stata dimostrata la resistenza a training set costituiti fino al 15% da dati outliers. [18]

Tra i svantaggi troviamo la non interpretabilità del modello e l'alto numero di risorse computazionali necessarie. Quest'ultimo problema è stato mitigato dall'uso di tecniche di elaborazione parallela eseguite su processori grafici.

2.7 Deep Learning

Il Deep Learning (**DL**) è un sottoinsieme del Machine Learning e comprende quegli algoritmi suddivisibili in livelli che generano, a partire dai dati in ingresso, caratteristiche di basso livello che andranno a costituire l'input per i livelli più in alto nell'algoritmo; a loro volta i livelli più alti generano caratteristiche sempre più astratte, in una catena che inizia con i dati in input e termina con la predizione della classe di output [19].

Le applicazioni più diffuse di queste tecniche si trovano in ambiti dove i dati da analizzare sono di per sé complicati, come nel caso di immagini, audio e video. L'implementazione di gran lunga più diffusa riguarda le reti neurali "profonde" formate da livelli di tipi differenti. Ai fini di questa tesi si approfondiranno i concetti alla base delle **Reti Neurali Convoluzionali** (CNN), la tecnica allo stato dell'arte per la classificazione di immagini.

2.7.1 Architetture Shallow e Deep

Nell'ambito del riconoscimento di immagini, la differenza tra le tecniche tradizionali dette "shallow" (*superficiale*) e quelle dette "deep" si ha nella fase di **estrazione delle feature**.

Un *classificatore di immagini tradizionale* è composto da una prima fase di estrazione delle feature a partire dall'input e da una fase successiva a cui si fornisce ad un generico classificatore i dati relativi alle feature dell'oggetto. Le tecniche come **SIFT** (Scale Invariant Feature Transform) e **HOG** (Histogram of Oriented Gradients) hanno lo scopo di estrarre le feature per l'individuazione di oggetti, basandosi sui gradienti e di conseguenza sul riconoscimento dei bordi. Esse sono progettate "a mano": non avviene una fase di apprendimento automatico dei dettagli di un oggetto, le tecniche devono essere create in base al tipo di oggetto da riconoscere.

Un *classificatore "deep"* non ha una componente dedicata all'estrazione delle feature, ma la struttura gerarchica multilivello elabora da sé le feature di basso livello (come parti del viso umano) e le trasforma in altre di livello più alto (viso umano) fino a restituire il risultato di confidenza delle classi (uomo o donna).

2.8 Convolutional Neural Networks (CNN)

Nella sezione riguardante la qualità dei dati (2.4.3) si è introdotto il concetto di "Maledizione della Dimensionalità" relativo al numero eccessivo di feature da analizzare. Le reti neurali sono particolarmente affette da questo problema.

I livelli *fully connected* collegano tutti i neuroni del livello precedente a tutti quelli del livello corrente. Questa è una potenziale causa di overfitting. Livelli collegati localmente possono ridurre la complessità, ma da soli non sfruttano il principio di località spaziale delle immagini.

Una soluzione ideata negli ultimi anni è quella di aggiungere dei **livelli convoluzionali** nella rete con il compito di ridurre le feature, aggregandole in feature di livello più alto, come mostrato in figura 2.7. Le risultanti **mappe delle feature** saranno formate da pixel associati a un intervallo di pixel dell'input. In figura 2.8a è mostrato un esempio di gerarchia di feature.

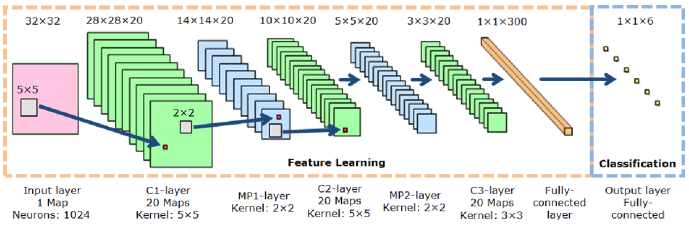


Figura 2.7: "Esempio di CNN che alterna livelli convoluzionali e max-pooling" [20]

2.8.1 ReLU

In ambito DL, la non linearità è offerta dall'uso della funzione di attivazione $max(0, v)$, detta Rectified Linear Unit (**ReLU**). Il grafico di questa funzione è mostrato in figura 2.8b.

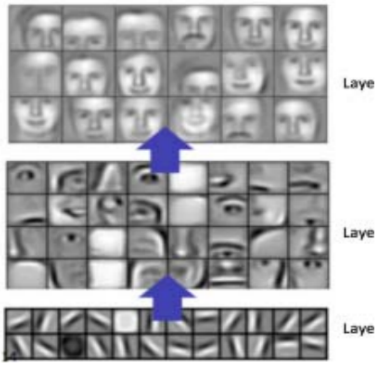
La scelta di questa funzione è dovuta al fatto che, a differenza di altre funzioni come la sigmoide, non ha un limite superiore e quindi non satura la rete. Questo permette al gradiente di "fluire" più facilmente in fase di training ed evita che input con caratteristiche differenti tra loro causino lo stesso tipo di output.

Leaky ReLU

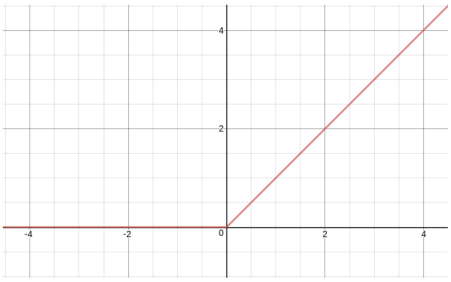
In una rete che usa la funzione di attivazione ReLU, neuroni che ricevono sempre input negativi restano inattivi e non contribuiscono alla classificazione. Una possibile soluzione è costituita dalla funzione **Leaky ReLU** che per valori negativi di input restituisce, anziché zero, valori negativi in output ma ridotti di un fattore arbitrario.

2.8.2 Livelli Convoluzionali

In un **livello convoluzionale** (*Conv Layer*) l'operazione eseguita è la convoluzione delle mappe di feature del livello precedente, che possono essere più di una, con uno o più filtri del livello corrente. Il **filtro**, o *kernel*, è una finestra solitamente quadrata, che va da 1x1 fino a valori come 10x10 ma che solitamente assume dimensione 3x3. Ad un livello convoluzionale possono essere associati più filtri. In tal caso il numero di mappe di feature in output è moltiplicato per il numero di filtri.



(a) "Feature apprese in una CNN" [22]



(b) Grafico della funzione ReLU [21]

Figura 2.8

La convoluzione è calcolata così:

$$\overbrace{h_j^n}^{output\ feature\ map} = \max(0, \sum_{k=1}^{K} \overbrace{h_k^{n-1}}^{input\ feature\ map} * \overbrace{w_j^n}^{kernel}) \tag{2.13}$$

dove K è il numero di mappe di feature in input, n è il livello corrente e j è l'indice di uno dei filtri del livello corrente. Nel caso banale di una mappa di feature in ingresso e un filtro, il valore di un pixel nella finestra della mappa di feature in input è moltiplicato per il valore del corrispettivo pixel nella stessa posizione nel filtro; questi valori sono sommati e il risultato andrà a costituire il valore del pixel della nuova mappa di feature. Il filtro parte da un angolo dell'immagine; la mappa di feature sarà completa una volta che il filtro analizza tutti i possibili pezzi di immagine, scorrendo di una certa quantità di pixel detto passo (*stride*). Il passo solitamente è di un pixel, ma può assumere valori più alti. In figura 2.9 è mostrato un esempio di convoluzione con passo 1.

Considerando il caso di più feature map in input, si sommano i valori dei singoli pixel nella stessa posizione di tutte le mappe calcolate a partire dalle mappe in input. Nel caso di più filtri si ripete l'operazione vista cambiando i valori del filtro considerato.

In figura 2.10a è mostrato graficamente un esempio di convoluzione: a sinistra c'è un immagine a colori (3 valori per pixel) di dimensione 32x32, a destra più filtri che osservano lo stesso pezzo di immagine. L'output sarà formato da tante mappe di feature quanti sono i filtri.

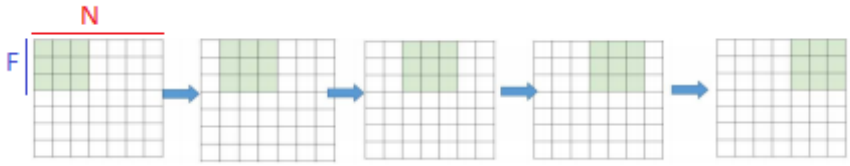
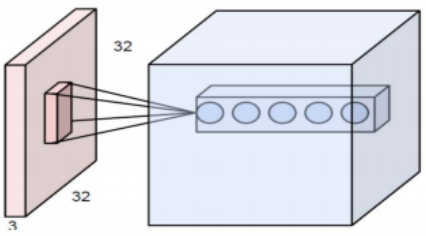
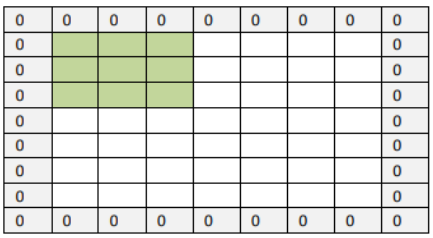


Figura 2.9: Convoluzione e spostamento della finestra con passo 1 [22]



(a) Esempio di convoluzione [22]



(b) Zero-padding [22]

Figura 2.10

La dimensione della mappa di feature (detta anche **mappa di attivazione**) è calcolata nel seguente modo:

$$(N - F)/stride + 1 \tag{2.14}$$

Qualora il risultato non risultasse intero, si procede con l'aggiunta di zero padding ai bordi, come mostrato in figura 2.10b. La nuova formula diventa così:

$$(N - F + 2 \cdot PAD)/stride + 1 \tag{2.15}$$

La scelta degli iperparametri (N, F, PAD, $stride$) è determinata dalla natura dei dati: più dati implicano valori più alti.

2.8.3 Pooling Layers

Architetture più avanzate di CNN sono costituite da un altro tipo di livello, chiamato di **max pooling**. Il compito principale del "pooling" è quello di ridurre la risoluzione delle mappe di feature (*downsampling*), rendendole più semplici da gestire.

Nel caso più frequente mostrato in figura 2.11, con filtro 2x2 e passo 2, blocchi contigui e non sovrapposti di dimensione 2x2 sono ridotti ad un singolo valore, il più alto tra i valori del blocco. Quando gli input hanno una profondità maggiore di uno, l'operazione di max pooling è effettuata indipendentemente per ogni "fetta". La mappa risultante avrà la stessa profondità della mappa in input.

Questa fase causa una perdita di informazioni, quindi è usata solitamente per segnalare la presenza di un informazione all'interno del blocco.

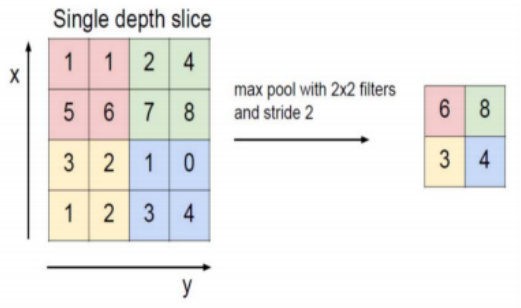


Figura 2.11: Esempio di max pooling con filtro 2x2 e passo 2[22]

Average Pooling

Mentre il max pooling è applicato su blocchi di dati piccoli (*small pooling*), per l'applicazione dello stesso concetto a blocchi di dati più grandi si preferisce calcolare il valore medio anziché il massimo. L'**average pooling** consente di rappresentare meglio la distribuzione dei valori di un blocco. In alcune architetture sostituisce i livelli Fully Connected. [19]

2.8.4 Fully-Connected Layers

Nelle CNN i livelli tradizioali di tipo **fully-connected** (**FC**) sono presenti solitamente alla fine della catena di perceptron, dove le feature sono ridotte e quindi più gestibili.

2.8.5 Regolarizzazione

La **regolarizzazione** dei dati si rende necessaria poiché la scelta errata di campioni può portare la CNN all'apprendimento di rumore, causando fenomeni di overfitting. Diverse tecniche sono state messe in atto per limitare questo rischio.

Regolarizzazione L2 / Weight Decay

Il principio della **regolarizzazione L2** è quello di svantaggiare la presenza di pesi troppo alti attraverso la limitazione del quadrato dei pesi. Ciò si ottiene modificando la funzione loss su cui fare la discesa del gradiente:

$$L'(\mathbf{w}) = L(\mathbf{w}) + \lambda/2 \cdot \mathbf{w}^2 \tag{2.16}$$

Di conseguenza è modificata la regola di aggiornamento dei pesi:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i} - \eta \lambda w_i \tag{2.17}$$

La componente aggiunta $\eta \lambda w_i$ è detta **Weight Decay** (*Decadimento del peso*).

L'effetto è quello di rendere il training più regolare e di conseguenza ridurre l'adattamento del modello a errori di scelta dei campioni.

Metodo Dropout

L'overfitting dovuto a **co-adattamento** (*co-adaptation*) è causato da livelli molto estesi e densi, come nei livelli FC. La soluzione al co-adattamento senza ridurre il numero di perceptron è il **metodo dropout** (*buttar fuori*): consiste nell'escludere una percentuale di perceptron in fase di training (solitamente la metà) e considerarli tutti in fase di test. Il tempo di training è maggiore, ma il vantaggio offerto lo rende molto usato.

Batch Normalization

In CNN molto profonde, i dati passanti nei vari livelli FC e Conv subiscono variazioni anche importanti della loro media e varianza. Questo fenomeno, detto del **gradiente evanescente** (*vanishing gradient*), rende la fase di training più difficoltosa. La soluzione ideata è quella dell'aggiunta di un livello di **Batch Normalization** dopo ogni livello Conv e FC. I dati sono normalizzati secondo la seguente formula:

$$x_{norm} = \frac{x - E[x]}{\sqrt{Var[x]}} \tag{2.18}$$

La media e la varianza sono calcolati per ogni batch in fase di training e applicati in fase di test. Questa tecnica richiede più memoria ma accelera la convergenza, rendendola molto conveniente da usare. [23]

2.8.6 Casi Studio di CNN

Le CNN sono una tecnologia relativamente recente. A partire da LeNet sono state sviluppate architetture sempre più profonde e accurate, fino a rendere le CNN la tecnologia con risultati migliori nell'ambito del riconoscimento di immagini. In figura 2.12 è mostrata l'evoluzione nell'accuratezza delle reti neurali sempre più profonde.

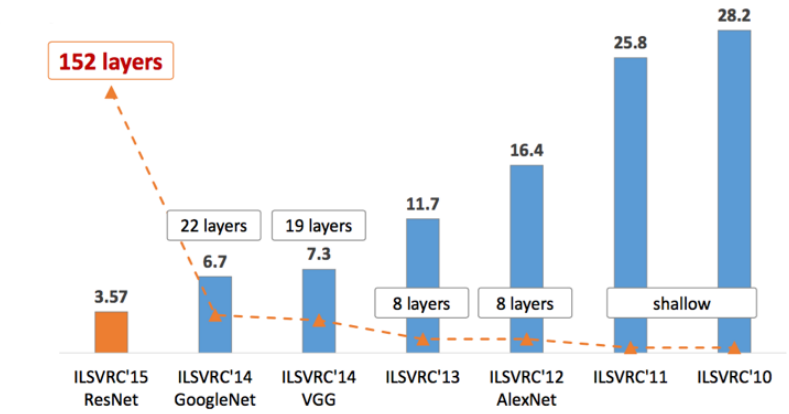


Figura 2.12: Risultati della ImageNet Large Scale Visual Recognition Challenge[24]

LeNet-5

La prima CNN nota in letteratura è LeNet-5 (figura 2.13), progettata nel 1998. [25] Il compito per cui è nata è il riconoscimento di cifre scritte a mano e ha trovato nel tempo utilizzo in ambiti bancari per la lettura automatica di cifre sugli assegni. La risoluzione delle cifre in input è di 32x32 pixel. È costituita da cinque livelli interni, di cui tre convoluzionali e due di subsampling.

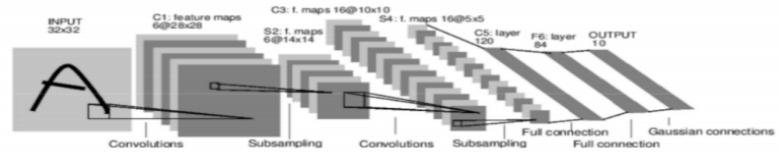


Figura 2.13: Architettura di LeNet-5 [25]

AlexNet

AlexNet venne sviluppata nel 2012 e fu una delle architetture più importanti (in figura 2.14) nella storia delle CNN per via delle prestazioni ottenute nella ImageNet Challenge. Ottenne un tasso di errore con distacco di 10 punti percentuali dalla seconda classificata. Questi risultati hanno contribuito a spingere la ricerca sulle CNN fino ad oggi.

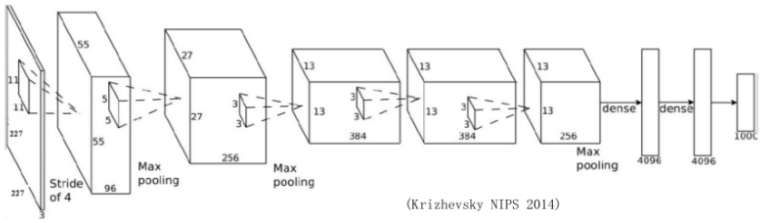


Figura 2.14: Architettura di AlexNet [26]

Questa rete riceve in input immagini a colori di 227x227, facendo un grosso passo avanti rispetto a LeNet. Il filtro del primo livello convoluzionale è di 11x11

con passo 4, in modo da ridurre velocemente il numero di parametri e di conseguenza l'overfitting. Nei livelli successivi di tipo convoluzione e pooling abbiamo filtri di 5x5 e 3x3. Questa CNN è stata la prima ad aver usato la funzione di attivazione ReLU. Tra le altre tecniche di normalizzazione introdotte abbiamo il metodo dropout.

Inception Networks

Sviluppate da Google, le reti **Inception** sono costituite da **Inception Layers**. L'approccio di questi livelli consiste nella parallelizzazione di più livelli convoluzionali con filtri di diverse dimensioni. In figura 2.15 ne è mostrato un esempio.

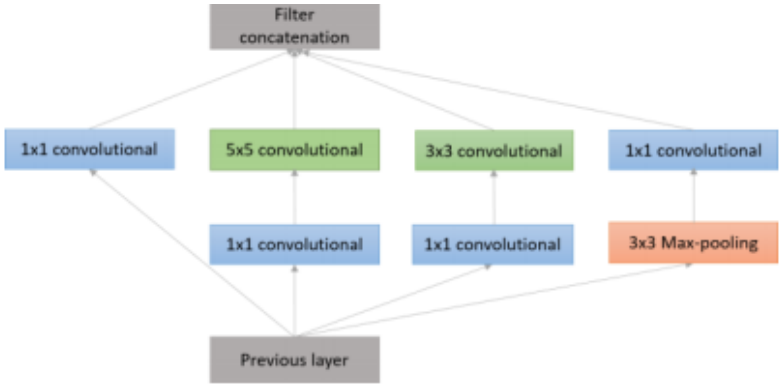


Figura 2.15: "Inception Layer con riduzione dimensionale" [27]

Il vantaggio fornito da questa disposizione degli elementi nella rete è quello di fornire un più alto numero di livelli, riducendo allo stesso tempo i parametri della rete e di conseguenza la complessità di training. Rispetto ad AlexNet, infatti, il numero di parametri passa da 60 milioni a 4 milioni. Tra le altre tecniche usate ricordiamo la normalizzazione batch. Le versioni più recenti (v3 e v4) introducono l'uso di unità residuali come in ResNet, offrendo un'accuratezza ancor maggiore.

Residual Networks (ResNet)

Nelle ResNet, introdotte nel 2015, sono presenti 152 livelli (figura 2.16). Questo numero di livelli è sufficientemente alto per causare un forte fenomeno di *Vanishing Gradient*, anche con l'uso della Batch Normalization.

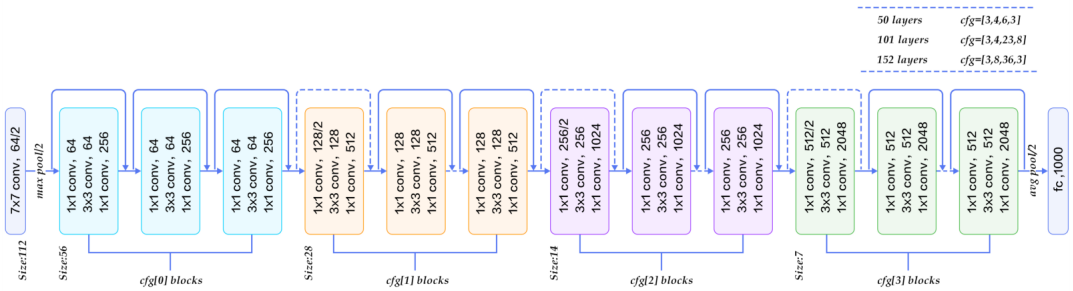


Figura 2.16: Architettura di ResNet [28]

Il metodo usato da questa rete per ridurre il problema è basato sulle **Skip Connections** (figura 2.17): l'output di un blocco diventa l'input non solo del livello

successivo, ma anche per livelli più avanti nella catena. Di conseguenza, l'output di un modulo è dato dall'output classico di una sequenza di livelli più l'input del modulo stesso.

$$y = F(x) + x \tag{2.19}$$

Quello che la rete impara e ottimizza non è l'output completo y, ma il residuo $F(x)$. Gli autori di questa rete affermano che effettuare il *learning* sul residuo è più semplice per via del percorso del gradiente durante l'addestramento di ResNet.

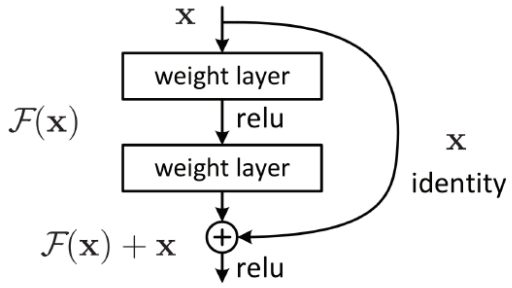


Figura 2.17: Blocco di una rete residuale [28]

In fase di training è possibile saltare le funzioni di attivazione non lineari utilizzando le skip connections. Durante la backpropagation, il gradiente ha un percorso più chiaro e diretto passando dalle skip connections verso il livello base.

Con la sua caratteristica "Ultra Deep" di 152 livelli, la rete ResNet è riuscita ad arrivare prima nella ImageNet Challenge in modo netto, in particolare nelle ImageNet Detection e Localization è stata rispettivamente il 16% e il 27% migliore della seconda arrivata. Con il tasso di errore di 3.57 la rete supera in accuratezza la classificazione umana, che è stimata essere il 5% [24].

2.8.7 Tecniche di Data Augmentation e Transfer Learning

Verranno qui presentate due tecniche per ottenere il massimo dalle deep neural networks, dalla riorganizzazione dei dati al riuso di reti già addestrate.

Data Augmentation

Le reti convoluzionali hanno bisogno di un gran numero di dati in ingresso durante il training per dare un'accuratezza maggiore. Poche immagini causano una più frequente analisi delle stesse durante il training, causando come è noto overfitting.

Indicativamente una rete convoluzionale inizia ad avere un accuratezza accettabile a partire da 1000 immagini usate per il training. Nei casi in cui l'acquisizione di un grande dataset di training risulti complicato, esistono tecniche che permettono di estrarre variazioni mediante trasformazioni artificiali delle immagini.

Le **caratteristiche visive** di un oggetto in un immagine sono diverse: luminosità, messa a fuoco, rotazione (angolo), distanza dal punto di vista, sfondo, forma e colore. Esistono diverse trasformazioni applicabili alle immagini:

- **Flip (capovolgimento) e rotazione**

il flip può essere sia orizzontale che verticale. La scelta di uno o entrambi dipende dalla caratteristica dell'oggetto: il riconoscimento di auto non capovolte non beneficia molto da un flip verticale, mentre per un oggetto posizionato su di un piano orizzontale entrambi i flip possono essere considerati utili. Similmente, la scelta di ruotare un oggetto o meno in fase di data augmentation è dettata da come si suppone esso sia disposto in testing. In figura 2.18a sono mostrati esempi di rotazioni di un immagine.

- **Crop (ritagli) e ridimensionamento**

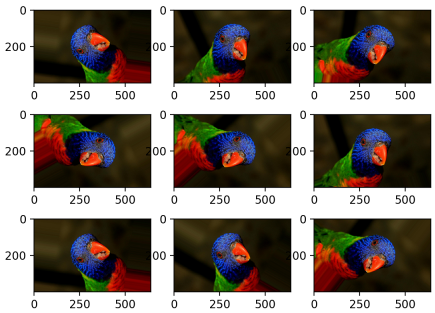
  il ritaglio casuale di pezzi più piccoli a partire da un immagine consente alla rete di adattarsi al caso in cui sia richiesta la classificazione di un oggetto di cui si è acquisita solo una parte. Ad esempio, per il riconoscimento di animali, i random crop permettono ad una rete di riconoscerli anche avendo a disposizione solo la faccia. Un modo tipico di procedere è quello di acquisire immagini ad una risoluzione maggiore di quella richiesta e prendere dei ritagli con una risoluzione pari a quella di input della rete.

- **Luminosità e Contrasto**

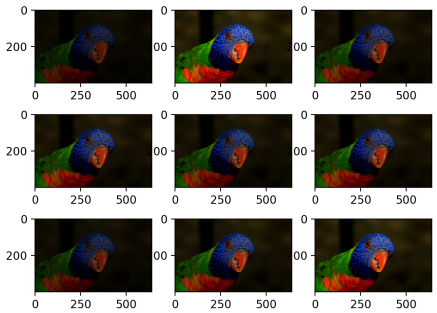
  le condizioni di luminosità possono influire anche in modo importante nel riconoscimento di un oggetto. Pertanto, oltre che acquisire immagini sul campo con diverse condizioni di luce, è possibile agire modificando in modo casuale luminosità e contrasto dell'immagine. In figura 2.18b è mostrato un esempio di immagini generate con luminosità differenti.

- **Distorsione**

  proprietà che potrebbe essere interessante considerare per oggetti sottoposti a distorsione di vario tipo (stretching) o per oggetti acquisiti da prospettive leggermente differenti.



(a) Variazioni casuali di rotazione [29]



(b) Variazioni casuali di luminosità [29]

Figura 2.18

La scelta delle trasformazioni ideali da considerare è data dalle caratteristiche del test set. Nel contesto di questa tesi, ovvero quello produttivo, ho ipotizzato l'acquisizione di immagini su un nastro trasportatore con sfondo neutro: le trasformazioni utili sono flip (oggetto simmetrico), rotazione, leggere distorsioni. Ho

scelto di escludere il crop poiché ho ipotizzato l'acquisizione corretta delle immagini (distanza sufficientemente grande tra camera e nastro).

La regola generale da considerare è di **massimizzare** la variazione delle trasformazioni degli oggetti all'interno di ogni classe e **minimizzare** la variazione delle stesse tra classi differenti. L'uso di Data Augmentation causa una convergenza di training del modello più lenta, fattore che è irrilevante a fronte della maggiore accuratezza in fase di testing.

Transfer Learning

Una rete inizializzata a partire da valori casuali richiede un certo tempo di training.

Il principio che sta dietro il **Transfer Learning** è quello di ridurre i tempi di training sfruttando reti già addestrate per il riconoscimento di oggetti con feature generiche. Sono disponibili **modelli pre-trained** da cui si può proseguire con il training specializzato (*fine tuning*). https://modelzoo.co/ è un sito che offre diversi modelli pre-trained open source.

Per piccoli dataset è consigliabile permettere la modifica dei parametri degli ultimi livelli fully connected, mentre per dataset più corposi è accettabile modificare i pesi dei Conv Layer di più alto livello. I livelli di base della rete solitamente estraggono feature comuni a tutti gli oggetti, come i bordi.

2.9 Deep Learning as a Service (DLaaS)

Un'azienda manifatturiera di grandi dimensioni, con un alto budget in investimenti su sviluppo e ricerca, ha la capacità finanziaria di acquisire hardware e software dedicato per tecnologie di frontiera. In particolare, per il riconoscimento di immagini con reti di tipo CNN, sono necessari calcolatori con GPU di alta fascia e sistemisti in grado di configurare l'ambiente, oltre che di data scientist per il training dei modelli su diversi framework esistenti.

Al contrario, aziende di piccole o medie dimensioni sono disincentivate dal supportare l'intelligenza artificiale nei processi produttivi per via degli alti costi di mantenimento. La nascita delle architetture cloud ha permesso alle aziende di effettuare l'**outsourcing** di servizi informatici di vario genere. Le tecnologie deep learning hanno visto il loro sviluppo in tempi recenti e molte aziende informatiche affermate nel campo hanno colto l'opportunità di investire su di esse. Pertanto il concetto di *Everything as a Service* entra anche in quest'ambito, offrendo quello che è possibile chiamare **Deep Learning as a Service** (DLaaS) [30].

La figura 2.19 mostra le fasi ad alto livello della costruzione di un modello in un servizio DLaaS: data preprocessing, training, deploy e uso del modello.

Figura 2.19: Fasi della gestione di un modello [30]

2.9.1 Architettura

L'architettura di un sistema di riconoscimento immagini basato su Deep Learning è strutturata in diverse componenti. La gestione dell'hardware e dei framework di deep learning è delegata al fornitore di servizi.

A seconda del livello di astrazione offerto, l'utente fruitore potrà gestire la fase di training fino ad un certo dettaglio. Ad esempio, IBM offre la possibilità con Watson Studio di progettare la rete e scegliere le risorse necessarie da utilizzare [31]. Oltre a Watson Studio, IBM offre servizi di IA dedicati al campo di applicazione. Per l'ambito del riconoscimento di immagini è disponibile il servizio di **Visual Recognition**, che semplifica la gestione del processo di classificazione.

Essendo un servizio cloud, le comunicazioni avvengono interamente sulla rete verso i server che fanno da gateway al servizio. Il protocollo utilizzato è HTTP basato su **REST API**. In figura 2.20 è mostrato un esempio di scambio di informazioni tra l'utente e il gateway del servizio. L'utente invia prima una richiesta POST per effettuare il retraining del modello e il servizio risponde con l'esito; successivamente l'utente richiede di classificare un immagine con una richiesta GET che include l'immagine e il servizio restituisce i punteggi di classificazione.

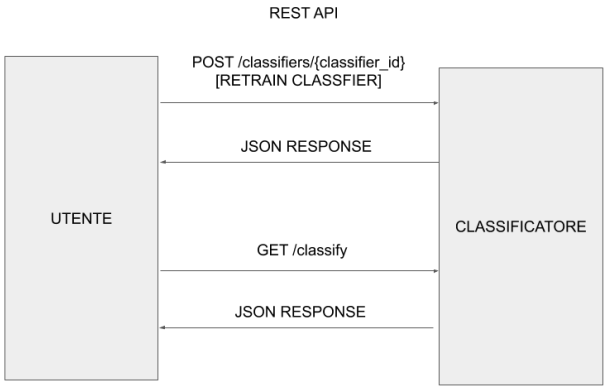


Figura 2.20: Schema di comunicazione tra utente e classificatore

Le operazioni a pagamento offerte dai servizi di questo genere sono solitamente: upload di immagini suddivise in classi per il (re)training di modelli personalizzati e upload di una o più immagini destinate ad essere classificate.

Il **meccanismo di pagamento** differisce in base al fornitore, ma esistono principalmente i seguenti metodi:

- **Pagamento per uso**

  gli addebiti avvengono in base a quali e quante volte le API sono state richiamate.

- **Pagamento per disponibilità**

  all'utente sono offerte un certo quantitativo di risorse computazionali che è libero di usare, ma che costituiscono un costo fisso.

La scelta della giusta modalità dipende dall'intensità dell'uso che se ne vuole fare.

2.9.2 IBM Watson Visual Recognition

Il servizio scelto che andremo ad utilizzare è **IBM Watson Visual Recognition**.

I costi dichiarati a maggio 2019 per il piano standard sono i seguenti:

- Classificazione immagine: $0.002/immagine
- Training di un modello personalizzato: $0.10/immagine

È offerta sia la classificazione generica di oggetti, sia quella basata su classificatori personalizzati. Quella di nostro interesse è la seconda.

Il training del modello deve seguire le specifiche *Best Practices* comuni. Abbiamo già parlato nella sezione sulla Data Augmentation (2.8.7) di quali siano i metodi da seguire in generale per trarre il meglio dai dati posseduti.

A quanto detto aggiungiamo degli appunti specifici al servizio offerto da IBM. Il numero minimo consigliato di immagini da assegnare per un ciclo di training è di 50 per classe, anche se un corretto bilanciamento tra tempo di training e accuratezza è dato da 200 immagini [32].

Considerando il caso di riconoscimento binario (classe positiva/negativa) di difetti, le immagini prese devono risultare classificabili da un operatore umano nell'ipotesi che questo avvenga con un tasso di errore di circa il 5%.

È importante anche considerare **quanto il difetto da riconoscere sia grande** in proporzione all'intera immagine: una percentuale troppo bassa, in quanto difficile da riconoscere dall'occhio umano, potrebbe portare a risultati subottimali. Una possibile soluzione a questo problema è quella di suddividere l'immagine in blocchi più piccoli da analizzare singolarmente.

La corretta gestione del processo di training ha permesso ad altri utenti di raggiungere percentuali di accuratezza maggiori del 98%. [32]

Per le tecniche di valutazione dei risultati di classificazione e di training si rimanda al capitolo dedicato (5.3).

2.9.3 Servizi di classificazione visiva alternativi

Esistono servizi alternativi ad IBM Watson Visual Recognition offerti da compagnie note nel settore, come Microsoft e Google. Di seguito sono mostrati i costi dichiarati ad ottobre 2019:

Microsoft Azure Custom Vision

I costi per il piano standard sono [33]:

- Classificazione immagine: EUR 0.001687/immagine
- Training EUR 16.866 per ora di calcolo:

Google Cloud AutoML Vision

I costi per la classificazione di immagini sono [34]:

- Classificazione immagine: $0.003/immagine
- Training: $20 all'ora

Capitolo 3

Metodologia: progettazione di un sistema di Quality Inspection

In questo capitolo verrà esposta la metodologia adottata per la progettazione del sistema di Quality Inspection.

L'obiettivo del tirocinio svolto in azienda è quello di creare un Proof of Concept (**PoC**) di un sistema di quality inspection, per cui il caso studio considerato è quello di un generico contesto produttivo dove si effettua la produzione di oggetti in serie.

I casi pratici da considerare sono l'analisi di qualità:
- in tempo reale mediante camere montate su linee di produzione
- in una fase dedicata successiva alla produzione

3.1 Architettura

Le macrocomponenti dell'architettura del sistema di Quality Inspection sono tre: la **Web Application**, il **client** installato sul computer edge collegato alla camera e il servizio di deep learning offerto da IBM (figura 3.1) .

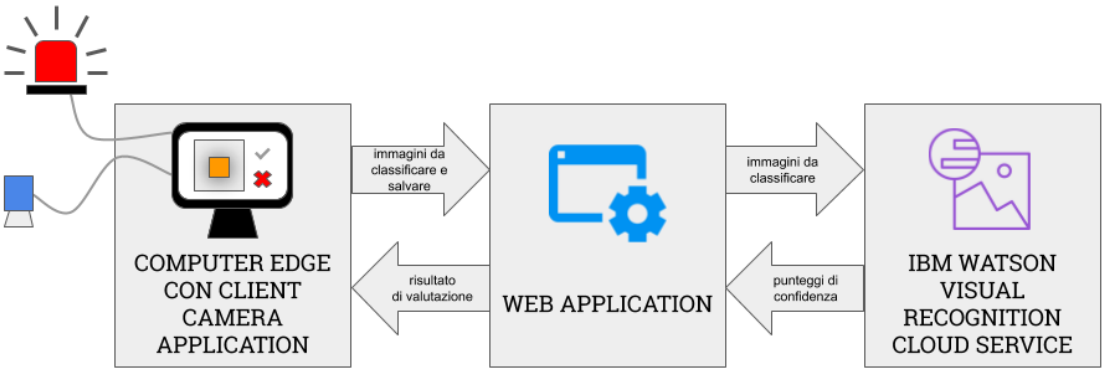


Figura 3.1: Macrocomponenti dell'architettura

- **Camera Client Application**

  La classificazione in tempo reale richiede il supporto hardware necessario: un *computer edge* a cui è collegato una *machine vision camera* che acquisisce le immagini dei prodotti in linea di produzione, ed eventualmente un sistema

di segnalazione (*Alarm Endpoint*) che venga azionato ogni qualvolta sia rilevato un prodotto difettoso. Sul computer edge è installato il software che permette alla camera di acquisire le immagini e inviarle alla web application per la classificazione. Il software ha anche il compito di mostrare i risultati di classificazione e nel caso comunicare la presenza di un difetto inviando un messaggio REST ad un Alarm Endpoint.

In particolare l'applicazione client, dopo aver effettuato l'autenticazione, permette l'acquisizione di immagini in modo temporizzato o manuale. I risultati sono mostrati nel log a video.

Il software è sviluppato in linguaggio **C#** sulla base dell'SDK offerto dalla machine vision camera.

- **Web Application**

  L'applicazione web è suddivisa in backend e frontend. Il *frontend* ha il compito di mostrare all'utente un'interfaccia navigabile dal browser per la gestione del sistema di quality inspection. Una volta autenticato l'utente, è mostrata una dashboard che offre una visione d'insieme dello stato dei classificatori definiti. La dashboard mostra le cifre più importanti da tenere sotto controllo, come le immagini da revisionare e quelle da usare per il prossimo retraining. Inoltre è presente un grafico che mostra l'evoluzione dell'incertezza nella classificazione nel tempo. Tra le altre operazioni possibili ci sono: mostrare la lista delle immagini acquisite e/o classificate, revisionare le immagini classificate in modo incerto, caricare ulteriori immagini, eseguire il retraining, vedere il riepilogo dei costi e configurare parametri del classificatore.

  Il *backend* ha il compito invece di fornire le API REST sia al frontend che alla client camera application. È possibile aggiungere, richiedere, aggiornare ed eliminare classificatori e immagini. Le immagini di cui è richiesta la classificazione sono inoltrate al servizio di Visual Recognition, che restituisce un risultato al backend. Il backend esegue un algoritmo decisionale e all'utente è mostrata la valutazione. È possibile inoltre registrarsi e accedere ad un account.

  In figura 3.2 sono mostrati i framework e servizi dell'applicazione web. Il framework scelto per la gestione delle API REST è **Node.js**, mentre per la presentazione grafica all'utente è stato scelto **Angular**.

  La persistenza dei dati è offerta da un database NoSQL basato su CouchDB, **Cloudant**.

  Le immagini sono salvate in un Cloud Object Storage (**COS**).

- **IBM Watson Visual Recognition**

  Il servizio di classificazione offerto da IBM offre la possibilità di creare nuovi classificatori inviandogli le immagini da usare per il (re)training, separate per classi, e di classificare le immagini inviate via REST. Per ogni immagine da classificare bisogna specificare il classificatore da usare. La risposta conterrà, per ogni classe definita in fase di training, un punteggio di confidenza compreso tra 0 e 1.

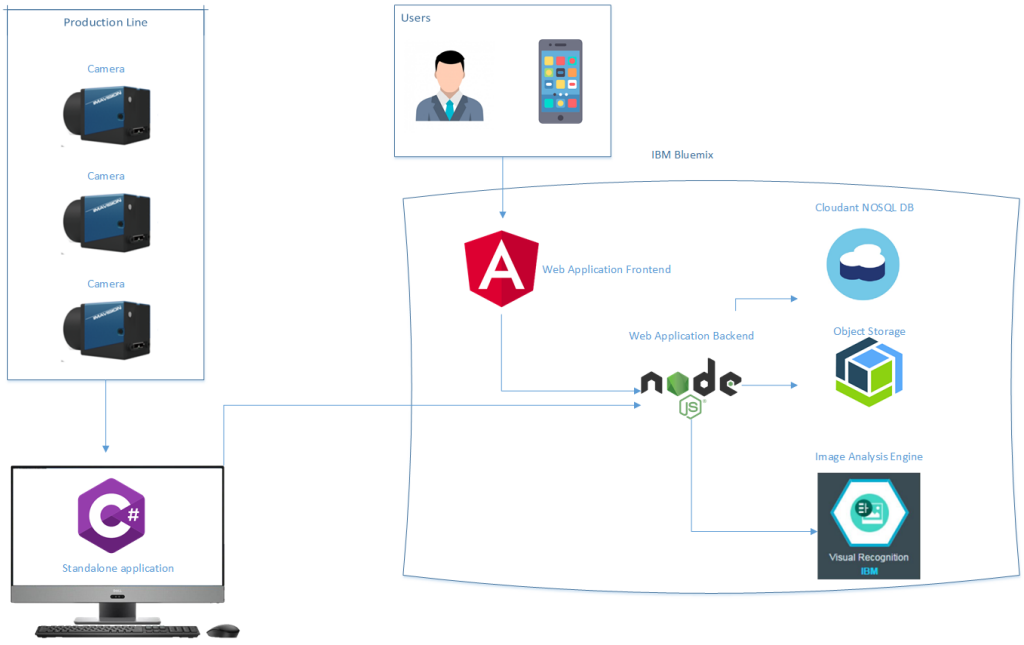


Figura 3.2: Architettura del sistema di Quality Inspection

3.2 Processi produttivi da ridefinire

Questa fase della progettazione è dedicata alla ridefinizione, successiva all'introduzione del sistema di Quality Inspection, dei processi produttivi. La notazione utilizzata è **BPMN** (*Business Process Model and Notation*), sviluppata da OMG (Object Management Group). [35]

I tre attori coinvolti nel processo sono la business function manifatturiera del cliente, il fornitore del servizio di Quality Inspection (Web Application - Backend) e del Deep Learning as a Service/DLaaS (IBM Cloud - Visual Recognition).

In questa tesi sarà specificata l'implementazione della Web Application intermediaria del servizio di classificazione.

3.2.1 Ispezione di Qualità in tempo reale

Il processo principalmente coinvolto riguarda l'ispezione durante la fase produttiva. La figura 3.5 ne mostra il flusso, mentre la figura 3.3 rappresenta la configurazione dell'hardware sul campo.

1. Il processo inizia con l'acquisizione dell'immagine del prodotto mediante la camera dedicata. Questa può essere effettuata manualmente da un operatore o automaticamente, in modo temporizzato o basato su sensori. L'attività di invio dell'immagine all'applicazione web è effettuata da un client installato su un computer edge, a cui è collegato una o più camere.

2. L'applicazione web, ricevuta una richiesta autenticata di classificazione, decide in base ai parametri di consumo definiti se effettuare o meno la classificazione.

3. Se questa primo controllo è superato, l'immagine è inoltrata al classificatore reale e da esso si attende la risposta con i punteggi di confidenza relativi alla difettosità o meno del prodotto.

4. Una volta ricevuta la risposta dal servizio di Visual Recognition, l'applicazione web esegue l'algoritmo di decisione basato sui punteggi ricevuti. I possibili risultati sono: **ok**, **defective** e **uncertain**.

5. Il risultato calcolato è restituito al cliente e archiviato insieme all'immagine relativa e ad altri parametri d'interesse, come i punteggi di confidenza e flag binari che indicano lo stato dell'immagine. Successivamente saranno mostrati i dettagli dello stato in cui possono trovarsi le immagini.

6. Infine la linea di produzione del cliente, collegata al client del sistema, reagisce in base al risultato di classificazione: se un prodotto risulta difettoso o incerto sono previste reazioni programmate come lo stop della linea di produzione, lo scarto automatizzato del prodotto difettoso o una segnalazione acustica.

3.2.2 Ispezione di Qualità posticipata

Questo processo, mostrato in figura 3.6, è molto simile alla controparte in tempo reale. In questo caso, le immagini sono acquisite offline, compresse in uno zip e caricate nella web app contemporaneamente. Una volta terminata la classificazione, i risultati sono visibili sulla web app.

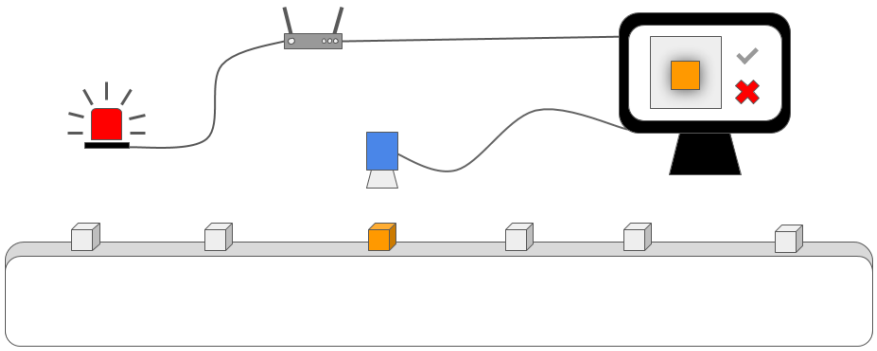


Figura 3.3: Schema stilizzato del sistema di Real Time Quality Inspection sul campo

3.2.3 Analisi di Classificazioni Incerte

Mentre si assume che per le immagini classificate "ok" o "defective" non ci sia bisogno di ulteriori controlli umani, per le immagini "uncertain" si richiede l'intervento di un **supervisore** umano che gli assegni manualmente la classe di difettosità. Si specifica che è comunque possibile modificare la classificazione di qualsiasi immagine, ma in assenza di un intervento le immagini non "uncertain" verranno usate dal modello per il retraining. In figura 3.7 è mostrato il processo in BPMN.

1. Il supervisore che ha il compito di visionare le immagini "uncertain" si autentica sulla web app ed entra nella sezione con le immagini da analizzare.

2. Il supervisore specificherà, attraverso l'interfaccia grafica dell'applicazione, se assegnare le immagini alla classe "ok" o "defective".

3. Il sistema salverà la nuova classificazione per ogni immagine e setterà i flag *analyzed* a TRUE e *retrained* a FALSE per segnalare che l'immagine dovrà essere usata per il retraining del modello.

3.2.4 Retraining del Modello

Il processo di retraining del modello in figura 3.8 è effettuato idealmente in un intervallo di tempo collocato fuori dai periodi di attività di classificazione, ad esempio durante il fine settimana. Il sistema è progettato per schedulare il retraining nel momento ritenuto più opportuno; inoltre è specificato un parametro di numero di immagini massime da usare per il retraining al fine di limitare i costi in base al budget disponibile.

1. Una volta iniziata la fase di retraining, il sistema recupera le immagini che non sono ancora state usate per il retraining (retrained=FALSE) e li raggruppa in due archivi zip separati, uno per ogni classe. Per evitare fenomeni come l'overfitting si richiede un numero minimo di immagini per entrambe le classi (50). Inoltre si risolve la sovrarappresentazione della classe "ok" effettuandone l'**Undersampling**, vincolando il numero di immagini "ok" nel training set ad essere al massimo pari al numero di immagini "defective". L'utente può decidere di applicare tecniche di **Data Augmentation** caricando manualmente varianti di immagini reali pronte per il retraining.

2. La Web Application invia le immagini al classificatore e sospende la disponibilità del modello a produrre classificazioni. Il periodo di attesa può arrivare a circa un'ora con un training set di 500 immagini.

3. Terminato il retraining, l'utente può classificare nuovamente le immagini con il modello appena riaddestrato.

3.3 Modello dei Dati

La successiva fase di progettazione riguarda la definizione del modello dei dati del sistema. In figura 3.4 è mostrata la rappresentazione UML dell'organizzazione dei dati dell'applicazione.

- **User**: per la gestione dei classificatori è richiesta l'autenticazione dell'utente. Pertanto sono salvati l'email e l'hash della password di tutti gli utenti autorizzati.

- **Classifier**: ai classificatori è associato un nome per la loro rapida identificazione dagli utenti.

  Vengono salvate le date di creazione e ultima modifica del classificatore: updated_local è aggiornato ad ogni operazione su di esso, mentre updated fa riferimento alla data di ultimo retraining, restituita direttamente dal fornitore del DLaaS insieme allo stato (status) del classificatore.

Lo stato può essere: "*ready*" (pronto per la classificazione), "*retraining*" (in fase di addestramento) e "*not trained*" (non ancora addestrato).

ok_class_weight è il parametro che bilancia l'influenza degli score delle classi "ok" e "defective" (per quest'ultima classe il peso è 1-ok_class_weight). I valori accettabili sono nell'intervallo [0,1]. Un valore pari a 0.5 non favorisce nessuna delle due classi, mentre un peso inferiore a 0.5 riduce le classificazioni di prodotti nella classe "ok" in casi di ambiguità.

cron_schedule è un'espressione **cron** [36], un formato usato da unix per la definizione di eventi periodici, che definisce la cadenza dei retraining. È possibile stabilire una frequenza settimanale, mensile o annuale andando a definire l'ora e il giorno della settimana/mese/anno.

max_images_retrain è il numero massimo di immagini utilizzabili per un singolo retrain del classificatore.

- **PaidEvent**: vengono memorizzati tutti gli eventi (event) a pagamento, come *retraining* e *classificazione*, il loro costo, l'istante (timestamp), e solo per gli eventi di tipo *retraining* sono salvati il numero di immagini usate per ogni classe.

- **Image**: le immagini sono salvate fisicamente su di un **Cloud Object Storage** e ne viene salvato il percorso (imagePath) sul database, oltre che il timestamp di salvataggio.

  L'immagine caricata è ridimensionata alla risoluzione di 256 x 256, poco superiore a quella minima richiesta dal servizio Visual Recognition di IBM, pari a 224 x 224.

  L'utente che inserisce l'immagine può specificare un titolo (title), una descrizione (content) e l'identificativo della camera che ha acquisito l'immagine (cam_id).

  È memorizzato il risultato di classificazione (classification_result) che può assumere valori *ok*, *defective* o *uncertain*.

  I flag reviewed e retrained indicano rispettivamente se l'immagine è stata controllata da un supervisore e se è stata usata per l'addestramento del classificatore.

- **Score**: sono memorizzate per ogni immagine classificata le coppie (class,score) che rappresentano il punteggio di confidenza per ogni classe.

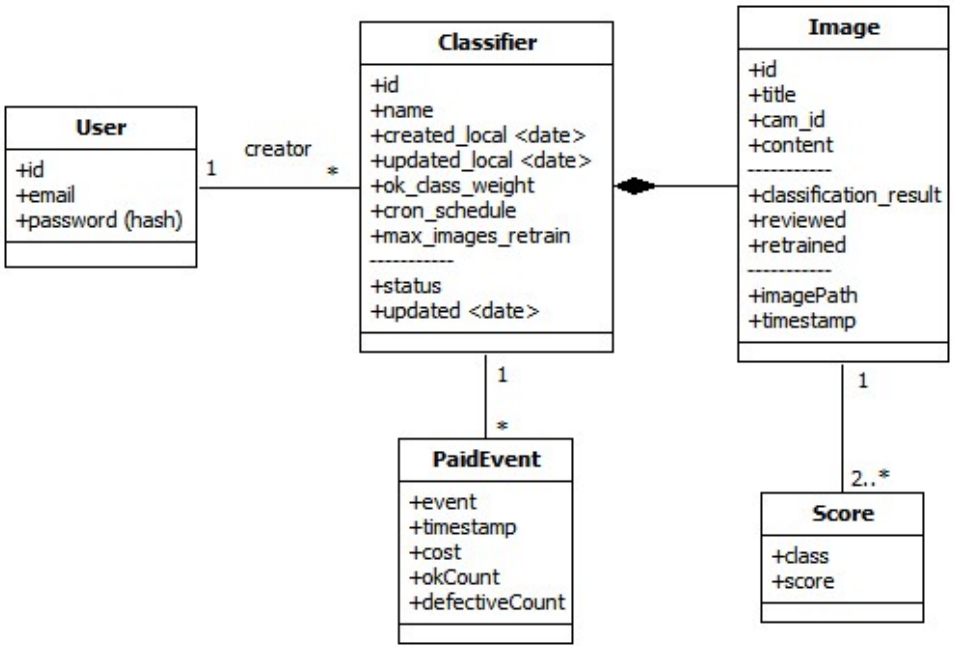


Figura 3.4: Modello UML dei dati del sistema

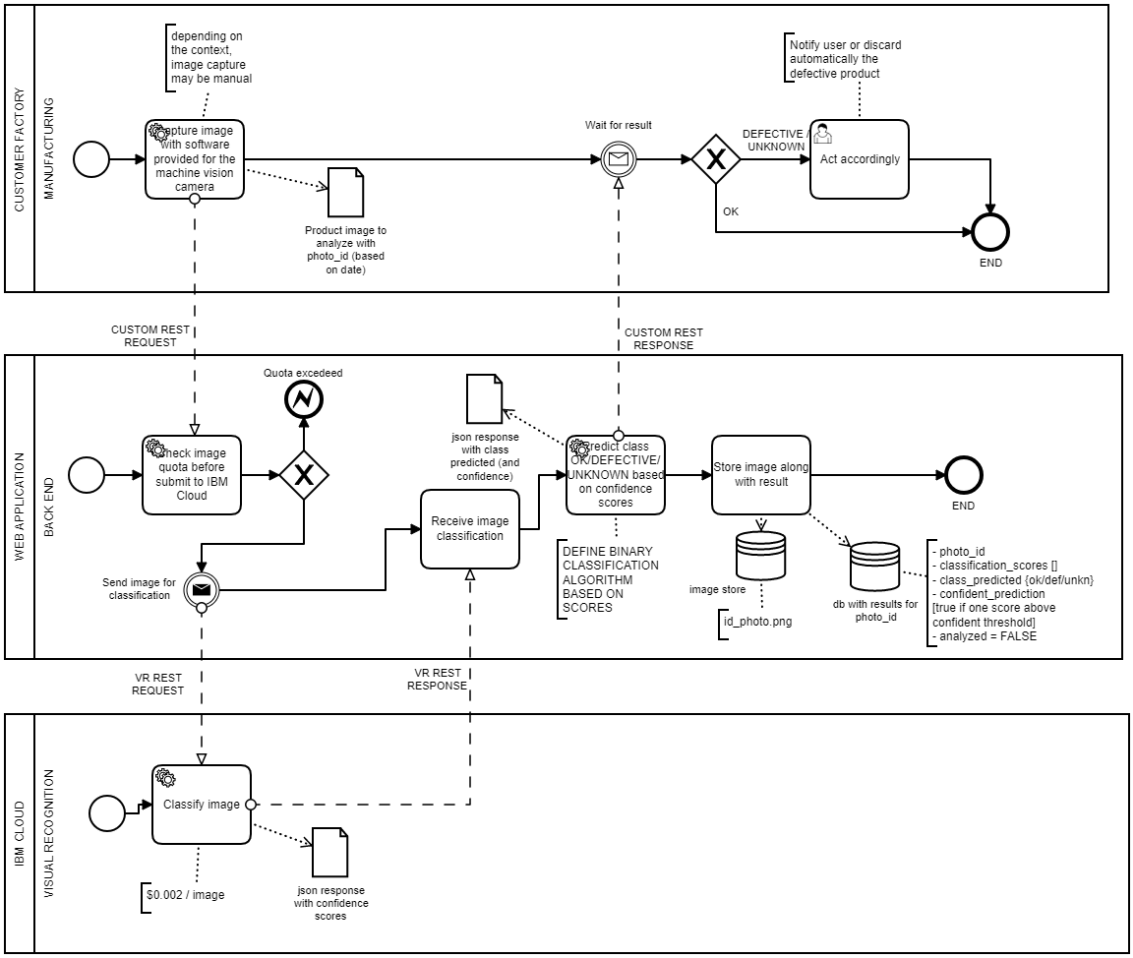


Figura 3.5: Processo di Real Time Quality Inspection

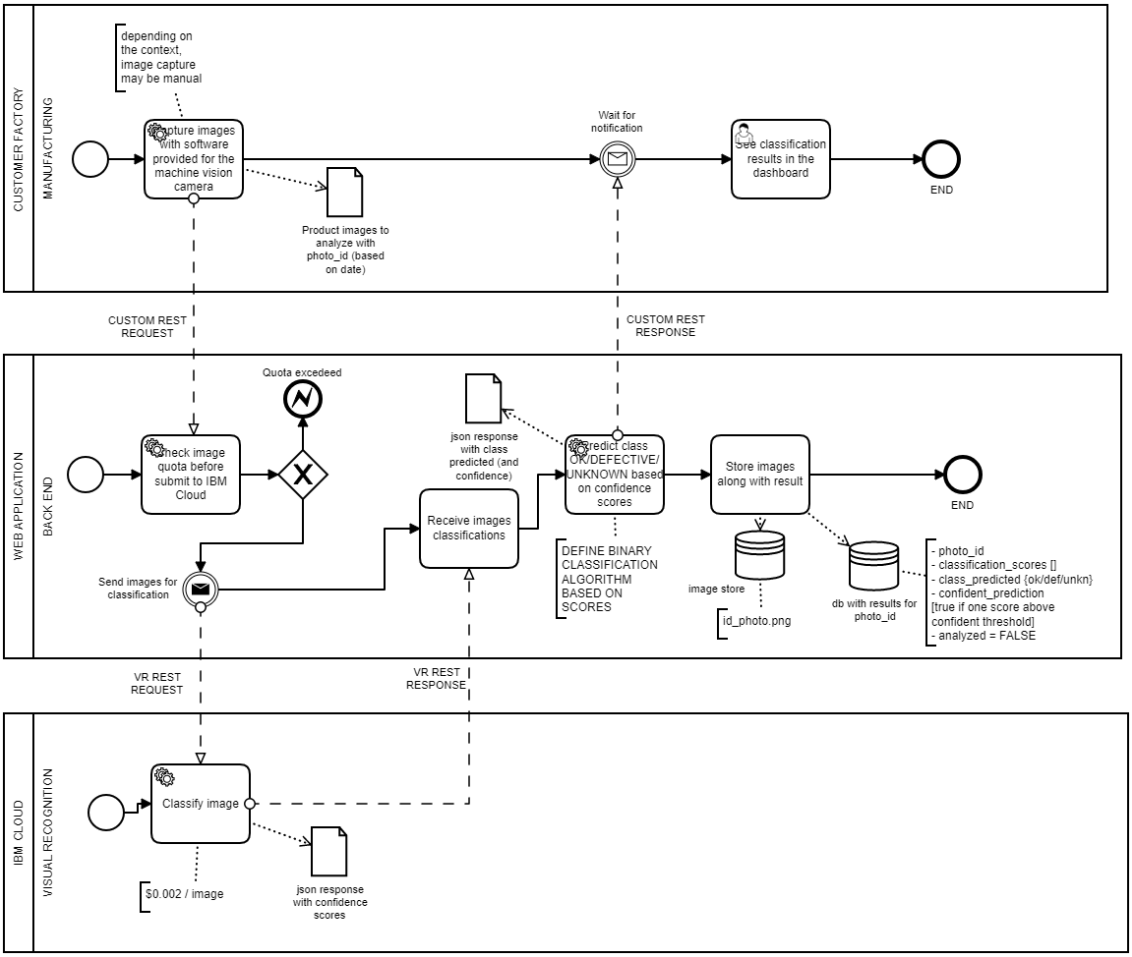


Figura 3.6: Processo di Postponed Quality Inspection

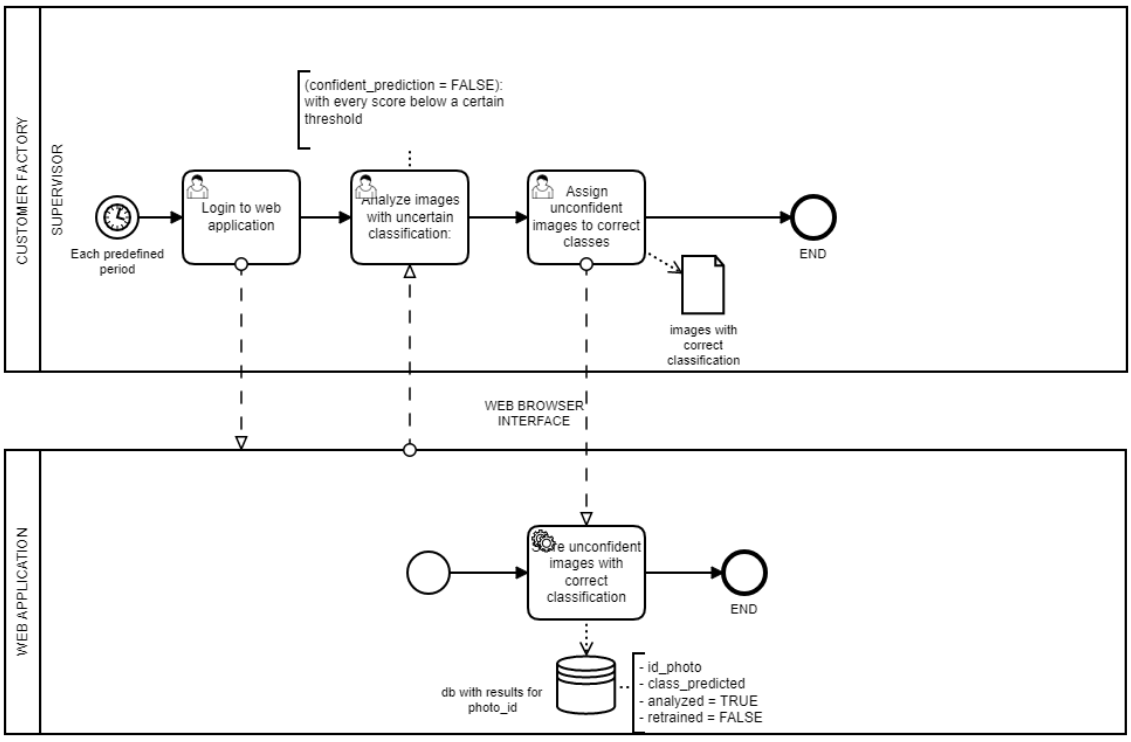


Figura 3.7: Processo di Analisi di Immagini classificate in modo incerto

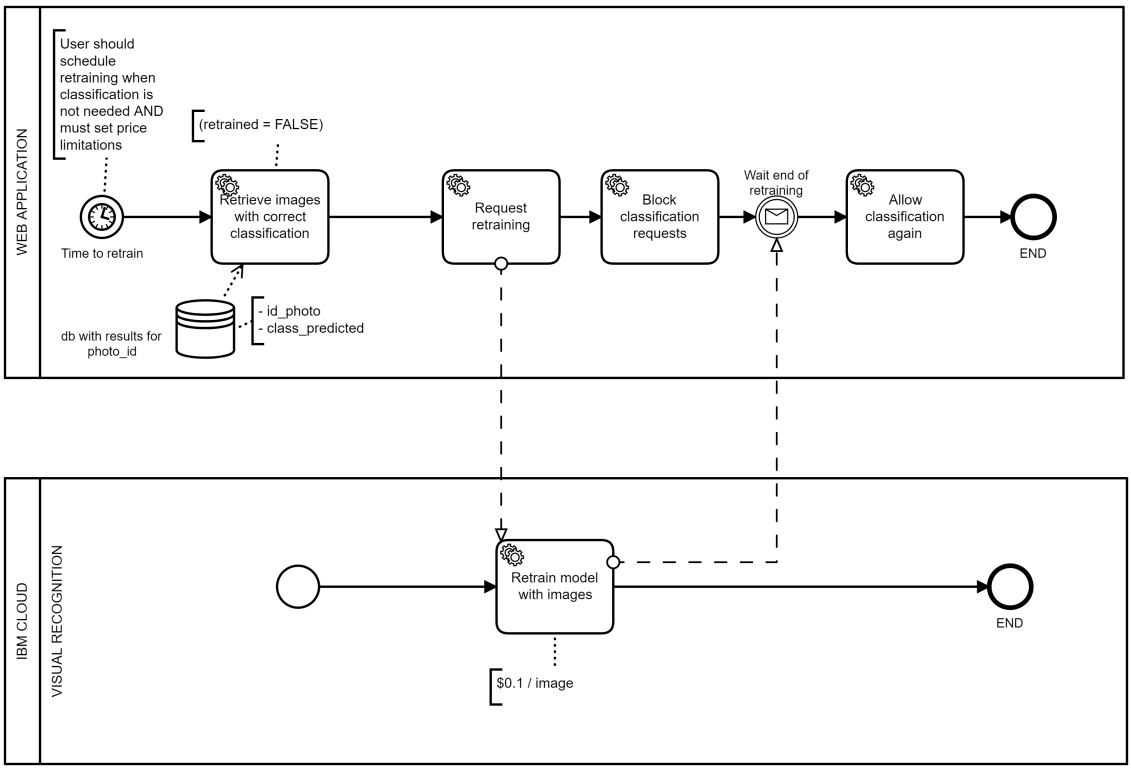


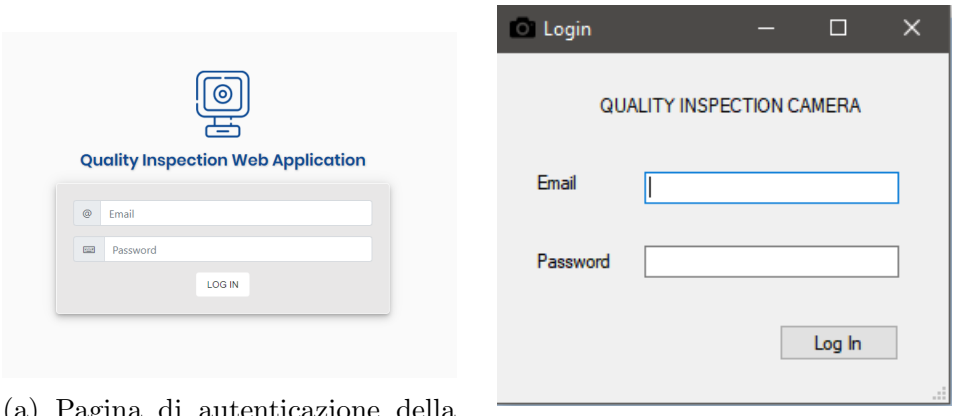
Figura 3.8: Processo di Retraining del Modello

Capitolo 4

Implementazione del sistema di Quality Inspection progettato

In questo capitolo sono mostrati in dettaglio le componenti principali del sistema di quality inspection.

4.0.1 Autenticazione



(a) Pagina di autenticazione della web application

(b) Autenticazione della camera client application

Figura 4.1

La pagina di benvenuto della web app richiede l'inserimento delle credenziali fornite dal gestore dell'applicazione ai clienti del servizio (figura 4.1a). Sono richieste le stesse credenziali anche per l'applicazione client della camera (figura 4.1b).

Il sistema di autenticazione, in entrambi i casi, è basato su **JSON Web Token**. [37] Una volta inserite e confermate le credenziali, il browser o il client invia una richiesta POST via https all'endpoint dedicato. La risposta del server conterrà un *token* e i secondi di validità di questo. Il token è una stringa contenente le informazioni codificate sull'utente autenticato e la firma del server. L'utente, attraverso il browser o il client, dovrà specificare il token nell'header di Bearer Authentication per tutte le richieste HTTP che richiedono autorizzazione. L'autenticazione dovrà essere rieffettuata ogni qualvolta scade il token.

4.1 Web Application

La schermata principale dell'applicazione web è mostrata in figura 4.2. A sinistra è possibile selezionare, mediante la barra di menu, la sezione d'interesse.

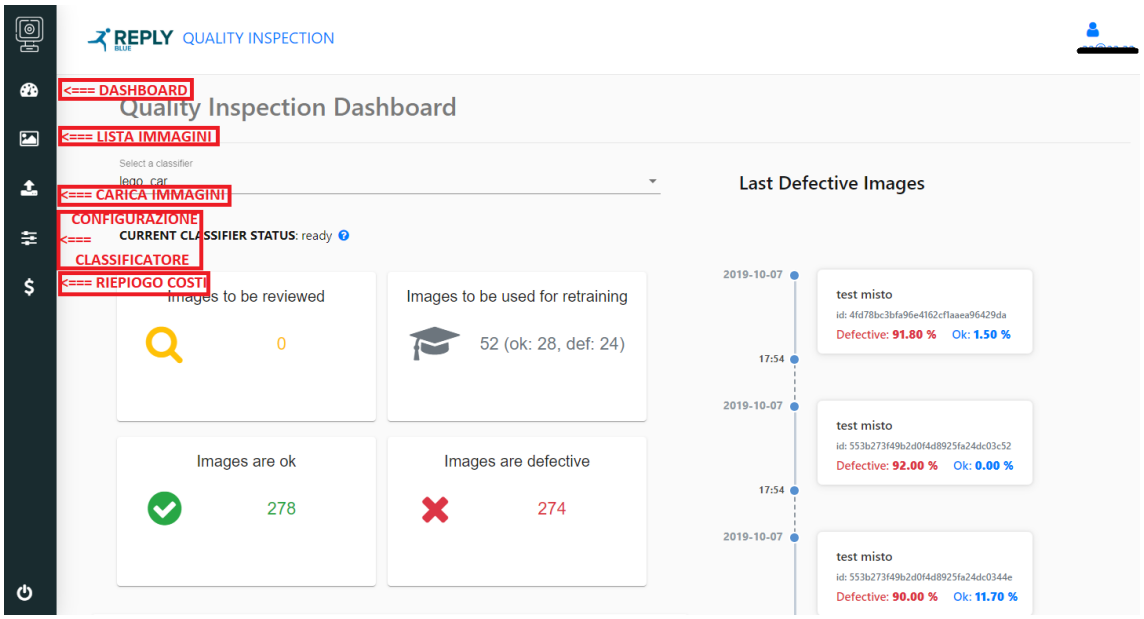


Figura 4.2: Home page della web application

4.2 Dashboard

La dashboard è la sezione principale dell'applicazione e ha lo scopo di mostrare a colpo d'occhio la situazione per ogni classificatore, selezionabile da un menu a tendina in alto.

Card riepilogative

- Immagini da revisionare

  la card mostra il numero di immagini che hanno ricevuto la classificazione "uncertain" e che non sono state usate per il retraining. L'interazione con la card reindirizza alla lista di immagini da revisionare, facilitando il compito al supervisore. Una volta che le immagini sono state correttamente catalogate dall'utente (figura 4.3a), sono selezionate per il prossimo retraining.

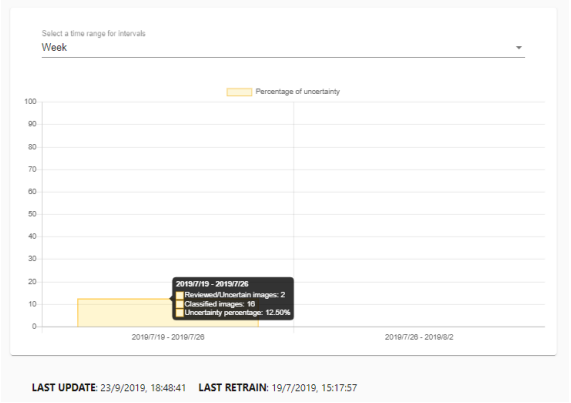
- Immagini da usare per il retraining

  accanto al numero di immagini destinate al training set è specificata la distribuzione di esse tra le due classi. Questo insieme di immagini contiene quelle che non sono ancora state usate per il retraning.

- Immagini "ok" e "defective"

  è mostrato il numero totale di immagini totali "ok" e "defective".

(a) Esempio di immagini da revisionare



(b) Grafico di incertezza nella classificazione

Figura 4.3

Last Defective Images

A destra appare una lista cronologica delle ultime immagini catalogate come difettose, permettendo agli operatori di far riferimento velocemente ad esse.

4.2.1 Grafico di Incertezza

In fondo alla pagina web è mostrato un grafico che mostra la **percentuale di incertezza** del classificatore in range temporali selezionabili (giornaliero, settimanale, mensile). Il grafico è mostrato in figura 4.3b.

La percentuale di incertezza è definita come:

$$\frac{immagini\ "uncertain"\ o\ "reviewed"}{immagini\ classificate} \tag{4.1}$$

Il trend atteso è discendente, poiché un classificatore addestrato meglio ha meno incertezza nel dare il risultato corretto. Di conseguenza il numero di immagini revisionate ("uncertain" e non) tende a diminuire.

Questa misura corrisponde all'opposto dell'accuratezza, cioè $1 - accuratezza$, in quanto il numero di immagini "uncertain" o "reviewed" costituisce quanto non classificato in modo corretto.

4.3 Lista Immagini

In figura 4.4 sono mostrate la lista di immagini, ordinate dalla meno alla più recente, associate ad un classificatore di test.

In alto, oltre al classificatore, è possibile scegliere i parametri di filtraggio delle immagini: per **stato** e **risultato di classificazione**.

Gli stati selezionabili sono "To Review" (da revisionare), "To Retrain" (da usare per il retraining) e "Classified" (immagini con classe assegnata dal classificatore). I risultati di classificazione selezionabili sono "ok" e "defective".

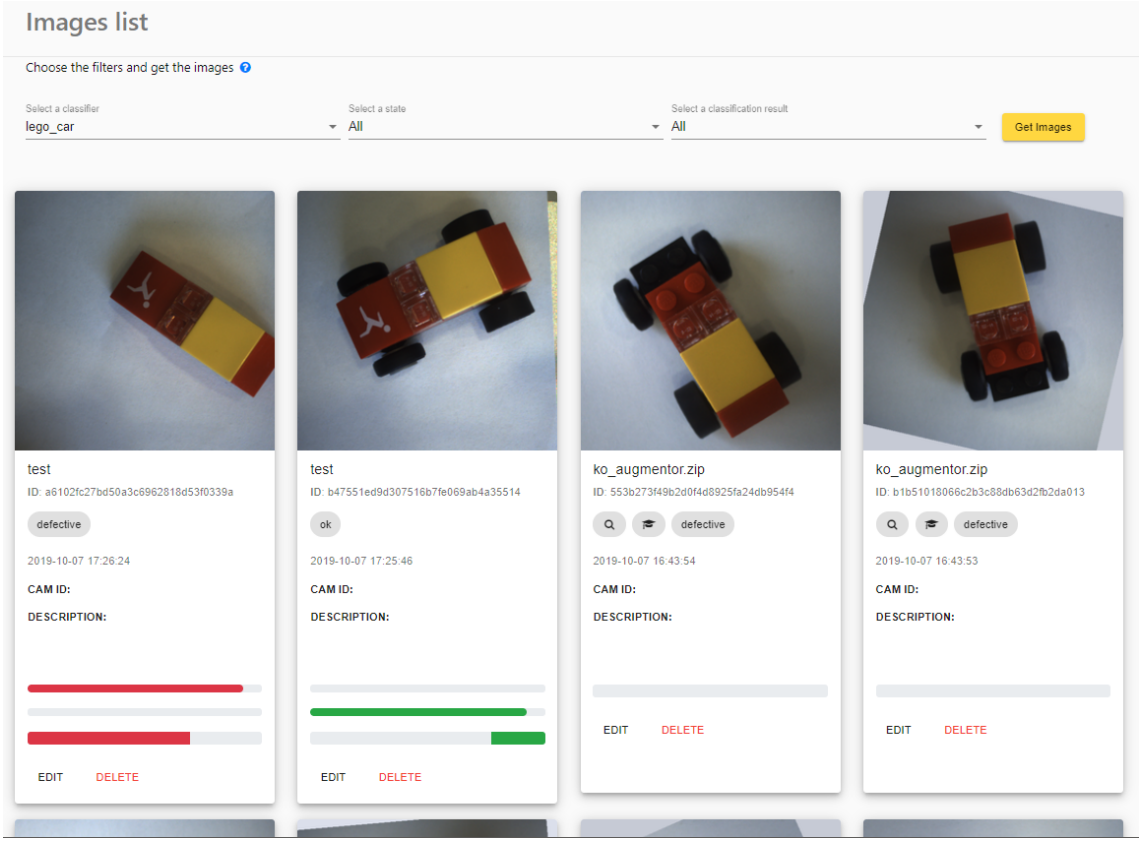


Figura 4.4: Lista di immagini associate ad un classificatore

Le immagini che hanno l'icona della *lente d'ingrandimento* e del *cappello accademico* indicano rispettivamente che l'immagine è stata sottoposta a revisione e che sono state usate dal modello per il riaddestramento.

Le due barre di colore rosso e verde indicano i punteggi di classificazione assegnati dalla rete neurale. Per la determinazione del risultato di classificazione (ok, defective o uncertain) si fa riferimento alla terza barra, di *evaluation result*, che assume valori compresi tra -100% e 100%.

4.3.1 Evaluation result

La formula che determina l'assegnazione della classe prevista all'immagine è la seguente:

$$EvalRes(img) = img.defScore * (1 - okWeight) - img.okScore * okWeight \quad (4.2)$$

dove $okWeight$ è un parametro del classificatore che bilancia l'influenza dei punteggi.

L'assegnazione della classe avviene secondo questo algoritmo:

```
se EvalRes(img) < 0
    img.classificationResult <- "ok"
se EvalRes(img) >= 0 AND EvalRes(img) < uncertaintyThr
    img.classificationResult <- "uncertain"
```

```
se EvalRes(img) >= uncertaintyThr
    img.classificationResult <- "defective"
```

"uncertaintyThr" è un altro parametro del classificatore che segna la soglia sotto il quale un immagine passa da "defective" ad "uncertain".

4.4 Caricamento Immagini

Nella pagina di caricamento delle immagini mostrata in figura 4.5 è possibile caricare una o più immagini in formato .png o .zip, specificandone i dettagli.

Le azioni da eseguire al caricamento dell'immagine/i sono:

- **Classifica**

  il processo di quality inspection posticipata termina con il caricamento delle immagini acquisite per la loro classificazione.

- **Segna come Ok/Defective**

  il primo training del classificatore richiede il caricamento di un certo numero di immagini a cui è già assegnata una etichetta di classe.

4.5 Configuratore Classificatore

In questa pagina (figura 4.6) sono configurabili i diversi parametri del classificatore visti in precedenza. Inoltre è possibile eseguire manualmente il retraining di un classificatore.

4.6 Riepilogo Costi

In base agli eventi di training e classificazione vengono calcolati e mostrati i costi previsti (figura 4.7).

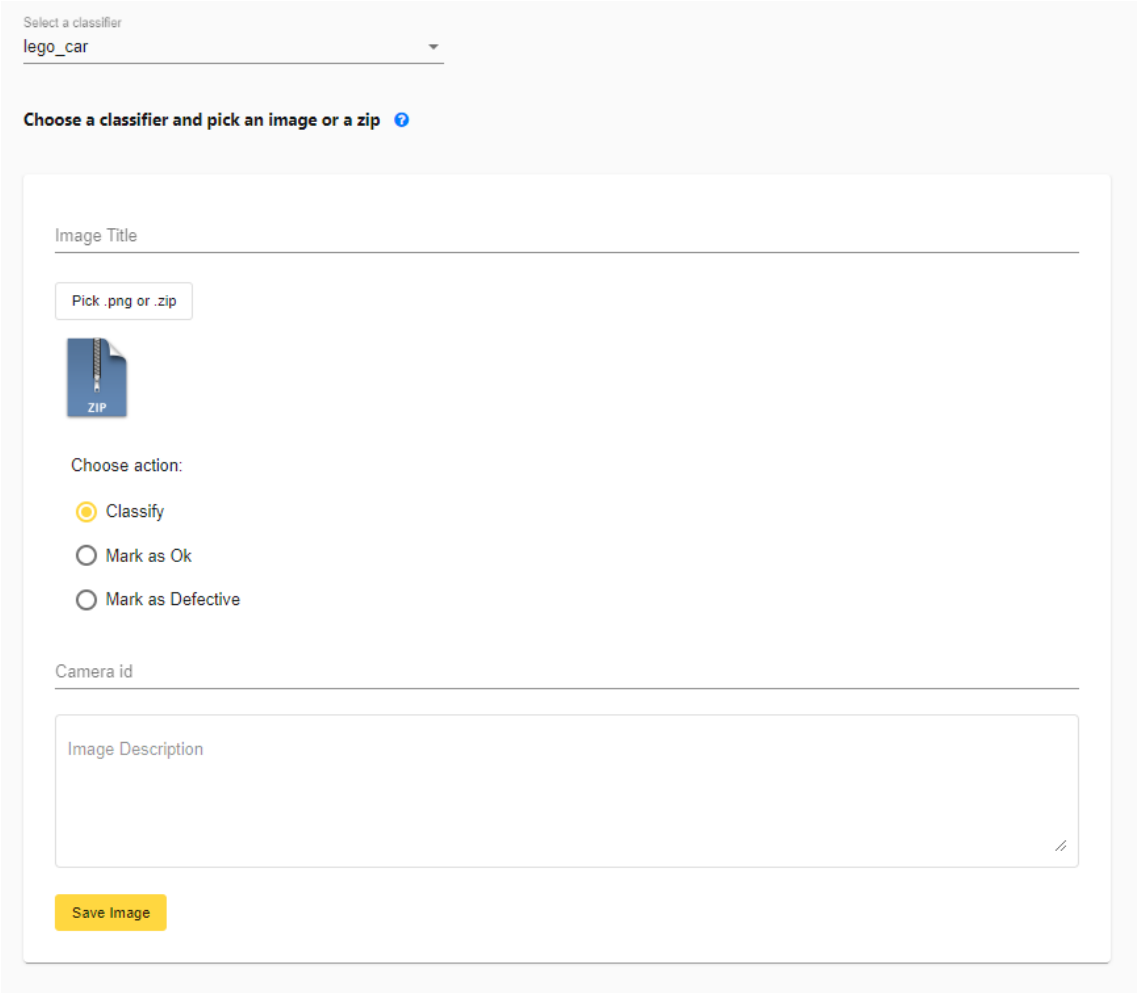


Figura 4.5: Sezione per il caricamento di immagini

4.7 Camera Client Application

Lo sviluppo del software per gestire l'invio delle immagini dalla camera al server è legato all'SDK (Software Development Kit) offerto dalla machine vision camera scelta.

4.7.1 Machine Vision Camera scelta

La camera scelta è *Mercury MER-041-436U3C-L*, con una lente LCM-5MP-06MM-F2.0-1.8-ND1 [38] [39]. Nella tabella 4.1 sono mostrate le specifiche più importanti della camera scelta. In figura 4.8 è mostrata la camera montata nel supporto usato in fase di test.

4.7.2 Interfaccia Utente

La GUI (Graphical User Interface) dell'applicazione è mostrata in figura 4.9. È stata sviluppata in C#, uno dei linguaggi offerti dalla SDK della camera Mercury. [40]

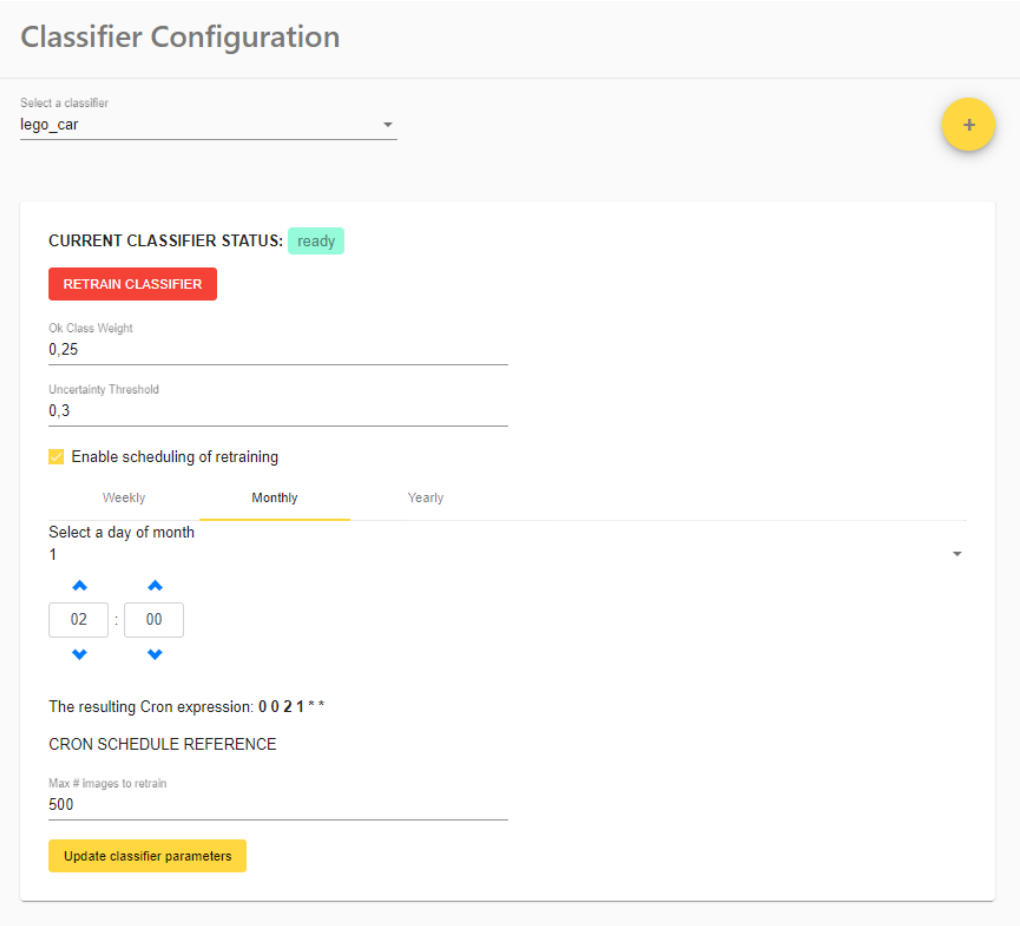


Figura 4.6: Schermata per la configurazione del classificatore

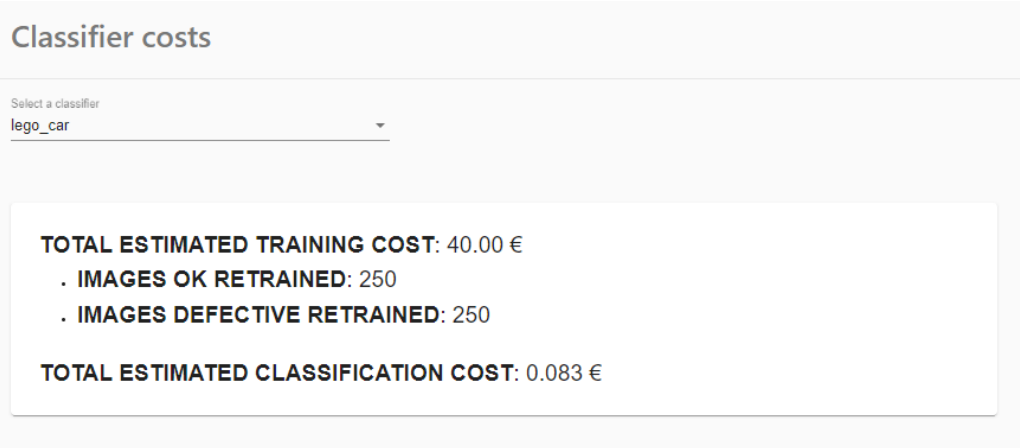


Figura 4.7: Riepilogo dei costi

Caratteristica	Valore	Note
Color/Mono	Color	
Tipo di Sensore	Sony IMX287	
Resolution	720x540	La risoluzione è sufficiente per acquisire un immagine quadrata con uno zoom massimo di 2,10%.
Frame Rate	436fps	L'alto numero di frame acquisiti al secondo rende questa camera utile in ambienti con oggetti in movimento.
Tipo di Shutter	Global Shutter	Evita l'apparizione del fenomeno di flickering dell'immagine.
Controlli programmabili	Image size, gain, Exposure time, etc.	Interfaccia che permette la trasmissione di un alta risoluzione e di un alto numero di frame.
Interfaccia Dati	USB3.0	Interfaccia che permette la trasmissione di un alta risoluzione e di un alto numero di frame.

Tabella 4.1: Caratteristiche della *Mercury MER-041-436U3C-L*[38]

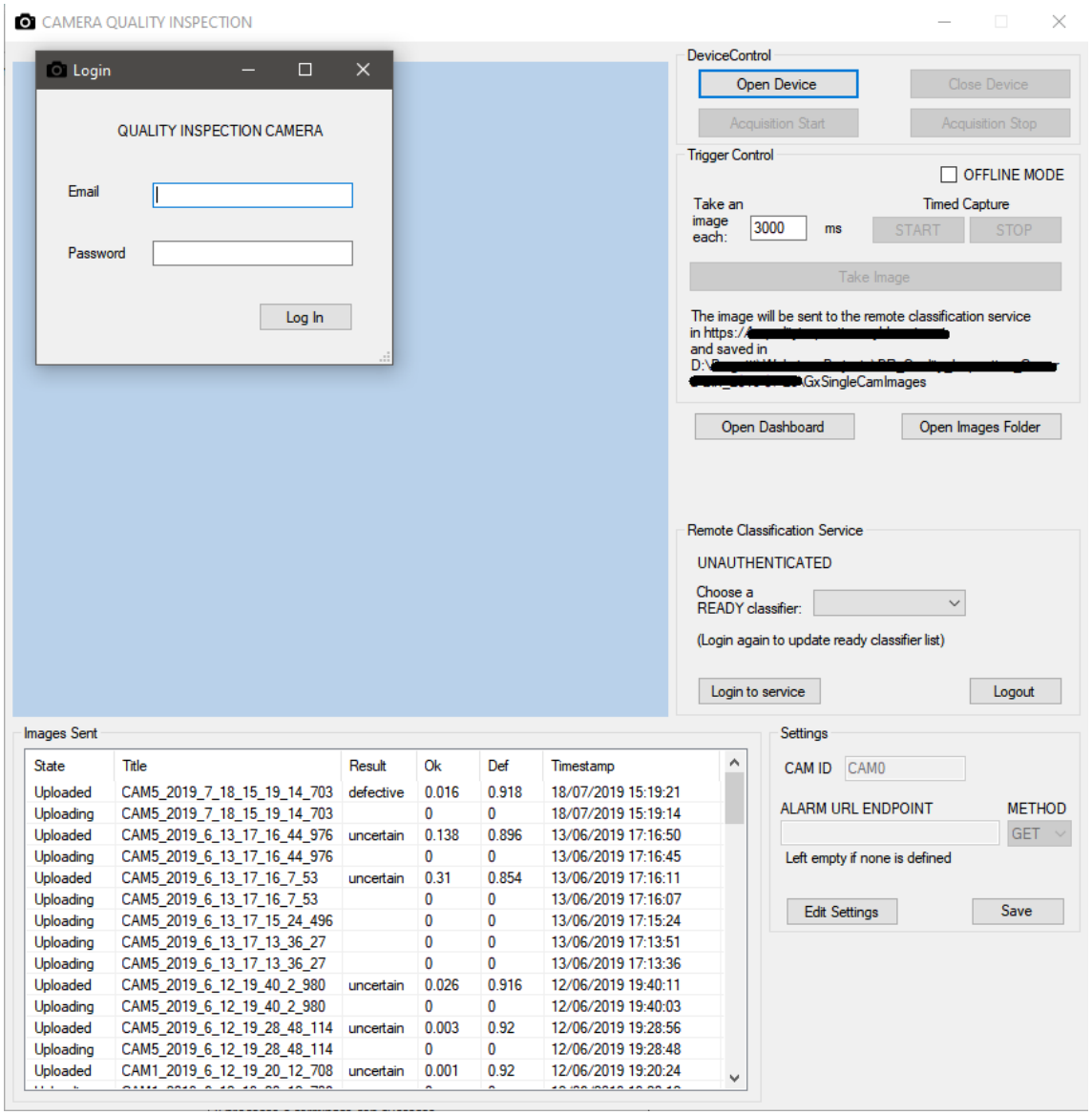
Figura 4.8: Configurazione di test della camera

Device Control

Prima di poter utilizzare il client è necessario installare il driver della camera presente nella sezione download del produttore. [40] L'applicazione richiede al sistema operativo (SO) di ottenere il controllo esclusivo della camera; se il SO riesce a soddisfare la richiesta sarà possibile iniziare l'acquisizione del flusso di immagini dalla camera.

Trigger Control

In questa sezione del programma è possibile acquisire le immagini manualmente cliccando su "Take Image". Il processo è semiautomatizzabile impostando il tempo tra un acquisizione e l'altra e iniziando la "Timed Capture". Nel caso in cui non sia



State	Title	Result	Ok	Def	Timestamp
Uploaded	CAM5_2019_7_18_15_19_14_703	defective	0.016	0.918	18/07/2019 15:19:21
Uploading	CAM5_2019_7_18_15_19_14_703		0	0	18/07/2019 15:19:14
Uploaded	CAM5_2019_6_13_17_16_44_976	uncertain	0.138	0.896	13/06/2019 17:16:50
Uploading	CAM5_2019_6_13_17_16_44_976		0	0	13/06/2019 17:16:45
Uploaded	CAM5_2019_6_13_17_16_7_53	uncertain	0.31	0.854	13/06/2019 17:16:11
Uploading	CAM5_2019_6_13_17_16_7_53		0	0	13/06/2019 17:16:07
Uploading	CAM5_2019_6_13_17_15_24_496		0	0	13/06/2019 17:15:24
Uploading	CAM5_2019_6_13_17_13_36_27		0	0	13/06/2019 17:13:51
Uploading	CAM5_2019_6_13_17_13_36_27		0	0	13/06/2019 17:13:36
Uploaded	CAM5_2019_6_12_19_40_2_980	uncertain	0.026	0.916	12/06/2019 19:40:11
Uploading	CAM5_2019_6_12_19_40_2_980		0	0	12/06/2019 19:40:03
Uploaded	CAM5_2019_6_12_19_28_48_114	uncertain	0.003	0.92	12/06/2019 19:28:56
Uploading	CAM5_2019_6_12_19_28_48_114		0	0	12/06/2019 19:28:48
Uploaded	CAM1_2019_6_12_19_20_12_708	uncertain	0.001	0.92	12/06/2019 19:20:24

Figura 4.9: Interfaccia grafica dell'applicazione client collegata ad una camera

necessario conoscere il risultato in tempo reale è possibile selezionare l'*OFFLINE MODE* e salvare le immagini soltanto nella cartella locale (Open Image Folder).

Remote Classification Service

Il riquadro specifica lo stato attuale di autenticazione al servizio di classificazione. Lo stato è inizialmente non autenticato ed è possibile cambiarlo ad autenticato effettuando il login all'avvio dell'applicazione o cliccando su *Login to Service*. Una volta autenticati è mostrata la durata della sessione.

Images Sent

La tabella nella sezione mostra gli eventi riguardanti le immagini caricate. È possibile tenere traccia dell'ordine e dei tempi di caricamento delle immagini, oltre che visualizzarle con doppio clic sulla riga scelta. Sono mostrati i dettagli di classificazione quali i punteggi delle classi e il risultato previsto.

Settings

È possibile specificare l'ID della camera, così da permettere al sistema di distinguere le immagini in base al loro luogo di acquisizione.

L'*alarm URL endpoint* è il riferimento ad un dispositivo collegato alla rete che può ricevere richieste HTTP per attivare un segnale sonoro.

4.8 Data Augmentation Script

Per il training del modello di classificazione può essere sufficiente utilizzare le immagini acquisite dei prodotti, senza particolari modifiche. Ciò è accettabile quando si ha già a disposizione un gran numero di immagini prese in tutti i possibili contesti. Nelle fasi di test del sistema non ho avuto a disposizione dei prodotti reali in un contesto produttivo reale, quindi ho dovuto simulare dei prodotti e difetti relativi. Ho usato uno script Python per effettuare l'*oversampling* delle immagini. Lo script usa la libreria di data augmentation chiamata **Augmentor** [41].

4.8.1 Pipeline di data augmentation

Di seguito è mostrato lo pseudocodice della pipeline di data augmentation.

```
# Tutte le immagini possono essere distorte, capovolte
# ruotate e cambiate in luminosità e contrasto
- distorci di 0.1 con probabilità del 10%

- capovolgi orizzontalmente con probabilità del 50%
- capovolgi verticalmente con probabilità del 50%

- ruota casualmente di 90/180/270 gradi con probabilità del 70%
- ruota senza ritagli di un valore tra -35 e 35 gradi
    con probabilità del 70% e riempi lo sfondo di grigio chiaro

- cambia casualmente la luminosità in un valore tra 80% e 150%
    della luminosità originale con probabilità del 50%
- cambia casualmente il contrasto in un valore tra 80% e 100%
    del contrasto originale con probabilità del 50%

# Il primo insieme di immagini può ruotare senza ritagli
# ai bordi, aggiungendo uno sfondo leggermente grigio
- campiona n/2 immagini

# Il secondo insieme di immagini può ruotare
# ritagliando i bordi
- rimuovi l'eventuale rotazione senza ritagli eseguita prima
- ruota con ritaglio di un valore tra -12 e 12 gradi
    con probabilità del 70%
- campiona n/2 immagini
```

Le trasformazioni che uso sono quelle mostrate nel paragrafo dedicato alla data augmentation. Nello script sono usati distorsione (*skew*), flip, rotazione, modifica luminosità e contrasto.

La rotazione finale è la combinazione di una rotazione di 0/90/180/270 gradi e di un offset di massimo 35 gradi per la rotazione senza ritaglio e di massimo 12 gradi per quella con ritaglio. In figura 4.10 sono mostrati due esempi di rotazione con e senza ritaglio: alla versione senza ritaglio è aggiunto uno sfondo leggermente grigio.

A ogni trasformazione è associata la probabilità che essa venga effettuata, e per alcune di queste sono associati dei range di valori casuali scelti. Esempi di range sono la percentuale di luminosità oppure la rotazione minima o massima.

Le immagini generate sono successivamente caricate nell'applicazione web, marcate con la loro classe e utilizzate per il successivo retraining.

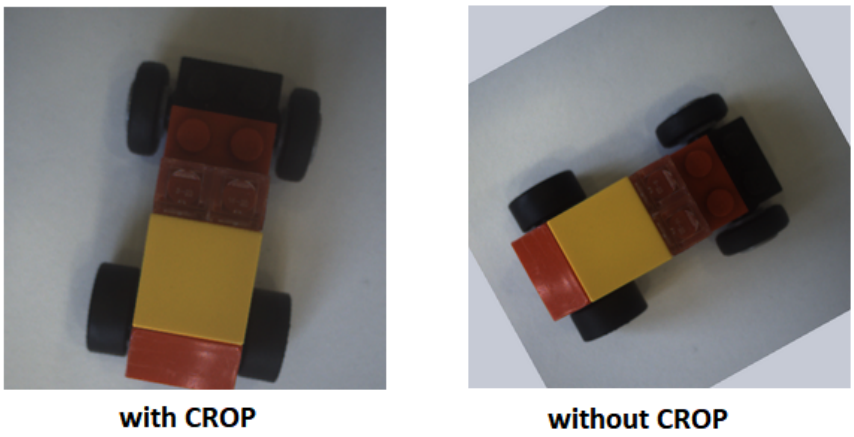


Figura 4.10: Esempio di immagine a cui è stata effettuata una rotazione con e senza crop

Capitolo 5

Valutazione dei risultati sperimentali

Il sistema di Quality Inspection progettato è stato sottoposto, durante e al termine dell'implementazione, a diversi test sull'accuratezza dei risultati. Prima di mostrare quanto ottenuto sono presentate le tecniche più comuni per la valutazione di classificatori sia in generale che nel caso particolare di classi non bilanciate.

5.1 Valutazione Classificatori

I modelli costruiti per la classificazione devono poter essere valutati per stabilire l'efficacia dell'applicazione di essi in un sistema reale.

Tra le **tecniche di partizionamento** per training set e test set necessarie per effettuare la valutazione di un modello le più note sono**Holdout** e **Cross-Validation**. [13]

Una volta scelto uno dei metodi suddetti è possibile applicare determinate metriche di valutazione, tra cui la più nota è l'**accuratezza**. Esistono ulteriori metriche, **richiamo** e **precisione**, che risultano più adatti in casi particolari. [42]

Infine, per la comparazione di classificatori binari (con sole due classi) è diffuso l'uso della **curva ROC**. [42]

5.1.1 Fattori che influiscono sulla qualità

Sono svariati i fattori da cui dipende la qualità: principalmente l' **algoritmo di training** usato ma anche **quanti dati sono presenti nei training set e test set** e **come le classi sono distribuite** all'interno di essi. Nel grafico in figura 5.1 il trend dell'accuratezza mostra come un numero troppo basso di elementi nel training set rende la classificazione incerta. Le linee blu indicano la deviazione standard dell'accuratezza. Dall'analisi di questi dati si deduce che l'accuratezza del modello considerato si stabilizza sopra i 100 elementi nel training set.

In contesti come la previsione di difetti o malattie, gli **errori di classificazione** non hanno **lo stesso costo**. In ambiente medico, difatti, un falso positivo è meno grave di un falso negativo: un soggetto che emerge positivo ad un test di una qualsiasi malattia ma che successivamente scopre di non esserlo è a livello sociale un prezzo da pagare inferiore rispetto al non diagnosticare per tempo una malattia.

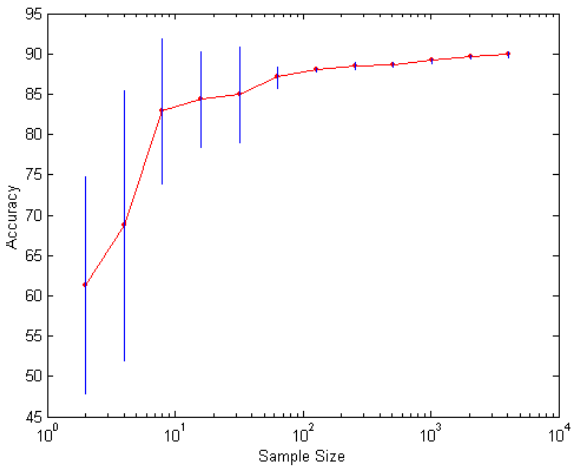


Figura 5.1: Andamento dell'accuratezza al variare della dimensione del training set [13].

5.2 Metodi per la Valutazione delle Prestazioni di un Modello

La costruzione di un modello richiede la raccolta di dati da distribuire tra training e test set. Ma come effettuare la ripartizione? Per entrambi i metodi proposti è consigliabile che ogni classe sia essere rappresentata in egual maniera per ogni sottoinsieme da definire, perciò si applica il **campionamento stratificato**.

5.2.1 Holdout

Nel metodo **holdout** si effettua una ripartizione fissa tra training e test set. Questa soluzione è indicata per i grandi insiemi di dati poiché più semplice da effettuare.

Di contro, il modello non sarà allenato sui dati del test set. Un training set troppo piccolo può causare una grande varianza del modello, mentre un test set troppo piccolo può restituire una misura di accuratezza non veritiera.

5.2.2 Cross-Validation

Con la **Cross-Validation** l'insieme di tutti i dati è suddiviso in k sottoinsiemi disgiunti (*fold*).

A turno ogni *fold* è scelto come test set e i restanti $k-1$ *fold* sono uniti nel training set. Per il calcolo dell'errore totale si sommano tutti gli errori di ogni passo della validazione. Il vantaggio principale di questa tecnica è dovuto al fatto che tutti gli elementi sono testati a turno.

Nel caso k sia uguale al numero totale di elementi, il test set sarà costituito da un singolo elemento. Questa variante è detta **Leave-One-Out**. Il training set, formato da $n-1$ elementi, permette di usare più dati possibili per il training ma risulta computazionalmente oneroso per dataset eccessivamente grandi.

5.3 Problema dello squilibrio tra classi: Metriche Alternative di Valutazione

Il caso studio di questa tesi impone un problema da valutare nella progettazione di un classificatore: lo squilibrio nella distribuzione delle classi. È chiaro che in un applicazione reale il numero di prodotti difettosi è decisamente inferiore a quello dei prodotti non difettosi.

Definiamo l'**accuratezza** come:

oggetti classificati correttamente / # totale di oggetti classificati

Questa misura non è sempre adatta nel nostro caso. Ipotizzando che l'1% dei prodotti è difettoso, un classificatore che non rileva correttamente nessun prodotto difettoso risulta avere comunque un'accuratezza del 99%.

5.3.1 Matrice di confusione

In un problema di classificazione binario diremo positivo il caso raro e negativo il caso frequente [42]. Il caso positivo è quindi più rilevante di quello negativo e le metriche considerate devono tener conto di ciò.

	actual positive	actual negative
predicted positive	TP	FP
predicted negative	FN	TN

(a) Confusion Matrix

Recall	=	$\frac{TP}{TP+FN}$
Precision	=	$\frac{TP}{TP+FP}$
False Positive Rate	=	$\frac{FP}{FP+TN}$

(b) Definitions of metrics

Figura 5.2: Definizione di matrice di confusione e delle misure di richiamo, precisione e del tasso di falsi positivi [42]

Si introducono le seguenti misure assolute:

- **True Positive** (TP)

  numero di casi positivi correttamente previsti.

- **True Negative** (TN)

  numero di casi negativi correttamente previsti.

- **False Negative** (FN)

  numero di casi positivi erroneamente previsti come casi negativi.

- **False Positive** (FP)

  numero di casi negativi erroneamente previsti come casi positivi (falso allarme).

Queste metriche possono essere organizzate, come in figura 5.2, in una matrice che prende il nome di **matrice di confusione**. Da queste è possibile definire le seguenti misure percentuali mostrate in figura 5.2:

- **Tasso di False Positive**

  quanti negativi sono stati erroneamente rilevati come positivi.

- **Richiamo**

  quanti positivi lo sono effettivamente sul totale dei positivi reali.

- **Precisione**

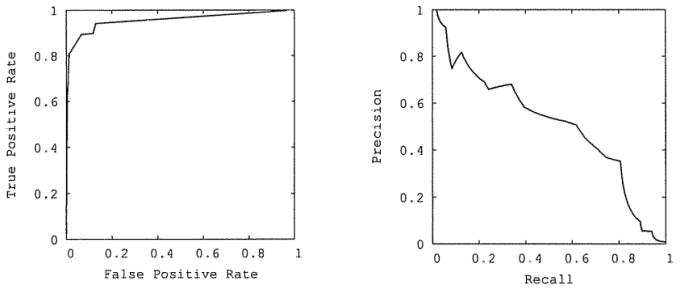
  quanti positivi lo sono effettivamente sul totale di positivi rilevati (inclusi i falsi positivi).

5.3.2 Richiamo e Precisione

Le misure di **richiamo** e **precisione** sono spesso usate in concomitanza nei classificatori binari e l'obiettivo è quello di massimizzare entrambe queste misure.

Queste misure possono essere unite nella F-measure, definita come la media armonica tra richiamo e precisione:

$$F = \frac{2rp}{r+p} \tag{5.1}$$



(a) Sample ROC curve (b) Sample PR curve

Figura 5.3: "La stessa curva mostrata nello spazio ROC e PR (richiamo-precisione)" [42]

5.3.3 Analisi Curva ROC

L'analisi della curva receiver operating characteristic (**ROC**) permette di visualizzare in modo grafico come bilanciare i tassi di true positive e false positive. La figura 5.3 rappresenta un esempio di curva ROC e la sua trasposizione nello spazio di richiamo-precisione. La curva è vincolata ai punti (0,0) e (1,1). I casi limite riscontrabili nella curva sono:

(0,0): tutti gli elementi sono rilevati come appartenenti alla classe negativa

(1,0): caso ideale

(1,1): tutti gli elementi sono rilevati come appartenenti alla classe positiva

La curva di un modello ben funzionante deve tendere il più possibile al punto (0,1), mentre un modello che effettua scelte a caso corrisponderà alla diagonale.

Il calcolo dell'area sotto la curva (AUC) restituisce la bontà del modello nel range [0.5, 1]. Nel caso in cui la curva abbia un area sotto la curva inferiore a 0.5 basta scambiare i positivi con i negativi e viceversa.

5.3.4 Approccio basato su Costi

Similmente a quanto fatto per la matrice di confusione, definiamo una **matrice dei costi** (figura 5.4) che indica quanto penalizzare gli errori di previsione del modello analizzato. [13]

		Predicted Class	
		Class = +	Class = −
Actual Class	Class = +	−1	100
	Class = −	1	0

Figura 5.4: Esempio di matrice dei costi [13]

La mancata rilevazione di un difetto ha associato un costo generalmente superiore alla erronea rilevazione di un difetto in un prodotto non difettoso. La scelta dei costi è legata all'importanza relativa di rilevare i correttamente casi positivi rispetto quelli negativi.

Il costo totale di un modello di cui è definita la matrice di confusione è:

$$Cost_t(M) = TP \cdot Cost(+,+) + FP \cdot Cost(-,+) + FN \cdot Cost(+,-) + TN \cdot Cost(-,-). \tag{5.2}$$

dove $Cost(i,j)$ è il costo della rilevazione di un elemento di classe i come uno di classe j.

Esistono diversi approcci per l'integrazione del concetto di costo nell'apprendimento di un classificatore. Per l'argomento di questa tesi risulta interessante applicare la tecnica dei costi nella scelta della classe di output a partire dai punteggi di confidenza restituiti da un classificatore (come per quelli basati su reti neurali).

Dato un oggetto in input i da classificare, si definisce **confidenza** $Conf(i,c)$ il punteggio che un classificatore assegna su quanto sia "sicuro" che l'oggetto i appartenga alla classe c.

L'approccio naive nella selezione della classe da assegnare può essere definito come:

$$OutClass(i) = argmax_{c \in \{+,-\}}(Conf(i,c)) \tag{5.3}$$

Per quanto questo approccio possa essere già soddisfacente, assegnare dei costi alla scelta di una classe rispetto ad un altra concede all'utente la possibilità di correggere il tasso di falsi negativi. Possiamo pertanto modificare la scelta della classe di output in questo modo:

$$OutClass(i) = argmax_{c \in \{+,-\}}(Conf(i,c) \cdot \frac{1}{Cost(c,-c)}) \tag{5.4}$$

5.3.5 Undersampling e Oversampling

L'altro metodo per mitigare lo sbilanciamento delle classi è l'**Approccio Sampling-Based**, in cui per le classi sovrarappresentate si procede con l'**Undersampling** e per le classi sottorappresentate con l'**Oversampling**.

L'**Undersampling** consiste semplicemente nel selezionare un numero inferiore di elementi della classe negativa. Ciò può rendere il modello meno allenato per le

caratteristiche presenti solo negli elementi esclusi nella fase di subsampling. C'è da dire che questo è un rischio limitato nei contesti manifatturieri in cui i prodotti sono di principio uguali tra loro.

L'**Oversampling**, dal canto suo, consiste nella replica degli elementi della classe positiva fino al raggiungimento dell'equilibrio tra le due classi. Per l'analisi di immagini è possibile scansare il rischio di overfitting di questa pratica andando ad applicare la tecnica di *Data Augmentation*. Si salvano copie di un immagine variando in modo aleatorio caratteristiche dell'oggetto come rotazione, traslazione, luminosità.

L'uso congiunto di queste due tecniche evita che il modello non riesca più a distinguere i casi positivi e negativi per via di dettagli non sufficientemente analizzati in fase di training.

5.4 Competizione per Ispezione Ottica Industriale

Il primo test è stato effettuato su un dataset parte di una competizione del 2007 per la scelta del migliore algoritmo di "Weakly Supervised Learning for Industrial Optical Inspection" [43]. Il test è stato svolto prima del termine dello sviluppo e valuta le prestazioni del classificatore addestrato su IBM Visual Recognition. Il modello è stato generato con il supporto di IBM Watson Studio.

Le classi considerate sono "class2", la classe negativa e "class2_def", quella positiva. Le immagini sono generate artificialmente, ma sono simili a casi reali. Il difetto consiste nella presenza di un graffio sul pattern.

Il dataset originale è composto da 1000 immagini "class2" e 150 per "class2_def". Ho applicato la **Holdout Validation** considerando un rapporto di 9:1 tra training set e test set. Per far ciò ho suddiviso in 10 parti i set delle classi: 9 sono stati scelti per il training e 1 per il test set.

Ho effettuato il training completo, ma per motivi di limitazione del piano questo primo test è stato ridotto ad un paio di casi particolari.

In figura 5.5 sono considerate immagini originariamente parte del test set di 100 immagini negative e 15 positive. Le immagini 1.png, 991.png e 146.png sono campioni originali, mentre l'immagine 902_bad.png deriva da 902_good.png. In particolare è stato aggiunto a 902_good.png un difetto e l'immagine risultante è stata rinominata 902_bad.png. È possibile notare che il punteggio massimo restituito dal classificatore VR (Visual Recognition) è di 0.92. I risultati di questa prima fase di test sono in linea con l'aspettativa, ma è necessario considerare casi reali per poter dare un giudizio complessivo.

5.5 Board

La categoria di prodotti considerata in questa sezione è un chip soggetto a diversi tipi di difetti. Il caso negativo (senza difetti) è costituito dalla board con il chip coperto dal logo. In figura 5.6 sono mostrati esempi di board considerate difettose e non.

Il caso positivo si ha quando sono presenti uno dei seguenti difetti:
- assenza del logo

Figura 5.5: Risultati del test della competizione[43]

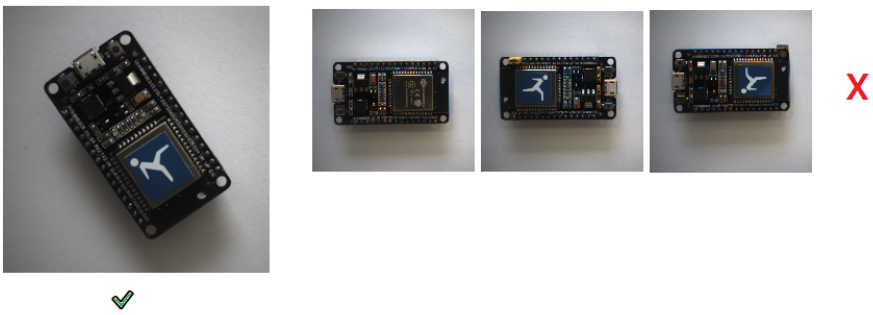


Figura 5.6: Esempio di prodotti "ok" e "defective"

- presenza di un jumper giallo
- con un pezzo di nastro adesivo

Per convenzione i prodotti rilevati come "uncertain" sono assegnati alla classe positiva. Per il calcolo dei valori della matrice di confusione sono considerate, dopo la revisione delle immagini uncertain, le immagini con le seguenti caratteristiche:

- True positive e true negative: immagini non revisionate

- False positive: immagini revisionate e che sono state originariamente rilevate dal classificatore come defective/uncertain

- False negative: immagini revisionate e che sono state originariamente rilevate dal classificatore come ok

Ricordiamo che il *richiamo* rappresenta la misura di quanto si è evitato di far passare difetti non rilevati, mentre la *precisione* indica quanto in percentuale si è

evitato di fermare la linea di produzione per un falso allarme.

5.5.1 Senza augmentation

Il primo modello è stato addestrato senza alcun tipo di data augmentation e acquisendo 150 immagini, 92 per il caso "ok" e 58 per il caso "defective". È stato scelto di non procedere nel training con data augmentation per poter successivamente valutare le differenze causate dal suo uso.

Per il test sono state considerate immagini riacquisite e, a partire da queste, generate con data augmentation. Il test set ha dato i risultati mostrati nella tabella 5.1:

	+ reale	**- reale**
+ previsto	**TP = 41**	**FP = 22**
- previsto	**FN = 8**	**TN = 56**

Tabella 5.1: Matrice di confusione del classificatore Board senza data augmentation

- Accuratezza = $\frac{TP+TN}{TP+FN+FP+FN} = 97/127 = 0,76$
- Richiamo = $\frac{TP}{TP+FN} = 41/49 = 0,84$
- Precisione = $\frac{TP}{TP+FP} = 41/63 = 0,65$
- F-measure = $\frac{2rp}{r+p} = 1,092/1,49 = 0,73$

I risultati sono sotto le aspettative di un sistema di quality inspection funzionante, ma ciò è giustificabile dal basso numero di immagini prese per la classificazione.

5.5.2 Con augmentation

Il secondo test differisce dal primo per l'uso di data augmentation e la riduzione dei difetti della board prodotta. L'unico difetto considerato è l'assenza del logo. Questa scelta è stata dettata dall'esigenza di dimostrare che fosse possibile creare un classificatore funzionante. In figura 5.7 sono mostrati campioni di immagini già sottoposte a data augmentation e usate per il training.

Figura 5.7: Prodotti "ok" e "defective" augmented per il training

In questo caso sono state usate 500 immagini per il training: 250 a testa per ogni classe. Il test set, costituito da 217 immagini per la classe "ok" (negativa) e 180 per i "defective" (positiva), ha dato i risultati nella tabella 5.2. Tra parentesi il numero di uncertain inclusi nel totale a fianco.

	+ reale	**- reale**
+ previsto	**TP = 178 (36)**	**FP = 1 (1)**
- previsto	**FN = 39**	**TN = 179**

Tabella 5.2: Matrice di confusione del classificatore Board con data augmentation

- Accuratezza = $\frac{TP+TN}{TP+FN+FP+FN} = 357/397 = 0,89$
- Richiamo = $\frac{TP}{TP+FN} = 178/217 = 0,82$
- Precisione = $\frac{TP}{TP+FP} = 178/179 = 0,99$
- F-measure = $\frac{2rp}{r+p} = 1,6236/1,81 = 0,89$

Analizzando l'impatto delle immagini "uncertain" sulla valutazione è possibile notare che il sistema è stato in grado di richiamare circa il 16% di prodotti che altrimenti sarebbero stati classificati come "ok".

D'altro canto è presente un numero non indifferente di falsi negativi. In figura 5.8 sono mostrati esempi di falsi negativi e true positive: la tendenza sembra essere quella di considerare "ok" prodotti "defective" che sono posizionati in un certo modo nell'immagine.

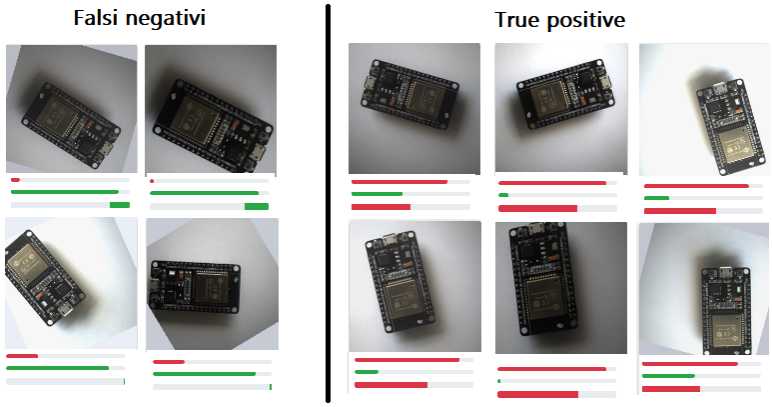


Figura 5.8: Campioni di falsi negativi e true positive del classificatore board con data augmentation

Aumentando ulteriormente il numero di immagini coinvolte nel training è previsto che questi fenomeni di overfitting si vadano ad assottigliare.

È possibile dire che le tecniche di data augmentation sono risultate utili per via della semplicità con la quale è possibile generare rotazioni di vario genere.

5.6 Macchina Lego

Un altro tipo di prodotto considerato per questa PoC è una macchina lego, facilmente utilizzabile poiché è possibile simulare i difetti cambiando le disposizioni dei blocchi dell'oggetto. In questo test utilizzeremo di nuovo del data augmentation e introdurremo diversi tipi di difetti.

In figura 5.9 si può notare sia il caso "ok", cioè la macchina senza alcun pezzo rimosso, e i casi "defective", con vari pezzi rimossi. In particolare i difetti sono:
1) ruote posteriori mancanti
2) ruote anteriori mancanti
3) ruote mancanti
4) cofano con logo mancante
5) tettuccio mancante

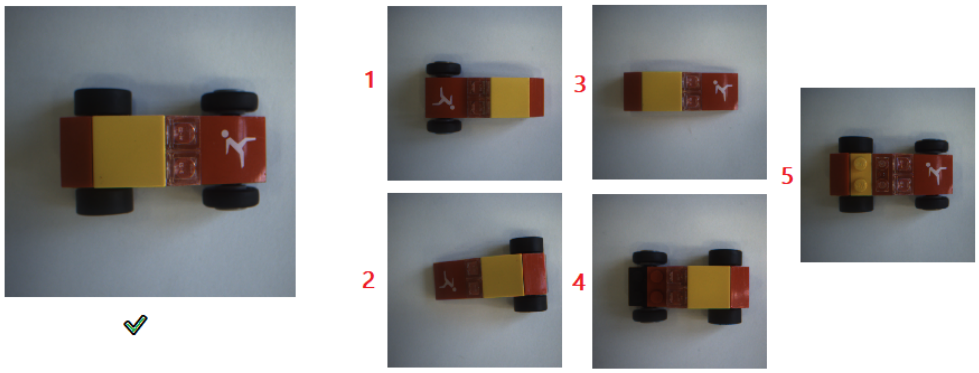


Figura 5.9: Esempi di macchina lego senza difetti e con i vari tipi di difetti

Anche in questo caso le immagini da usare per il retraining sono 250 a testa per ogni classe. Il test set, costituito da 215 immagini per la classe "ok" (negativa) e 189 per i "defective" (positiva), ha dato i risultati mostrati nella tabella 5.3.

	+ reale	**- reale**
+ previsto	**TP = 184 (7)**	**FP = 26 (15)**
- previsto	**FN = 5**	**TN = 189**

Tabella 5.3: Matrice di confusione del classificatore di macchine lego

- Accuratezza = $\frac{TP+TN}{TP+FN+FP+FN} = 373/404 = 0,92$
- Richiamo = $\frac{TP}{TP+FN} = 184/189 = 0,97$
- Precisione = $\frac{TP}{TP+FP} = 184/210 = 0,87$
- F-measure $\frac{2rp}{r+p} = 1,6878/1,84 = 0,92$

Questo classificatore ha, all'opposto di quello precedente, una maggiore capacità di richiamo ma una precisione inferiore.

L'accuratezza e l'f-measure sono molto simili e arrotondati risultano uguali. Prendendo come riferimento l'accuratezza, si nota come le prestazioni di questo modello siano superiori a quelli precedenti avvicinandosi molto alla soglia ideale di 0,95. Nei casi di test sono state aggiunte delle varianti di difetti somiglianti a quelli elencati ma non considerati nel training. In figura 5.10 ne sono mostrati tre esempi. Queste varianti non sono state considerate nel calcolo delle metriche di valutazione.

Figura 5.10: Casi particolari non inclusi nel training set

L'analisi delle caratteristiche dei falsi positivi e negativi, di cui sono mostrati degli esempi in figura 5.11, ha mostrato che per alcuni falsi positivi la motivazione è da ricondurre alle condizioni di luminosità o al ritaglio dell'oggetto che non è incluso completamente nell'immagine; mentre tra i falsi negativi è presente una consistente percentuale di macchine senza ruote posteriori.

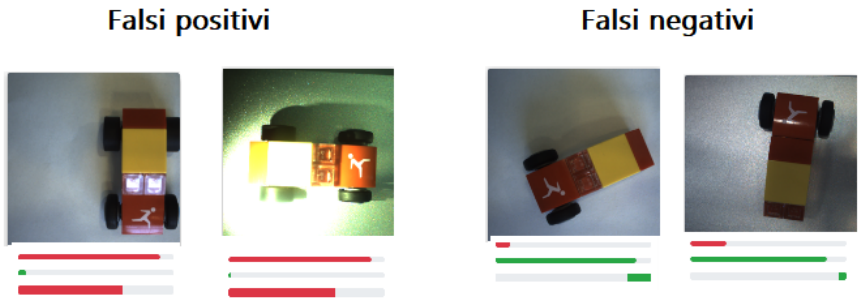


Figura 5.11: Comparazione tra falsi positivi e falsi negativi

Ci si aspetta che un retraining ulteriore possa raffinare i risultati rendendoli ottimali per un'implementazione reale.

Capitolo 6

Conclusioni

L'obiettivo posto all'inizio di questa tesi era quello di sviluppare la PoC di un sistema di quality inspection basato su cloud che fosse effettivamente utilizzabile in un contesto reale. Nei capitoli centrali è stata posta particolare attenzione alla descrizione delle componenti principali del sistema sviluppato, che ricordiamo essere l'applicazione web e il software client di cattura immagini.

L'applicazione web è accessibile in modo autenticato da più dispositivi contemporaneamente e gestisce il ciclo di vita di più classificatori, a partire dal loro primo training fino alla classificazione, catalogazione e archiviazione delle immagini caricate. L'applicazione supporta anche il processo di revisione delle immagini non classificate in modo esatto.

Il software installato sul computer edge, d'altro canto, è basato sul Software Development Kit offerto dai produttori della camera scelta per questa PoC. Le funzioni principali svolte da questo software sono l'acquisizione manuale o temporizzata delle immagini, l'invio in tempo reale di richieste di classificazione all'applicazione web mediante interfaccia REST, log dei risultati ed eventuale segnalazione di un prodotto difettoso mediante invio di un messaggio HTTP ad un alarm endpoint.

Questi due software sono stati sviluppati nell'ottica di poter dimostrare che fosse possibile applicare il principio di outsourcing dei servizi di deep learning. Considerando i processi BPMN mostrati nel *Capitolo 3*, è possibile dire che il sistema proposto riesce ad implementarli appieno. Nel *Capitolo 5* della valutazione è stato dimostrato come fosse possibile applicare il servizio ad un caso pratico con un accuratezza accettabile. Le misure di valutazione ottenute sono migliorabili con un eventuale uso continuo della classificazione e del retraining.

Una delle limitazioni di questo sistema consiste nel numero di classi per ogni classificatore: limitare il numero di esse a due ha senza dubbio permesso di semplificare lo sviluppo e la valutazione della metodologia, ma ciò non permette di distinguere i diversi tipi di difetti possibili. Inoltre il processo di data augmentation è risultato essere importante nei casi studi affrontati, però le tecniche relative non sono state integrate nell'applicazione web ma esternamente.

Le limitazioni esposte possono essere dei punti di partenza a partire dal quale estendere le funzionalità del sistema. La progettazione è avvenuta con lo scopo di rendere il sistema indipendente dal fornitore di servizi di deep learning: un'altra possibile estensione è data dall'inclusione di ulteriori fornitori di servizi di classificazione, permettendo all'utente finale di scegliere l'offerta più conveniente. Restano migliorabili diversi aspetti in comune con le applicazioni web in generale, come

la gestione delle autorizzazioni degli utenti oppure l'uso di filtri e di ordinamenti personalizzati per le liste di immagini.

Bibliografia

[1] Frederick Winslow Taylor. *The principles of scientific management*. Harper e Brothers, 1911.

[2] Douglas C. Montgomery. *Introduction to statistical quality control*. 6. ed. Wiley, 2009.

[3] Fiorenzo Franceschini et al. "Obtained Value Through Quality Inspection". eng. In: *In: FAIMA BUSINESS AND MANAGEMENT JOURNAL* (2017). ISSN: 2344-4088.

[4] Borkowski Stanisław e Knop Krzysztof. "Challenges Faced in Modern Quality Inspection". eng. In: *Management and Production Engineering Review* 7.3 (2016), pp. 11–22. ISSN: 20808208.

[5] Xiaomian Li e Yufeng Shu. "Research on glass surface quality inspection based on machine vision". eng. In: *Australian Journal of Mechanical Engineering* 16.1 (2018), pp. 98–104. ISSN: 1448-4846. URL: `http://www.tandfonline.com/doi/abs/10.1080/1448837X.2018.1545479`.

[6] Simon-Frédéric Désage et al. "Macroscopic exploration and visual quality inspection of thin film deposit". eng. In: vol. 9050. SPIE, 2014, 90502G–90502G–9. ISBN: 9780819499738.

[7] Xinman Zhang et al. "A High Precision Quality Inspection System for Steel Bars Based on Machine Vision". eng. In: *Sensors* 18.8 (2018), p. 2732. ISSN: 1424-8220.

[8] Vachara Peansupap Chollada Laofor. "Defect detection and quantification system to support subjective visual quality inspection via a digital image processing: A tiling work case study". In: (2012).

[9] Peterson Belan, Sidnei Araújo e Wonder Alves. "An Intelligent Vision-Based System Applied to Visual Quality Inspection of Beans". In: 9730 (lug. 2016), pp. 801–809. DOI: `10.1007/978-3-319-41501-7_89`.

[10] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[11] Redazione di Elettronica Open Source. *I sensori di immagine*. 2018. URL: `https://it.emcelettronica.com/i-sensori-di-immagine`.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] Kumar Tan Steinbach. *Introduction to data mining*. Pearson, 2006.

[14] RE Bellman. "Dynamic programming". In: (1957).

[15] Naveen Venkat. "The Curse of Dimensionality: Inside Out." In: (set. 2018). URL: https://www.researchgate.net/publication/327498046_The_Curse_of_Dimensionality_Inside_Out.

[16] P Vimaladevi Manjula Kamalahasan K Vijayarekha. "REVIEW ON CLASSIFICATION ALGORITHMS IN IMAGE PROCESSING". In: (nov. 2017). URL: https://www.researchgate.net/publication/331319583_REVIEW_ON_CLASSIFICATION_ALGORITHMS_IN_IMAGE_PROCESSING.

[17] Kamber Han. *Data mining; Concepts and Techniques*. Morgan Kaufmann, 2006.

[18] Khalid Haron Azme Khamis Zuhaimy Ismail e Ahmad Tarmizi Mohammed. "The Effects of Outliers Data on Neural Network Performance." In: (2005). URL: https://scialert.net/abstract/?doi=jas.2005.1394.1398.

[19] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] Jawad Nagi et al. "Max-pooling convolutional neural networks for vision-based hand gesture recognition". In: *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)* (2011).

[21] Abien Fred M. Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: (2018).

[22] Saad Albawi, Tareq Abed Mohammed e Saad ALZAWI. "Understanding of a Convolutional Neural Network". In: (ago. 2017). DOI: 10.1109/ICEngTechnol.2017.8308186.

[23] Christian Szegedy Sergey Ioffe. "Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift". In: (2015). URL: https://arxiv.org/pdf/1502.03167.pdf.

[24] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[25] Y. Bengio Y. LeCun L. Bottou e P. Haffner. "Gradient-based learning applied to document recognition." In: (1998).

[26] Geoffrey E. Hinton Alex Krizhevsky Ilya Sutskever. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012).

[27] Christopher Yakopcic et al. Md Zahangir Alom Tarek M. Taha. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches". In: (mar. 2018). URL: https://arxiv.org/abs/1803.01164.

[28] X. Zhang et al. K. He. "Deep residual learning for image recognition". In: (dic. 2015). URL: https://arxiv.org/abs/1512.03385v1.

[29] Jason Brownlee. *How to Configure Image Data Augmentation in Keras*. 2019. URL: https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/.

[30] Naveen Panwar Utkarsh Desai. *Getting Started with Deep Learning as a Service (DLaaS) on the IBM Cloud*. 2019. URL: https://medium.com/@utk.is.here/getting-started-with-deep-learning-as-service-dlaas-on-the-ibm-cloud-c1c081bc4008.

[31] IBM. *IBM Watson Studio for Deep Learning*. URL: https://www.ibm.com/cloud/deep-learning.

[32] Kevin Gong. *Best practices for using custom classifiers in Watson Visual Recognition*. 2017. URL: https://developer.ibm.com/articles/cc-build-with-watson-tips-best-practices-custom-classifiers-visual-recognition/.

[33] Microsoft. *Servizio visione artificiale personalizzato*. URL: https://azure.microsoft.com/it-it/pricing/details/cognitive-services/custom-vision-service/.

[34] Google. *Google Cloud AutoML Vision*. URL: https://cloud.google.com/vision/automl/pricing?hl=it.

[35] OMG. *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011. URL: http://www.omg.org/spec/BPMN/2.0.

[36] baeldung. *A Guide To Cron Expressions*. URL: https://www.baeldung.com/cron-expressions.

[37] J.Bradley et al. M. Jones Microsoft. *RFC7519 JSON Web Token (JWT)*. URL: https://tools.ietf.org/html/rfc7519.

[38] *MER-041-436U3C-L, IMX287, 720X540, 436FPS, 1/2.9" GLOBAL SHUTTER CMOS, COLOR*. URL: https://www.get-cameras.com/Vision-Camera-Sony-IMX287-MER-041-436U3C-L.

[39] *LCM-5MP-06MM-F2.0-1.8-ND1, LENS C-MOUNT 5MP 6MM F2.0 1/1.8" NON DISTORTION*. URL: https://www.get-cameras.com/LENS-C-mount-5MP-6MM-F2.0-1/1.8-LCM-5MP-06MM-F2.0-1.8-ND1.

[40] Daheng Imaging. *PROGRAMMING DRIVERS AND MANUALS DAHENG IMAGING*. URL: https://www.get-cameras.com/Downloads.

[41] Andreas Holzinger Marcus D Bloice Peter M Roth. "Biomedical image augmentation using Augmentor". In: (). URL: https://doi.org/10.1093/bioinformatics/btz259.

[42] Mark Goadrich Jesse Davis. "The Relationship Between Precision-Recall and ROC Curves". In: (gen. 2006). URL: https://minds.wisconsin.edu/bitstream/handle/1793/60482/TR1551.pdf.

[43] University of Heidelberg. "29th Annual Symposium of the German Association for Pattern Recognition". In: (2007). URL: http://resources.mpi-inf.mpg.de/conferences/dagm/2007/prizes.html#industry.