ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di informatica-Scienza e Ingegneria

Corso di laurea in Ingegneria Informatica Magistrale

TESI DI LAUREA MAGISTRALE

Analisi, Sviluppo e Sperimentazione di Sistemi di Visual Inspection in Contesti Industriali

CANDIDATO:

Danilo Belvedere

RELATORE:

Prof. Ing. Paolo Bellavista

CORRELATORE:

MSc. Dott. Pietro Leo

IBM Italia S.p.A

Anno Accademico 2020/2021

PREFAZIONE

Questa tesi è stata redatta durante il tirocinio svolto presso **IBM Italia** S.p.A (Bologna) e in collaborazione con l'azienda **Automobili Lamborghini S.p.A** che mi ha consentito di sperimentare i risultati ottenuti in un caso di studio industriale reale in ambiente Automotive.
Durante il tirocinio sono entrato a far parte del gruppo **Active Intelligence** di IBM sotto la guida di Pietro Leo, correlatore di questa tesi, con lo scopo di approfondire diverse tematiche legate al tema della Visual Inspection, che racchiude un insieme di metodi di computer vision impiegati per il controllo qualità di prodotti e/o step di processi in ambiente industriale attraverso verifiche visive.
L'obiettivo specifico di questa tesi è stato quello di formare una figura esperta che potesse sviluppare un sistema di riconoscimento (Object detection e Object recognition) al fine di individuare potenziali difetti o anomalie presenti all'interno di fotografie e video.
Sebbene il controllo qualità rallenti nel complesso l'immissione di nuovi pezzi da parte delle aziende sul mercato, esso è necessario non solo per aumentare la soddisfazione nel tempo dei clienti ma anche per evitare gli enormi costi potenzialmente causati dai richiami di prodotti difettosi. Solitamente le ispezioni visive sono condotte manualmente e sono soggette ad un tasso di errore molto elevato causato dalla ripetitività con la quale i controlli sono svolti dagli operatori, sono pochi relativamente i settori di industria che impiegano queste in modo sistemico.
Tipicamente l'uso di tecniche di ispezione visiva vengono applicate in settori dove la numerosità dei prodotti è molto alta o quando si è in presenza di prodotti che richiedono controlli di sicurezza molto stringenti, si pensi al controllo di prodotti in ambito farmaceutico ad esempio.
Grazie alle più recenti tecniche di Machine Learning nell'ambito dei sistemi di Computer Vision oggi è possibile sviluppare sistemi in grado di apprendere costantemente il concetto di "difetto" apportando sostanziali miglioramenti in una miriade di ambienti e al contempo ampliando la platea degli utilizzatori industriali di queste tecniche.
Avere la possibilità di delegare ad una macchina il cruciale compito delle ispezioni visive non solo riduce in maniera netta i costi del settore di controllo qualità ma garantisce all'azienda anche il riconoscimento di qualità ed eccellenza da parte dei clienti.
In questa tesi verranno analizzati i principali software di Visual Inspection presenti nell'industria: si valuteranno le performance di piccole aziende specializzate in sistemi di visual inspection e sistemi

proposti da grandi aziende come Microsoft, Amazon, Google ed IBM per poi spostarsi su alcuni casi di studio reale sviluppati sia in ambiente Open Source che con gli strumenti messi a disposizione da IBM Italia durante questo tirocinio.
Nell'ultimo capitolo della tesi verrà trattato un caso di studio reale affrontato in collaborazione con l'azienda **Lamborghini S.P.A** con cui ho avuto modo di sperimentare le tecniche apprese durante questa esperienza in un caso di studio industriale.

Sommario

CAPITOLO 1 – INDUSTRIA 4.0 E VISUAL INSPECTION

1.1 Che cosa si intende per industria 4.0?

Il termine industria 4.0 è stato utilizzato per la prima volta nel 2011 alla Fiera di Hannover (Germania) con l'intento di descrivere una delle trasformazioni di maggiore impatto dell'era moderna.
Il nome cerca di suggerirci qualcosa di paragonabile ad una quarta rivoluzione industriale, ma come ben sappiamo cambiamenti di questa entità sono percepibili durante il corso di diversi anni e richiedono un'analisi critica e oggettiva sul fenomeno stesso.
Il concetto che sta alla base dell'industria 4.0 è l'**interconnessione** delle principali risorse dei **sistemi produttivi** come macchine, impianti, persone, materie prime e prodotti finiti.
Uno dei principali errori commessi quando si parla di industria 4.0 è quello limitarsi al considerare solamente il processo **produttivo** aziendale senza abbracciare le dimensioni di distribuzione, produzione, sviluppo di nuovi prodotti e soprattutto i processi che si andranno ad interfacciare con il consumatore finale.
Industria 4.0 quindi non significa solamente innovare nel settore produttivo, ma prendere parte ad un intero e nuovo ecosistema interconnesso in grado di abbracciare molteplici dimensioni differenti.
Questa nuova visione delle cose è nata grazie alla continua evoluzione e permeazione nel mercato dei principali attori che prendono parte a questa quarta rivoluzione industriale.
Basti pensare alle già affermate infrastrutture cloud che ci consentono di disporre di enormi risorse di calcolo senza investire sul lato hardware o ai dispositivi IOT (Internet of Things) che grazie ai molteplici sensori sono in grado di monitorare e trasferire incredibili quantità di dati analizzabili al fine di estrarre conoscenza per migliorare le prestazioni di un sistema industriale o di un processo produttivo. La riduzione dei costi delle nuove tecnologie ICT e l'affermarsi di queste realtà ha permesso di migliorare notevolmente la velocità e l'efficienza di una considerevole tipologia di sistemi produttivi industriali ma anche di alimentare lo **sviluppo di nuovi prodotti** a partire dall'analisi delle abitudini dei consumatori.
Alcuni dei nuovi meccanismi che iniziano ad essere utilizzati sempre di più dalle aziende che hanno

preso parte a questa rivoluzione sono ad esempio l'uso dei dispositivi di realtà aumentata, la profilazione di clienti (analisi dei big-data), l'utilizzo di sensori smart, l'inserimento di assistenti robotizzati nelle aziende o le stampe 3D.
Uno dei principali obiettivi dell'industria 4.0 è il tentativo di **digitalizzare** ed espandere i prodotti già esistenti, rendendoli oggetti in grado di interagire con il consumatore tramite rinnovate interfacce uomo-macchina.

1.2 Benefici applicativi in ambito industriale

L'industria 4.0 si distingue come mezzo per generare significativi guadagni in termini di produttività: viene migliorata la **qualità** dei prodotti, vengono snelliti i **processi produttivi**, ridotti i costi di **sviluppo** e dei tempi di **consegna**, accelerati i tempi di **ingresso** nel mercato e creati nuovi **servizi** con l'obiettivo di migliorare l'efficienza complessiva. La rivoluzione tecnologica portata dall'industria 4.0 è apprezzabile in diversi campi industriali, ognuno dei quali cerca di trarre i maggiori guadagni dai complessi apparati tecnologici in esame.
La lista *Fortune 500*, che include i primi 500 gruppi economici mondiali per fatturato, ha sottolineato che le aziende in grado di sostituire istruzioni cartacee all'interno di una catena assemblativa con tecniche di realtà aumentata che guidino le fasi di assemblamento di un prodotto, hanno migliorato la capacità di ottenere pezzi corretti al primo tentativo di oltre dieci punti percentuali[1]. Una delle aziende ad aver sfruttato questi meccanismi di realtà aumentata è stata BMW con l'impiego di un particolare modello di occhiali AR. A partire dal 2019 l'azienda tedesca ha infatti iniziato ad utilizzare questi dispositivi nelle sessioni di formazione e assemblaggio delle proprie unità di motori. Questi occhiali sono in grado di guidare gli addetti al montaggio attraverso tutte le fasi del processo, fornendo loro importanti informazioni difficilmente trattabili con il solo mezzo cartaceo.
Anche l'efficienza della formazione del personale BMW è migliorata notevolmente grazie all'impiego di questa tecnologia: i trainer che prima erano in grado di supervisionare solo un nuovo addetto alla


Figura 1 BMW assemblaggio AR

volta, ora sono in grado di seguirne fino a 3 contemporaneamente in vere e proprie sessioni di training virtuale. [2]

Un significativo aumento della capacità produttiva e di riduzione dei costi è visibile anche nel caso di studio di *Tool Gauge*, un'azienda statunitense che lavora nell'industria aeronautica e aerospaziale a Tacoma, Washington.
Per semplificare i processi di assemblaggio nei propri impianti, l'azienda ha iniziato ad installare sulle proprie catene produttive diversi robot collaborativi (cobots UR5) che sono stati impiegati in affiancamento agli operai. Questi robot sono facilmente trasportabili lungo tutto l'ambiente produttivo aziendale, sono definiti robot "universali" perché possono essere programmati per svolgere qualsiasi tipo di compito, anche senza esperienza di programmazione. I cobots UR5 sono in grado di supportare gli operatori nelle operazioni più complesse da eseguire, riducendo il rischio di infortunio a cui sono esposti giornalmente gli addetti.
Prima di questo cambiamento, l'azienda era costretta ad utilizzare per questi processi assemblativi, fino a 4 operai per produrre 200 unità di prodotto finito. Grazie all'aiuto dei cobots il numero di unità prodotte è salito a 400 mentre il numero di operai dispiegati è stato ridotto solamente ad uno. È migliorato notevolmente anche il tasso di parti scartate per errato assemblamento che è passato dal 15% al 3%. Il supporto dei cobots consente agli operatori di concentrarsi sulla fase di ispezione e di risparmiare più di 9000$ per ogni ordine completato.[3]

Figura 2 Cobots UR5

Gli esempi riportati precedentemente cercano di presentare i miglioramenti delle aziende su fattori specifici della linea produttiva come la riduzione dei **costi,** l'aumento della **velocità di produzione di pezzi** oppure **efficienza** generale del complesso aziendale.

1.3 Mercato Industria 4.0 e smart manufacturing

Attualmente il mercato globale dell'industria 4.0 è in costante crescita. *IoT Analytics*, leader nel settore degli approfondimenti e delle indagini di mercato in ambito Industria 4.0, ha pubblicato un report analizzando 350 offerte di prodotti e 38 casi di studio con l'obiettivo di definire diverse tendenze per le aziende che mirino ad approcciarsi a questa rivoluzione tecnologica. Si prevede che il mercato dei prodotti e servizi in ambito Industria 4.0 crescerà fino a 310 miliardi di dollari entro il 2023.[4]

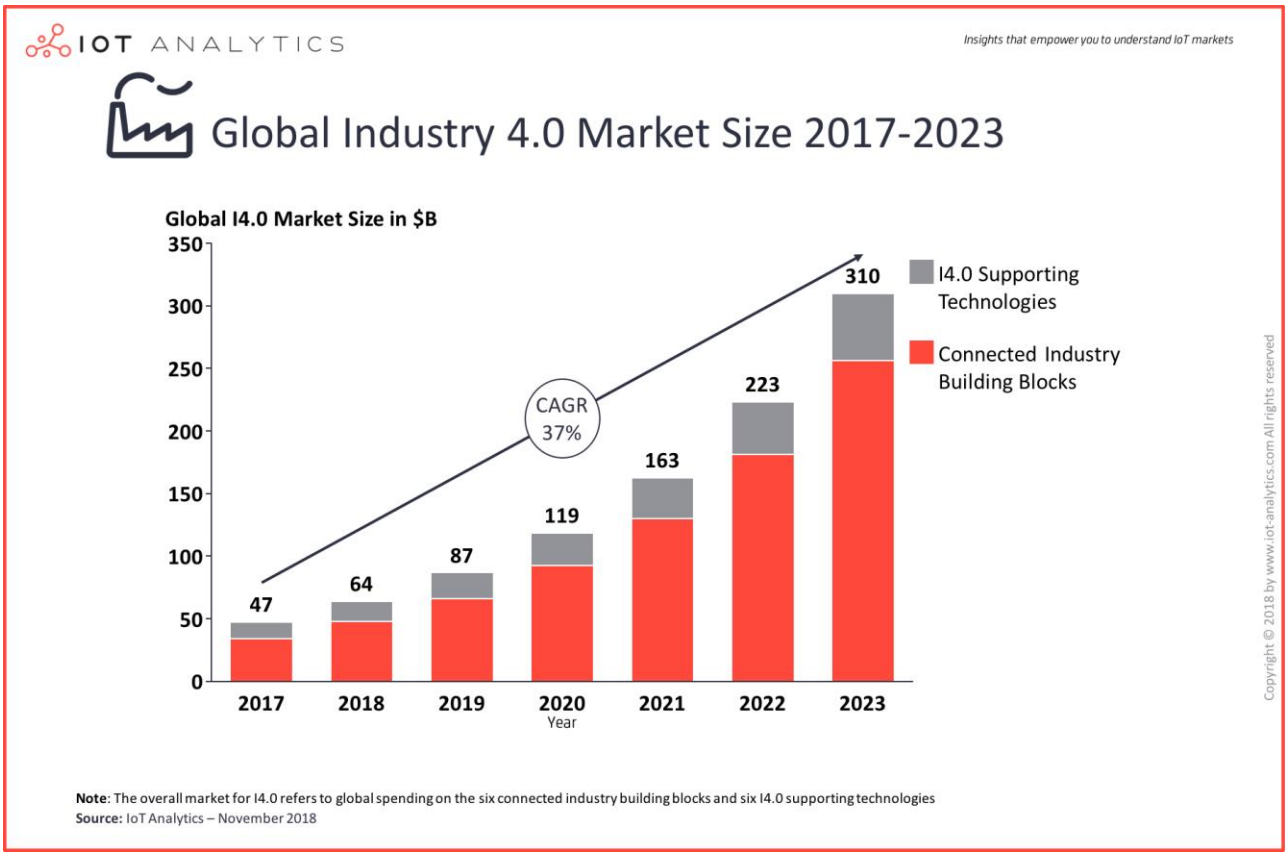


Figura 3 Crescita mercato industria 4.0

Applicando consapevolmente le tecniche più recenti è possibile migliorare alcuni aspetti che sì, fanno parte del comparto produttivo, ma che si concentrano sulla digitalizzazione della lunga e dispendiosa fase di **verifica** della **qualità dei prodotti**.

1.4 Controllo qualità dei prodotti

Negli scenari industriali odierni sono presenti forti componenti di **ispezione**, **verifica** e **analisi** della **qualità** di un prodotto.

Tali verifiche possono essere eseguite adottando diversi approcci, alcune vengono automatizzate

mediante l'ausilio di robot e macchine, mentre altre richiedono l'intervento di un operatore umano. Alcuni esempi di controlli eseguiti dalle aziende per verificare e analizzare la qualità di un prodotto sono:

- Controlli di tipo meccanico (l'azienda *Valbruna* dispone ad esempio di moderni e attrezzati laboratori in grado di eseguire prove meccaniche su campioni d'acciaio sia a freddo che a caldo per eseguire **certificazioni** richieste dai clienti[5]).
- Controlli chimici (l'azienda *Merieux NutriSciences* offre analisi chimiche di laboratorio e test microbiologici per rilasciare certificazioni su prodotti farmaceutici e dispositivi medici[6]).
- Ricerche Radiografiche/Magnetiche/Ecografiche/Laser (l'azienda *Tec-Eurolab* applica controlli per localizzare discontinuità superficiali e sub-superficiali in materiali che vengono prima magnetizzati e poi sottoposti ad un campo magnetico che, grazie ad una lampada speciale, è in grado di localizzare la presenza di eventuali rotture nel pezzo analizzato[7]).
- Visive (fanno uso della computer vision o della vista umana per ispezionare visivamente un prodotto)

Tali verifiche rallentano nel complesso l'immissione di nuovi pezzi da parte delle aziende sul mercato ma sono evidentemente necessarie per evitare eventuali richiami di prodotti difettosi.
Si basti pensare che i costi associati ai richiami sono più elevati per le aziende rispetto all'incrementare le capacità di verifica a monte, ovvero prima che i prodotti arrivino nelle mani del consumatore finale.
Un esempio molto famoso di problematica del genere è avvenuto nel 2016 con il modello di smartphone Galaxy Note 7 prodotto dall'azienda *Samsung*. Questi dispositivi erano soggetti a gravissimi problemi di surriscaldamento a causa di un problema di manifattura delle batterie.
Le batterie originarie, prodotte dall'azienda affiliata *SDI,* avevano problemi di tolleranza troppo bassa del vano in cui erano collocate all'interno dello smartphone, mentre le batterie sostitutive (realizzate dopo i primi richiami da *Amperex Technology*) avevano veri e propri difetti produttivi. A causa dell'elevato numero di richieste di sostituzione, l'azienda decise di sacrificare il controllo qualità a discapito di una più rapida produzione generando però solamente più costi di quelli stimati precedentemente.
Ad ottobre dello stesso anno l'azienda fu costretta a ritirare dal mercato tutti i dispositivi di quel tipo e a sospenderne la produzione. Secondo alcune stime, la spesa per il gruppo ammontò a circa

5.3 miliardi di dollari.[8]

1.5 Visual inspection e penetrazione nel mercato

La visual inspection, chiamata anche **Visual Testing (VT)**, è il metodo tradizionalmente più semplice con il quale è possibile verificare la presenza di eventuali difetti qualitativi sul prodotto ispezionato considerando la sua apparenza e/o morfologia da un punto di vista esterno.
Questo tipo di controllo è spesso di tipo visivo, può essere eseguito a monte o a valle della fase produttiva ed è solitamente assegnato ad operatori esperti dotati di grande esperienza nel campo in esame.
L'azienda *Condomett* che lavora nel settore dei controlli non distruttivi possiede ad esempio diversi periti che effettuano esami visivi alla ricerca di caratteristiche difettose superficiali o dimensionali. Tali controlli vengono effettuati in maniera diretta dall'operatore oppure in modalità remota, utilizzando l'ausilio di un drone professionale che trasmette all'addetto le immagini da analizzare.[9]
In questo caso di studio si può notare come l'elemento tecnologico (drone) non sia sostitutivo dell'operato umano e che alla fine è sempre e solo grazie **all'esperienza** visiva **dell'addetto** che si è in grado di riconoscere la presenza di un difetto oppure no.
Sebbene le ispezioni visive non possano produrre la stessa qualità di risultato ottenibile tramite i controlli non distruttivi più complessi, esse sono parecchio utili all'interno di campi dove si cerca di dare priorità al contenimento dei costi e all'ottenimento di una stima sullo stato dell'oggetto in esame. La tecnica della visual inspection può essere applicata in maniera efficace anche senza l'intervento umano in diversi ambiti tra cui il retail, manufacturing, airport screening, food industry o medicine. Per dimostrare quanto la visual inspection abbia permeato i più disparati scenari di mercato, nel seguito, verranno presentati alcuni possibili scenari applicativi e casi d'uso.

Automotive: visual inspection for Automotive solutions

Il mondo automotive si presta particolarmente all'applicazione del Visual Testing in quanto ogni step di processo è composto da una lunga lista di controlli con l'obiettivo di verificare la qualità di ogni fase produttiva.

Figura 4 Visual inspection bulloni ruote

- Carrozzeria del veicolo

Visual inspection applicabile per analizzare i **componenti** in arrivo che saranno **assemblati**, per una verifica sulla qualità delle **saldature** o per l'efficacia di prodotti **sigillanti**.

-Verniciatura del veicolo

Visual inspection applicabile per individuare eventuali **difetti** di verniciatura come aree non coperte sufficientemente di prodotto o zone sprovviste di vernice.

-Assemblaggio

Visual inspection applicabile per verificare la **corretta installazione** dei pezzi (guarnizioni, finiture...) o per valutare la bontà delle **connessioni elettriche** nel veicolo.

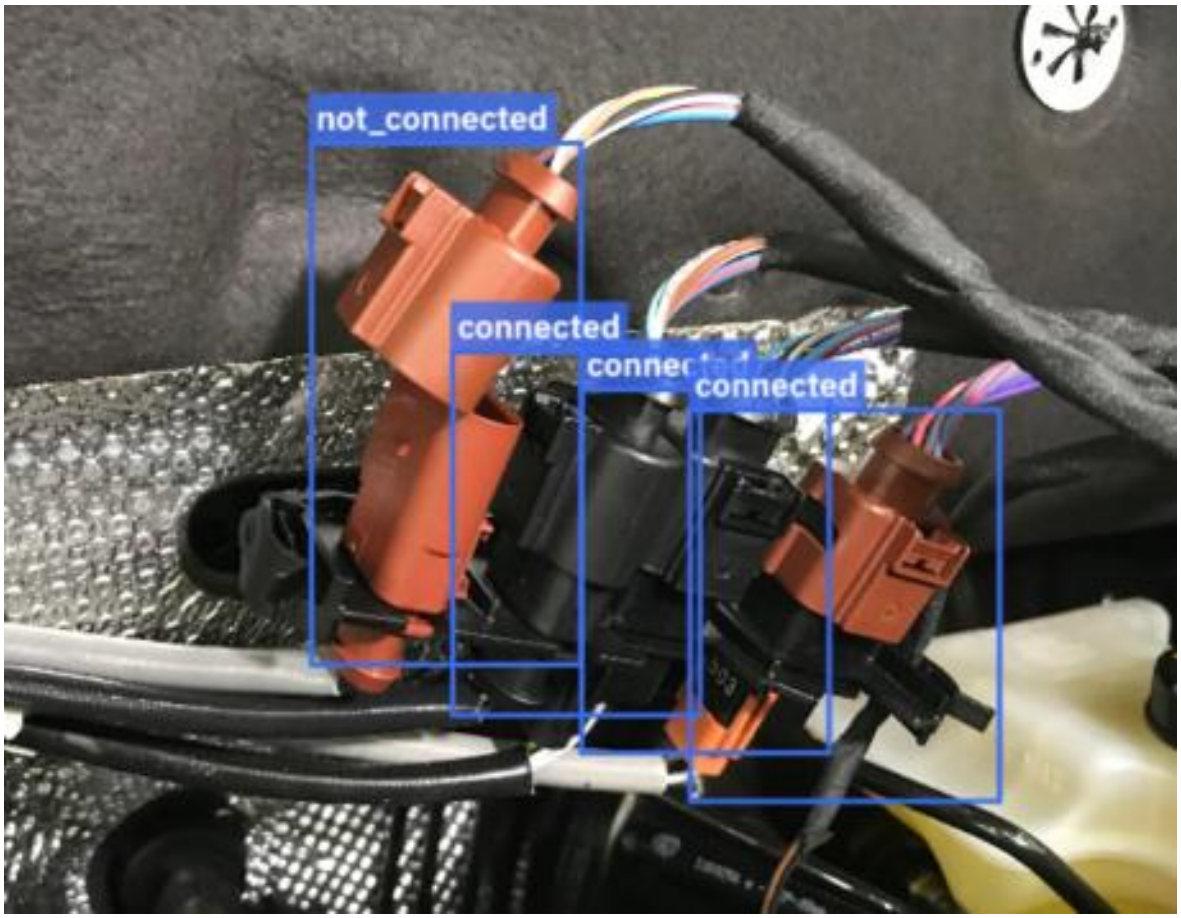


Figura 5 Visual inspection connettori

Automotive use case:

L'azienda *Shelton Vision* possiede un sistema in grado di effettuare Visual Testing sulle diverse parti di un veicolo a partire dalla cablatura elettrica fino alla componentistica assemblata.
Shelton Vision utilizza diverse telecamere in grado di effettuare:

- Computazioni real time delle immagini ricevute e scarto dei pezzi difettosi.
- Analisi e tracciamento dei risultati nel tempo.

I dati sono elaborati dal sistema WebSPECTOR che consente di classificare i difetti e, con l'ausilio di una persona finale guidata dai risultati ottenuti del sistema, è in grado di garantire la qualità durante tutto il ciclo di vita dei veicoli, dalla produzione delle parti al prodotto finito.
Quando viene rilevato un difetto dal sistema, esso viene automaticamente classificato per tipo ed etichettato con un corrispondente grado di gravità. Il sistema può essere programmato per scartare solamente i pezzi più danneggiati oppure, se si vuole una precisione maggiore, adottare politiche di tolleranza più basse.
Nello use case presentato dall'azienda stessa, viene analizzata la problematica relativa alle ispezioni visive del **tessuto** che riveste il tetto, la seduta o altri componenti di un veicolo.
Questa operazione veniva fatta tradizionalmente da persone che ispezionavano manualmente il tessuto che scorreva su un tavolo illuminato. A causa della forte luminosità e del ritmo con la quale venivano prodotti i pezzi, la probabilità effettuare errori era veramente elevata. I clienti insoddisfatti e i conseguenti reclami si traducono spesso in costi aggiuntivi molto più elevati del valore del tessuto stesso per un'azienda.
Con Shelton Vision lo stesso lavoro viene eseguito da una macchina ad una velocità dieci volte superiore a quella umana, incrementando la produzione aziendale e abbattendo notevolmente i costi produttivi.[10]

Farmaceutico: visual inspection for pharmaceutical solutions

In campo farmaceutico sono molto comuni veri e propri **documenti** (check lists) progettati per aiutare gli operatori sanitari a svolgere ispezioni visive dei medicinali al fine di rilevare segni di contraffazione, imballaggi inadeguati o etichettature errate dei prodotti. Uno di questi documenti è stato redatto dall'International Council of Nurses in collaborazione con la Farmacopea degli stati uniti (USP).[11] Applicare consapevolmente la visual inspection potrebbe sensibilmente migliorare l'efficacia di questi controlli e favore l'introduzione sul mercato di medicinali che hanno subito elevati controlli di qualità.

- Packaging

Verifica dell'effettiva chiusura e dell'imballaggio del farmaco. Il controllo potrebbe essere fatto sia all'esterno (involucro contenitore per il farmaco) che internamente alla scatola che lo racchiude.

- Labeling

La verifica dell'etichettatura di un prodotto farmaceutico è di vitale importanza. La visual inspection potrebbe aiutare a convalidare la provenienza del medicinale e a verificare in tempo reale se il medicinale è stato registrato correttamente nello stato di destinazione. In questo modo si potrebbero evitare prodotti venduti illegalmente o marchi contraffatti. Un ulteriore controllo verrebbe fatto anche sul nome dei principi attivi oppure sul colore del logo presentato (alcuni loghi cambiano colore se visti da differenti angolazioni). Degni di interesse anche i controlli sull'indirizzo del produttore (sempre verificabile real time), la forza del medicinale (mg/unit), il numero di unità per scatola o le date di scadenza.

Figura 6 Visual inspection presenza di medicinali

- Caratteristiche fisiche del medicinale

Tutti i medicinali devono essere sottoposti a rigidi controlli al fine di individuare tempestivamente segni di sporco, umidità, erosione, crepe o qualsiasi altra forma di manomissione esterna.

La visual inspection potrebbe essere utilizzata per verificare la dimensione dei farmaci, l'uniformità del colore e la disposizione delle textures.

Potrebbero essere eseguiti controlli anche per rilevare eventuali crepe nei prodotti o segni di contaminazione. Particolare attenzione viene data al rilevamento dei difetti nei farmaci liofilizzati poiché di natura polverosa e ai medicinali particolarmente viscosi come l'insulina.

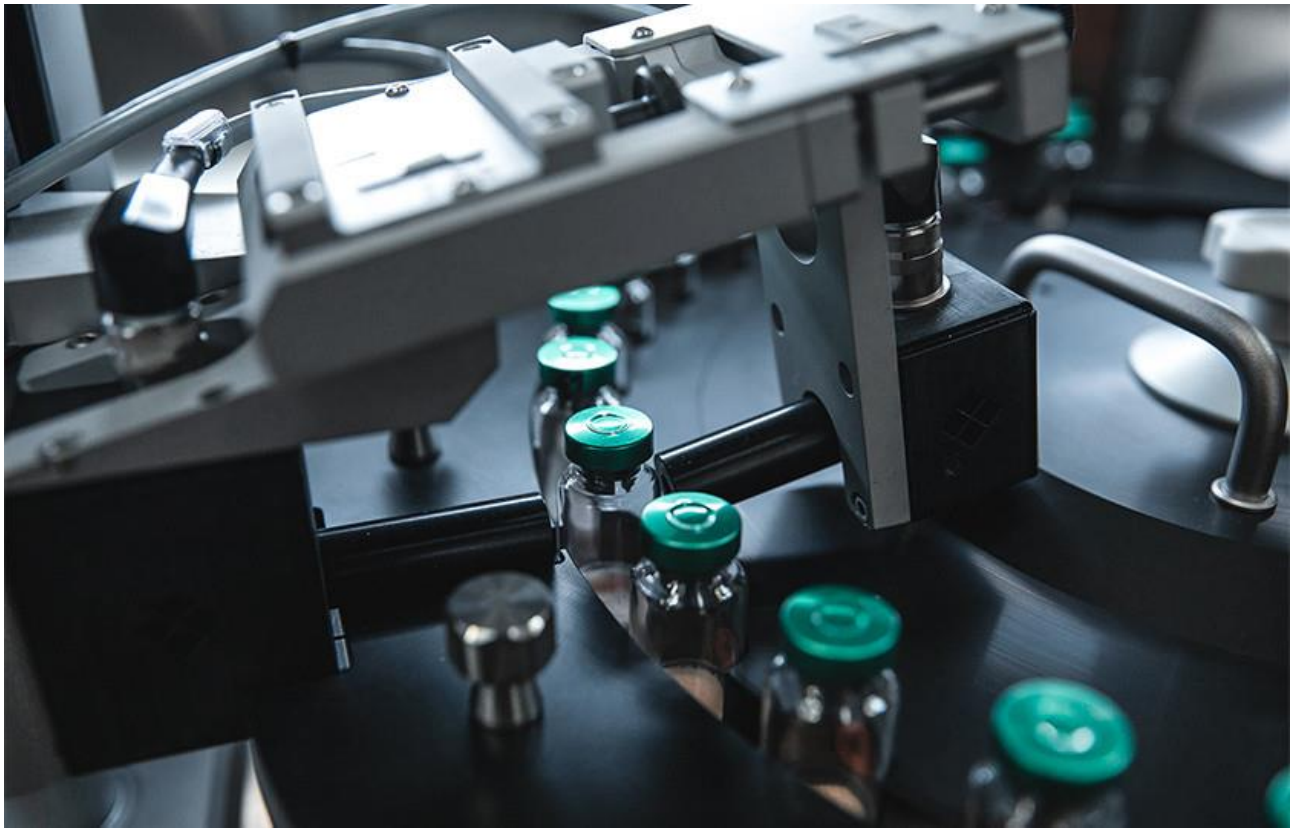


Figura 7 Telecamera di visual inspection per uso farmaceutico

Farmaceutico use case:

In Italia così come in altri stati, i sistemi di ispezione automatica per il rilevamento di difetti nei prodotti farmaceutici sono moderni e all'avanguardia. Il controllo su questa tipologia di prodotti è molto più rigido di altre categorie in quanto occorre rispondere a requisiti tecnici molto esigenti imposti da standard e comunità internazionali. Il gruppo *Stevanato Group* offre, ad esempio, una gamma completa di prodotti progettati come soluzione specifica per rispondere alle esigenze di clienti che desiderino effettuare ispezioni particellari, rilevamento di difetti cosmetici e test d'integrità di flaconi e contenitori. Le telecamere offerte ai clienti sono in grado di catturare sequenze molto rapide di immagini anche se i pezzi scorrono davanti ad esse a velocità elevatissima. Anche la trasparenza dell'eventuale liquido ispezionato non è un problema, questi macchinari riescono infatti a rintracciare anche particelle vetrose residue in fluidi trasparenti. Per farlo vengono utilizzati modelli probabilistici di movimento delle particelle estranee come quelle di vetro, in quanto esse si comportano statisticamente in un modo misurabile e di conseguenza individuabile.

Per effettuare ispezioni su medicinali particolarmente torbidi, si fa ruotare il contenitore del farmaco molto rapidamente in maniera da far ruotare allo stesso modo le particelle estranee nel liquido. Le telecamere sono in grado di rilevare queste impurità e segnalarlo tempestivamente al sistema che provvederà a scartarle. Ogni telecamera offerta dal gruppo possiede il proprio specchio che consente di effettuare più ispezioni contemporaneamente.

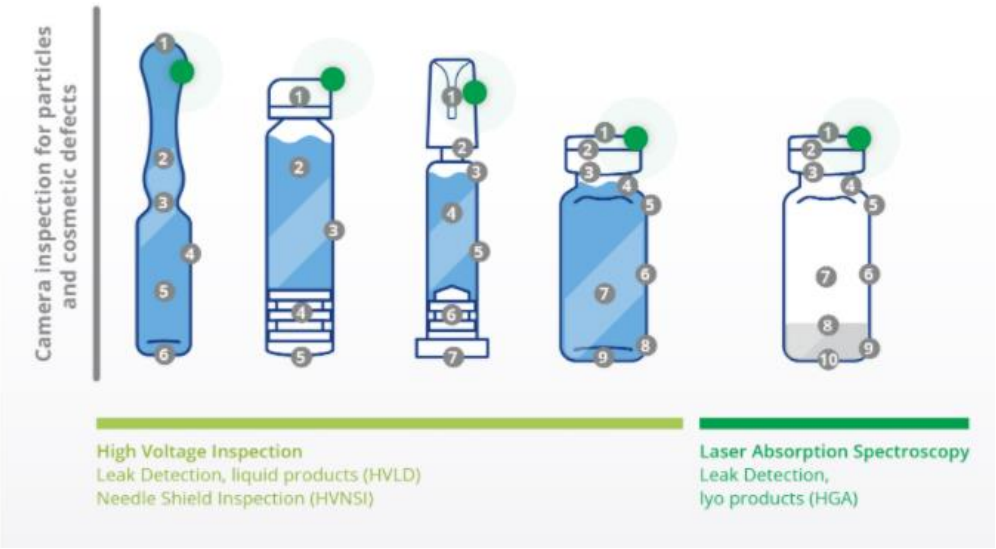


Figura 8 Visual inspection contenitori farmaceutici

Un altro esempio di test effettuato riguarda eventuali rotture o perdite all'interno dei contenitori che racchiudono i medicinali, come le siringhe: grazie alle immagini ad altissima definizione si è in

grado di rilevare difetti sul corpo della siringa e addirittura controllare il livello di liquido presente con precisione millimetrica.[12]

Elettronica e Meccanica: visual inspection for electronics solutions

In un settore dove componenti dalle dimensioni ridotte fanno da protagonisti, la visual inspection ha già assunto un ruolo fondamentale per il controllo qualità durante la fase produttiva e nelle fasi di imballaggio e spedizione. Da diversi anni le aziende utilizzano le più disparate tecniche per effettuare un controllo visivo al fine di rilevare difettosità all'interno di componenti elettronici. L'utilizzo sempre maggiore di componenti SMD (small Surface devices) richiede strumenti molto precisi in grado di valutare efficacemente la bontà di un componente o una scheda con precisione millimetrica. Questa operazione, svolta fino ad alcuni decenni fa direttamente a mano, viene portata a termine con l'ausilio di telecamere collegate solitamente ad un sistema di elaborazione.

- Saldature e pin

Vengono ispezionate accuratamente le **saldature** per evitare sovrapposizioni e uso di troppo materiale (bridging) e si effettuano controlli sullo stato dei **pin** dei componenti al fine di scartare i pezzi con piedini piegati in fase di produzione.

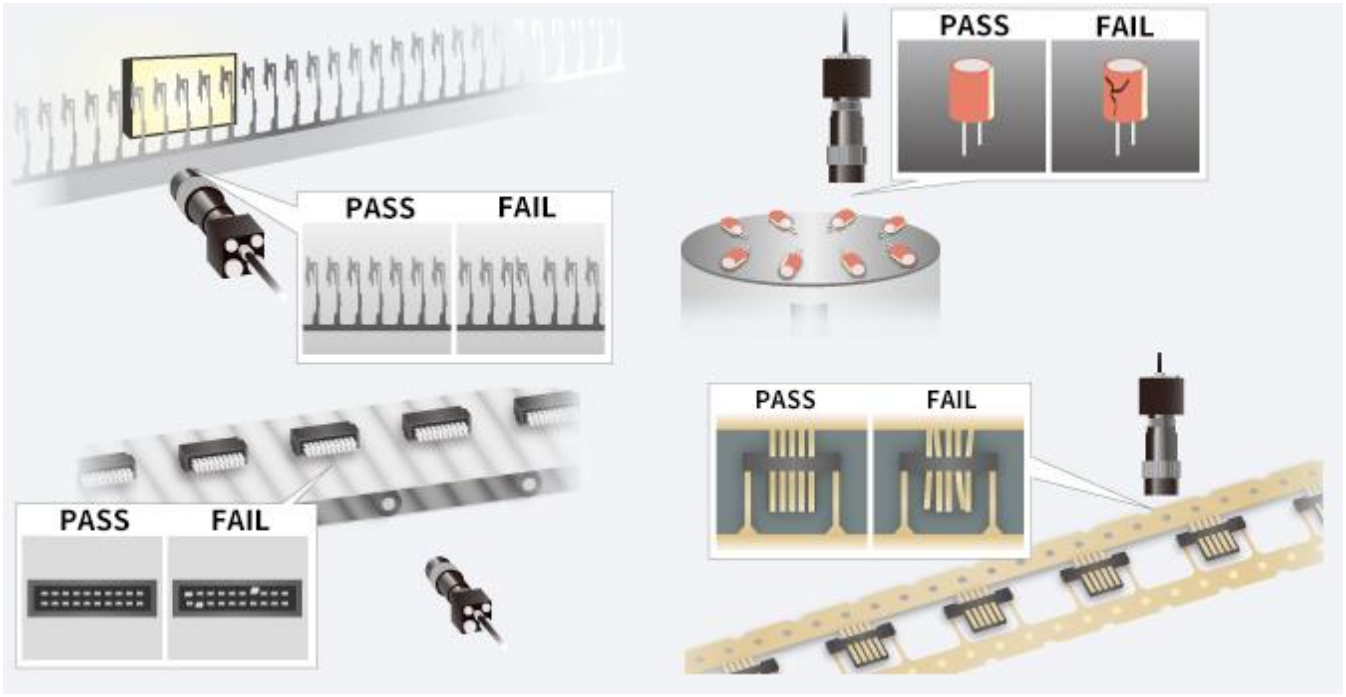


Figura 9 Alcuni controlli sui componenti

- Schede ed etichettatura

Si cerca di verificare la presenza o la mancanza di un dato **componente** all'interno di una scheda o di un circuito stampato. Oggi risulta molto semplice verificare la posizione di **interruttori** oppure il disallineamento di alcuni pezzi. Alcuni sistemi sono in grado di verificare anche l'integrità strutturale dei componenti esaminati; è infatti possibile rilevare se un condensatore si è gonfiato in maniera eccessiva durante il proprio utilizzo in modo da sostituirlo. Un'altra attività importante è quella del controllo per le etichette al fine di verificare certificazioni e provenienze del componente esaminato: grazie alla presenza di database che identificano il luogo d'origine di ogni pezzo è possibile tracciare e verificare l'autenticità di ogni elemento che compone una scheda.

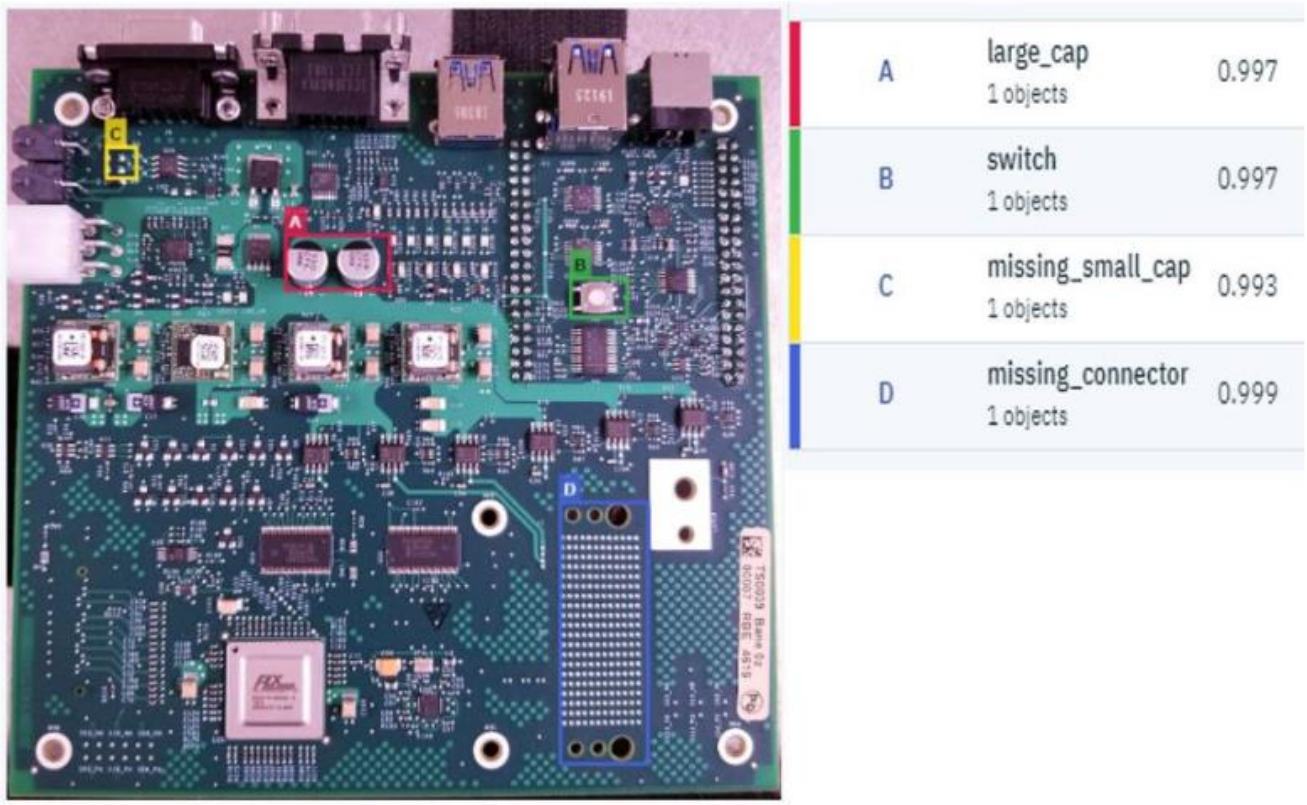


Figura 10 Visual inspection componenti su scheda

Elettronica e Meccanica use case:

L'azienda *Sipotek* è specializzata nell'effettuare visual inspection di prodotti e componenti elettronici e meccanici da 13 anni. Tutte le soluzioni implementate sono specifiche per trattare una particolare categoria di problemi, si passa dall'ispezione di induttori al controllo di wafer utilizzando macchine molto costose installabili in loco presso l'azienda del cliente.

I software impiegati effettuano calcoli e verifiche sulle dimensioni del pezzo esaminato al fine di rilevare imperfezioni sulla struttura e procedere all'eliminazione di componenti difettosi.[13]

Figura 11 Sipotek controllo dimensionamento

Food industry visual inspection for food industry solutions:

Prima di raggiungere le nostre tavole, il cibo subisce una serie di controlli necessari per salvaguardare la qualità e la sicurezza di noi consumatori finali. Ogni azienda che intenda immettere prodotti consumabili sul mercato è obbligata a contattare centri specializzati nel controllo degli alimenti che rilasciano certificazioni di qualità. La computer vision e la visual inspection digitalizzata possono aiutare anche in questo campo:

- Struttura degli alimenti

Il cibo viene controllato ricercando corpi estranei, ammaccature o irregolarità sul corpo esterno del cibo. In questo modo è possibile scartare una mela che si è danneggiata durante la fase di raccolta oppure cibo contaminato da parassiti o insetti.

- Confezionamento, Labeling e data di scadenza

Come già visto in altri settori, anche qui è possibile verificare visivamente tutta la fase di confezionamento, labeling e di scadenza del cibo. È molto importante garantire che sul mercato vengano immessi solamente cibi tracciabili e con date di scadenza corrette.

- Colore e analisi iperspettrale

L'occhio umano è in grado di distinguere il mondo come somma di tre colori RGB. A partire dall'analisi iperspettrale di un alimento vengono captate informazioni invisibili all'occhio umano come la capacità di riflettere una certa tipologia di luce o la lunghezza d'onda dell'elemento analizzato.

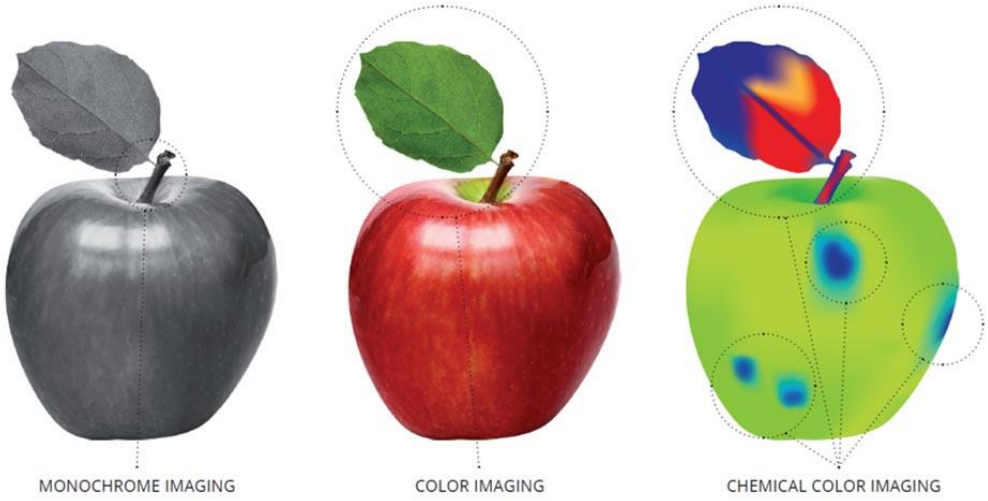


Figura 12 Analisi visiva di una mela

Food industry use case:

EyePro System è un'azienda leader globale nelle ispezioni visive che sfrutta la visual inspection per il controllo qualità di prodotti di consumo come il pane. Il loro compito è quello di aumentare la qualità complessiva del cibo che viene trasferito al consumatore, ottimizzare e snellire il processo di controllo qualità e ridurre gli sprechi mantenendo basso il numero di richiami.
Dopo aver acquisito immagini ad alta definizione del cibo, vengono controllate e confrontate online diverse informazioni come le misure di diametro e lunghezza.
Successivi step di controllo cercano di classificare e riconoscere elementi contaminanti a partire dall'analisi iperspettrale dei colori. In questo modo è possibile rilevare elementi che sarebbero difficili da percepire rapidamente sfruttando solamente la vista.

Grazie all'analisi iperspettrale anche una striscia di plastica inserita all'interno di una pizza surgelata può essere facilmente individuabile dal software di questa azienda. L'elemento più interessante è che tutte le analisi vengono effettuate in tempo reale ma soprattutto online, sfruttando gli enormi database di features che l'azienda mette a disposizione come confronto.[14]

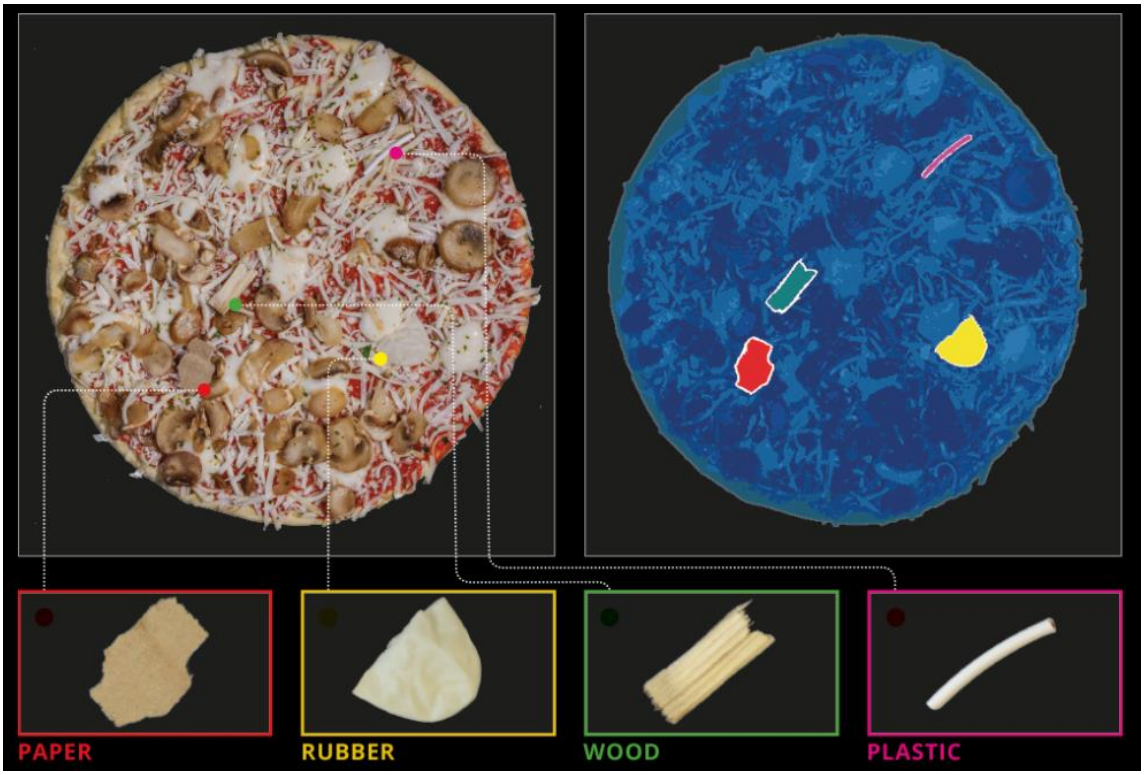


Figura 13 Analisi iperspettrale Eyeprosystem

Insurance visual inspection for insurance solutions:

Esistono potenziali sviluppi anche in ambito assicurativo; si pensi ad esempio alla possibilità di ottenere una rapida stima dei danni a seguito di un sinistro stradale solamente con l'utilizzo di una semplice fotografia. Le ultime innovazioni nel campo dell'intelligenza artificiale e del deep learning sono infatti in grado di supportare queste operazioni e di ottenere risultati percepibilmente migliori dell'impiego di un operatore in grado di affidarsi solamente alla vista.
Alcune aziende hanno iniziato a sviluppare supporti di ogni genere in grado di riconoscere, affidandosi al deep learning e all'intelligenza artificiale, situazioni di pericolo, difetti o problematiche di vario genere.

Insurance use case:

Con l'obiettivo di snellire l'aspetto burocratico e ottenere rapide stime a seguito di sinistri stradali, l'azienda *Nanonets* ha sviluppato un software in grado di effettuare riconoscimento e classificazione di immagini. Il confronto con l'aspetto convenzionale di ispezione assicurativa è interessante: solitamente la procedura classica richiederebbe lo spostamento on-site da parte dell'operatore, l'esecuzione di un video rappresentante i danni, l'upload sui server della compagnia assicurativa e l'analisi manuale del sinistro.

Figura 14 Nanonets assicurazioni

Questa fase richiede in media una settimana di tempo e solamente se i danni rientrano nei limiti espressi sui contratti assicurativi si procederà allo step successivo.
Con Nanonets il tempo di verifica viene abbattuto da una settimana ad un giorno. Il cliente non ha bisogno dell'ausilio dell'operatore per ottenere la stima dei danni, basta infatti utilizzare il proprio smartphone con l'app dedicata e lasciare che il software elabori la richiesta del cliente. Nanonets è in grado di rilevare con precisione ammaccature, graffi e ruggine, assegnando al contempo una valutazione su 3 livelli (low, medium e high) all'entità del danno subito. Il tutto viene trasformato in un file JSON che racchiude la lista dei danni, le loro dimensioni e lo score attribuito dal sistema.
L'impatto di Nanonets sui costi e sulla felicità del cliente è veramente elevato: con l'approccio classico il costo di ispezione può arrivare a costare fino a 7.26$ per auto, mentre con il software fornito dall'azienda abbatte il tutto a soli 0.59$ ad ispezione.[15]

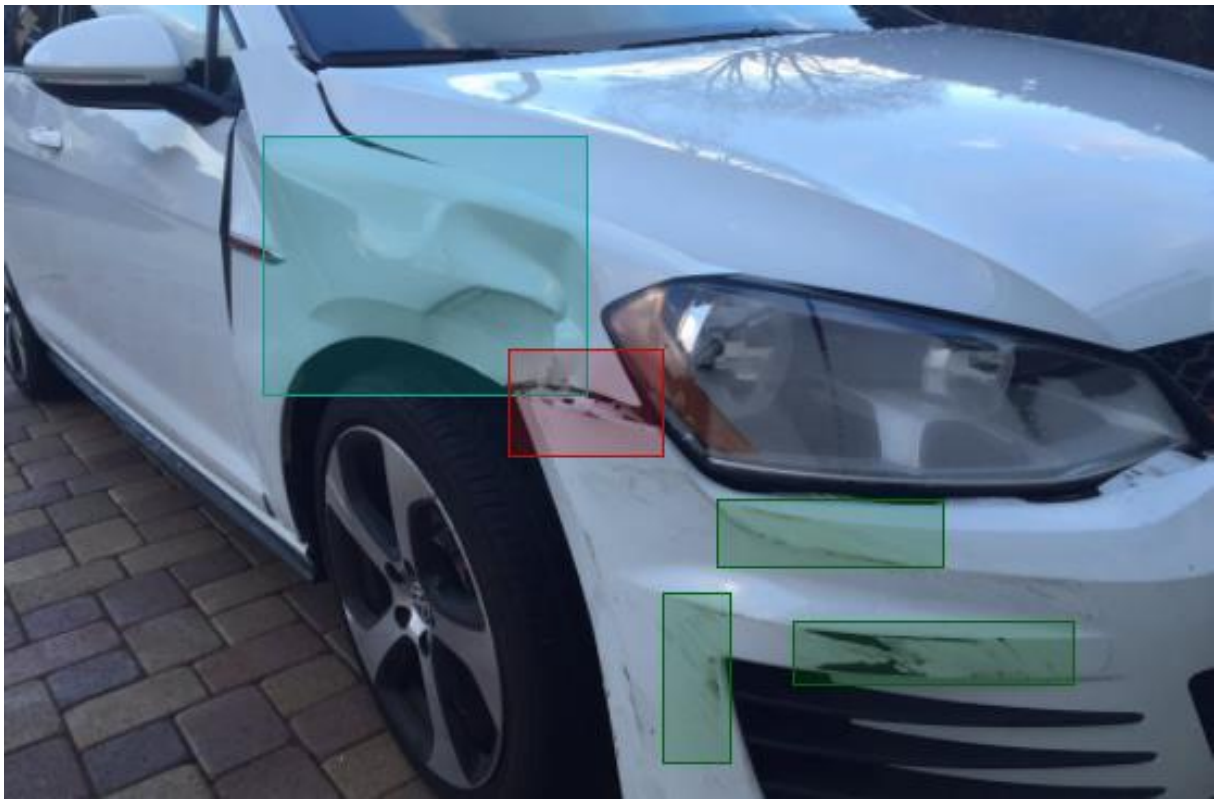


Figura 15 Immagine di input al sistema Nanonet

1.6 Deep Dive in un'unità robotica di ispezione visiva in ambito **farmaceutico**

Per un'azienda che lavora in ambienti critici come quello farmaceutico, il concetto di verifica è spesso rappresentato da un vero e proprio **processo**: l'analisi di un vaccino o di un medicinale richiede infatti diversi step intermedi sia di tipo meccanico (traslazioni e rotazioni) che chimico (modifiche di temperatura o pressione). Questa tipologia di sistemi è costruita su misura per il cliente e raggiunge un elevatissimo grado di **specializzazione**. *Stevanto Group* ha sviluppato telecamere custom in grado di effettuare fino a 10.000 esposizioni al secondo al fine di rilevare pattern relativi ai cambiamenti di posizione delle particelle spurie presenti nel medicinale analizzato.

Figura 16 Telecamera a 10.000 esposizioni di Stevanto Group

Grazie al loro algoritmo di tracciamento delle particelle, l'azienda è in grado di rilevare in modo efficiente la presenza di contaminanti in una vasta gamma di prodotti farmaceutici liquidi.

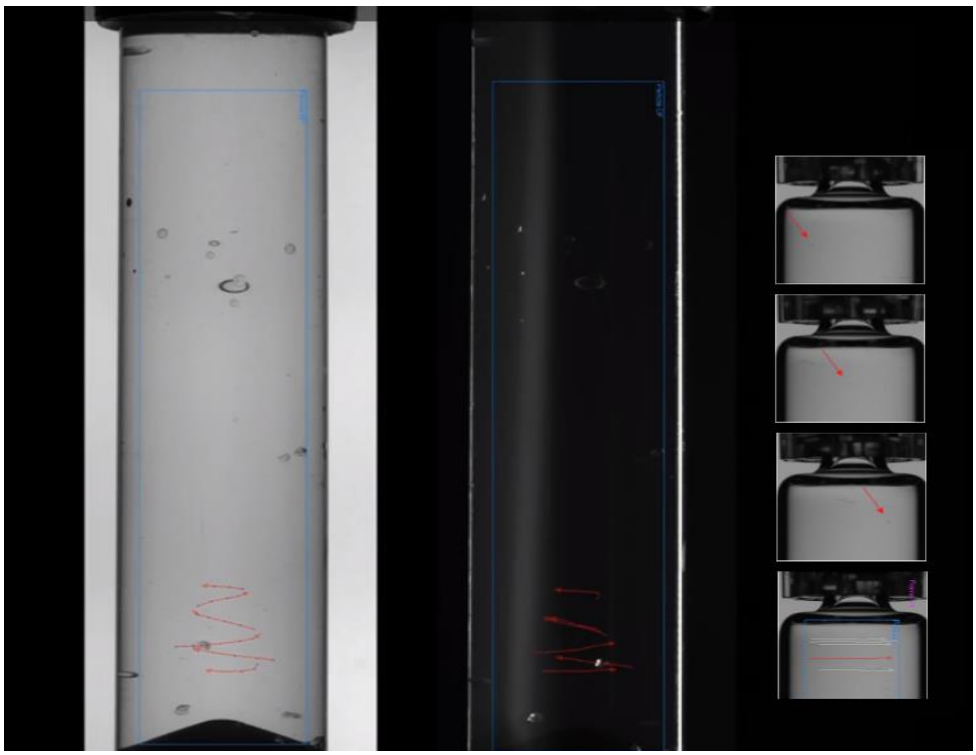


Figura 17 Analisi della posizione delle particelle

Sebbene l'industria farmaceutica sia restia sull'adozione dell'ispezione a raggi X, perché teme che le radiazioni possano compromettere la stabilità del prodotto, studi recenti dimostrano che la composizione chimica e l'efficacia dei prodotti farmaceutici non sono compromesse dall'esposizione ai raggi X utilizzati dai sistemi di ispezione.

Stevanto Group ha consolidato questi studi utilizzando i raggi X come strumento per ispezionare e campionare prodotti biologici e liofilizzati. Uno degli esempi è l'utilizzo del metodo spettroscopico Near infrared (NIR). Con questo metodo una sostanza viene illuminata con un ampio spettro a diversa frequenza di luce del vicino infrarosso, che può essere assorbita, trasmessa, riflessa o diffusa dal campione. Le telecamere identificano le particelle presenti sulla superficie o nel fondo di un farmaco liofilizzato e le radiazioni NIR possono essere utili per individuare i contaminanti presenti all'interno del medicinale, permettendo di distinguere il frammento di particella (vetro, carta, ecc.) dalla sostanza stessa. Quando si ispezionano i prodotti parenterali, può succedere che i prodotti buoni vengano respinti come difettosi perché le telecamere classificano erroneamente le **bolle** come particelle. Stevanto Group ha messo a punto con successo un metodo efficace per separare le bolle d'aria dalle particelle : facendo ruotare il contenitore ad alta velocità, le bolle vengono confinate al centro e la ricerca dei difetti si concentra in tutte le altre aree.

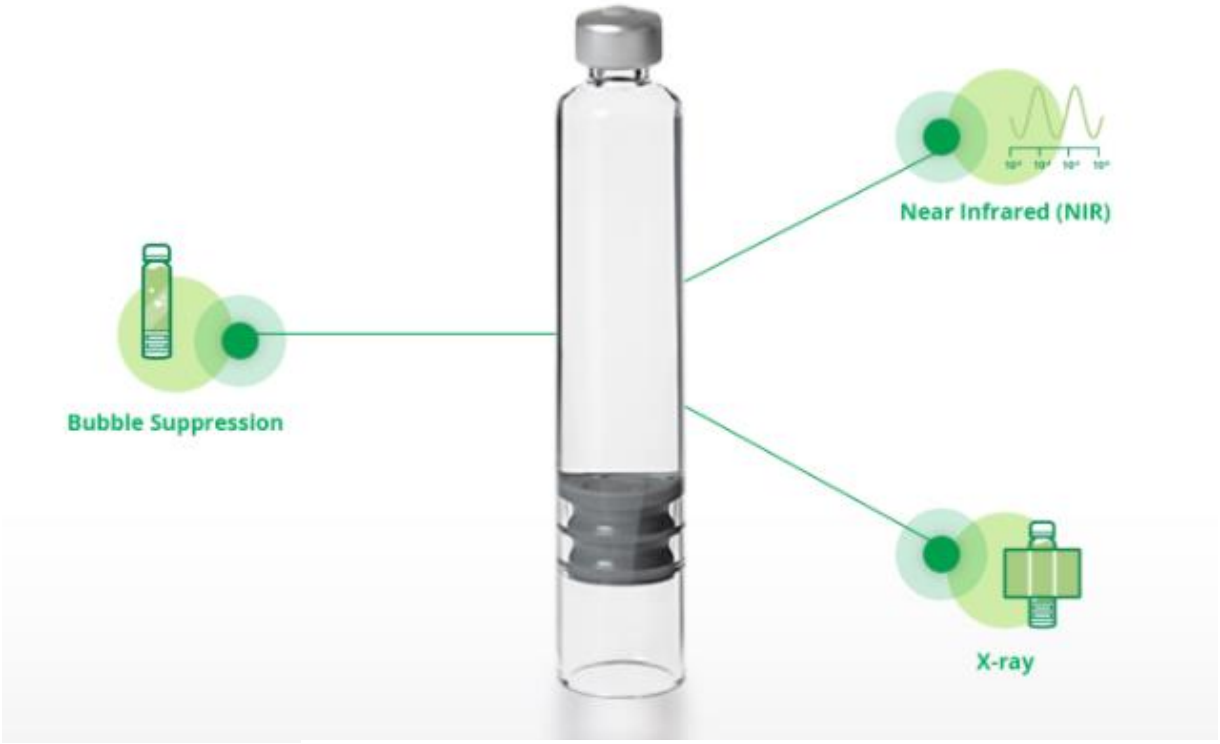


Figura 18 Stevanto group ispezione bolle d'aria

Un caso di studio affrontato dall'azienda ha riguardato il rilevamento di difetti all'interno dei farmaci liofilizzati. Questa particolare categoria di farmaci è spesso soggetta a falsi rilevamenti di contaminazione in quanto i difetti, a causa della natura della propria composizione, si presentano sottoforma di polvere presente sui lati e sul fondo del flacone (difficilmente distinguibile dal prodotto in quanto anch'esso della stessa natura).
Stevanto Group ha risolto la problematica utilizzando tecniche di **illuminazione** in grado di far catturare immagini che il sistema riuscisse a distinguere in maniera più precisa. Le immagini catturate discriminavano nettamente le particelle di vetro dalle crepe e i difetti del prodotto dai problemi cosmetici del contenitore.
Dopo l'adozione dell'apparecchiatura, i falsi scartati sono stati ridotti del 50% della produzione totale.

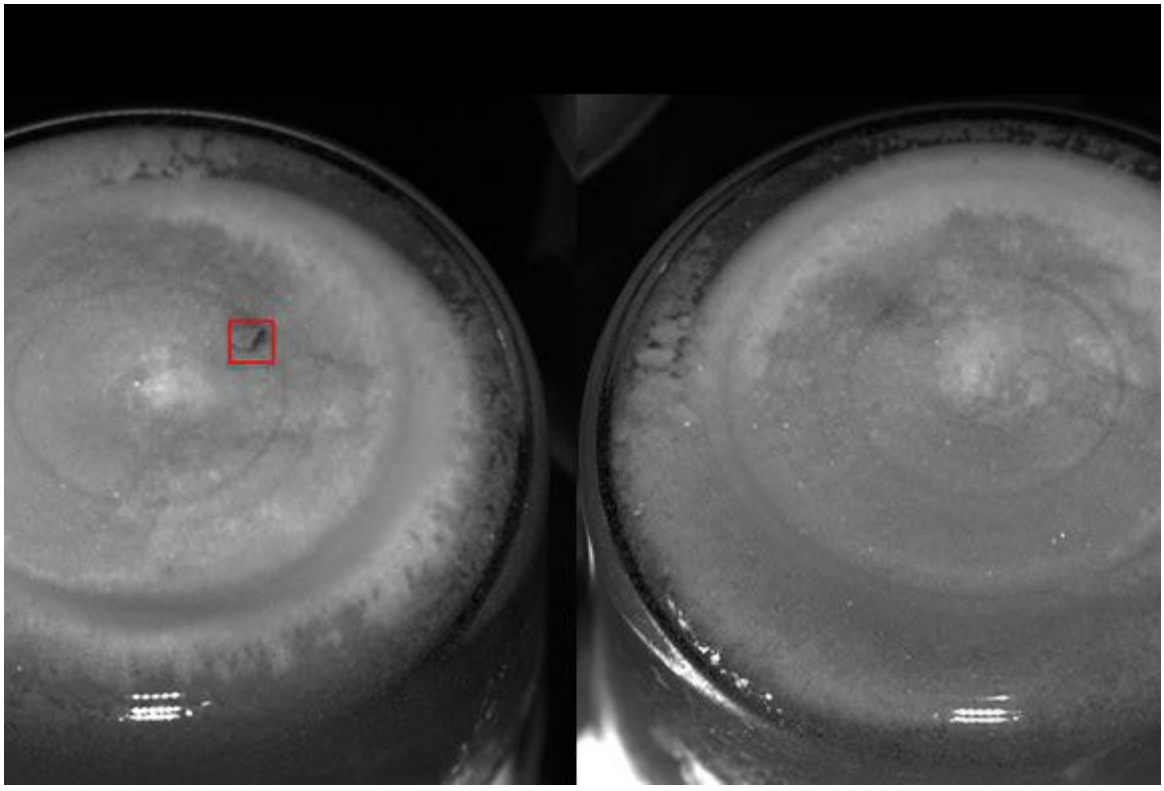


Figura 19 Difetto di un farmaco liofilizzato

Tutte queste verifiche sono incapsulate all'interno della **Vision Robot Unit** (VRU), un sistema modulare per l'ispezione di farmaci parenterali. Questo sistema sfrutta principi dell'intelligenza artificiale abbinati alla conoscenza profonda di esperti di settore ed è in grado di effettuare ispezioni completamente automatizzate senza l'intervento umano.
Tutti i componenti di questo sistema sono plug and play, è infatti possibile aggiungere unità alla macchina per aumentare il rendimento complessivo e adattarsi alle più disparate esigenze industriali. L'IA del prodotto è in grado di reagire al cambiamento delle caratteristiche del medicinale analizzato e di **apprendere autonomamente** la presenza di **nuovi difetti** riducendo i falsi scartati e riqualificando i falsi positivi.

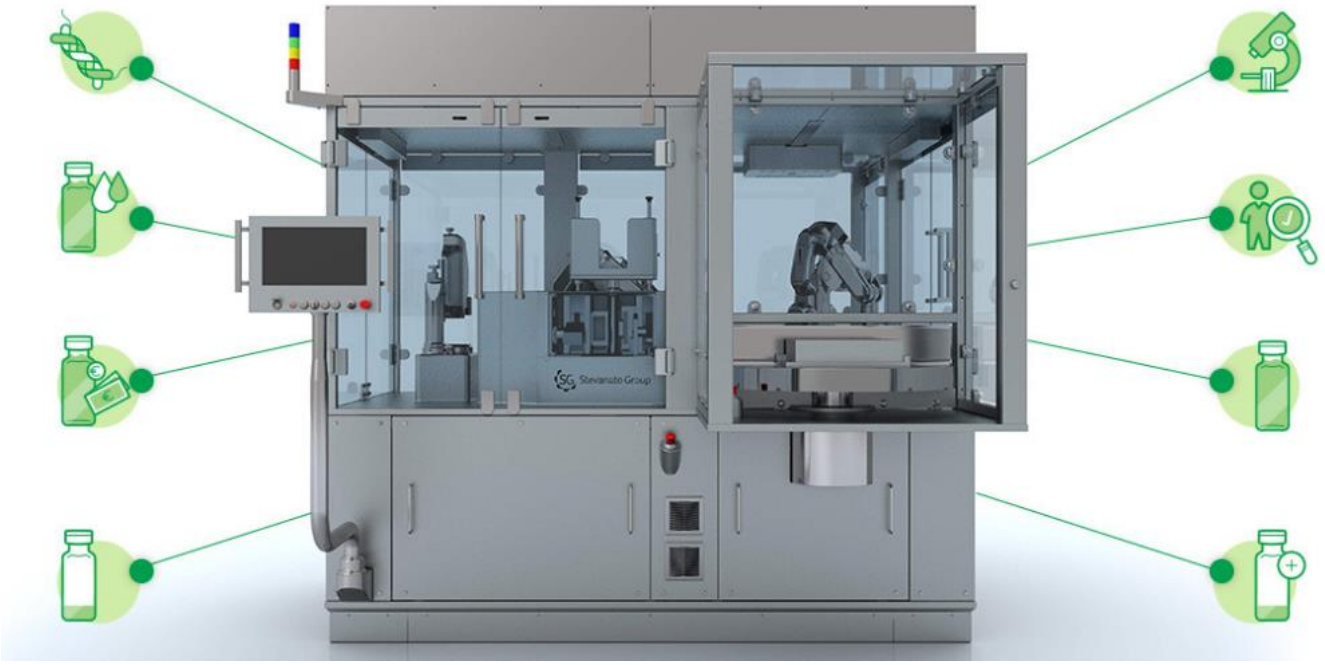


Figura 20 VRU Stevanto Group

La macchina tratta tranquillamente diverse tipologie di farmaci dai vaccini a prodotti dall'elevatissimo valore economico. Le tecniche di Stevanto Group sono state brevettate per ispezionare in modo affidabile anche le sospensioni più complesse o i prodotti più torbidi come l'insulina. Sia che si tratti di un'azienda intenzionata ad ispezionare farmaci liofilizzati che un colosso che richiede molteplici controlli su diverse tipologie di prodotto, la Visual Robot Unit è in grado di essere configurata per soddisfare tutte le esigenze del cliente. [16]

1.7 **Computer vision** e **visual inspection**

Nella sezione precedente sono stati presentati diversi problemi, casi d'uso e possibili soluzioni offerti da aziende che su di esse hanno costruito veri e propri business.
Il tentativo di innovare la visual inspection affidandosi alla **computer vision** è in realtà un problema molto antico che risale alla fine degli anni 90. L'obiettivo della computer vision in ambito visual inspection era quello di velocizzare i processi di verifica di difettosità e ridurre gli scarti causati da errori umani (falsi positivi). Sono stati fatti molti studi sulla **velocità** con la quale un essere umano era in grado di rilevare dei difetti affidandosi solamente alla propria vista. Un interessante paper realizzato da ricercatori di IBM del 1973 racconta di alcuni esperimenti volti a comprendere e migliorare l'ispezione visiva di alcuni chip effettuata da ispettori addestrati. Dalla ricerca emerse che il tempo medio di rilevamento di difettosità su un pezzo era di circa 200ms, una tempistica molto elevata se si pensa alla mole con la quale i chip venivano prodotti. Il risultato più interessante della ricerca risiedeva nel fatto che il 23% dei chip contenenti anomalie veniva comunque considerato come corretto, mentre solo il 2% dei chip senza anomalie veniva scartato.[17]
Sono state pubblicate negli anni numerose pubblicazioni su argomenti più critici come l'ispezione manuale di prodotti farmaceutici : a partire dagli anni 80 si stava tentando di fornire mezzi statistici per valutare le **prestazioni** degli ispettori, al fine di fornire alle aziende suggerimenti per un corretto svolgimento delle ispezioni manuali.
Storicamente gli studi su visual inspection e computer vision hanno dapprima cercato di migliorare la capacità di ispezione visiva umana fornendo strumenti **statistici** in grado di ridurre i falsi positivi nel tempo. Questo tipo di approccio ha iniziato a subire una ulteriore evoluzione con l'avvento dei **sistemi di visione artificiale** (CVS).
La caratteristica principale di questi sistemi era l'integrazione di apparecchi visivi (come le telecamere) nei sistemi di ispezione, al fine di acquisire immagini ed elaborare tali informazioni tramite l'ausilio di un software.

L'approccio di rilevamento di una difettosità all'interno di un pezzo seguiva tradizionalmente questi 4 passi fondamentali:
1) Acquisizione dell'immagine.
2) Pre-elaborazione (correzione di luminosità, contrasto, saturazione del colore etc.)

3) Estrazione delle **features** da analizzare.

4) Classificazione e confronto con modelli conosciuti.

Tutte le informazioni acquisite dai sistemi di visione artificiale dovevano essere digitalizzate e discretizzate per consentire ad un computer di elaborarle. A causa del poco spazio disponibile durante quegli anni iniziarono a nascere le prime tecniche di compressione (lossless, lossy, ibride) alla quale seguirono le prime standardizzazioni di formato come il JPEG o l'MPEG.
In particolare, le features estratte dalle immagini al fine di rilevare difetti erano solitamente:

- Textures

- Forme

- Colori (pixel)

Inizialmente il lavoro del software si basava sul **confronto diretto** tra un'immagine di un pezzo "ben riuscito" e un pezzo "difettoso" basandosi solamente sulle features estratte nella fase iniziale. All'interno degli use case presentati vi sono esempi di confronto del genere come quello della rondella metallica effettuata dal software dell'azienda Sipotek. In quel caso, infatti, la soluzione per rilevare la difettosità prendeva in esame solamente i pixel dell'immagine, confrontando la disposizione degli stessi al fine di valutare la bontà della forma del componente analizzato.
Storicamente, i metodi di paragone tra i pixel di un immagine erano effettuati strutturando i pixel in un vettore di K-elementi utilizzando lo spazio di **colori** HSV; in questo modo si era in grado di ottenere invarianza rispetto alla luce (azzerando il canale V= 0) mantenendo allo stesso tempo il colore e la saturazione originali della fotografia da analizzare. Il confronto tra i colori dei pixel era effettuato con differenti tecniche tra cui:
- **LP-NORM**: Confronto 1-1 con i pixel delle due fotografie in input senza considerare la **correlazione** tra differenti colori (ad esempio non si considera che rosso e arancione sono più simili di rosso e blu)

$$L_p(p,q) = \left(\sum_{i=1}^{D} (|p_i - q_i|)^p \right)^{\frac{1}{p}}$$

Figura 21 p e q vettori di pixel confrontati con LP NORM

- Weighted Eucliden Distance: Oltre ai due vettori rappresentanti i pixel delle immagini veniva dato in input al sistema anche un vettore rappresentante i pesi, ovvero valori indicativi dell'interesse di un colore rispetto ad un altro. Se per esempio la difettosità era di un colore rosso acceso, era possibile dare più importanza all'analisi di tale colore che al resto!

$$L_{2,W}(p,q,W) = \left(\sum_{i=1}^{D} w_i(|p_i - q_i|)^2\right)^{\frac{1}{2}}$$

Figura 22 Due vettori di pixel p e q abbinati al vettore peso w

1.7.1 Feature **engineering** vs Feature **learning**

L'incredibile aiuto offerto dai sistemi di visione artificiale spinse le aziende ad effettuare investimenti e ricerca al fine di sviluppare tecniche sempre più complesse per ottenere conoscenza da immagini e filmati. Non vennero perfezionati solamente gli algoritmi di estrazione di features (che includevano ad esempio forme e textures) ma si iniziarono ad applicare gli studi di Fourier per effettuare analisi di **figure e forme** anche nel dominio delle frequenze.
Tutte queste tecniche di vecchia generazione presupponevano requisiti di pre-elaborazione ed estrazione delle caratteristiche necessarie a confrontare le immagini. Ogni distinto problema era infatti risolto grazie all'intervento di **esperti** di settore in grado di formulare un **catalogo di regole** che definivano formalmente quando un pezzo era difettoso oppure no. Questa attività di ingegnerizzazione è chiamata **feature engineering** ed è formalmente definita come quel processo con il quale è possibile estrarre conoscenza a partire dai dati grezzi e dalle loro features.
Se per esempio un'azienda fosse stata interessata a scrivere un algoritmo in grado di riconoscere e scartare mele marce, avrebbe dovuto creare un sistema che, a partire da certe features, fosse in grado di discriminare situazioni positive da quelle negative (in pseudocodice):

```
scarta SE:
        mela.dimensione < X  ||  mela.colore != "Rosso"
```

Da come si può notare, queste soluzioni **dipendono fortemente** dal dominio applicativo e dal problema da risolvere. È richiesta una **profonda conoscenza tecnica** delle features che rappresentano l'elemento in esame e ciò si traduce nella necessità di assumere figure specializzate che mettano a disposizione le proprie conoscenze per la costruzione di questi sistemi. Gli esperti devono essere in grado di definire formalmente quando un pezzo non si trova in condizioni perfette e questo si traduce in **modelli specializzati** per la risoluzione di un singolo problema. Sebbene scriverle fosse molto dispendioso a livello di tempo, in queste piccole regole si sedimentavano conoscenze di tantissime persone che nel tempo avevano lavorato sul problema, analizzando e raffinando le soluzioni al fine di aumentare la precisione con la quale il modello era in grado di riconoscere difetti.

Le tecniche moderne hanno compreso la difficoltà nel gestire un numero sempre maggiore di regole e soprattutto il problema di mantenerle nel tempo. Piuttosto che **sistemi specializzati**, oggi si sta cercando di creare soluzioni **generalizzabili** e applicabili a diverse classi di problemi.

Per questo motivo la visual inspection viene ad oggi approfondita con tecniche di apprendimento **automatico** per fare in modo che queste regole vengano ricavate attraverso una consultazione di **esempi** forniti da esperti umani. Questo processo è chiamato **feature learning** ed è un insieme di tecniche che consente ad un sistema di scoprire in maniera autonoma un modo per **classificare e prendere decisioni** analizzando i dati in input.

In ambito Machine learning esistono differenti approcci con la quale è possibile far **apprendere** ad un sistema una qualsiasi struttura:

- Apprendimento supervisionato: si forniscono dataset di elementi che possiedono certe caratteristiche. Il sistema cerca di **prevedere** ad esempio la specie di un certo animale a partire dal numero di zampe, dalla lunghezza del pelo etc.
- Apprendimento non supervisionato: un sistema è in grado di riconoscere determinati pattern a partire dai dati di input creando gruppi e agglomerati chiamati **clusters**. Si applica ad esempio per il riconoscimento facciale.
- Apprendimento con rinforzo: il sistema è in grado di ottimizzare il proprio comportamento grazie ad un sistema di **ricompense/punizioni**. Se il sistema prende una decisione corretta verrà ricompensato, nel caso in cui prenda una decisione errata allora sarà punito.

La tecnica più moderna applicabile in ambito computer vision per addestrare un sistema a riconoscere situazioni di varia natura è chiamata **Deep Learning.**

Il Deep Learning comprende un insieme di metodi di Machine Learning, una branca dell'intelligenza artificiale che si occupa di costruire sistemi che si ispirano alle capacità percettive e interpretative tipicamente umane. Questi metodi si ispirano alla biologia sottostante al nostro cervello ed è caratterizzata dalla presenza di **reti neurali** a più livelli in grado di processare ed elaborare i dati in input.

In letteratura, il primo esempio che si fa per analizzare questa tecnica è il riconoscimento automatico della scrittura manuale: dato in input un numero, il sistema deve essere in grado di riconoscerlo, anche con calligrafie diverse. Per farlo, il software si avvale di una rete neurale composta da un **layer di input**, da alcuni **layer nascosti** e da un **layer di output**.

Il primo layer ha il compito di prendere i pixel dell'immagine (associati con un valore nella corrispondente scala di grigi) e trasferire tale conoscenza al primo layer nascosto che si occupa del riconoscimento di **angoli e segmenti.** Una volta riconosciuta la presenza di queste strutture, i neuroni del layer numero 2, trasferiranno la conoscenza al layer successivo, che avrà il compito di riconoscere **cerchi e linee**. Grazie all'analisi di questi pattern, il layer di output avrà tutte le informazioni necessarie per riconoscere quale numero è stato dato in input dall'utente.

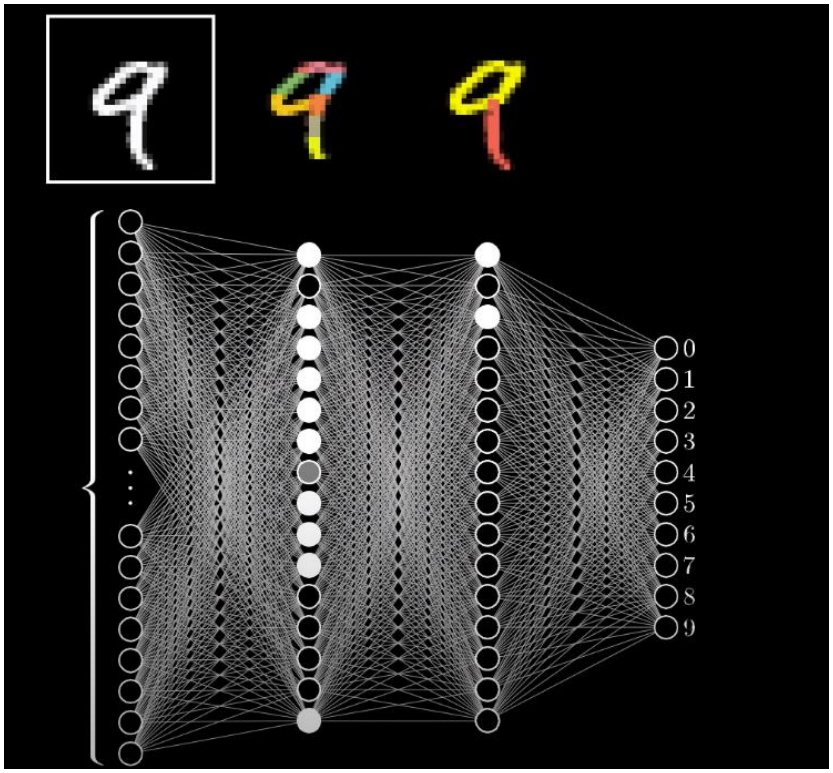


Figura 23 Deep learning e riconoscimento scrittura

Applicando consapevolmente questa tecnica in ambito industriale sarebbe possibile creare sistemi **generici** che, a partire da un certo dataset di esempi rappresentativi, siano in grado di adattarsi a diverse tipologie di problemi anche di categoria molto diversa tra loro.
I sistemi ad apprendimento **general purpose** sono sì in grado di applicare il proprio operato su tantissimi casi d'uso, ma rimarranno sempre ad un livello **superficiale** rispetto all'unità di elaborazione farmaceutica di turno che viene utilizzata per una miriade di controlli specifici.
Purtroppo, i sistemi di deep learning per operare richiedono spesso milioni di immagini di addestramento per raggiungere la precisione richiesta in ambito industriale; sebbene questa disciplina sia infatti molto affascinante, risulta complesso sradicare completamente gli approcci a regole che sono stati adottati dalle industrie negli ultimi decenni. Per quanto "cablare regole nel codice" sia considerata per letteratura una cattiva pratica di programmazione, essa risulta comunque il metodo più adottato per la risoluzione rapida di problemi di questa natura.
Come riferito dal CEO di Landing AI Andrew NG, azienda di intelligenza artificiale :
"Le soluzioni di ispezione visiva basate sull'intelligenza artificiale hanno dimostrato chiari vantaggi rispetto ai metodi convenzionali, ma l'adozione complessiva è **lenta** poiché molte aziende rimangono bloccate dopo alcuni progetti di prova su piccola scala".
Le ispezioni visive proposte da Landing AI consentono anche ad utenti non esperti di addestrare e distribuire un modello con pochi click, consentendo alle industrie di non essere vincolate a team di terze parti per effettuare addestramento e deploy di queste reti. Ad oggi però solamente il 5% dei produttori ha riferito di avere strategie di IA all'interno della propria catena produttiva.[18]
La soluzione migliore sarebbe quella di unire l'approccio tradizionale basato su **regole** con sistemi di **apprendimento** consapevolmente modificati al fine di sfruttare quelle regole in maniera tale da far risparmiare alla rete molte ore in fase di addestramento e la necessità di avere un dataset di milioni di immagini, è questo l'approccio ad esempio seguito da IBM Research nelle ricerche più recenti che prendono il nome di Neuro-symbolic AI.
Come è semplice intuire, il Deep Learning è in grado di superare molti dei problemi che gli algoritmi CV tradizionali non possono risolvere. Il problema fondamentale risiede **nell'alta variabilità** dei dati di **input** la cui qualità viene influenzata dalla presenza di elementi come l'ora del giorno o la presenza di ostacoli davanti all'oggetto da esaminare. Se si tentasse di identificare un'auto in un'immagine utilizzando come features solo la presenza di quattro ruote e di un parabrezza l'approccio si rivelerebbe infatti fallimentare e poco generalizzabile. Sebbene tutte queste caratteristiche possano risultare semplici da riconoscere per un essere umano, gli algoritmi

di computer vision si trovano spesso in difficoltà a dettagliare accuratamente queste situazioni. Il Deep Learning è in grado di superare questi limiti degli algoritmi convenzionali suddividendo il problema in più livelli e costruendo concetti complessi basati su associazioni molto dettagliate. Unico prezzo da pagare? Lunghi tempi di addestramento della rete e dataset di immagini molto grandi, forse non adatti ad uno scenario industriale.

L'obiettivo e lo scopo di questa tesi è quello di **esplorare** queste soluzioni al fine di rinnovare la catena produttiva e il suo intero ecosistema, cercando di introdurre strumenti digitali come il **Deep Learning** che facilitino l'ispezione **visiva** di prodotti o semilavorati nell'industria.

CAPITOLO 2 – SOLUZIONI TECNOLOGICHE DI COMPUTER VISION E VISUAL INSPECTION

In questo capitolo verranno messe a stretto confronto le tecniche di apprendimento **general purpose** con i **sistemi specializzati** che hanno caratterizzato gli ambienti industriali dello scorso ventennio nell'ambito della computer vision. Si presenteranno sia sistemi presi disponibili da grandi aziende che hanno costruito veri e propri business sfruttando la computer vision ma anche realtà più piccole che hanno cercato di **verticalizzarsi** sull'utilizzo di tecnologie più specifiche. Particolare attenzione, in quanto oggetto della tesi, sarà volta all'analisi di quelle aziende che hanno avuto una certa rilevanza nel campo della **visual inspection** analizzando le loro soluzioni dal punto di vista critico per farsi un'idea dello stato dell'arte ad oggi di questa branca della computer vision. A fine capitolo verranno effettuati alcuni rapidi confronti di tool di **visual recognition, di tipo generalizzato,** offerti dalle aziende MICROSOFT, IBM e GOOGLE per vedere e testare le API offerte gratuitamente da queste grandi aziende anche per comprendere il perimetro della loro applicabilità nell'ambito della visual inspection.

2.1 Sistemi **general purpose** VS sistemi **specializzati**

Fino a circa 15 anni fa problemi come la traduzione automatica tra due lingue avevano come elemento di alta specializzazione la possibilità di scansionare enormi dizionari legati tra loro da complesse regole grammaticali create da esperti.
Tali sistemi, sebbene facessero uso di alcune tecniche di intelligenza artificiale, erano **altamente specializzati** e si portavano dietro moltissimi problemi tra cui il mantenimento delle proprie regole interne oppure gli eventuali conflitti nati nei casi di possibili scelte multiple.
Con l'evoluzione del Deep Learning l'approccio si è spostato sull'accumulo di features anche apparentemente distanti e sovrabbondanti tra loro come la presenza di maiuscole, elementi di punteggiatura oppure la separazione tra le parole. Continuando ad allenare le reti e fornendo diversi esempi, i layer e i pesi della rete si sono stabilizzati ed oggi abbiamo traduttori efficienti come Google Translate non più legati a regole fisse ma a reti neurali performanti.
Alcune delle domande che oggi ci poniamo sono:

> Quanto è possibile spingere un sistema general purpose ad apprendimento rispetto ad un sistema specializzato?
> È possibile superare un sistema specializzato utilizzandone uno generico?

In alcuni campi gli algoritmi general purpose sono stati in grado di **superare** anche l'operato umano e garantire prestazioni persino superiori a quelle ottenute dai sistemi specializzati.
Un esempio di quanto detto è il sistema GPT-3, sviluppato dal laboratorio di ricerca *OpenAi,* con l'obiettivo di produrre testi simili a quelli prodotti dagli umani. Questa enorme rete ha la capacità di gestire **175 miliardi** di parametri di apprendimento automatico al fine di elaborare il linguaggio naturale e generare testi di qualità talmente elevata da essere difficilmente distinguibili da scritti umani.[19] GPT-3 è un classico esempio di *brute force* applicata all'AI; si tenta di utilizzare come features non più le singole e localizzate informazioni rilevanti ma data set enormi e insiemi di feature estremamenti grandi, grazie alla disponiblità di materiale accedibile via **internet**. Questa gigantesca rete che non nasce per uno scopo preciso, può essere applicata sia per sintetizzare testi che per fare altre cose come l'analisi grammaticale o sintattica di alcune frasi.
Questa breve introduzione al secondo capitolo è stata presentata per mostrare che il sorpasso dei sistemi general purpose rispetto a quelli specializzati è in realtà già avvenuto per alcune categorie di problemi molto ristrette (vedi il caso dell'analisi e della generazione del testo).
Il problema fondamentale di questi sistemi è che per addestrare una rete da 175 miliardi di parametri ci vogliono **risorse computazionali** veramente impressionanti: per il modello che sta dietro a Google Translate ci vogliono diversi milioni di dollari e una quantità elevatissima di GPU. Se per addestrare una rete e generare un modello occorre coinvolgere un enorme network di computer oppure **dataset di milioni di immagini**, la cosa non risulta conveniente dal punto di vista dei costi. Se si volesse fare un paragone, si potrebbe dire che i modelli **general purpose** allo stato dell'arte attuale sono uno **specchio di esempi** fagocitati a forza bruta dalle più disparate fonti di informazioni.
I sistemi di oggi tendono ad **abbattere questi costi** creando diversi **modelli generalizzati** che vengono **pre-addestrati** e venduti dalle grosse aziende come Microsoft, Amazon, IBM e Google. Questi modelli, già capaci di risolvere alcuni sotto-problemi, sono in grado di diventare performanti nelle mani del cliente con l'aggiunta di qualche **esempio specifico** nel campo applicativo nella quale si vogliono utilizzare.
Un caso d'esempio potrebbe essere un sistema generico allenato a riconoscere **forme, linee e curve** applicabile in diversi campi come il riconoscimento della scrittura, degli alimenti oppure di

pezzi elettronici.
Quando un utilizzatore finale riceve un modello pre-addestrato, solitamente deve aggiungere alcuni **esempi specifici** (magari eliminando magari qualche layer della rete iniziale), addestrare nuovamente la rete a risolvere il suo problema e poi procedere ad utilizzarla.
Oggi tutti i grossi produttori sfruttano una **base generalizzata di sistemi** di partenza che va poi completata con lo specifico servizio richiesto dall'utente finale.
La possibilità di addestrare una rete con un determinato dataset, al fine di *trasferire* tali conoscenze in un diverso problema con **elementi atomici comuni** al problema di partenza, è detto **Transfer-Learning.** Un classico esempio di Transfer-Learning è quello che si fa imparando a riconoscere le auto: utilizzando la conoscenza acquisita durante questa fase sarebbe possibile riconoscere anche i camion oppure le motociclette.[20]

2.2 Soluzioni di computer vision e visual inspection **nell'industria**

In questa sezione cercheremo di approfondire il **mondo industriale della computer vision** e capire se alcune aziende di grosse o piccole dimensioni si occupano di **visual inspection.** L'obiettivo di questa sezione è quella di dare un'idea generale dei produttori che si stanno occupando dell'argomento ed analizzare in che modo queste soluzioni differiscono tra loro. Verranno prese in esame sia aziende di grosse dimensioni che piccole-medie imprese dando spazio anche ad alcuni framework open source. Il focus delle grandi aziende sarà fatto sui servizi di Machine Learning e IA che esse mettono a disposizione per i propri clienti (in cloud o locali). Si esamineranno brevemente le offerte proposte dalle big companies e si cercherà di valutare quanto tali soluzioni possano essere applicate in ambito **visual inspection** e soprattutto CHI si occupa di questa particolare branca della computer vision.

2.2.1 **AMAZON** AWS REKOGNITION

La piattaforma dove Amazon offre i propri servizi Cloud computing, che rappresenta il 58% dei guadagni totali del colosso di Bezos, è chiamata **Amazon AWS**. In ambito visual recognition, Amazon offre il servizio ***Rekognition***, un sistema nato per identificare oggetti, persone, testo scenari e attività basandosi sul **deep learning**.

Figura 24 Riconoscimento oggetti Amazon Rekognition

Il servizio viene offerto ai clienti anche senza competenze specifiche di machine learning ed è in grado di svolgere innumerevoli compiti in base allo scenario in cui viene impiegato.

- **Rilevare oggetti**: Rekognition riesce ad indentificare qualsiasi tipologia di **oggetto** all'interno di un'immagine oppure di un video. Dopo il riconoscimento e l'assegnazione di un punteggio che rappresenta il grado di affidabilità, consente di effettuare ricerche e filtrare i risultati a partire dall'output generato.

- Riconoscere i volti, analisi e controllo facciale: Rekognition è in grado di riconoscere **volti** umani e creare un indice delle persone che presentano determinate caratteristiche come gli occhi aperti oppure il sorriso. Tali immagini possono essere utilizzate per calcolare la probabilità che un volto in immagini **diverse** appartenga alla stessa persona. È basato su un punteggio di somiglianza che identifica ogni utente in ciascuna foto in tempo reale. La tecnologia di riconoscimento facciale di Amazon risulta ad oggi una delle più avanzate del mondo e sotto alcuni aspetti anche migliore di alcuni prodotti offerti da altre aziende competitors.

Figura 25 Analisi volti Amazon Rekognition

- Rilevamento di immagini inappropriate: Rekognition è in grado di capire se, all'interno di una determinata fotografia o video, sono presenti contenuti espliciti inappropriati al fine di segnalarli ed eventualmente rimuoverli.

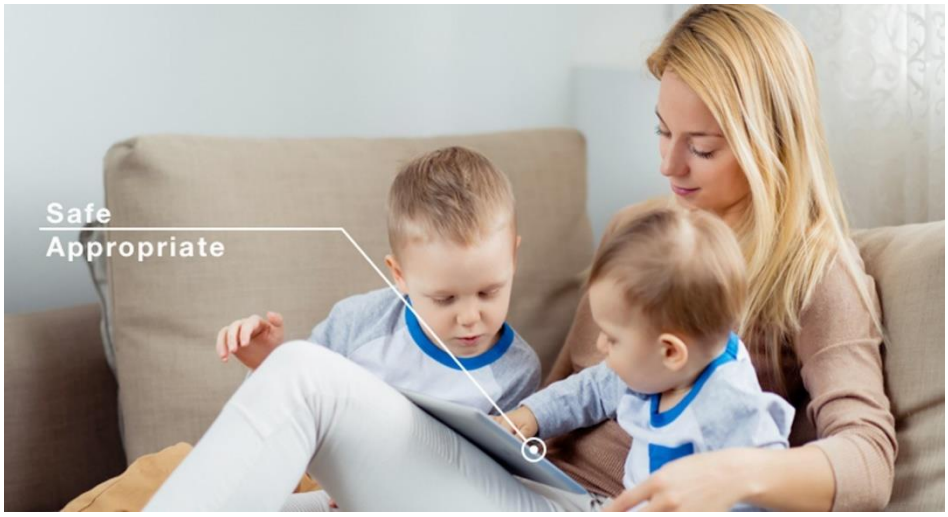


Figura 26 Riconoscimento di contenuti inappropriati Amazon Rekognition

- Estrazione di testo: Rekognition è in grado di effettuare estrazione di **testo** all'interno delle immagini di qualsiasi forma. Le informazioni ottenute vengono catalogate con un'etichetta che rappresenta la probabilità che una certa parola sia stata rilevata correttamente. Le estrazioni possono essere effettuate su qualsiasi tipo di superficie e non solo su testi cartacei o digitali.[21]

Figura 27 Estrazione di testo Amazon Rekognition

Amazon Rekognition è stato venduto a moltissime agenzie governative Americane e durante la storia del proprio utilizzo è stato impiegato per **identificare** e **analizzare** i volti di diversi sospettati colpevoli di reati anche gravi. L'adattabilità e la generalità di questo sistema hanno permesso ad Amazon Rekognition di permeare diversi settori di mercato. Il software è stato anche utilizzato per gestire i problemi di immigrazione (schedare nuovi immigrati) oppure in ambito urbano per riconoscere rapidamente le infrazioni compiute dai veicoli (divieto di sosta, di parcheggio etc..). Il prodotto ha subito diverse evoluzioni e critiche nel tempo, soprattutto a causa della **bassa precisione** con la quale venivano effettuati i riconoscimenti facciali: nelle sue prime versioni il software era in grado di dare all'utente finale un unico possibile risultato (best match), invece che un ampio ventaglio sulla quale effettuare una scelta. Nel 2019 alcune ricerche mostrarono che Amazon Rekognition ha avuto molte difficoltà a riconoscere correttamente le donne con carnagione scura (35% error rate) rispetto ai competitor IBM e Microsoft.[22] Questi gravi problemi di profilazione razziale hanno fatto sorgere diversi dubbi e hanno allarmato gli azionisti di Amazon che decisero di votare a favore del divieto di vendere il software alle agenzie governative. Nel 2020 il fondatore e CEO di Amazon, Jeff Bezos, ha parlato a sostegno del movimento Black Lives Matter decidendo di vietare temporaneamente alla polizia di utilizzare la sua tecnologia di riconoscimento facciale per un'anno.[23] Amazon **NON** si occupa direttamente di Visual inspection.

2.2.2 **MICROSOFT** AZURE VISIONE ARTIFICIALE

Azure è la piattaforma di Microsoft che, analogamente per quanto visto con Amazon e AWS, offre servizi di cloud computing di diverso genere a moltissimi clienti nel mondo. Tra i diversi prodotti proposti dalla piattaforma esistono soluzioni anche in ambito computer vision per automatizzare l'**estrazione** di **testo,** oppure **analizzare immagini** e **video** in tempo reale. Analogamente a quanto visto nella sezione 2.2.1 con l'azienda competitor, Microsoft non obbliga i propri clienti ad avere competenze di machine learning per poter utilizzare i propri prodotti. La documentazione Microsoft è molto completa e guida sia il cliente che lo sviluppatore con esempi e demo in diversi linguaggi di programmazione tra cui .NET, Python e Java.[24]

Alcune delle funzionalità offerte dal servizio di **Visione artificiale** Microsoft includono:

- **Estrazione di testo**: È possibile riconoscere testo e analizzare documenti e immagini con lingue e stili di scrittura diversi. Utilizza i modelli più recenti funzionando anche su diverse superfici e sfondi.

- **Comprensione delle immagini**: Microsoft basa l'analisi delle proprie immagini su ontologie dotate di più di 10.000 concetti differenti. È possibile identificare e assegnare tag agli elementi visivi di un'immagine non limitandosi ai soli soggetti principali ma anche agli scenari interni ed esterni. Come diversi altri produttori in ambito computer vision, anche Microsoft restituisce le **coordinate** per ogni tag applicato sfruttando raggruppamenti a rettangolo di tutti gli elementi visivi presenti nell'immagine in esame. Dopo l'esecuzione del software è possibile associare all'immagine analizzata una descrizione generata dal sistema e una lista di tag associati. Successivamente le immagini ottenute dal sistema possono essere **classificate** utilizzando una tassonomia di categorie organizzate in relazioni padre/figlio.

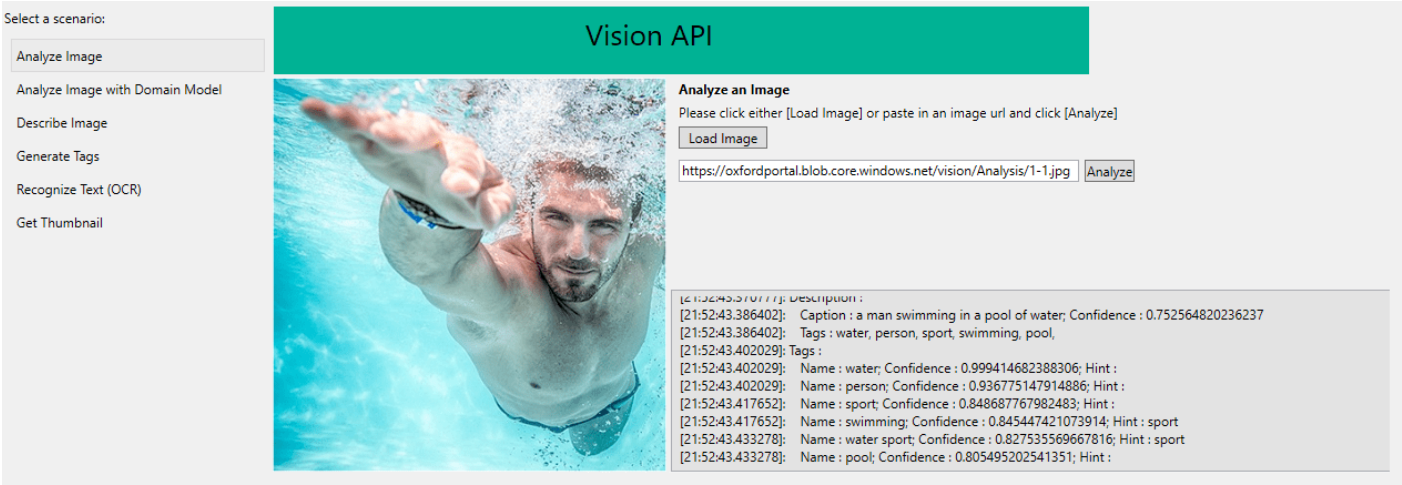


Figura 28 Vision API Microsoft

- Rilevamento dei volti: Microsoft è in grado di rilevare il volto di un soggetto restituendo le coordinate, il sesso e la possibile età di ogni soggetto. Il servizio in grado di svolgere questo compito è chiamato ***Viso*** ed anch'esso, come Amazon AWS, è attualmente utilizzabile solo da enti privati ma non da agenzie governative per la difesa.

Microsoft si occupa direttamente di **visual inspection** con il proprio software **spyglass visual inspection** (MARINER) e di effettuare **multitagging** di elementi all'interno di una fotografia con il proprio servizio di **visual recognition** chiamato **Vision**.

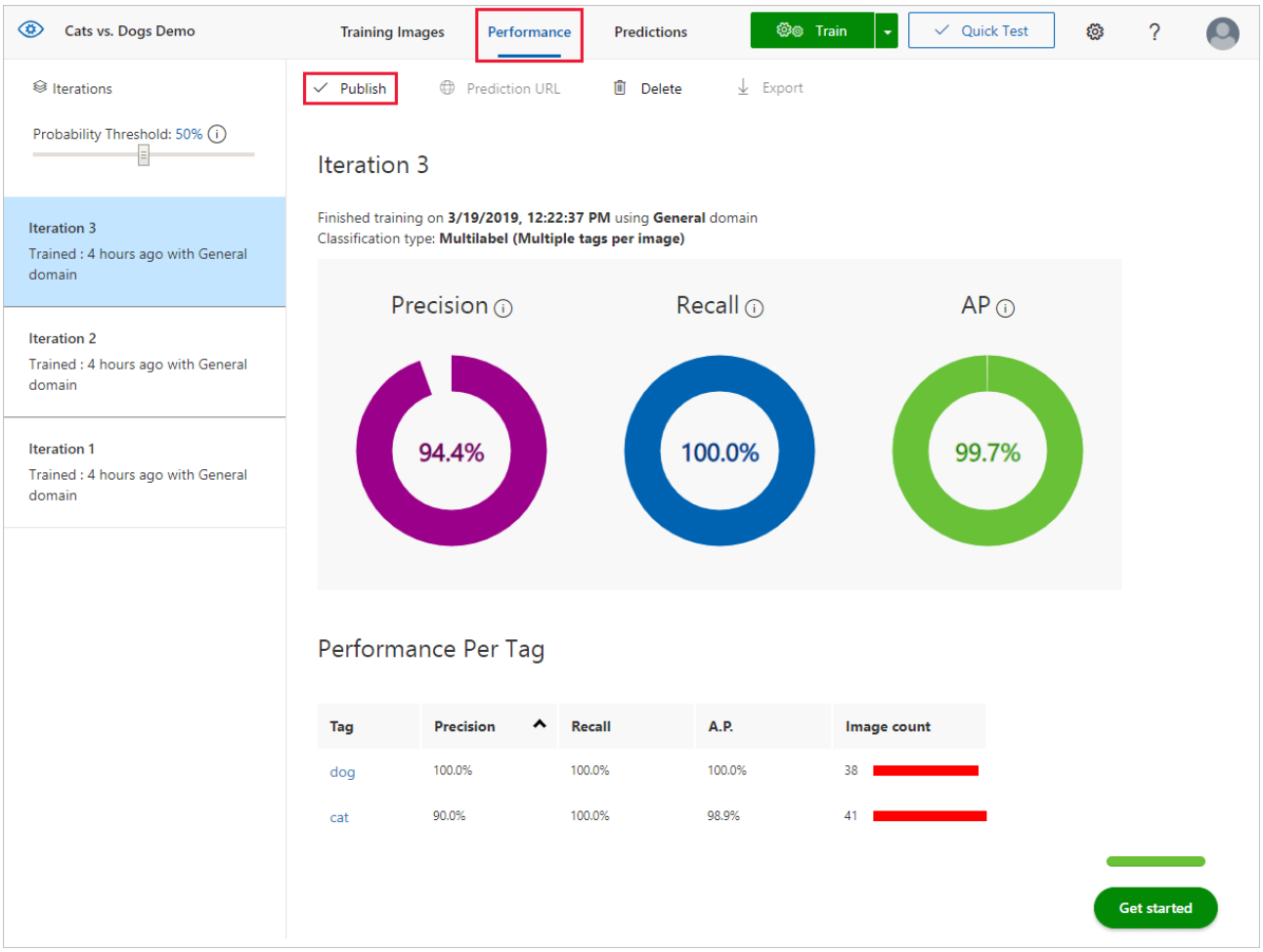


Figura 29 Microsoft vision

2.2.3 **GOOGLE** VISION AI E AUTOML

Google Cloud è uno dei più ricchi ecosistemi dalla quale le aziende leader attingono per portare a termine i propri progetti. Più del 50% delle prime 10 aziende nei settori di telecomunicazioni, media, retali, software, logistica e automotive si affidano alle soluzioni offerte da Google Cloud. In ambito visual recognition e visual inspection, Google dispone di diversi strumenti che fanno tutti parte della suite AutoML.

AutoML ha l'obiettivo di guidare l'utente finale nell'addestramento di **modelli personalizzati** di machine learning anche per clienti con limitata esperienza in questo campo. Le prestazioni di Cloud AutoML sono garantite dall'utilizzo di tecnologie **proprietarie** di Google e si basano sulle più recenti tecniche di **Transfer Learning** e **Neural Architecture Search (NAS)**.

Il **NAS** rappresenta il futuro del deep learning in quanto è in grado di automatizzare il processo di **creazione dell'architettura** per i sistemi di machine learning. In parole semplici questo sistema ha la capacità di fornire all'utente finale la migliore architettura possibile per ottenere ottimi risultati in termini di performance.[25]

I prodotti AutoML sono strutturati in 3 differenti macrocategorie:

- **Linguaggio (AutoML Natural Language)**: Google è in grado di rilevare la struttura e il significato del testo tramite machine learning e tradurre in modo dinamico da una lingua all'altra. Le features offerte comprendono l'estrazione di parole dal testo, l'individuazione di sentimenti da parte di chi ha scritto tale testo, la sintassi ed eventuali categorie.[26]

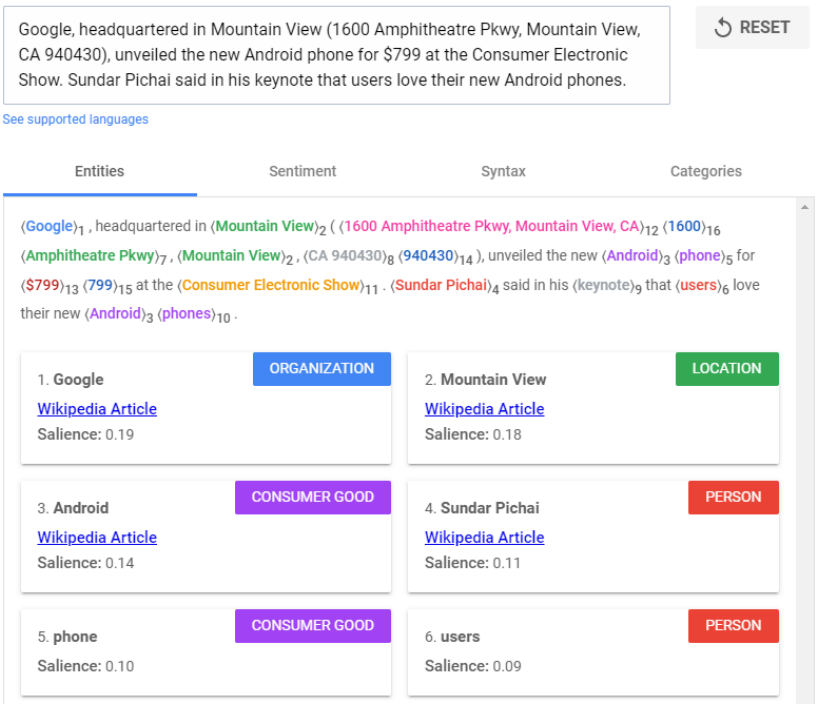


Figura 30 Google AutoML text extraction

- **Dati Strutturati (AutoML Tables)**: Google è in grado, a partire da dati strutturati, di creare modelli di machine learning per effettuare previsioni in diversi campi applicativi. Tutto ciò che deve fare il cliente è caricare un file .csv che rappresenta il dataset di partenza, premere qualche pulsante e infine attendere il training della rete. Grazie a questo sistema è possibile effettuare previsioni su quando avverrà il prossimo pagamento all'interno di una banca oppure prevedere quando un macchinario sarà in procinto di essere sostituito. Google offre la possibilità di allenare la rete generata per un tempo variabile e consentire all'utente di scegliere l'opzione più interessante per lui.[27]

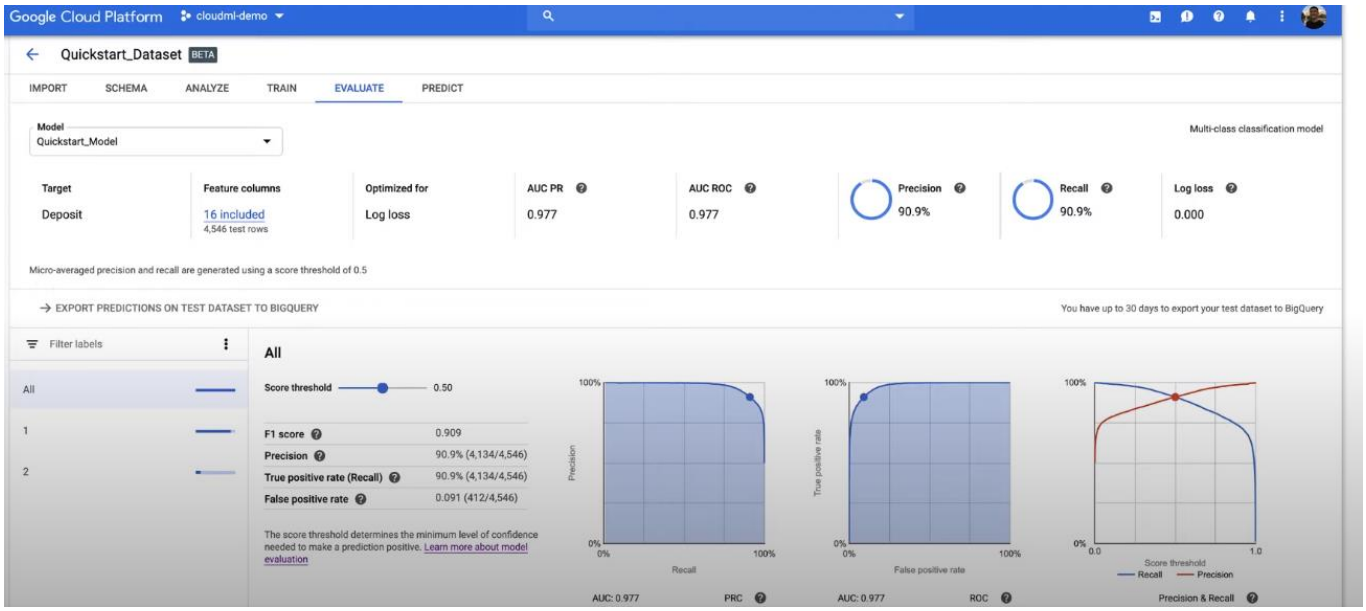


Figura 31 Google AutoML Tables after Training

- **Visivo (AutoML Vision & AutoML Video intelligence)**: L'ambito visivo è ovviamente quello che interessa di più ai fini di questa tesi, e per tale motivo verrà analizzato più nel dettaglio. Google ricava informazioni dalle immagini o dai video sfruttando il potere computazionale del Cloud oppure lavorando in locale.

- **AutoML Vision [APPROFONDITA NEL CAPITLO 3]:** L'utente carica all'interno del sistema una serie di immagini oppure brevi video rappresentanti l'oggetto che si desidera riconoscere.

Una volta raccolti ed organizzati i dati si passa all'interfaccia di AutoML Vision dove basterà semplicemente cliccare su **Train** per sviluppare il modello desiderato. In questo esempio sono state utilizzate immagini di sedie, biciclette e tavoli per ottenere questi risultati. Quando il modello si troverà davanti immagini che rappresentano situazioni mai viste dal sistema, utilizzerà i dati raccolti per cercare di capire di che cosa si tratta.

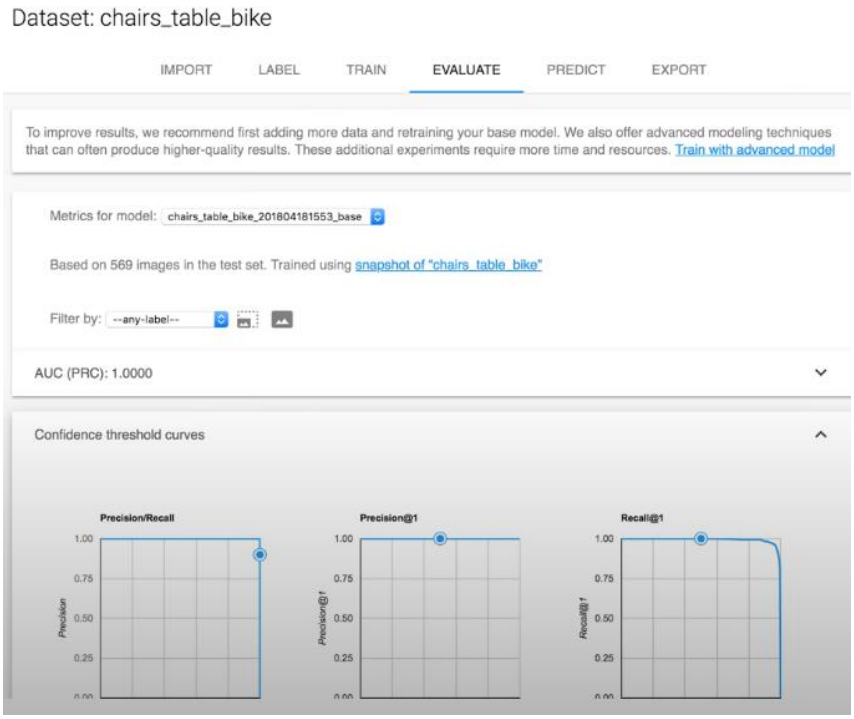


Figura 32 AutoML Training model with images

Alcuni esempi di test effettuati da Google mostrano i limiti del sistema: se il soggetto dell'immagine è posto **davanti** all'obiettivo della camera, il modello allenato è in grado di riconoscerlo assegnando ad esso un alto punteggio di precisione. Se tuttavia sono presenti diversi soggetti nell'immagine a volte il sistema sbaglia e riconosce elementi che in realtà non sono presenti nell'immagine in esame. Lo stesso sistema può essere utilizzato per riconoscere la presenza di elementi all'interno di diversi video: ogni momento che interessa l'utente sarà taggato con una serie di etichette rappresentanti ciò che il sistema è in grado di vedere in tempo reale.

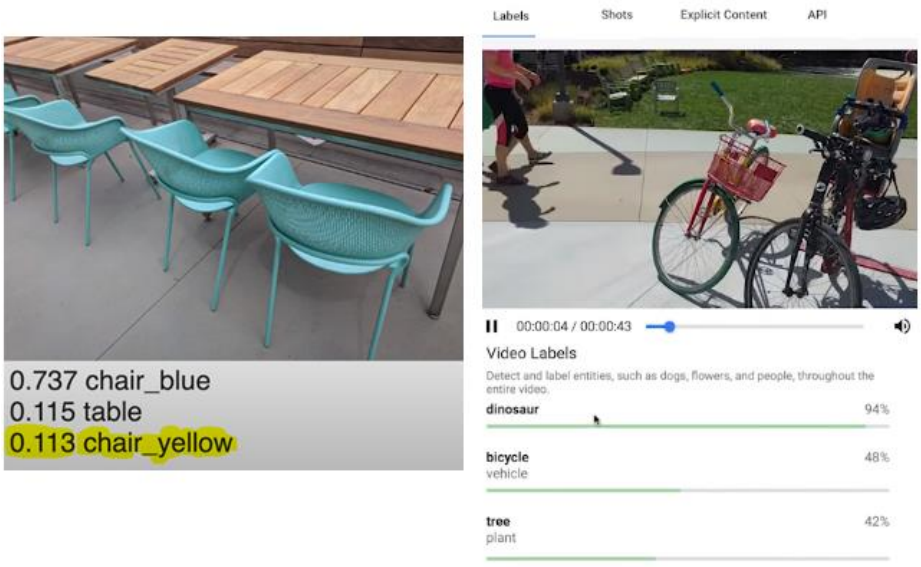


Figura 33 Google AutoML Vision common mistakes & Video

GOOGLE VISUAL INSPECTION: FLEXIBLE VISION

Delle aziende analizzate fin'ora Google è una delle poche che si occupa **direttamente** di **visual inspection**. Google ha riconosciuto tutti i problemi chiave nell'applicare modelli classici di **computer vision** alle realtà aziendali:

- Il numero di campioni per costruire dataset è **limitato.**
- Occorrono alte **competenze** specifiche per creare un sistema funzionante.
- Difficoltà a riconoscere elementi **organici** o particolarmente **variabili** come la frutta.
- Necessità di avere telecamere in grado di essere invarianti alla **luce** o ad altre **variabili ambientali.**
- Costi **molto elevati** per applicare approcci di feature egineering.
- Difficoltà a **mantenere** questi sistemi nel tempo.
- Richiesto l'utilizzo di **telecamere molto costose** per l'acquisizione di immagini.

In un primo momento Google ha analizzato le varie proposte **open source** per cercare la soluzione al problema e si è accorta che **TensorFlow, OpenCV, Node-Red, Zbar e TesseractOCR** non erano integrabili in maniera semplice con loro ed erano spesso soggetti a **curve di apprendimento** molto ripide in termini di tempo.

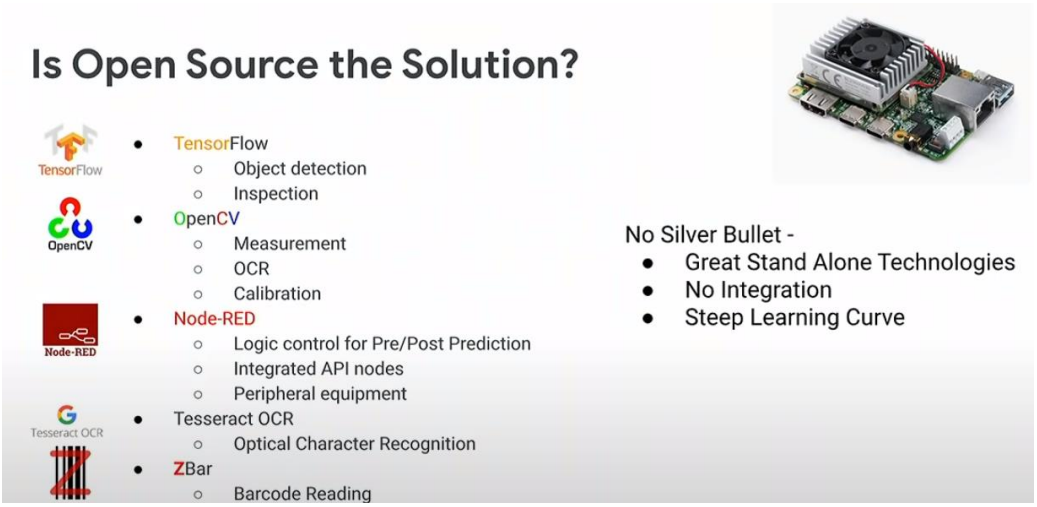


Figura 34 Soluzioni open source esplorate da Google

Google si è impegnata fermamente per **rimuovere le barriere** tecniche richieste per applicare il Machine Learning alla computer vision **semplificando** nettamente il setup iniziale e fornendo all'utente un'interfaccia grafica all-in-one flessibile ed efficace.

La soluzione dell'azienda di Mountain View risiede proprio nell'utilizzo di **sistemi general purpose** costruiti **ON-TOP** di architetture **open-source** per mantenere i **costi limitati** anche avvalendosi di dispositivi a basso uso di potenza come CPU e edge devices.
Il risultato ottenuto è stato chiamato **Flexible Vision,** un sistema con API **aperto** utilizzabile sia **in cloud** che **senza connessione** dove è possibile collegare **qualsiasi telecamera USB** senza alcun tipo di restrizione ed essere subito pronti a creare i propri modelli personalizzabili.
Una delle caratteristiche più peculiari di **Flexible Vision** è la possibilità di **esportare** e **importare** modelli di computer vision favorendo il concetto di **sharing** e la creazione di una community sensibilizzata all'utilizzo della CV. La possibilità di scambiare modelli e di comunicare direttamente con **PLC** e **Robot** rendono il sistema molto versatile in ambito industriale e quindi facilmente integrabile anche in campi dove la connessione è assente.

Il sistema offerto da Google è completamente personalizzabile e supporta telecamere USB fino ad una risoluzione 4K. Dopo aver collegato la propria telecamera essa sarà visibile e selezionabile all'interno del sistema. Dopo una breve calibrazione degli elementi da catturare risulta molto semplice **creare un progetto**; in questo esempio si analizzerà un sistema in grado di allenarsi a riconoscere arachidi di diversa natura.
Il primo step richiede di selezionare una telecamera, un modello ed **iniziare a scattare fotografie** degli elementi che si intende riconoscere.

Figura 35 Raccolta delle fotografie campione da parte del cliente

Subito dopo, occorre **taggare** manualmente le immagini appena prese con l'obiettivo di **istruire il sistema** sulla natura degli oggetti che comporranno il modello. Il software di google supporta anche **OCR** o la lettura di **barcode** con un livello di zoom tale da consentire all'utente di taggare efficacemente anche **piccolissimi componenti** presenti all'interno di una scheda elettronica.

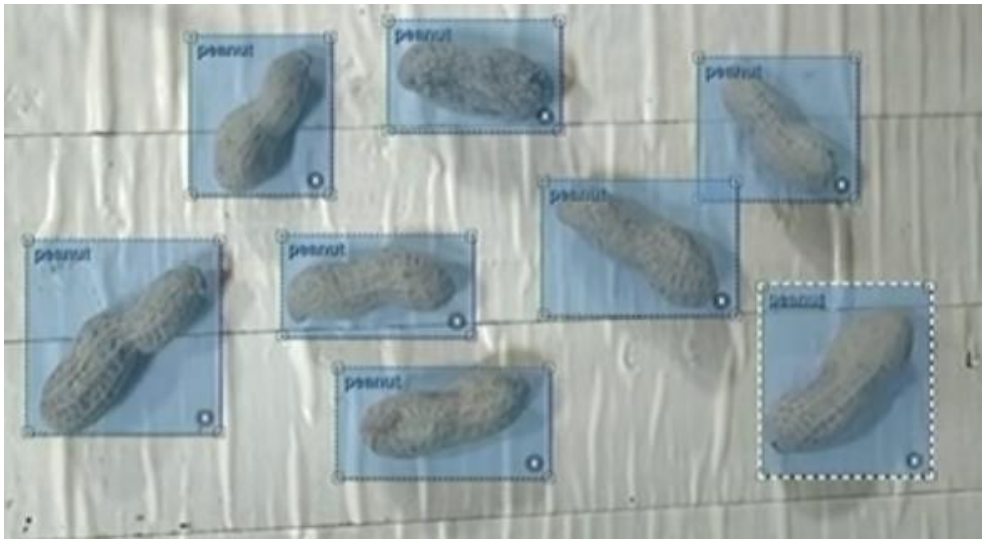


Figura 36 Tag delle fotografie da parte del cliente

Con appena **3 immagini** rappresentanti alcune noccioline, noci e anacardi il sistema è in grado di **generare un dataset in maniera autonoma** applicando all'input fornito diverse trasformazioni come la rotazione o l'aumento/riduzione di scala. Questa tecnica applicata da Google è chiamata "**Image Augmentation**" ed è in grado di abbattere i **costi** dovuti all'elevato numero di campioni richiesti per effettuare un training efficace della rete.

Dopo aver generato migliaia di immagini occorre **sottoporre il modello** alla fase di training della rete attraverso il servizio Cloud di Google, questa fase richiede dai venti minuti ad un'ora e mezza per essere completata con successo. Una volta che il training è stato completato è possibile testare la rete ed effettuare il riconoscimento delle immagini in maniera molto precisa. Una volta effettuato il training, il modello risulta esportabile e ri-utilizzabile anche **offline**!

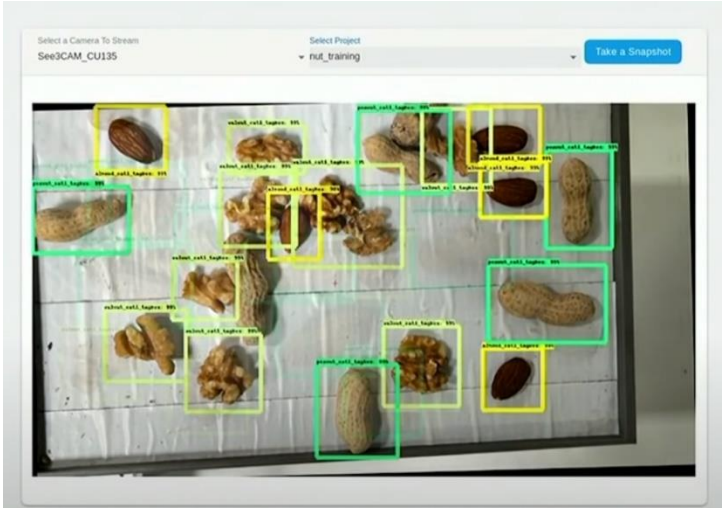


Figura 37 Riconoscimento della rete finale allenata

Analisi dell'architettura Flexible-Vision

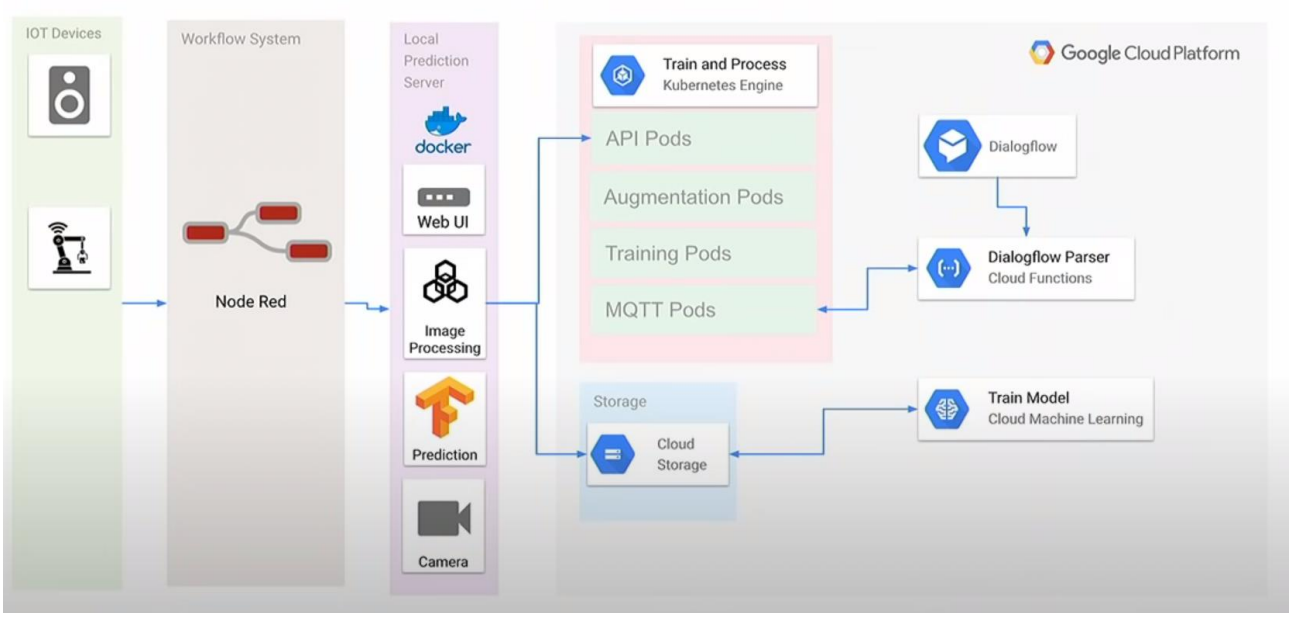


Figura 38 Architettura Flexible Vision

IOT Devices: Tutti i robot o dispositivi IOT che avranno il compito di manipolare i dati forniti dalle telecamere.

Workflow system: Rappresenta qualsiasi tool in grado di creare una rete/workflow di dispositivi connessi tra loro in grado di reagire a certi stimoli. Google ha utilizzato NodeRed, uno strumento di sviluppo basato sulla programmazione visiva.

Local prediction server: I micro-servizi richiesti per il sistema Flexible-Vision di google sono molto pesanti, per questo motivo è stato deciso di utilizzare per ciascuno di essi un **container docker.**

Google Cloud: I componenti presenti nel cloud di Google sono diversi

- Kubernetes è responsabile di effettuare **l'image augmentation.**
- Cloud storage è responsabile della **persistenza** e di comunicare con il componente che effettuerà il **training** della rete.
- Dialog flow e altre interfacce sono responsabili dell'integrazione di alcuni dispositivi nel sistema come Google Home.

Per effettuare la **localizzazione** degli oggetti all'interno di una fotografia, Google si è appoggiata alle API open source offerte da **TensorFlow.**
Questo framework mostra all'utente la classica struttura rettangolare in grado di racchiudere e localizzare i diversi soggetti all'interno di una fotografia.
I **modelli** per il **riconoscimento** degli oggetti sono fatti in 2 modi differenti:

- **Single shot detection**: Utilizza una rete di convoluzione che è in grado di fornire una lista di possibili candidati o tag che potrebbero trovarsi nella fotografia. I vari test effettuati in questa tesi utilizzando come input il chip elettronico utilizzano questo metodo in quanto molto veloce e non troppo dispendioso in termini di potenza richiesta. (Ottimo per cellulari o piccoli dispositivi che utilizzano edge CPU)
Il lato negativo di questa tecnica risiede nel fatto che fa molta fatica a riconoscere **piccoli oggetti.**

- **Faster R-CNN**: A differenza della tecnica SSD, invece che basarsi sul single-shot dell'immagine, questa tecnica effettua **post**-**processing** dell'immagine in input per cercare possibili candidati o tag che potrebbero trovarsi nella fotografia. Il tutto è eseguito cercando mediante algoritmo tutti i **punti di ancoraggio** presenti nell'immagine in modo da capire quali sono gli **elementi principali** di una fotografia o quelli che rappresentano lo **sfondo**. Dopo aver identificato i candidati si effettua la previsione allo stesso modo visto nell'SSD.

Esistono due differenti modelli per **estrarre** le features da un'immagine:
- **RESNET:** Una rete neurale molto **profonda** (più di 100 livelli di profondità) in grado di ottenere in maniera molto precisa oggetti all'interno di un'immagine.
- **INCEPTION:** Una rete neurale molto **larga** che funziona in maniera simile a **RESNET** ma lavorando orizzontalmente piuttosto che in profondità. [28]

Quale delle due è la migliore per effettuare il riconoscimento degli oggetti?

Dipende dal caso, ma un modo generale per farlo è utilizzare la **mean average precision** che viene calcolata valutando:

1) **Precision**: misura di esattezza e fedeltà, rappresenta il numero di oggetti **attinenti** recuperati da una ricerca.

2) **Recall**: il numero di elementi **attinenti** che sono stati recuperati dalla ricerca nell'immagine

3) **Intersezione delle aree contornate**: cerca di valutare quanta area effettiva viene selezionata dall'algoritmo per ogni oggetto rilevato nell'immagine. L'algoritmo non performa bene se la cornice rettangolare che racchiude gli elementi è troppo estesa o troppo piccola. [29]

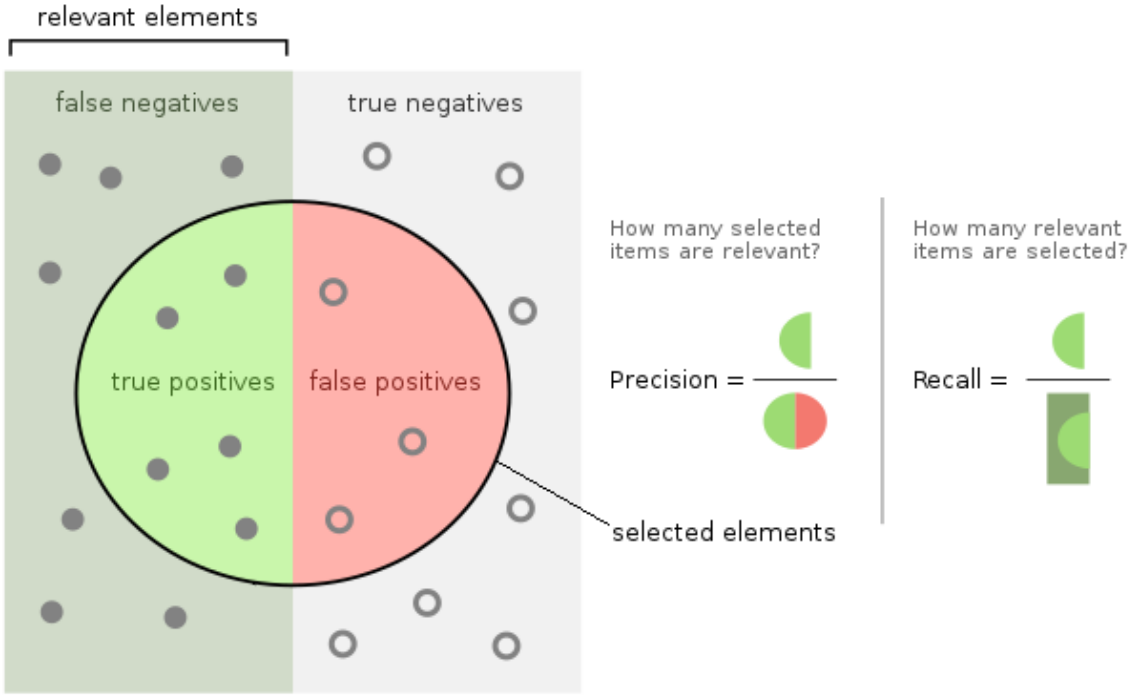


Figura 39 Precision and Recall

Una volta definito l'algoritmo, il sistema di Google Flexible-Vision cerca di **IDENTIFICARE** gli oggetti nello spazio mediante una **calibrazione** della telecamera che andrà a scattare le fotografie da analizzare. La calibrazione viene effettuata mediante una **griglia di pixel** che viene posizionata sulla superficie su cui si vuole effettuare l'analisi (possibilità di usare QR-codes).

Una volta identificati gli oggetti nello spazio si può effettuare:

- Scansione OCR
- Lettura di Bar-codes
- **Rilevare e scartare elementi difettosi** (precedentemente inseriti nel sistema e catalogati)

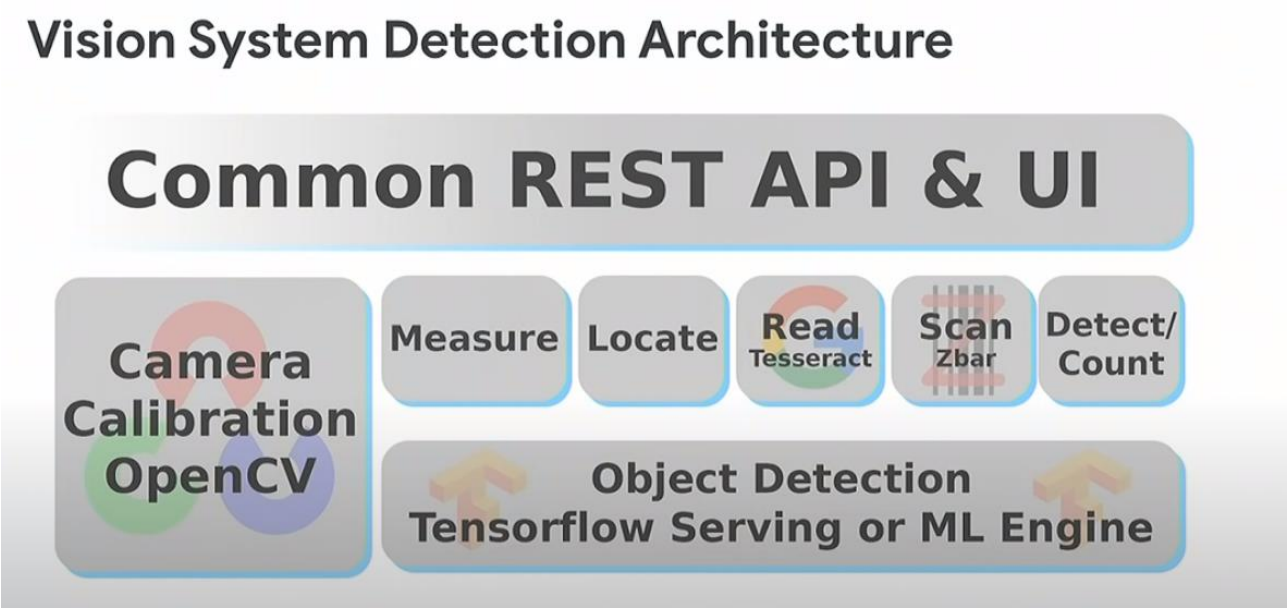


Figura 40 Vision System Detection Architecture

Google ha sintetizzato in maniera eccellente gli step necessari per costruire un sistema di riconoscimento di oggetti e per integrarlo correttamente sia all'interno di un contesto produttivo che in fasi di analisi e verifica della qualità dei prodotti. Questi step sono **gli stessi** per qualsiasi sistema che venga preso in esame e consistono sostanzialmente in:

1) A partire da alcune immagini di input si effettua **Tagging** dei differenti elementi riconosciuti dall'algoritmo che sono **rilevanti** per il cliente. Il **rilevamento** degli elementi può essere effettuato caricando le immagini direttamente sul **cloud** di Google.

2) Dopo aver taggato le immagini avviene la fase di **training** che allena la rete a **riconoscere** tali oggetti. Questa fase richiede **grossi dataset** che Google ottiene tramite il processo di **image augmentation** riducendo la variabilità causata dalle **variabili ambientali** (come luce o orientazione della telecamera). Nel caso di Google-Vision il training è fatto da **AutoML** che utilizza **32 GPU parallele** che consentono di ottenere velocità di completamento del training in circa un'ora (include la possibilità di effettuare un training aggiuntivo della rete ancora più velocemente). Il tempo per effettuare questa fase di addestramento su un PC tradizionale senza affidarsi ad AutoML è dell'ordine di 16 ore.

3) Si seleziona **un'azione** da compiere una volta che è stato completato il riconoscimento. Un'azione potrebbe essere quella di scartare il pezzo nel caso in cui si sia verificata la presenza di un difetto o un'anomalia precedentemente identificata in fase di tagging oppure richiedere ad un robot di consegnarci l'oggetto appena riconosciuto.

In questo esempio viene mostrato il flow di azioni eseguite per effettuare il riconoscimento di una mandorla a partire dall'input **vocale** effettuato con Google Home.
Dopo aver riconosciuto il messaggio esso viene consegnato a **DialogFlow**, una piattaforma di comprensione del linguaggio naturale utilizzata per tradurre e integrare comandi in un sistema complesso.

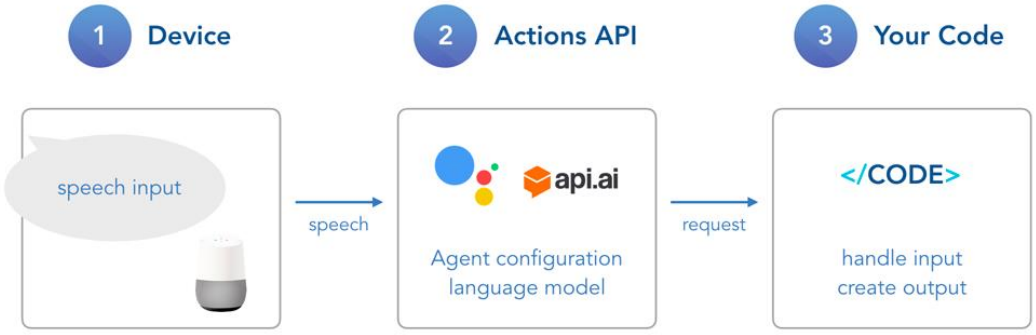


Figura 41 DialogFlow converte l'input vocale in un evento PUB/SUB

DialogFlow produce un evento **PUB/SUB** che viene ascoltato da **Node-Red** (o da qualsiasi gestore di workflow per sistemi IOT) il quale chiama le **REST API** presenti sul server Flexible Vision richiedendo di effettuare una fotografia e restituire i contorni rettangolari di tutti gli oggetti trovati nell'immagine. I rettangoli e gli oggetti sono riconosciuti grazie a **TensorFlow**, il quale consegna al sistema Flexible Vision tutte le informazioni relative alla **posizione in MILLIMETRI** di ogni oggetto, la **label** col nome riconosciuto ed eventualmente lo scan OCR / BarCode.
Tutti i dati sono restituiti infine a **NodeRed** in forma di **JSON** che comanderà il Robot ordinando di raccogliere la mandorla che si trova nella specifica posizione.

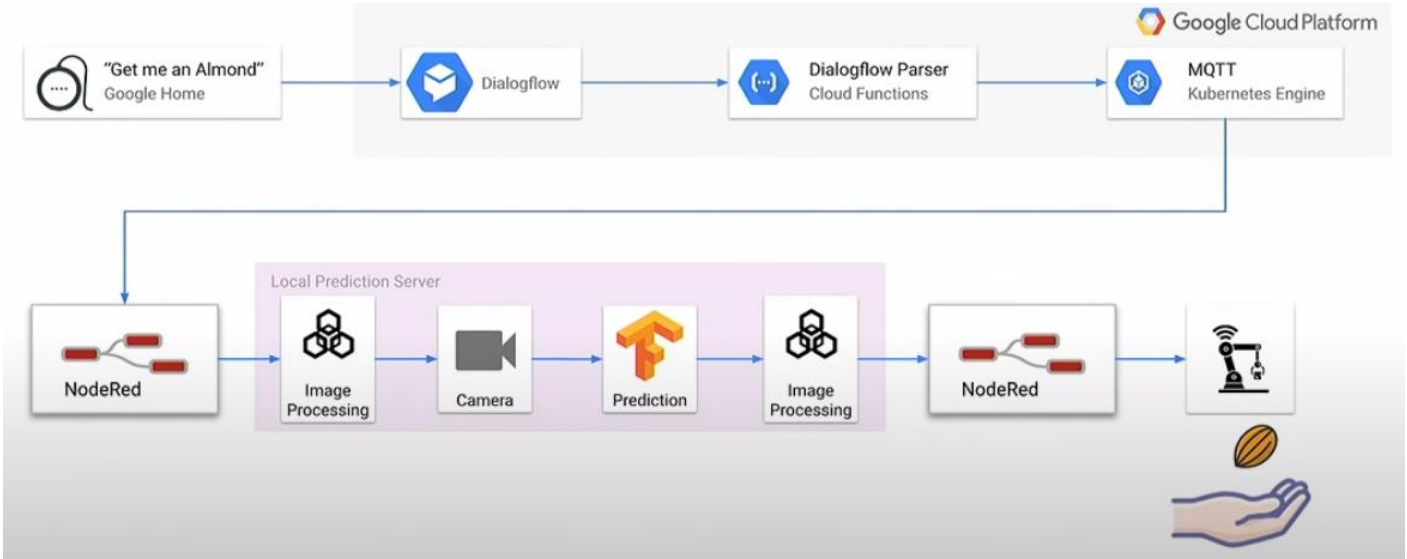


Figura 42 Ottenere e riconoscere una mandorla con un robot tramite comando Vocale

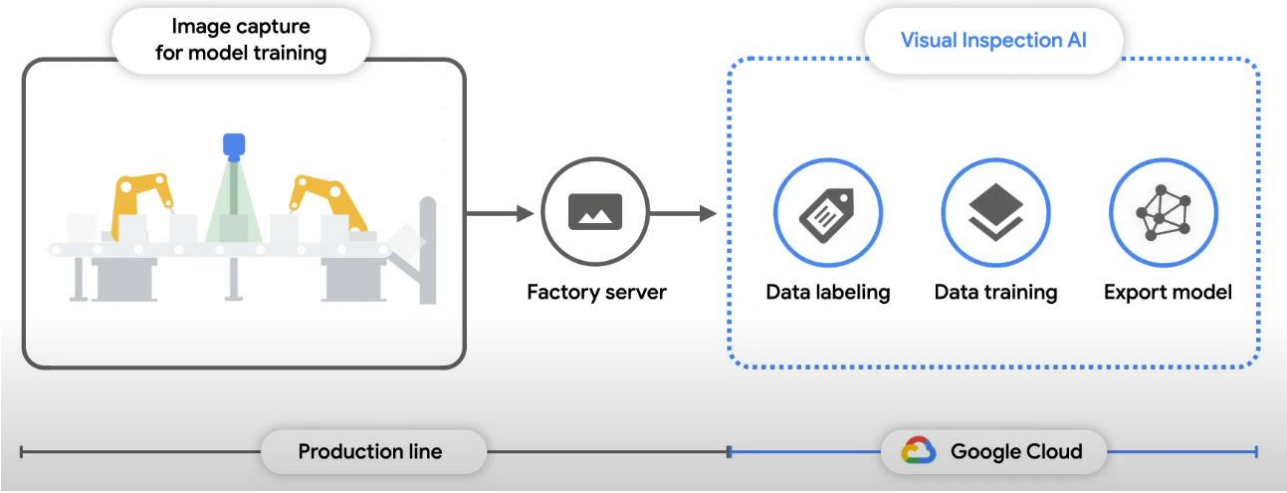


Figura 43 Google Visual Inspection

Google ha dimostrato che sono sufficienti dalle **3 alle 10 immagini** per effettuare un buon riconoscimento di un pezzo o di un oggetto nel classico ambiente industriale.
In altre situazioni dove occorre **riconoscere un pezzo difettoso** potrebbero essere necessarie diverse fotografie (sul **centinaio**) in quanto i difetti sono molto differenti gli uni dagli altri. Sebbene il processo di image augmentation di Google sia in grado di produrre ottimi dataset a partire dalle fotografie di partenza, occorre comunque identificare **a priori** tutti i possibili difetti che possono presentarsi.

I prodotti di Visual Inspection di Google sono applicabili nei più disparati campi aziendali; essi includono il settore **farmaceutico, automotive, semiconduttori** o anche **tessile**.[30]

Settore farmaceutico: Google si è occupata di fotografare e ispezionare dispositivi medici chiusi ermeticamente (tramite iniezioni di silicone) con l'ausilio di due telecamere poste perpendicolarmente l'una rispetto all'altra. Tramite il controllo visivo incrociato di questi apparecchi è stata in grado di capire se il dispositivo medico era utilizzabile oppure da scartare.

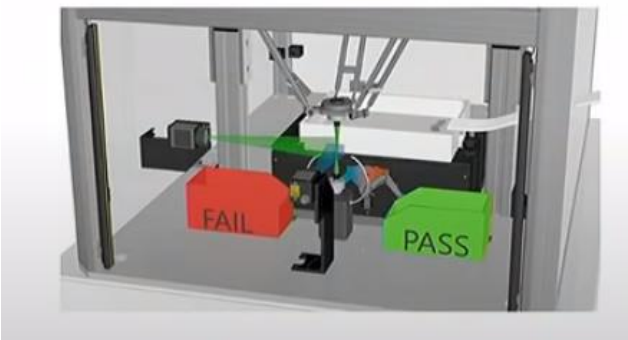


Figura 44 Riconoscimento di dispositivi medici con 2 telecamere Google

Settore automotive: Google ha utilizzato una **telecamera all in one** con un modello pre-addestrato a riconoscere la presenza o l'assenza di pasta termica sopra ad una scheda per uso elettronico. La scheda, che sarebbe stata assemblata su un veicolo, era prodotta in elevate quantità e per tale motivo risultava difficile da ispezionare manualmente da un operatore.

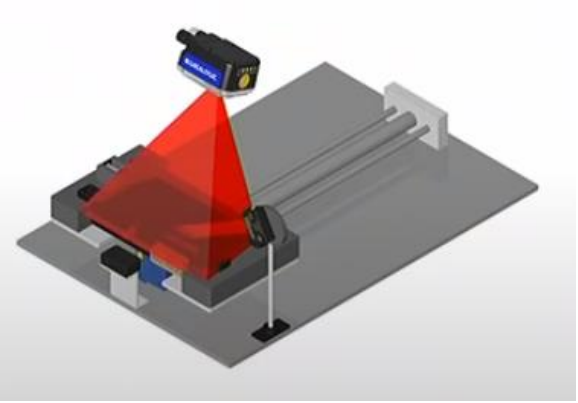


Figura 45 Riconoscimento pasta termica Google

2.2.4 **IBM** CLOUD E IBM VISUAL INSPECTION

La piattaforma Cloud di IBM offre una vasta gamma di servizi differenti ai propri clienti ed è in grado di fornire l'accesso alle proprie risorse ad una velocità molto elevata (2000GBit/s) grazie ai propri server con bassissima latenza ed alte prestazioni. Le risorse e i prodotti offerti alla propria clientela spaziano tra l'ambito mobile, cognitive, IA, Big data, IOT ma anche machine learning. L'aspetto più importante da sottolineare è la **versatilità** con la quale IBM è in grado di soddisfare le esigenze delle aziende: le implementazioni delle applicazioni lato cliente possono essere sviluppate in qualsiasi linguaggio senza compromettere in alcun modo l'accesso alle risorse Cloud. Gli strumenti della piattaforma offerta da IBM consentono di sviluppare, eseguire e distribuire soluzioni personalizzabili dotate delle più disparate configurazioni hardware e software.[31]
Alcuni servizi di AI/Machine learning proposti da IBM cloud includono:

- **Watson Assistant:** sistema di interfacce per costruire **chatbot** in grado di effettuare **conversazioni** in qualsiasi applicazione, dispositivo o canale. Alcune applicazioni tipiche possono essere il supporto al cliente post-vendita, prenotazioni di hotel oppure fornire indicazioni di vario genere all'interno di siti internet o

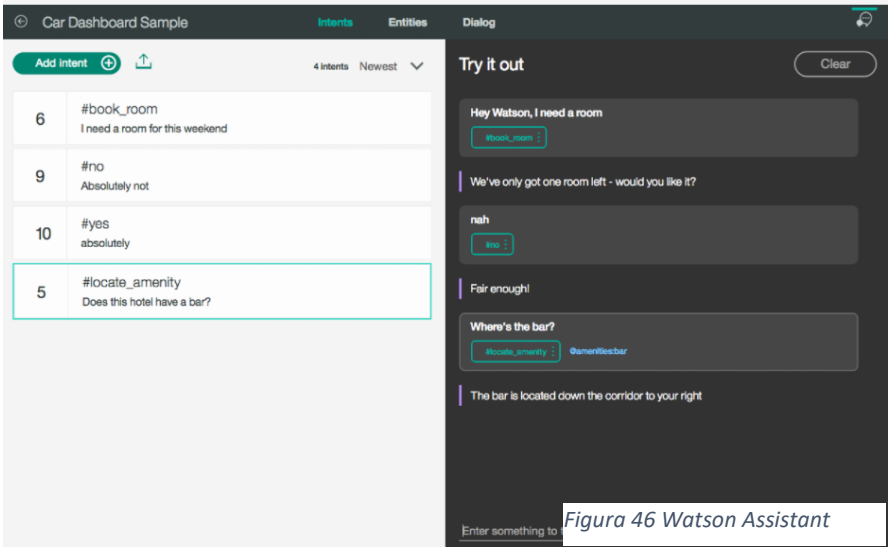


Figura 46 Watson Assistant

luoghi interattivi come musei, teatri, ospedali o cinema. I bot offerti da IBM possono rispondere e **interpretare** il testo fornito in input dall'utente per selezionare la migliore risposta per ogni evenienza. È importante sottolineare che il testo in input non viene mai preso direttamente come comando, esso verrà elaborato e interpretato per cercare di capire le intenzioni dell'utente. Se per esempio l'utente inserisce come input "**nah**" il bot è in grado di interpretarlo come un "**no**" e procedere con la conversazione.

- **Language translator e Natural Language Classifier:** sistema pre-addestrato che è in grado di gestire decine di lingue differenti. Passando un testo come input è in grado di effettuare la traduzione di questo testo.
È possibile addestrare il **traduttore** IBM anche per lingue completamente nuove inventate dal cliente, in questo caso il sistema è in grado di imparare come tradurre dalla lingua del cliente alle altre lingue del mondo. Il **classificatore** IBM è un sistema addestrabile per classificare i testi; se per esempio l'utente volesse classificare le proprie e-mail o SMS in base all'oggetto o al testo presente, questo classificatore è in grado di smistare tali email in base al loro contenuto nella cartella corretta e riconoscere persino testi potenzialmente pericolosi o contenenti virus.
Il classificatore è stato pre-addestrato preventivamente con tantissime lingue da parte di IBM: quando occorre addestrarlo per risolvere il **particolare problema** del cliente, lo sforzo di addestramento risulta **minimo** ovvero nell'ordine di decine di esempi!

- Personality Insights, speech to text e tone analyzer: fanno parte del filone della **sentiment analysis** che si occupa di capire se un testo o una persona parlano in maniera positiva o negativa di un fatto con un certo gradiente. L'obiettivo di questa branca dell'IA è quello di estrarre il maggior numero di informazioni possibili a partire da testi o conversazioni tra persone.
Personality insights è un sistema che è in grado di tracciare un profilo **psico**-**linguistico** della persona che ha redatto un **testo** estraendo da esso diverse features. Se abbinato ad altri servizi come lo **speech to text** oppure il **tone analyzer**, questo sistema è in grado di creare un vero e proprio workflow in grado di analizzare un'intera conversazione e capire le emozioni provate dagli interlocutori.

Tutti questi servizi offerti da IBM sono completamente **addestrabili** quindi **non si programmano!** In sintesi, non è richiesta una particolare esperienza di programmazione per utilizzare questi sistemi che risultano particolarmente user-friendly per i clienti che decidono di utilizzarli. Ognuno di questi servizi possiede un input e un output ben definito e può essere integrato con altri elementi per creare macro-applicazioni in grado di svolgere le più disparate funzionalità.

In ambito **ispezioni visive** IBM offre uno strumento chiamato **Visual Recognition** che possiede funzionalità simili a quelle viste per le altre aziende leader nel settore.

-**Visual Recognition:** Servizio pre-addestrato dotato di una serie di **modelli base** per effettuare il riconoscimento di immagini. Visual recognition possiede anche **modelli specializzati** a riconoscere particolari categorie di elementi come fotografie di **cibi** oppure immagini con contenuti sgradevoli o che potrebbero non risultare adatti per un pubblico comune.
Il cliente può inventare le proprie **categorie da riconoscere** e quindi creare i propri modelli personalizzati per risolvere i propri problemi nello specifico.

IBM VISUAL INSPECTION MAXIMO

Anche IBM si occupa direttamente di **visual inspection**. Il suo prodotto verrà approfondito nel capitolo 4 in quanto sarà oggetto della tesi anche dal punto di vista sperimentale.

2.2.5 OLTRE ALLE BIG COMPANIES – **COMPUTER VISION**

Una delle motivazioni principali che spinge un cliente ad orientarsi su una soluzione software piuttosto che un'altra quando deve decidere di acquistare un prodotto è il **prezzo** decrescente del sistema complessivo. I prodotti legati ad IA e machine learning offerti dalle big companies sono solitamente molto costosi e risultano spesso nuovi per le aziende in quanto gli acquirenti si trovano spesso lontani dal mondo della tecnologia. Basta pensare alle soluzioni di computer vision per settori come quello agricolo, tessile o alimentare per comprendere le possibili implicazioni di business e il valore aggiunto che queste soluzioni potrebbero fornire in questi campi. Una delle barriere più grandi da superare per poter sfruttare appieno i sistemi di computer vision senza

pregiudizi è sicuramente l'accuratezza complessiva del software impiegato. Il problema principale nell'impiegare queste tecnologie risiede nell'affidabilità variabile dei sistemi e la diffidenza nell'adottare soluzioni (anche molto costose) da parte di diversi clienti.
Per agevolare l'introduzione di queste tecnologie e accelerare il processo di adesione anche da parte di più utenti, sono nate diverse librerie e framework **open source** che supportano l'intero ecosistema della **computer vision**. Attualmente la libreria di computer vision open source più famosa è chiamata **OpenCV** ed è utilizzabile gratuitamente sui sistemi di Windows, Android, Linux e OS X. OpenCV (Open Source Computer Vision Library) è la libreria open source che può essere considerata il punto di riferimento gratuito per tutti coloro che desiderano sperimentare la produzione di applicazioni di computer vision. Al suo interno troviamo migliaia di algoritmi che sono stati ottimizzati e aggiornati nel corso del tempo che permettono di rilevare oggetti, volti e persino di classificare le azioni compiute dagli umani all'interno di video.

LUMINOTH

Tool-Kit completamente Open-Source scritto in Python che si appoggia a TensorFlow e Sonnet. È in grado di effettuare **object detection** combinando modelli **pre-addestrati** e lo sfruttamento di dataset molto grandi come **Coco** e **Pascal**.
Questo strumento consente al cliente o allo sviluppatore di accedere a differenti modelli di object detection già addestrati in grado di ottenere risultati ragionevolmente precisi in poco tempo.
Attualmente sono disponibili 2 differenti checkpoint scaricabili con LUMINOTH:
SSD + Pascal: Molto veloce ma difficoltà a riconoscere piccoli oggetti quindi poco preciso.
Faster R-CNN + Coco dataset: Più preciso ma ovviamente più lento.
Tali checkpoint possono essere già utilizzati per effettuare **previsioni** caricando le proprie immagini e ottenendo il risultato del riconoscimento direttamente in forma di documento JSON (possibile anche eseguire su GPU remote tramite WEB-Interface).
Per usare Luminoth è sufficiente spostarsi nella cartella cove sono presenti le immagini da analizzare e utilizzando la riga di comando lanciare
lumi predict immagine_da_riconoscere.jpg

Di default il toolkit utilizza l'algoritmo Faster R-CNN allenato con le immagini del dataset **Coco**
Lo stesso comando può essere lanciato per eseguire l'analisi frame-by-frame di un video.

Se l'utente non desidera utilizzare i dataset di default occorrerà munirsi di un buon numero di immagini abbinate alle corrispondenti label e coordinate rettangolari. Luminoth richiede che in fase di addestramento della rete sia specificata l'unità computazionale con la quale eseguire il training (come una GPU). Una volta iniziato il training è possibile interromperlo e riprendere in un secondo momento grazie ad un **checkpoint.** Questi punti di salvataggio si preoccuperanno di salvare i **pesi correnti** della rete ogni 600 secondi e consentiranno al modello di raggiungere la convergenza del training anche successivamente. Affiancando alla fase di training un tool come **TensorBoard** è possibile vedere visivamente i progressi del modello e accorgersi di eventuali perdite.[32]

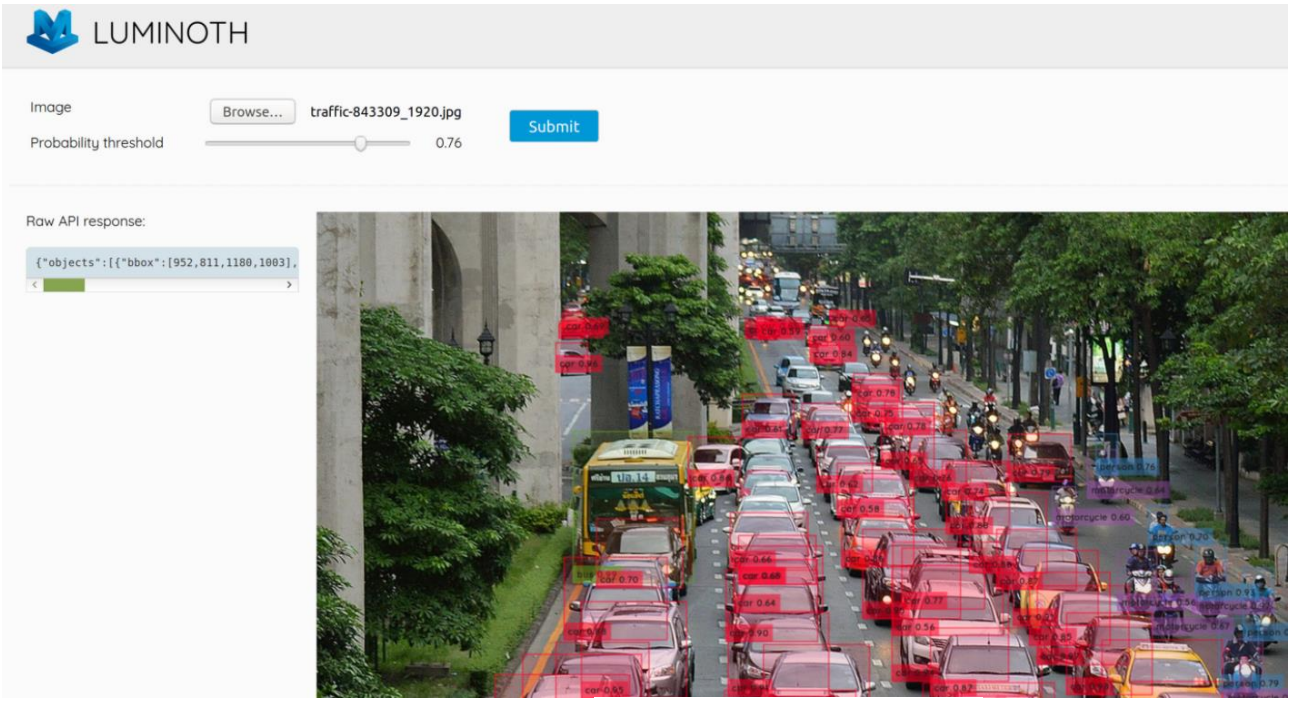


Figura 47 Interfaccia Web di Luminoth

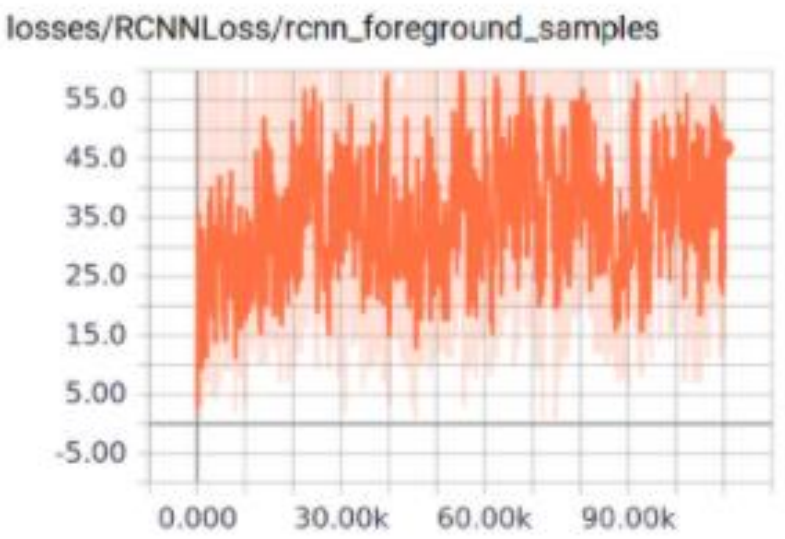


Figura 48 TensorBoard visual training

Sebbene alcuni clienti preferiscano adottare soluzioni Open Source per la risoluzione dei propri problemi, altri preferiscono rivolgersi ad aziende di **piccole dimensioni** con l'obiettivo di ricevere una consulenza mirata e allo stesso tempo **minimizzare** i **costi** elevati imposti dalle grandi aziende. L'international Data Corporation (**IDC**), uno dei principali fornitori globali di informazioni e analisi di mercato IT, ha presentato una lista di diverse aziende in ambito computer vision che possiedono un fatturato inferiore ai 100 milioni di dollari l'anno ma che riescono ad avere un enorme impatto sul mercato CV oggi. Tali aziende verranno citate nella successiva sezione con l'obiettivo di sensibilizzare il lettore nella comprensione di differenti realtà rispetto a quelle già presentate in precedenza.

Algolux

Azienda di Montreal con 40 impiegati fondata nel 2014. Si occupa di machine learning e computer vision, in particolare di auto a guida autonoma e robot. Hanno l'obiettivo di migliorare la precisione dei sistemi di visione artificiale sfruttando la loro **deep neural network** proprietaria. Il loro software è in grado di processare direttamente i dati **grezzi** (a differenza delle reti comuni che richiedono file di tipo JPEG per analizzare immagini) utilizzando meno energia e tempo di una rete tradizionale. L'utilizzo di dati grezzi è in grado di allenare la loro IA anche in condizioni sfavorevoli garantendo ottimi risultati di precisione anche in presenza di dataset non troppo puliti o con massiccia presenza di rumore.[33]

In questo esempio si può notare come il software Algolux sia in grado di captare molte più informazioni del sistema di bordo impiegato sulle Tesla di ultima generazione.[34]

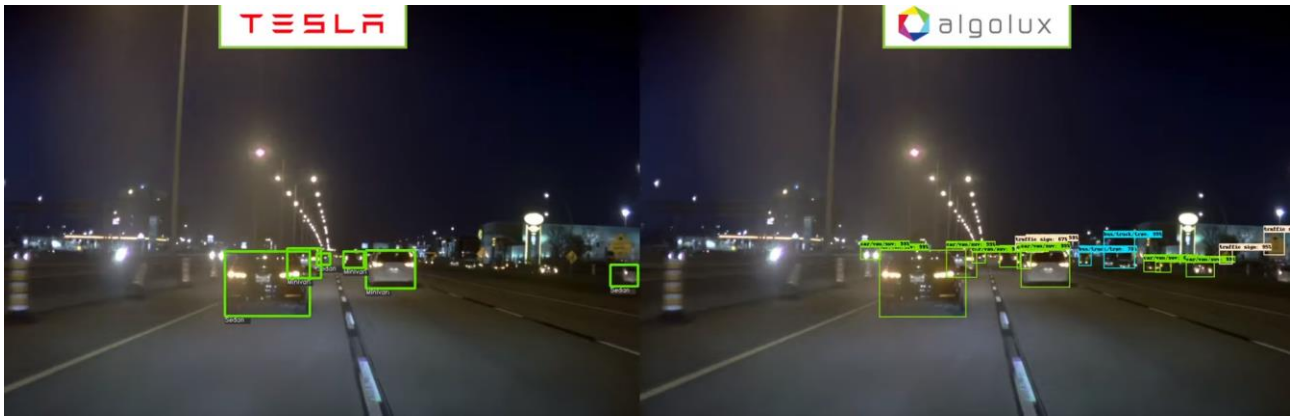


Figura 49 Algolux vs Tesla self-driving

Deep vision AI

Azienda di Costa Mesa con 20 impiegati fondata nel 2015. Si occupa di riconoscimento facciale e di oggetti con l'obiettivo di impiegare i propri prodotti in ambito sicurezza, marketing o advertising. L'azienda è specializzata nel riconoscimento di **veicoli** e ha la peculiare capacità di adottare modelli in grado di riconoscere anno, modello e tipologia di ogni veicolo da qualsiasi angolazione.[35]

Figura 50 Deep Vision AI riconoscimento veicoli

La piattaforma offerta da Deep Vision AI è integrabile all'interno di qualsiasi sistema hardware ed è per questo motivo molto indicata in scenari come le **smart city.** Oltre a questi possibili casi d'uso, Deep Vision AI si è concentrata nello sviluppare software in grado di riconoscere oggetti simili tra loro e risulta pertanto all'avanguardia anche nei campi industriali che richiedono flessibilità e notevoli capacità di adattamento.

Sighthound

Azienda di Winter Park con 50 dipendenti fondata nel 2013. Si occupa di sicurezza ed è specializzata nell'applicazione di filtri Blur sui volti delle persone, veicoli, oggetti e tutto ciò che si

ritenga essere protetto per questioni di **privacy**. Sono un'azienda che lavora a stretto contatto con l'università centrale della Florida e per questo motivo sfrutta la competenza e la preparazione di esperti che lavorano nel settore della computer vision. Sighthoud offre ai propri clienti la possibilità di selezionare differenti **classi** di oggetti predefinite (come veicoli o volti) al fine di

Figura 51 Sighthound blur faces all'opera in un negozio

applicare maschere di **oscuramento** solo sugli elementi scelti. Interessante anche la possibilità di ottenere la licenza d'uso per i propri software ad un prezzo ridotto per tutti i clienti in grado di fornire dataset anonimi che possano migliorare il modello ottenuto dopo la fase di training.[36]

ViSenze

Azienda di Singapore con 93 impiegati fondata nel 2012. Si occupa **ricerca visiva attiva** e di migliorare l'esperienza utente al fine di aumentare i guadagni provenienti dalle vendite nel mondo retail. I sistemi ViSenze consentono ai propri clienti di utilizzare una fotografia per confrontare un prodotto all'interno di un negozio e offrono agli acquirenti la possibilità di richiedere ed ordinare la taglia o il colore di capi o oggetti non presenti in magazzino. L'intelligenza artificiale di ViSenze è stata allenata quasi interamente su **prodotti reali** e data set forniti direttamente dai partner con la quale l'azienda collabora consentendo training di qualità e un elevato grado di precisione.

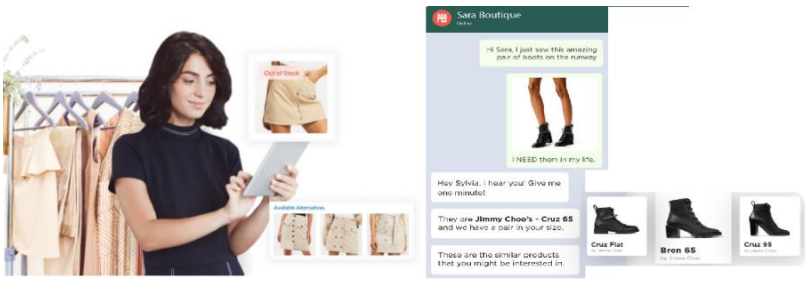


Figura 52 ViSenze immagini simili ed esperienza utente

Ogni oggetto o capo di abbigliamento in esame prevede la possibilità di analizzare anche differenti

alternative simili a quanto richiesto dal cliente in modo da guidarlo in un'esperienza interattiva durante tutta la fase di acquisto.[37]

Umbo CV

Azienda di Taipei con 50 impiegati fondata nel 2014. Si occupa di **videosorveglianza** e sfrutta modelli e IA allenate per riconoscere la forma umana e **comprendere le azioni** e le intenzioni di alcuni soggetti. Le soluzioni offerte da Umbo CV sono plug-and-play e richiedono pochissima configurazione per essere utilizzate. L'azienda spicca per la **precisione** e la **distanza** con la quale le proprie telecamere riescono ad identificare i soggetti in esame.
Le telecamere più avanzate di Umbo CV riescono a captare e ad analizzare persone fino a 50 metri di distanza con una precisione **molto superiore** a quella umana. Il modello di IA impiegato da Umbo CV è talmente generale da poter essere utilizzato in qualsiasi ambiente anche in presenza di zone poco illuminate. Attualmente il numero di falsi positivi rilevati dall'azienda è veramente basso e per tale motivo si è guadagnata una buona fetta di mercato nel settore della sicurezza e dell'analisi video. Alcuni degli ambiti in cui le telecamere di Umbo CV vengono impiegate spaziano dal rilevamento di **intrusioni** all'interno di negozi oppure all'individuazione tempestiva di casi di **aggressione** o **lesione** che avvengono su base giornaliera all'interno dei quartieri dove il tasso di criminalità risulta più elevato della norma.[38]

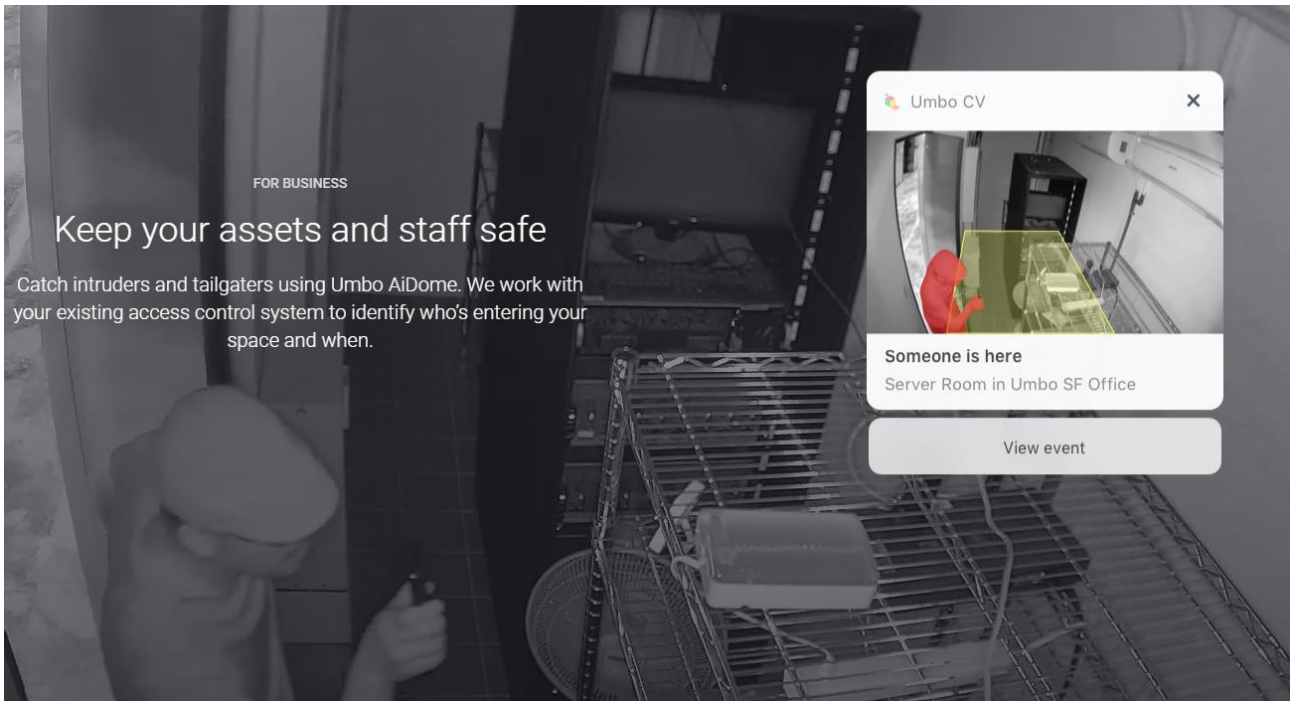


Figura 53 Umbo CV sistema di rilevamento intrusione ad eventi

2.2.6 OLTRE ALLE BIG COMPANIES – **VISUAL INSPECTION**

Landing AI – Landing Lens

Azienda di Palo Alto (California) che nasce con l'obiettivo specifico di supportare i clienti per creare e distribuire efficacemente soluzioni di **visual inspection** in ambito industriale. L'azienda è stata fondata da Andrew Ng, capo fondatore di Google Brain, specializzandosi nell'aiuto all'individuazione di parti e prodotti difettosi tramite l'applicazione di tecniche di Intelligenza Artificiale e Machine Learning.[39]
Landing AI opera nei più disparati settori:
- Manufacturing: Alta specializzazione nel costruire sistemi in grado di riconoscere parti graffiate, scheggiate o difettose.
- Automotive: Alta specializzazione nella categorizzazione e classificazione di difetti per semplificare il lavoro degli ispettori e migliorare il rendimento della produzione.
- Farmaceutico: Alta specializzazione nell'ispezionare microparticelle mediante modelli di intelligenza artificiale che illuminano e rendono più visibili i difetti.
- Telecomunicazioni: Alta specializzazione per connettere aziende e clienti con l'obiettivo di rendere l'esperienza utente più dinamica e coinvolgente.
- Agricoltura: Alta specializzazione nello sviluppo di modelli di IA per la massimizzazione del raccolto e alleggerire la pressione sugli operatori.

La piattaforma sviluppata da Landing AI specializzata nell'effettuare **visual inspection** end-to-end è chiamata **Landing Lens**, un sistema integrato che facilita la gestione, il riconoscimento e la classificazione di difetti da parte dei clienti.
L'obiettivo del progetto Landing Lens è quello di focalizzarsi sul problema industriale causato dalla mancanza di **dataset** specifici per il riconoscimento dei difetti in quanto essi si presentano in maniera sporadica e difficilmente vengono catturati con immagini o fotografie dai clienti.
Dalla ricerca effettuata da Landing AI il 58% dei clienti non è in grado di proseguire con progetti legati all'intelligenza artificiale proprio per la **mancanza di dati** con i quali effettuare i training delle reti.

Statisticamente le immagini di difetti che un cliente possiede sono limitate all'ordine delle decine; secondo la teoria del Machine Learning se solo il 10% dei dati forniti è taggato in maniera errata serve circa il doppio (1.88x) degli stessi dati per mantenere un certo livello di precisione.

Landing Lens è stato sviluppato tenendo in mente proprio il modo con la quale gli ispettori compiono le ispezioni visive: in ambito industriale si tendono ad utilizzare dei veri e propri **libri dei difetti** cartacei che vengono aggiornati ogni volta che un nuovo difetto viene individuato.
Sulla base di questa idea, l'azienda ha progettato una versione **digitale** del libro dei difetti che grazie alle più recenti tecniche di AI è in grado di **supportare** l'operato svolto dagli ispettori.
Un altro problema affrontato dall'azienda riguarda **l'ambiguità** con la quale l'IA dovrebbe essere in grado di riconoscere elementi come difetti: è possibile che due ispettori diano giudizi differenti sul medesimo graffio a causa della propria esperienza.
Landing Lens offre ai clienti degli strumenti ri-addestrabili in un click in grado di definire in **tempo reale** che cosa è difetto e cosa no, avvisando allo stesso tempo per eventuali **ambiguità** o **inconsistenze.**

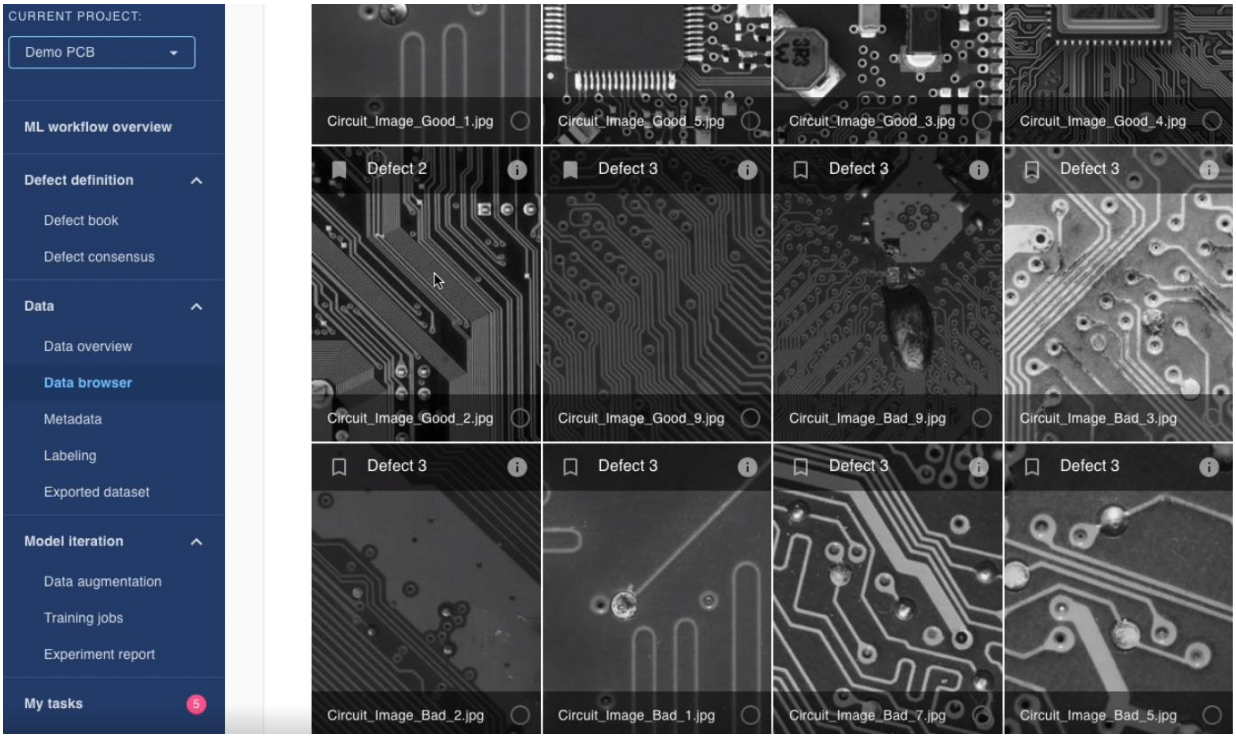


Figura 54 Libro dei difetti Landing Lens

L'architettura del sistema Landing Lens è stata strutturata in 3 macroaree: **Defect/Data management, model Iteration** e **continuous learning**

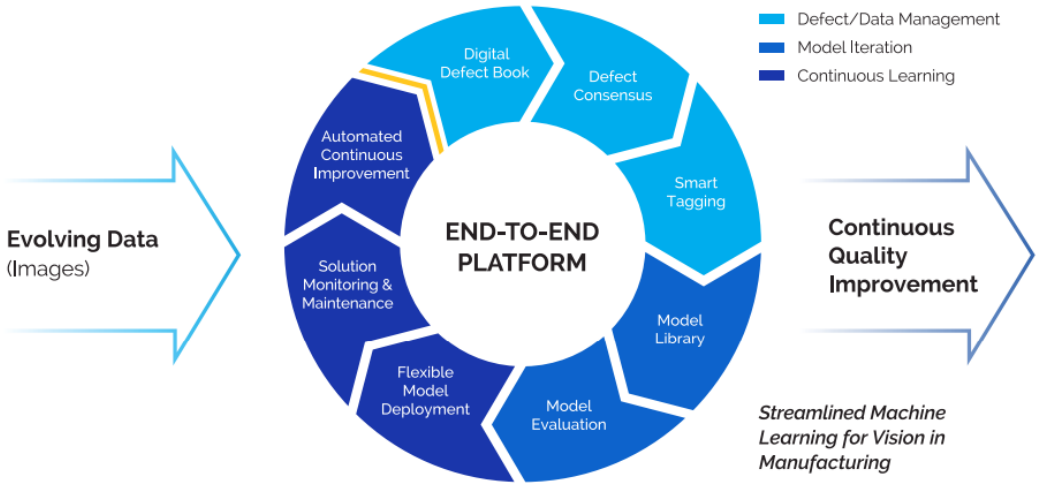


Figura 55 Sistema Landing Lens

La piattaforma è basata sul concetto di **libro dei difetti**, un catalogo contenente la **definizione precisa** di cosa si intende per ciascun difetto rilevato dal cliente. Ogni difetto viene mostrato a schermo ed è caratterizzato da un **tag,** da immagini d'esempio e dalla possibilità di risolvere eventuali **incongruenze** fatte da diversi ispettori in un'apposita tab chiamata "Defect Consensus". Questa utilissima feature offerta dal sistema è in grado di identificare se, nella fase di tag, due o più ispettori hanno identificato immagini simili come difettose oppure no creando una collisione che potrebbe alterare la bontà del dataset. Il Tool è in grado di identificare set di **difetti** che causano più confusione tra gli ispettori: solo dopo una revisione manuale sarà possibile inserire il difetto all'interno del libro virtuale.

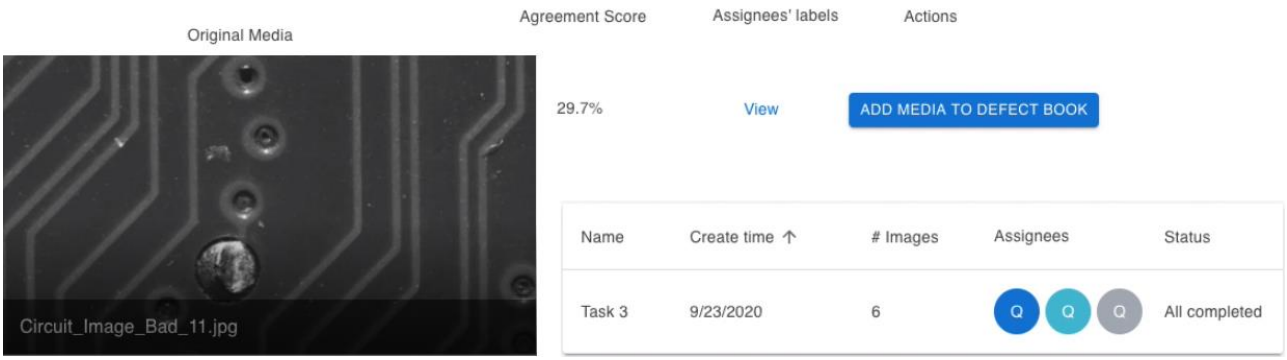


Figura 56 Difetto ambiguo rilevato richiede azione Manuale

La fase di acquisizione dei dati è molto semplice da effettuare: il sistema può ricevere come input sia immagini caricate manualmente dal cliente che interi file CSV (possibilità di acquisire dati anche via API).

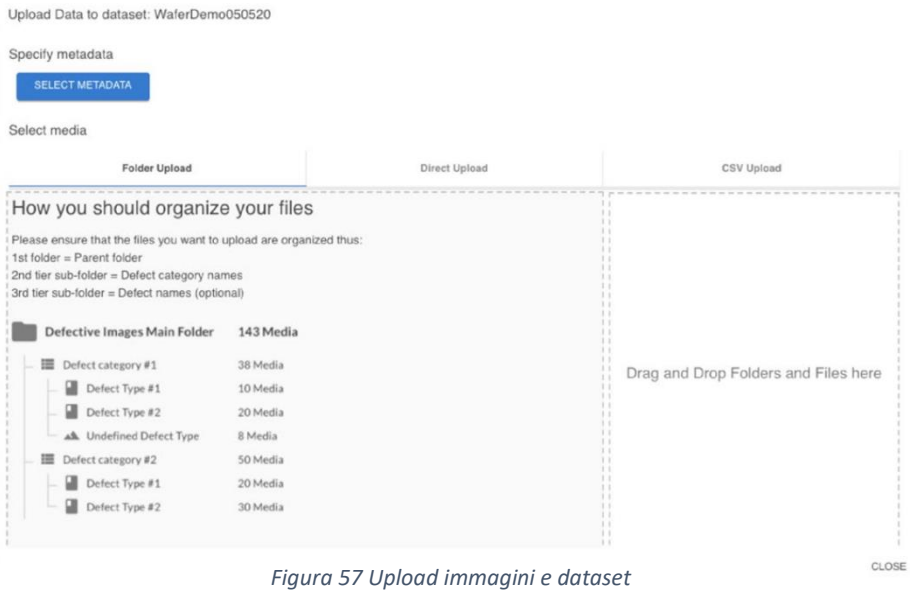


Figura 57 Upload immagini e dataset

Tutte le immagini possono essere annotate grazie al **labeling tool** che, analogamente a quanto visto con Google, consente al cliente di **classificare** ed effettuare il tag di tutti i difetti con l'ausilio dei classici bounding box. Il sistema di labeling è strutturato in maniera tale da avere un supervisore e diversi sottoposti: quando un sottoposto esperto nell'individuare difetti di un certo tipo riceve una **notifica**, egli potrà accedere al labeling tool e gestire la richiesta taggando correttamente il difetto oppure approvando il pezzo.

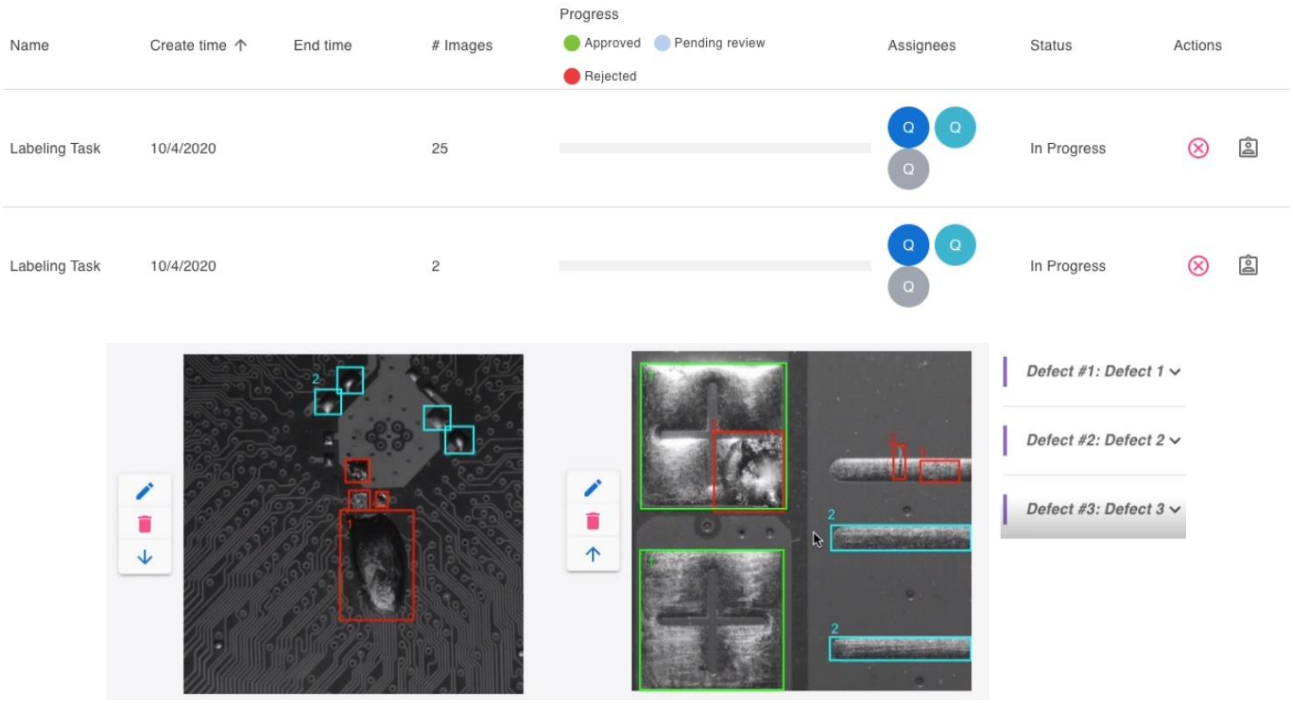


Figura 58 Tag e Labeling delle immagini e sistema di approvazione label per esperti

Dopo aver effettuato tag e labeling delle immagini di input, Landing Lens adotta le più moderne **tecniche** di image augmentation per rendere i dataset di dimensioni tali da favorire il processo di apprendimento e il training della rete. Prima di effettuare l'image augmentation il cliente è in grado di visualizzare alcune immagini che sarebbero prodotte dall'intero processo.
Tutti i dataset possono essere esportati, importati e sfruttano **architetture pre-addestrate** in grado di effettuare rapidamente **object-detection, semantic segmentation e image classification**.
Dopo la fase di training il modello viene salvato e può essere **confrontato** con i precedenti modelli in termini di precisione in modo da poter effettuare sostituzioni con le precedenti versioni meno performanti.[40]

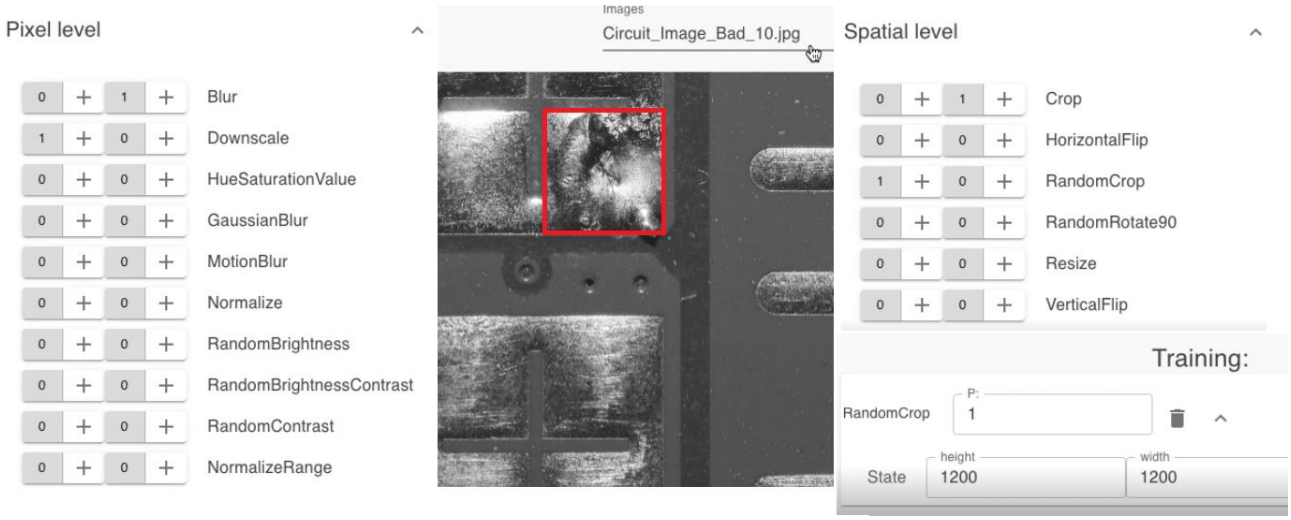


Figura 59 Image Augmentation Lending Lens

Cogniac

I sistemi offerti da Cogniac combinano le più recenti ricerche in ambito IA per rendere la visione artificiale più semplice, accurata e scalabile. Come altre aziende, anche Cogniac ha posto particolare attenzione nella progettazione di una **semplice interfaccia utente** adatta anche agli utenti non tecnici e allo sviluppo di sistemi semi-automatizzati che ricercano costantemente l'architettura e le variazioni di configurazione ottimali per ottenere il miglior risultato possibile a runtime.[41]

Alcuni casi d'uso affrontati da Cogniac:

- **Ispezioni ferroviarie automatizzate**: In America, una delle principali compagnie ferroviarie gestisce più di 32.000 miglia di binari che possono **logorarsi o espandersi a causa del calore** nel tempo e che richiedono costanti **ispezioni** visive da parte di operatori specializzati. Oltre alle ispezioni dei binari, sono di fondamentale importanza anche le ispezioni dei treni che richiedono il controllo di 24 ruote per carrozza con una media di **2400** per treno.

Senza ispezioni accurate e coerenti i treni rischiano di deragliare e causare ingenti danni a persone e costi per le aziende. Così come già visto in altri scenari, anche qui a causa del **basso numero di campioni** per addestrare i modelli i sistemi di **visione artificiale** risultano poco performanti e approssimativi. Cogniac grazie alla tecnica dell'**hyper parameter optimization** è stata in grado di sviluppare una IA auto-allenante che con sole **100** immagini iniziali è in grado di riconoscere **efficacemente i difetti**.

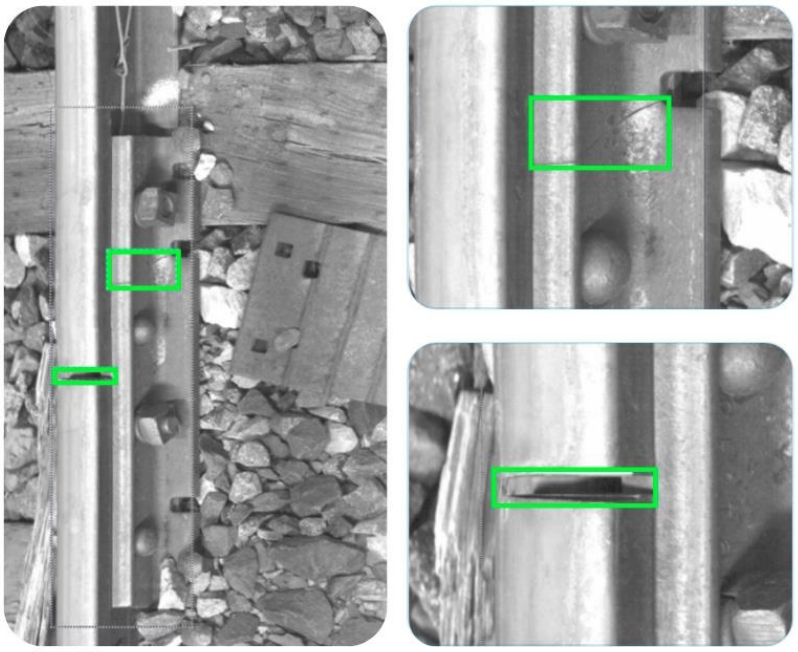


Figura 60 Rilevamento di fratture nella linea ferroviaria

- **Ispezioni industriali per il legno**: Ispezionare i **tronchi** di legno può essere un lavoro lungo e faticoso che richiede esperienza, tempo e soprattutto può essere soggetto ad errori umani. Ogni singolo mulino può ricevere tra i 75 e i 500 carichi di tronchi al giorno che vengono di norma ispezionati da ispettori professionisti. Mentre alcune aziende prendono in considerazione i sistemi LIDAR e di scansione del carico con radiazioni gamma, altre si sono affidate a Cogniac per l'analisi di questi prodotti.

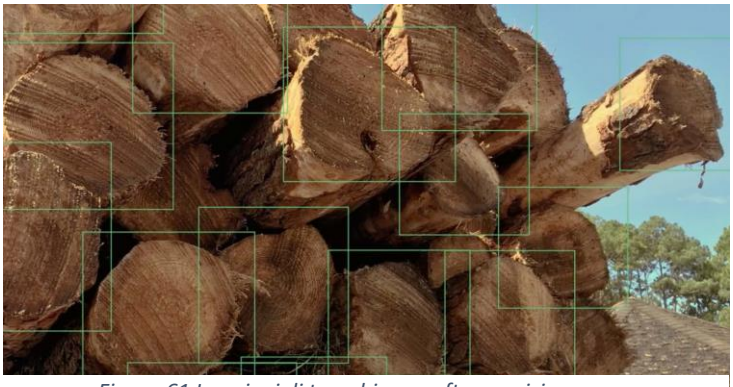


Figura 61 Ispezioni di tronchi con software visivo

- **Ispezioni industriali per automotive**: Oggi, moderne fabbriche sono in grado di produrre una portiera di auto ogni 4 secondi, un tempo molto più ridotto della velocità con la quale un essere umano può compiere un'ispezione. Utilizzando la tecnica dell'hyper parameter optimization, Cogniac è stata in grado di creare una IA che agisce come un data scientist virtuale.
Essa prova rapidamente diverse combinazioni di oltre 60 parametri indipendenti per ottenere la **massima precisione** possibile.

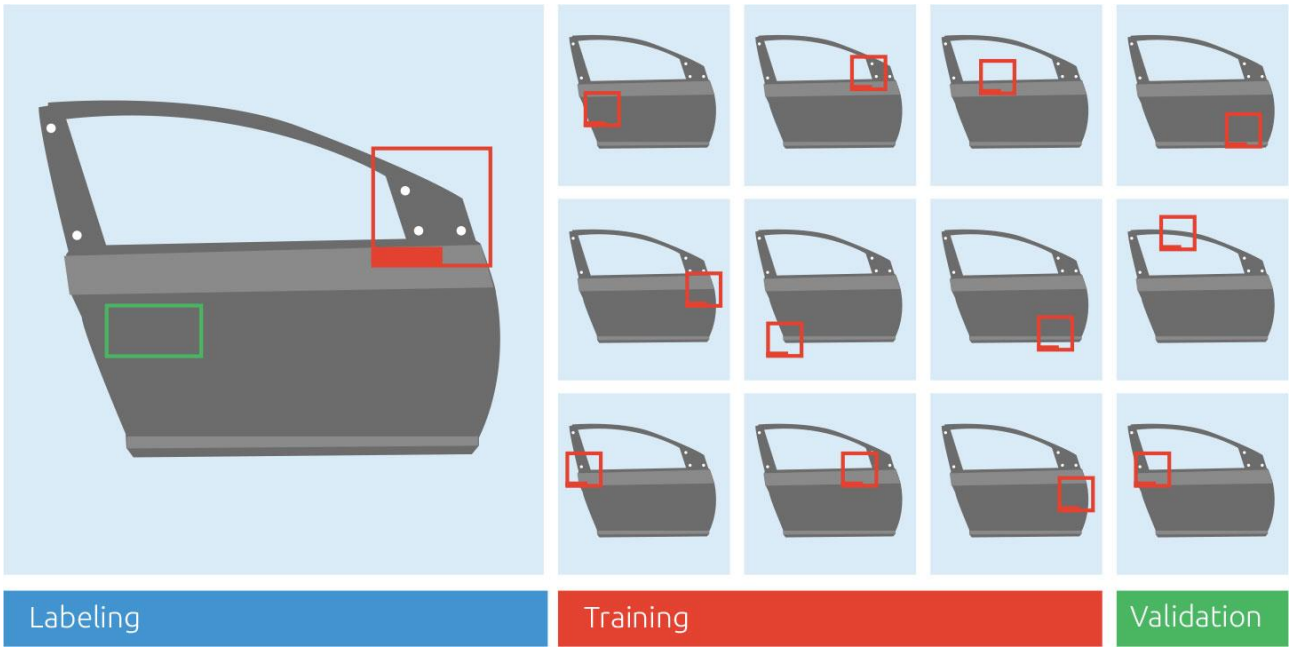


Figura 62 Visual inspection automotive Cogniac

PEKAT Vision

Si tratta di un software sviluppato dalla medesima azienda per effettuare **ispezioni visive industriali** su molteplici superfici.
Il prodotto astrae efficacemente ogni problematica e riconosce difetti su campioni di varia natura come legno, viti o lastre metalliche. [42]

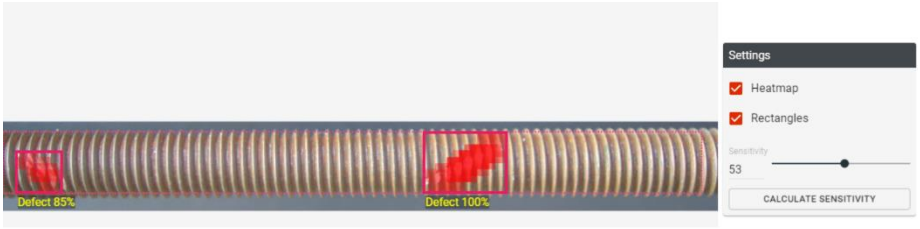


Figura 63 Pekat vision riconoscimento difetti su una vite

PEKAT Vision è in grado di rilevare **difetti mai visti prima** dal modello oppure lavorare con approccio **supervisionato** per cercare un difetto **specifico** come un graffio o la presenza di ruggine. Questi risultati sono ottenuti grazie ad un algoritmo **proprietario** di apprendimento **focalizzato** che va oltre al deep learning e fornisce risultati **in tempo reale** dello stato di un oggetto dal punto di vista dei difetti. Una delle novità introdotte da Pekat Vision consiste nella possibilità di **combinare** il lavoro svolto dalle **IA** con **regole interne** in grado di migliorare ulteriormente il risultato ottenuto. Se ad esempio il modello è stato allenato a riconoscere anomalie sul **legno**, il sistema PEKAT consente all'utente di indicare **la dimensione** per la quale queste anomalie sono da considerare come difetto oppure no. Rispetto ad un sistema di computer vision tradizionale i risultati sono molto più precisi e consentono ispezioni di qualità e precisione.
Il sistema viene addestrato mediante esempi **corretti** e privi di errori in quantità **molto limitata** rispetto ad un classico algoritmo di computer vision. È completamente indipendente da condizioni esterne come **luminosità** o **rotazione** ed è composto da diversi moduli che velocizzano l'addestramento del modello come:

- Modulo di **preprocessing**: per effettuare trasformazioni sulle immagini come ritagli, ridimensionamenti e normalizzazioni dello sfondo. Consente all'utente di **concentrarsi** su una singola parte di un'immagine ed effettuare ritagli a catena su migliaia di campioni.
- Modulo di **ispezione**: Consente l'intreccio di codice di scripting e consente una modifica della **sensibilità** con la quale il sistema rileverà i difetti.
- Modulo sulle **statistiche**: calcolano i successi delle applicazioni, la matrice di confusione e le metriche utili per eventuali addestramenti futuri.

L'intero sistema è stato concepito come un complesso **workflow** di operazioni effettuabili in cascata. La lista di operazioni che possono essere eseguite è veramente molto ampia e consente all'utente di scegliere il **modulo** alla quale egli è più interessato.

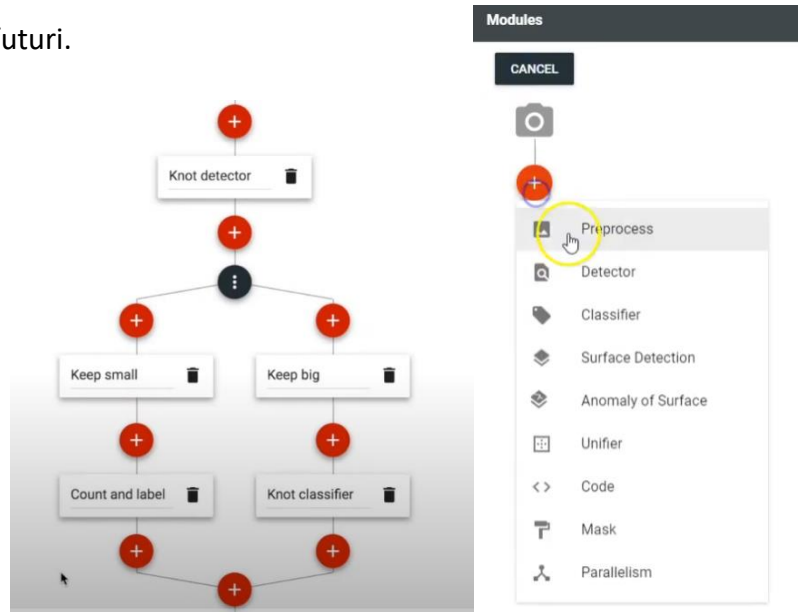


Figura 64 Sistema a workflow modulare

Una volta effettuata la scelta, i vari moduli possono essere **combinati** fino al raggiungimento del sistema completo. Qui si può notare

come l'utente abbia inserito un modulo che istruisce il sistema di **classificare come difetto** solamente parti che hanno una grandezza **inferiore o superiore** ad una certa dimensione.

La possibilità di aggiungere delle **regole** e integrare il sistema con l'**intelligenza** artificiale garantisce risultati molto migliori di quelli visti nei precedenti sistemi analizzati durante questa tesi. L'approccio ibrido risulta quindi una delle migliori soluzioni in ambito aziendale.

Qui si vede come l'utente è in grado di impostare una soglia > 230 pixel che indica al sistema un pezzo **difettoso a priori** e quindi da scartare:

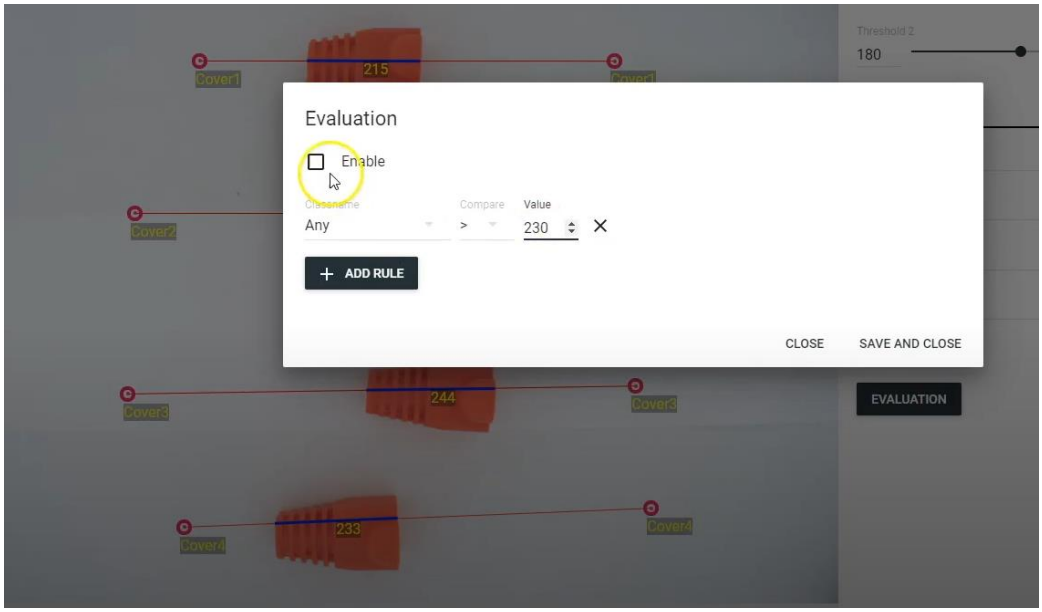


Figura 65 Aggiunta manuale di regole

Con appena **40 immagini taggate** il software ottiene risultati **ottimi** dal punto di vista del riconoscimento. Rispetto ad un sistema di computer vision tradizionale, il sistema PEKAT offre al cliente la possibilità di annotare difetti non solo con il classico strumento di selezione rettangolare ma anche con meccanismi di selezione **manuale** (magari utilizzando una tavoletta grafica).

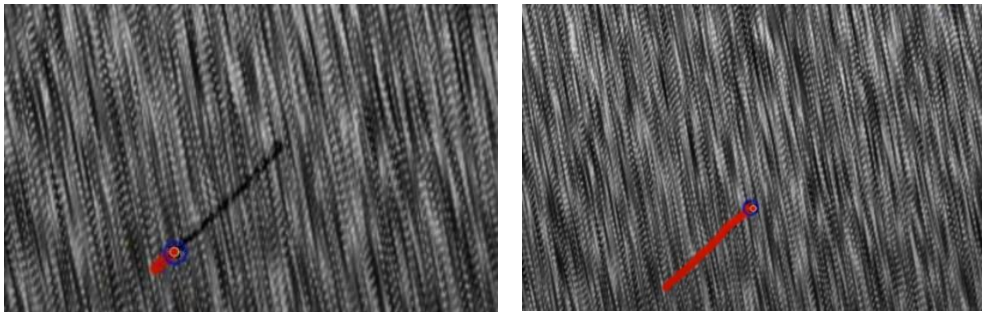


Figura 66 Annotazioni manuali e non solo bounding boxes

Un'altra interessantissima funzionalità di **PEKAT** consiste nella possibilità di effettuare l'upload di un video (magari proveniente dalla catena produttiva) e utilizzarlo col fine di **addestrare un modello** che riconosca la presenza di un certo pezzo. Il software consente di taggare certe porzioni di video e di apprendere le caratteristiche di un oggetto che transita, ad esempio, su un nastro:

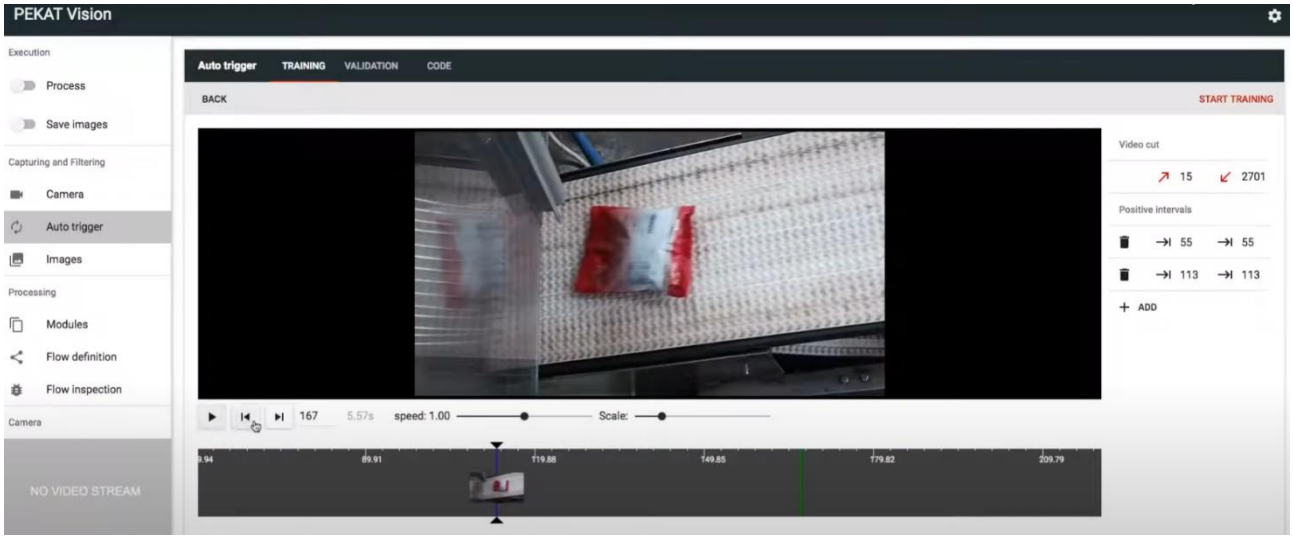


Figura 67 Annotazioni video PEKAT vision

2.3 **Confronto** tra le **grandi aziende**

Da quanto emerso dall'analisi effettuata in sezione 2.2, i vari player che si occupano di computer vision hanno sviluppato diverse soluzioni che privilegiano un approccio a modello generico in grado di adattarsi al problema del singolo cliente. La tendenza di aziende più piccole è invece quella di **verticalizzarsi** su una tecnologia in particolare ed offrire IA allenate con lo scopo di risolvere una specifica classe di problemi.

All'interno di questa parte verranno presentati dati e analisi mediante l'ausilio del lavoro svolto da *Forrester* che ha presentato un'interessante ricerca sulle aziende leader nel settore della Computer Vision.

Forrester offre ai leader mondiali del settore dei consumatori **consulenze** e supporto con lo scopo di informare e far emergere criticità o punti di forza di aziende, tecnologie e piani di business. Saranno prese in considerazione solamente le aziende presentate in questa tesi quindi Amazon (AWS), Google (AutoML), Microsoft (Azure) e IBM (IBM Cloud).[43]

Le analisi effettuate da Forrester si sono basate su 10 differenti criteri e 2 ore di briefing effettuate con ogni fornitore valutato.

1. Dati: Capacità di etichettare, proteggere e annotare i dati durante le fasi di pre-processing e

dopo il periodo di addestramento del sistema.

2. Capacità: Quanto l'azienda è in grado di supportare le operazioni più semplici effettuate sulle immagini e video.

3. Modelli pre-allenati: Quanto sono moderni e performanti i modelli pre-allenati che vengono forniti al cliente prima della modifica specifica in fase di utilizzo.

4. Sviluppo: Quanto l'azienda è in grado di sfruttare tecniche come il transfer learning per semplificare il riconoscimento e il lavoro svolto dalle reti neurali.

5. Distribuzione: Quanto l'azienda è in grado di semplificare l'utilizzo di modelli di visione artificiale da parte dei clienti e quanto è in grado di distribuire i propri prodotti su dispositivi di qualsiasi tipo.

6. Soluzioni: Quanto l'azienda è in grado di costruire modelli scalabili e gerarchizzabili.

7. Facilità d'uso: Quanto l'azienda è in grado di fornire interfacce semplici per gli utenti e cosa pensano gli utenti di quanto la piattaforma offre loro.

8. Visione: Quanto performano gli algoritmi e i sistemi richiesti dal cliente e se soddisfano i suoi requisiti.

9. RoadMap: Aspettativa di quanto l'azienda migliorerà i propri prodotti, adozioni sul mercato da parte dei clienti nei prossimi 12 mesi e capacità di sviluppare soluzioni migliori nel lungo termine.

10. Approccio al mercato: Quanto d'impatto è l'esposizione del prodotto offerto a livello di mercato.

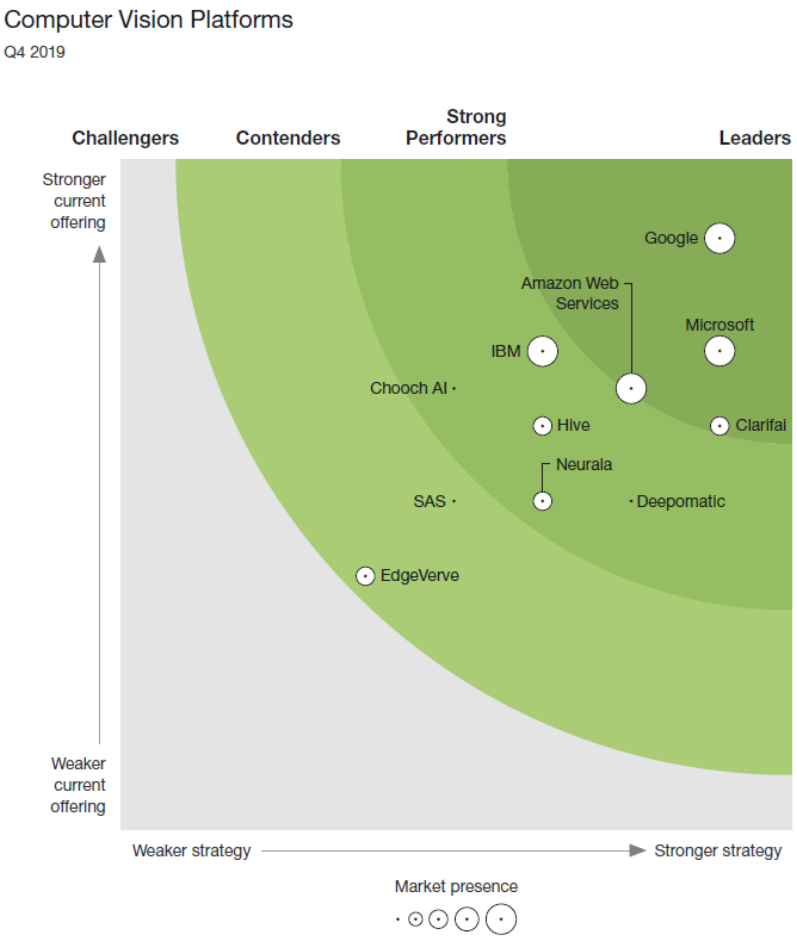


Figura 68 Presenza di mercato delle principali aziende di Computer Vision

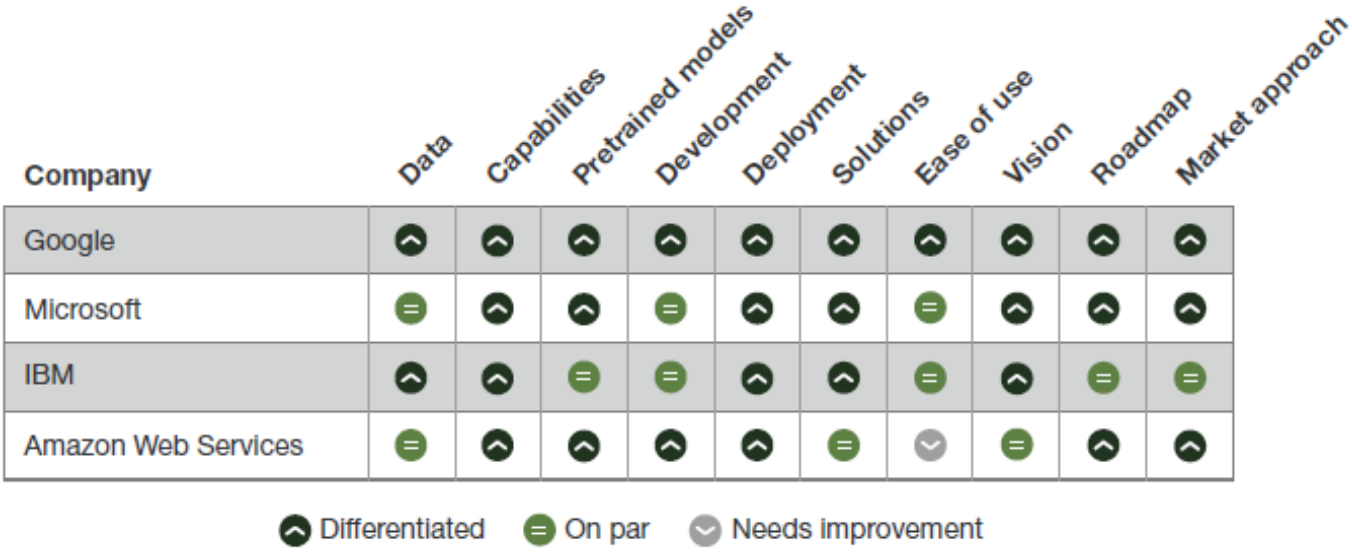


Figura 69 Analisi Forrester delle aziende trattate nella tesi

Google: Prodotti analizzati **Vision** e **Video intelligence (API e AutoML)**

L'azienda di riferimento di ogni cliente che desideri approcciarsi al mondo della computer vision; offre soluzioni free to use per studenti e complessi modelli addestrabili per le aziende. La facilità con la quale è possibile addestrare una rete, il supporto post-vendita e i costi ridotti fanno di Google la scelta migliore per la maggior parte delle aziende. Gli unici difetti dell'azienda risiedono nel non offrire tools specifici per il **riconoscimento facciale** a causa del fatto che considerano poco "etica" la pratica. I punti di forza dell'azienda risiedono nella capacità di costruire potenti modelli e nell'elevata scalabilità e applicabilità sul mercato.

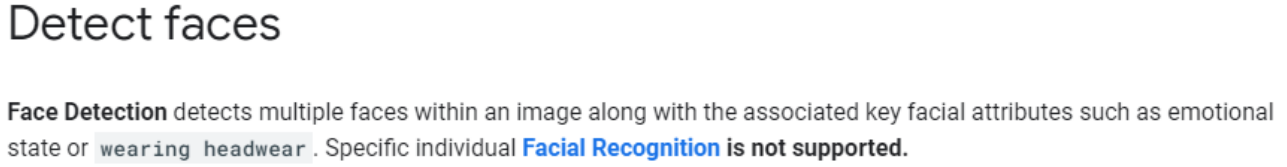


Figura 70 Rilevamento dei volti di Google non supportato

Microsoft: Prodotti analizzati **Azure Cognitive Services**

Azienda molto solida che grazie all'enorme network di clienti e alle proprie conoscenze risulta spesso il punto di partenza di moltissimi imprenditori che vogliono affacciarsi al mondo della

computer vision. I clienti sono soddisfatti dei prodotti Microsoft nel complesso ma richiedono alcuni miglioramenti nel riconoscimento di brand o nella generale semplicità nell'uso dei sistemi proposti.

Amazon: Prodotti analizzati **Amazon Rekognition**

Amazon possiede uno dei comparti di riconoscimento facciale più avanzati tra le aziende analizzate, la sua capacità di riconoscere volti, brand e oggetti unito alla possibilità di effettuare rapide ricerche con l'output del riconoscimento la inseriscono tra le migliori aziende a disposizione per i clienti. L'unica problematica è la propria difficoltà d'uso per gli utenti **non sviluppatori** e quindi la necessità di assumere, da parte delle aziende, figure specializzate che siano in grado di effettuare con maneggevolezza chiamate API e utilizzare notebook Python.

IBM: Prodotti analizzati **IBM PowerAI Vision, IBM Watson Visual Recognition**

IBM ha fatto diversi passi in avanti in ambito computer vision e offre ai propri clienti ottimi spunti per sviluppare e integrare la CV nei sistemi di riconoscimento visivo. Le recensioni dei clienti sono positive in termini di praticità d'uso e affidabilità mentre alcuni hanno sollevato perplessità sull'effettiva possibilità di sviluppare modelli personalizzabili. Nel complesso risulta un'azienda di riferimento con diverse soluzioni integrabili tra loro e un comparto CV all'avanguardia.[44]

2.3.1 Grandi aziende e **visual recognition** a confronto

Solitamente le grandi aziende mettono a disposizione sul proprio portale alcune demo e test dei prodotti e servizi che offrono ai propri clienti. Con l'obiettivo di effettuare un confronto coerente

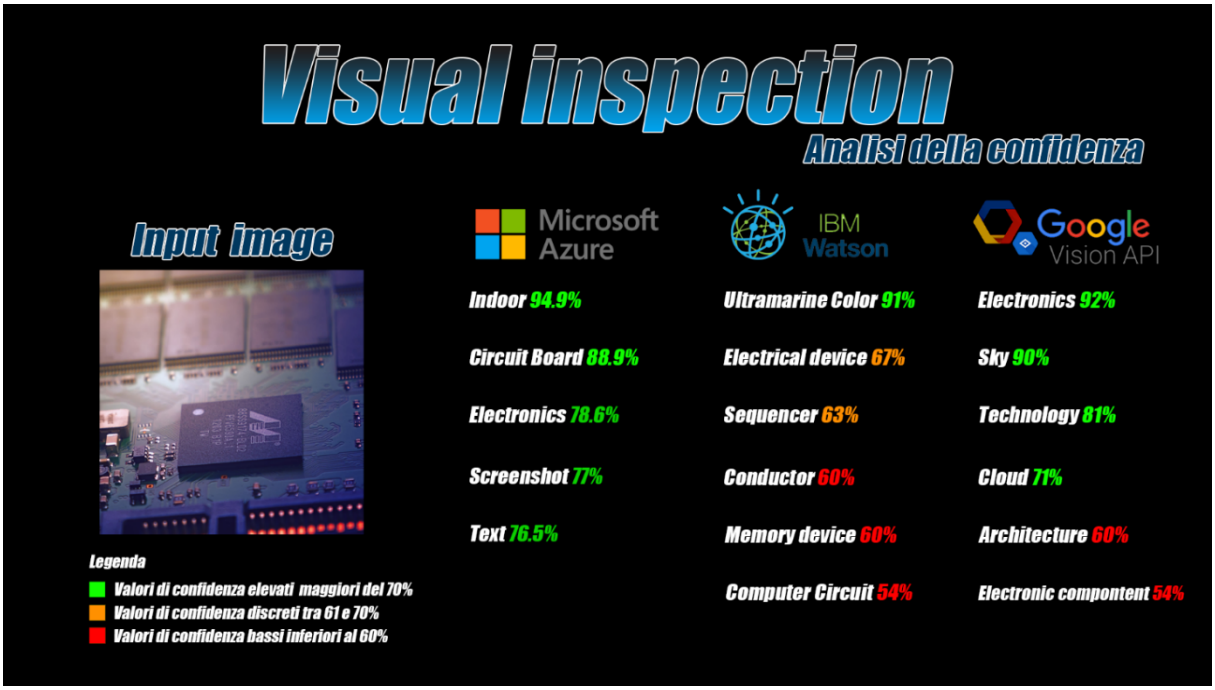


Figura 71 Confronto sul parametro di confidenza nelle demo di object recognition di Microsoft, Ibm e Google

tra le varie aziende che mettono a disposizione una demo per il **riconoscimento** di **immagini** verrà utilizzata come fotografia quella di una porzione di una scheda elettronica. I risultati seguenti sono stati ottenuti utilizzando il tool messo a disposizione da Microsoft Azure Vision, Google API e IBM visual recognition. In questa analisi è stata considerata solamente la **confidenza** con la quale il software di ogni demo è stato in grado di rilevare qualsiasi elemento all'interno dell'immagine. NON sono stati analizzati all'interno della fotografia i risultati a livello **semantico.** Da quello che è emerso in fase di sperimentazione, il software di **Microsoft Azure Vision** si è calibrato su valori di confidenza parecchio elevati nel complesso ed è stato in grado di riconoscere la presenza di un componente elettronico montato all'interno di una scheda. Purtroppo, ha confuso l'immagine con uno screenshot e ha assegnato come punteggio più elevato il tag "indoor" che non c'entra troppo con un componente elettronico.[45]

Il lavoro svolto da **IBM Watson visual service** si è concentrato principalmente sui **colori** (Ultramarine color 91%) ma è stato comunque in grado di distinguere e rilevare un generico "dispositivo elettronico" con un valore di confidenza pari al 67%. Interessante anche il fatto di essere riuscito a rilevare proprio l'oggetto "dispositivo di memoria" con un 60%.[46]

Google Vision API adotta un approccio più generico: sebbene riconosca la presenza di "Elettronica" con un 92% purtroppo confonde il chip con il "cielo" (90%) e riconosce un dispositivo elettronico solamente con una precisione pari al 54%.[47]

CAPITOLO 3 – VALUTAZIONE COMPARATIVA DI SOFTWARE DI COMPUTER VISION (INDUSTRIALI E OPEN SOURCE) APPLICATI AD UN PROBLEMA DI VISUAL INSPECTION

In questo capitolo si cercherà di analizzare il servizio offerto da **Google Cloud Vision, Microsoft, IBM visual inspection** e **YOLO** (open source) per la **classificazione** e il **riconoscimento** di immagini.
L'obiettivo di questo capitolo è valutare concretamente le soluzioni offerte da aziende e open source cercando di adottare differenti approcci sia nella generazione dei data set che nell'analisi dei modelli e dei risultati ottenuti. Per tutti sistemi in esame sarà sottoposto il medesimo problema da analizzare: si desidera riuscire a classificare correttamente alcuni **dadi a 20 facce** stampati con una **stampante 3D** che presentano alcuni **difetti**.
Alcuni dadi risultano **graffiati** mentre altri presentano un **buco** sulla faccia numero "1".
Il sistema deve essere in grado di riconoscere e classificare quando il dado **non presenta imperfezioni** oppure se presenta uno dei due **problemi** indicati precedentemente.
Il **buco** oppure il **graffio** potrà essere presente in **qualsiasi punto** del dado preso in esame.

Saranno presi in considerazione **solo riconoscimenti con valore maggiore del 50%** di confidenza.

Figura 72 Un dado corretto

Figura 73 Un dado graffiato

Figura 74 Un dado bucato

3.1 Google Cloud Vision

Per Google Cloud Vision sono stati eseguiti tre esperimenti diversi a partire da due differenti dataset ottenuti effettuando **scatti** o **video** ai dadi con uno smartphone.

ESPERIMENTO 1 CLASSIFICAZIONE DI IMMAGINI BRUTE-FORCE

Dataset

Ho preso 4 dadi in **perfette condizioni**, 4 dadi che presentavano un **buco** in corrispondenza della faccia numero "1" e 4 dadi con il medesimo problema però sottoforma di **graffio**. Ho eseguito video della durata di **30 secondi circa** per ciascun dado, facendo ruotare un piatto di cartone sopra ad un cuscinetto metallico per ottenere le diverse angolazioni necessarie.

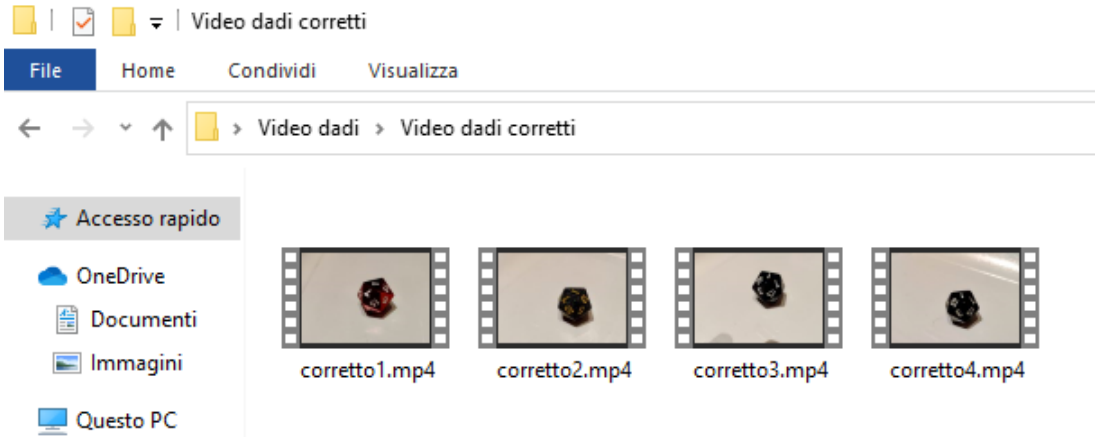


Figura 75 Video per il dataset catturati con lo smartphone

Dopo aver ottenuto i video (sempre registrati con il mio smartphone), ho trasformato i secondi di riprese utilizzando il tool **ffmpeg** in una serie di immagini.

Il seguente comando, preso in input un file **.mp4**, è in grado di convertirlo a livello di **frames** in tantissime immagini **.jpg**.

ffmpeg *-i corretto1.mp4 corretto1%03d.jpg*

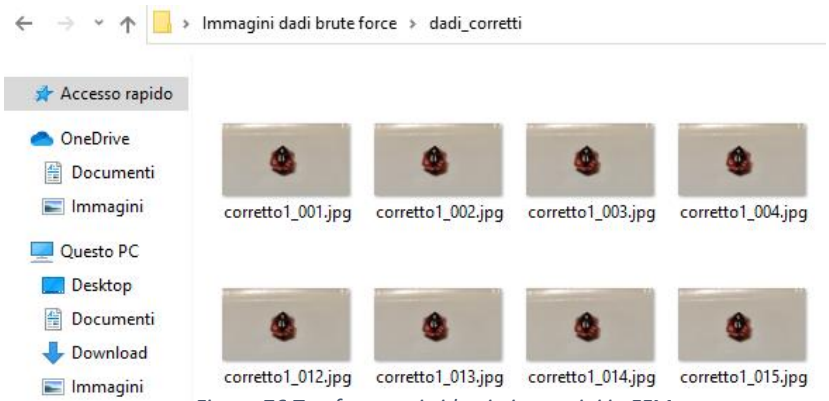


Figura 76 Trasformare i video in immagini in FFMpeg

La stessa cosa è stata fatta anche per i **dadi bucati** e per i dadi **graffiati**.

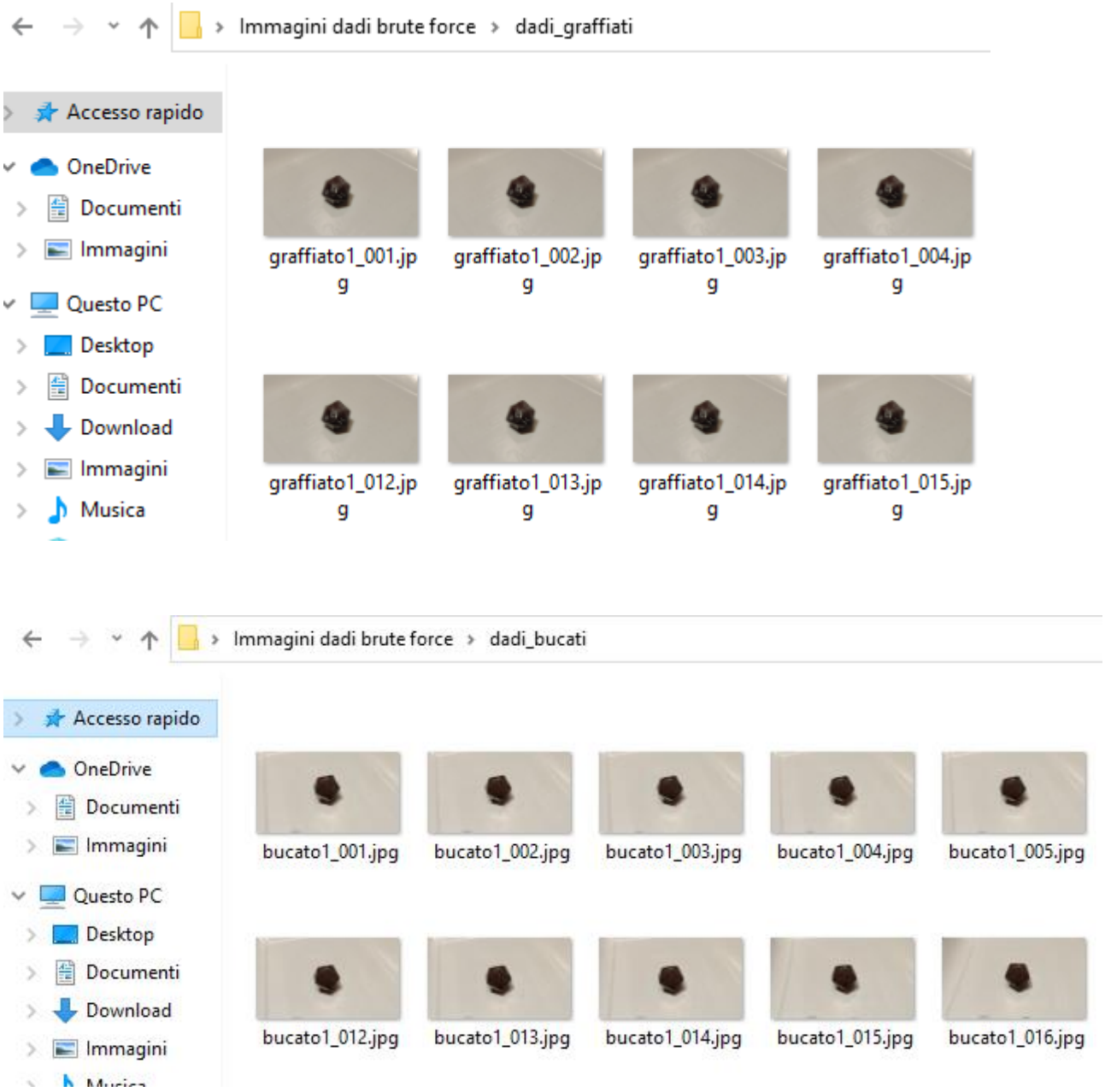


Figura 77 Dataset generati dai video suddivisi in cartelle

Durante la generazione di questo data-set **non è stata fatta particolare attenzione** ad alcun tipo di condizione ambientale come la **luce,** eventuali **ombre** oppure la **posizione** dello smartphone durante la ripresa del video.

Da come si nota in questa foto alcune riprese avevano anche molto rumore (**ombra**).

Figura 78 Presenza di rumore nel primo dataset

Importare le immagini

Una volta preparato il dataset ho creato un nuovo progetto all'interno di Google Cloud Platform Vision. Il sistema consente all'utente di testare il prodotto per una durata di 90 giorni (**250 euro** di credito utilizzabile).

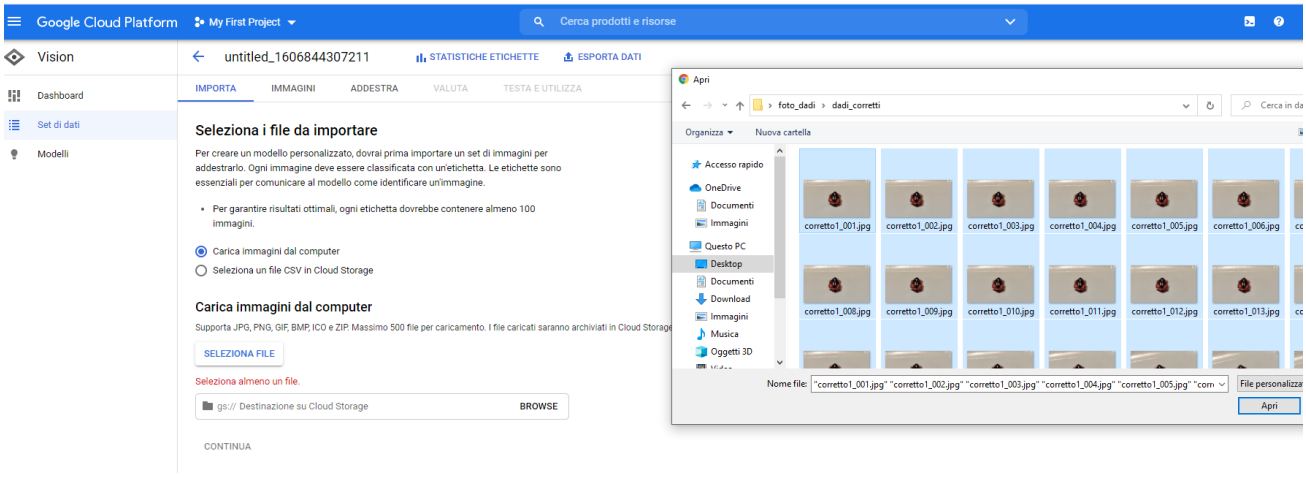


Figura 79 Importare i file su google cloud platform

Importare le immagini su Google Cloud può avvenire mediante il caricamento delle stesse dal computer oppure selezionando un file **CSV** dentro al Cloud Storage.
Il file deve esseere strutturato nel seguente modo:
<TRAIN> <VALIDATON> <TEST> , link_Immagine_Google_Cloud, eventuali etichette e tag

Nel mio caso ho deciso di caricare le immagini direttamente dal mio computer di casa creando un **Bucket** dentro al Cloud Google.

I **Bucket** sono particolari strutture dentro alla quale l'utente può caricare i propri files strutturandoli sottoforma di cartelle.

Nel mio caso ho creato **3 bucket** (uno per ogni differente esperimento, in questo caso chiamato dati_dadi) strutturati in **3 differenti cartelle**, una per ogni categoria di immagini (dadi_corretti, dadi_graffiati e dadi_bucati).

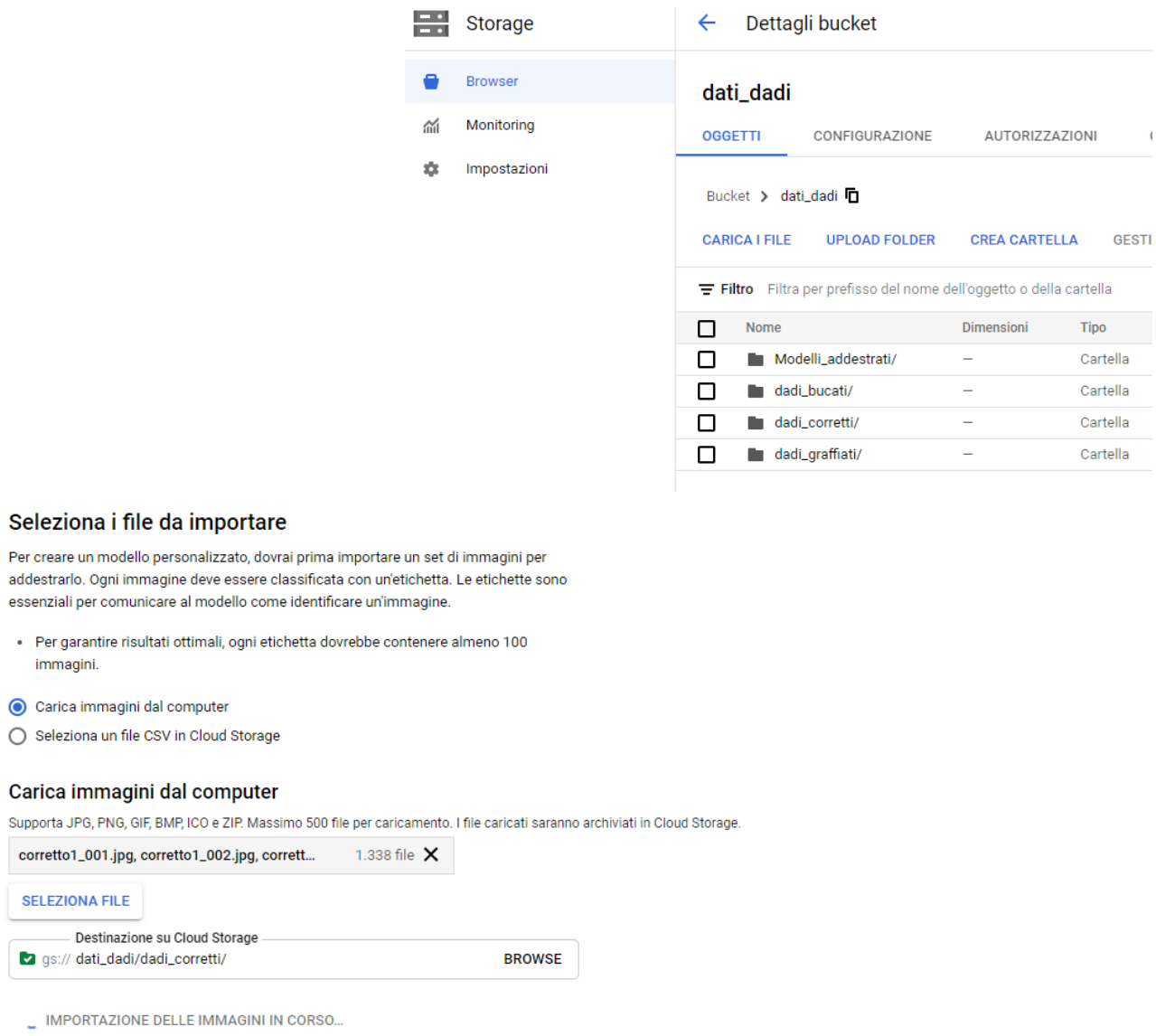


Figura 80 Creazione di un bucket e import delle immagini

Tagging

Una volta terminata l'importazione delle immagini all'interno del sistema occorre effettuare il **tagging** delle stesse. Prima di farlo ho creato 3 differenti etichette con i nomi

dado_corretto

dado_bucato

dado_graffiato

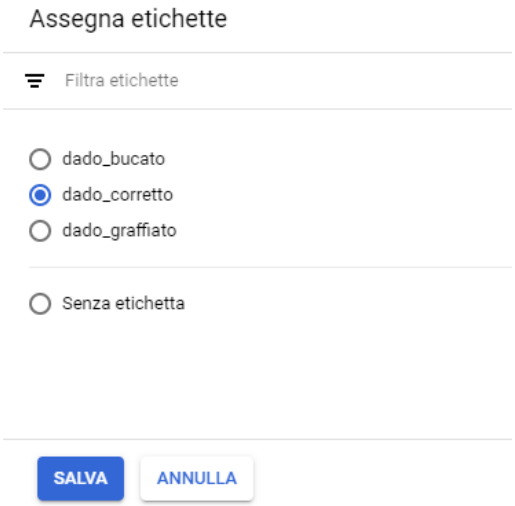


Figura 81 Definizione delle classi

Per ogni immagine presente all'interno del sistema di gestione dell'input è quindi necessaria un'azione manuale che istruisca il sistema sulla **natura** della fotografia esaminata. E' possibile assegnare fino a 50 etichette alla volta selezionando la spunta "seleziona tutto".
Il sistema, una volta etichettate correttamente le immagini le inserisce all'interno di un folder dividendo ogni elemento da quelli con un'etichetta differente.

Figura 82 Assegnamento delle etichette

Una volta terminato l'assegnamento delle etichette ecco la situazione complessiva dei dati relativi al primo esperimento:

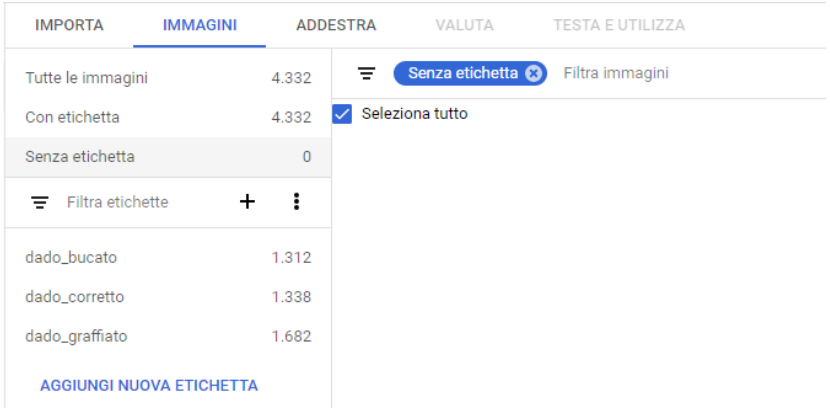


Figura 83 Dati raccolti per il primo esperimento

Da come si nota nell'immagine, 30 secondi di video per 12 dadi complessivi, hanno generato un dataset di **4332 immagini**.

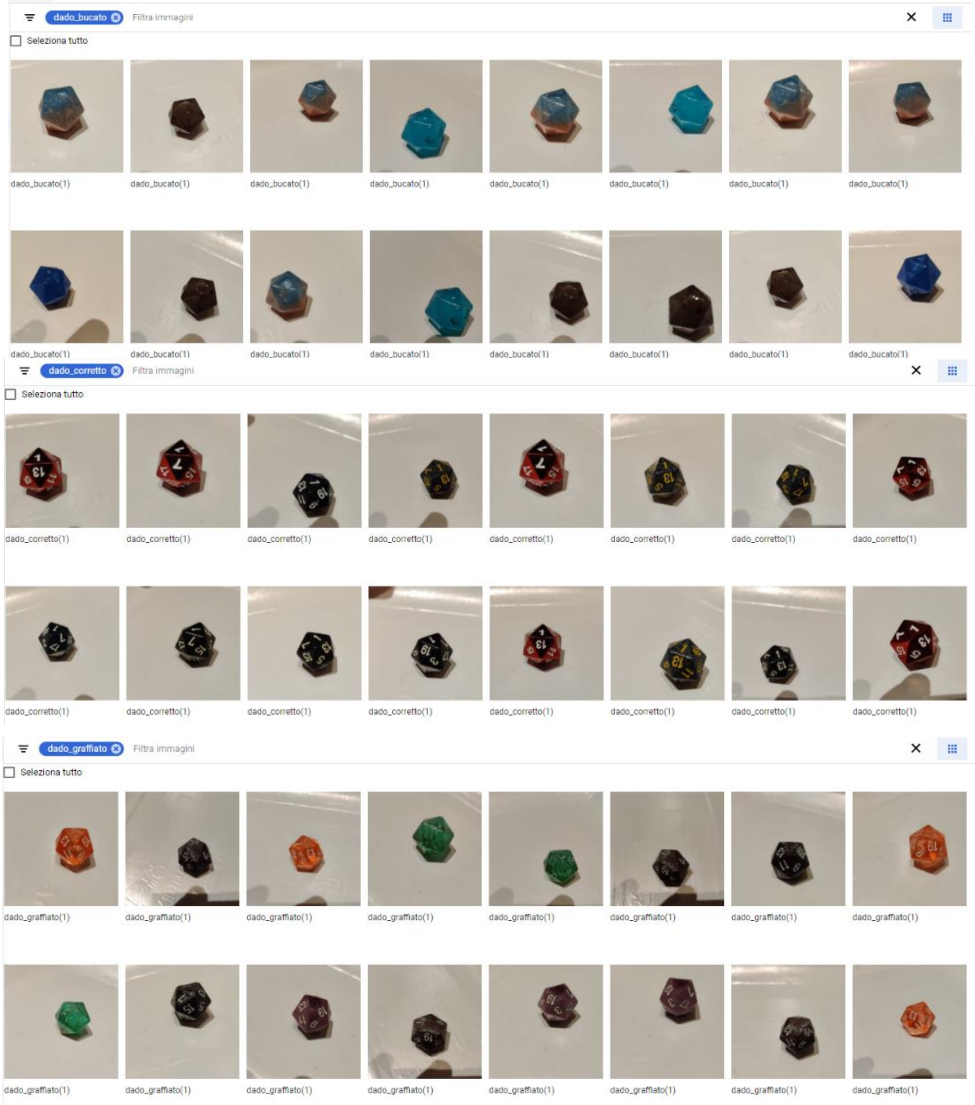


Figura 84 Primo dataset (con rumore) dadi corretti, bucati e graffiati

Addestramento

Ora che il dataset è completo ed è stato correttamente importato e taggato è necessario avviare l'**addestramento** della rete. Google consiglia di avere almeno **10 immagini per ogni etichetta** per evitare errori di precisione e richiamo approssimativo.

Le immagini date in input vengono **automaticamente suddivise** in 3 set:

-Addestramento (usate per addestrare la rete)

-Convalida (Usate per convalidare la rete)

-Test (Usate per testare la rete dopo l'addestramento)

Questi numeri rappresentano come Google ha diviso i miei dati in input in questi set.

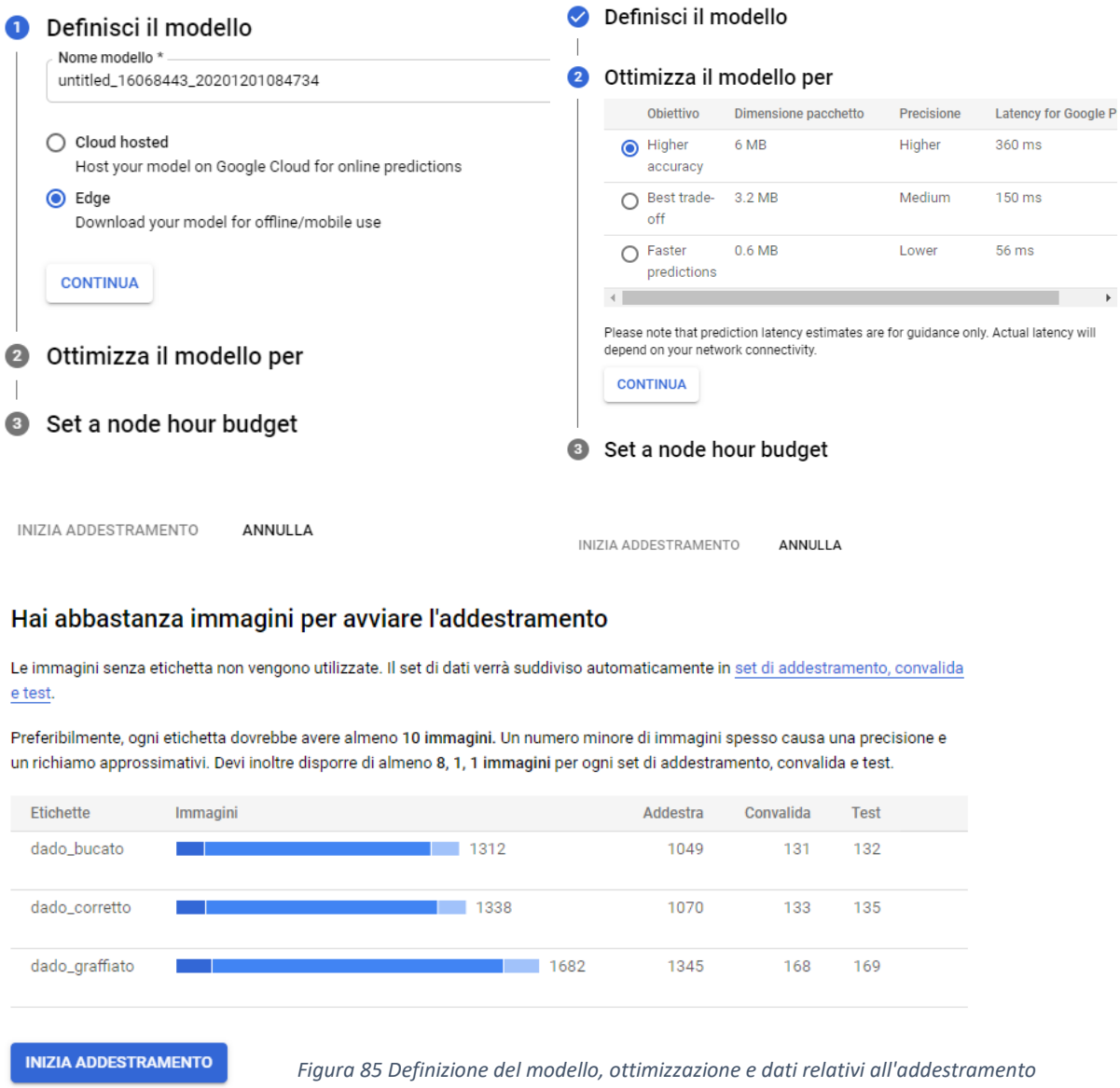


Figura 85 Definizione del modello, ottimizzazione e dati relativi all'addestramento

Per effettuare il training di questa rete sono stati utilizzati **3.15$** (1 Nodo/Ora) e il tempo di addestramento è stato di **1 ora e 30 minuti** circa.

Per la fase di test Google offre due differenti metodi con la quale poter testare il modello finale. Il primo consiste nel poter caricare in maniera automatica le immagini in un'interfaccia WEB. Il secondo invece consente di **esportare** il modello e **scaricarlo** per poterlo utilizzare anche **offline** su un qualsiasi dispostivo (anche mobile).

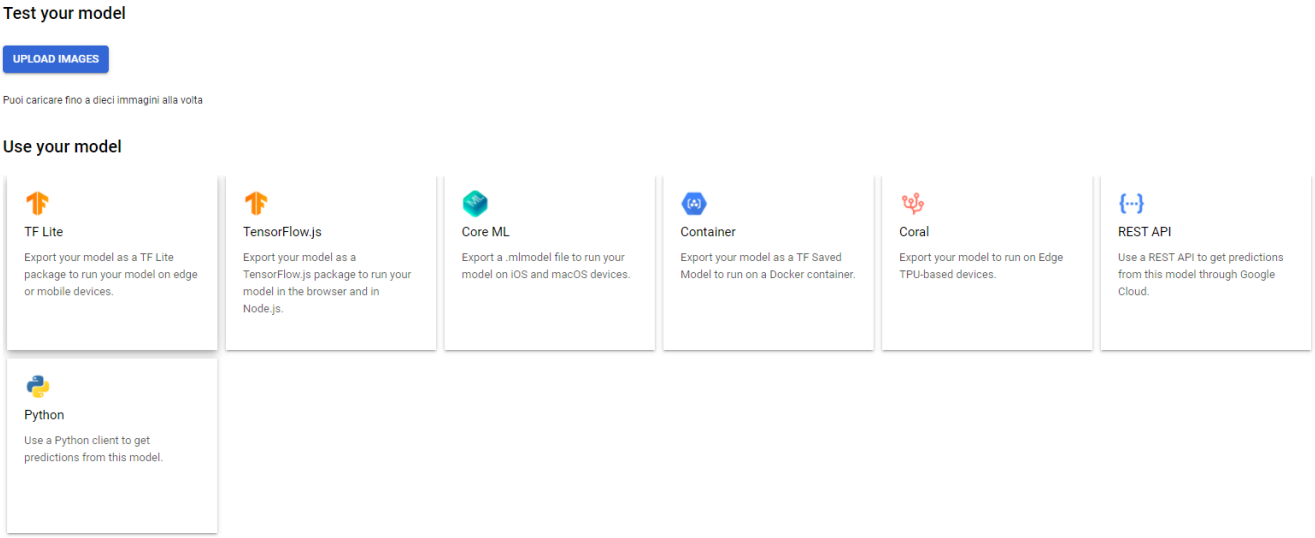


Figura 86 deployment e utilizzo del modello

Test e Risultati

I risultati della rete dopo l'addestramento sono stati i seguenti:

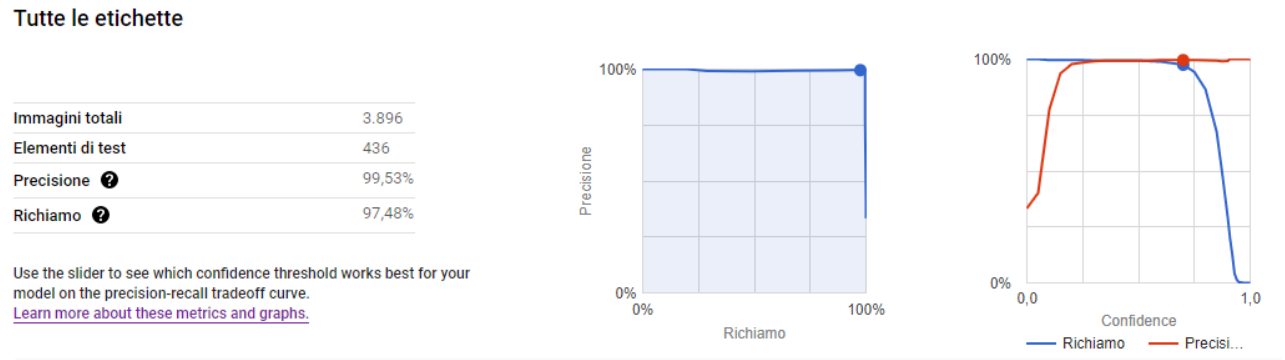
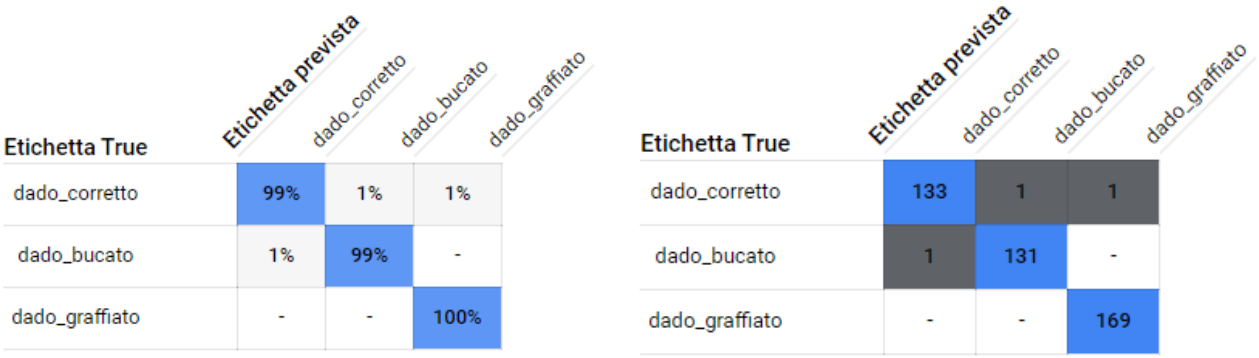


Figura 87 Precisione e richiamo del primo esperimento

Il sistema ha preso, delle 4332 immagini, un campione di **3896** fotografie per effettuare la fase di **training** e un campione di 436 immagini per la fase di **test.** Google effettua una divisione in **train-test** con il **rapporto 90%-10%**.



Etichetta True	dado_corretto	dado_bucato	dado_graffiato
dado_corretto	99%	1%	1%
dado_bucato	1%	99%	-
dado_graffiato	-	-	100%

Etichetta True	dado_corretto	dado_bucato	dado_graffiato
dado_corretto	133	1	1
dado_bucato	1	131	-
dado_graffiato	-	-	169

Figura 88 Confusion matrix esperimento 1

Da quello che si può notare dalla **Matrice di confusione** e dai valori di **precision e recall,** il sistema sembra performare veramente in maniera **eccellente** e ha risultati che superano il 99% di accuratezza.

1) Su **135** immagini di **dadi_corretto**, il sistema ha confuso la classificazione solamente **2 volte**.

2) Su **132** immagini di **dadi_bucato**, il sistema ha confuso la classificazione solamente **1 volta**.

3) Su **169** immagini di **dado_graffiato** il sistema non ha mai confuso la classificazione.

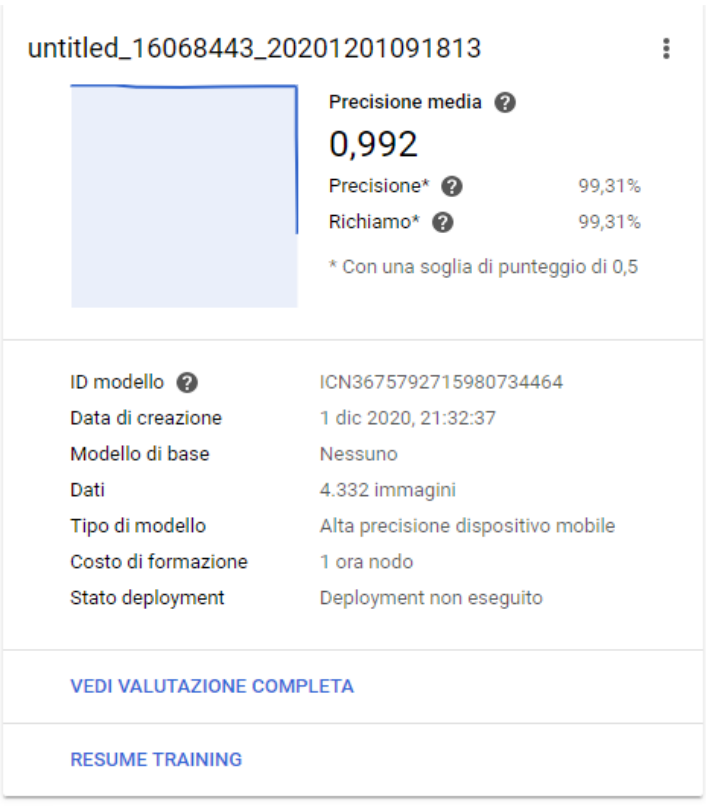


Figura 89 Precisione media e modello primo esperimento

Ecco alcuni risultati ottenuti dal modello dando in input **dadi mai visti dal sistema**:

Analisi di **dadi corretti** interpretati dal sistema:

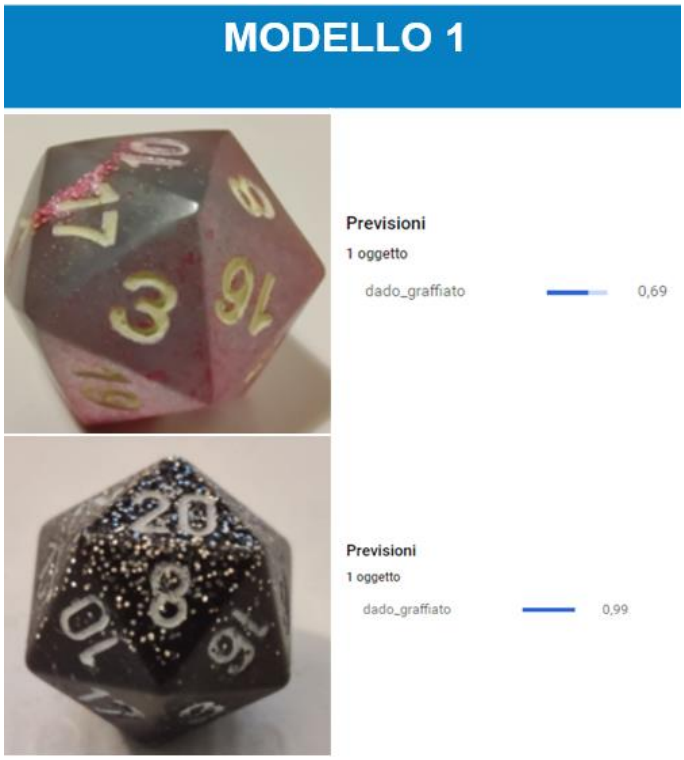


Figura 90 Dati corretti interpretati dal sistema come graffiati

Analisi di **dadi graffiati** interpretati dal sistema:

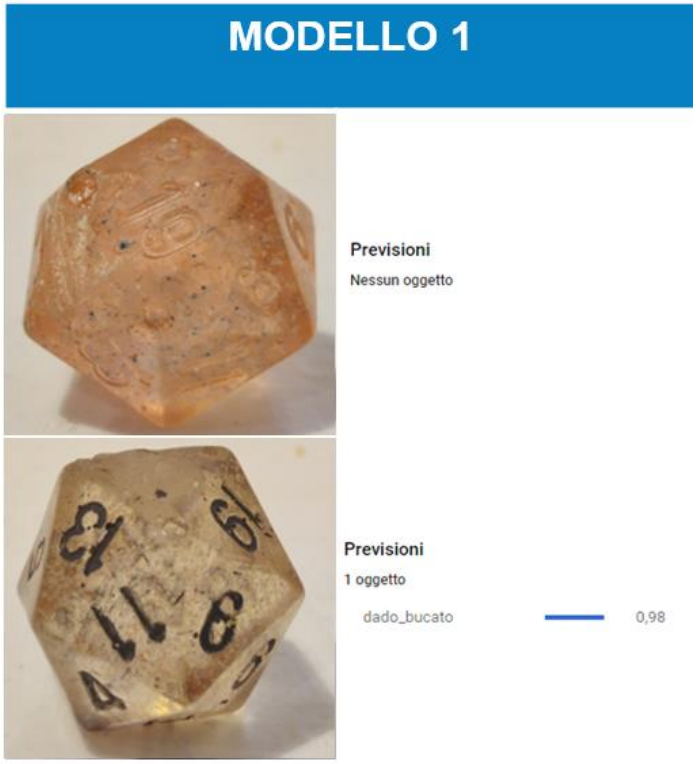


Figura 91 Dati graffiati interpretati dal sistema

Analisi di **dadi bucati** interpretati dal sistema:

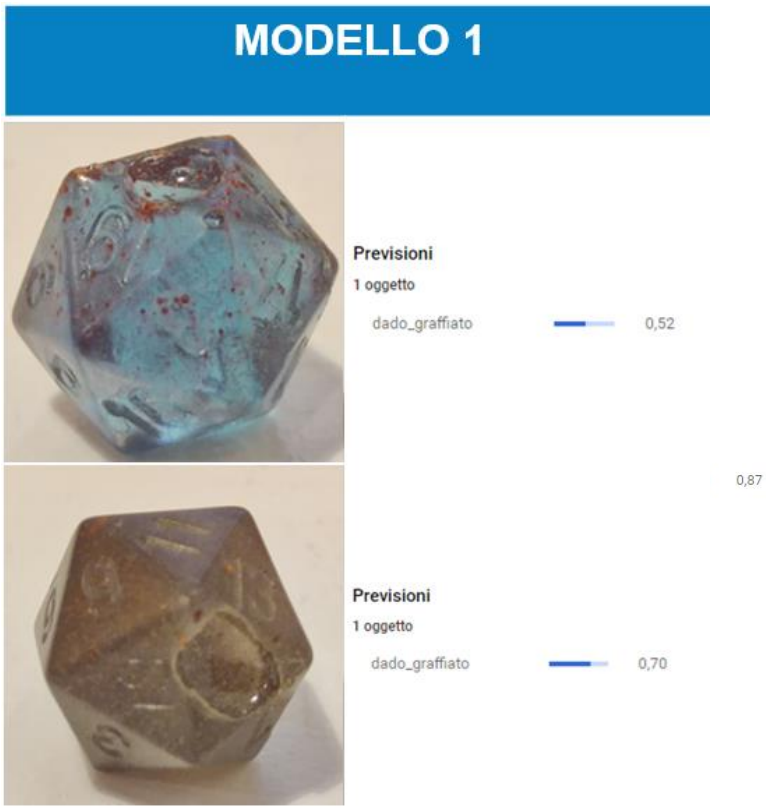


Figura 92 Dati bucati interpretati dal sistema come graffiati

Altri dadi di **vario** tipo:

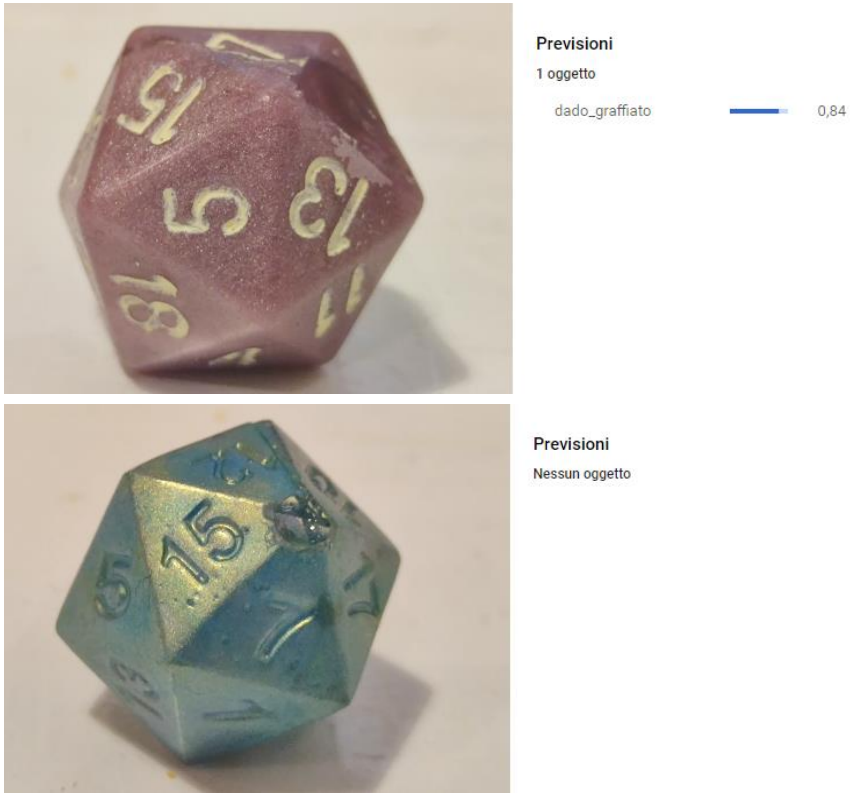


Figura 93 Altri riconoscimenti di dadi

Previsioni

1 oggetto

dado_corretto 0,53

Previsioni

1 oggetto

dado_bucato 0,81

Figura 94 Altri riconoscimenti di dadi

Alcune immagini di **gruppi** di dadi:

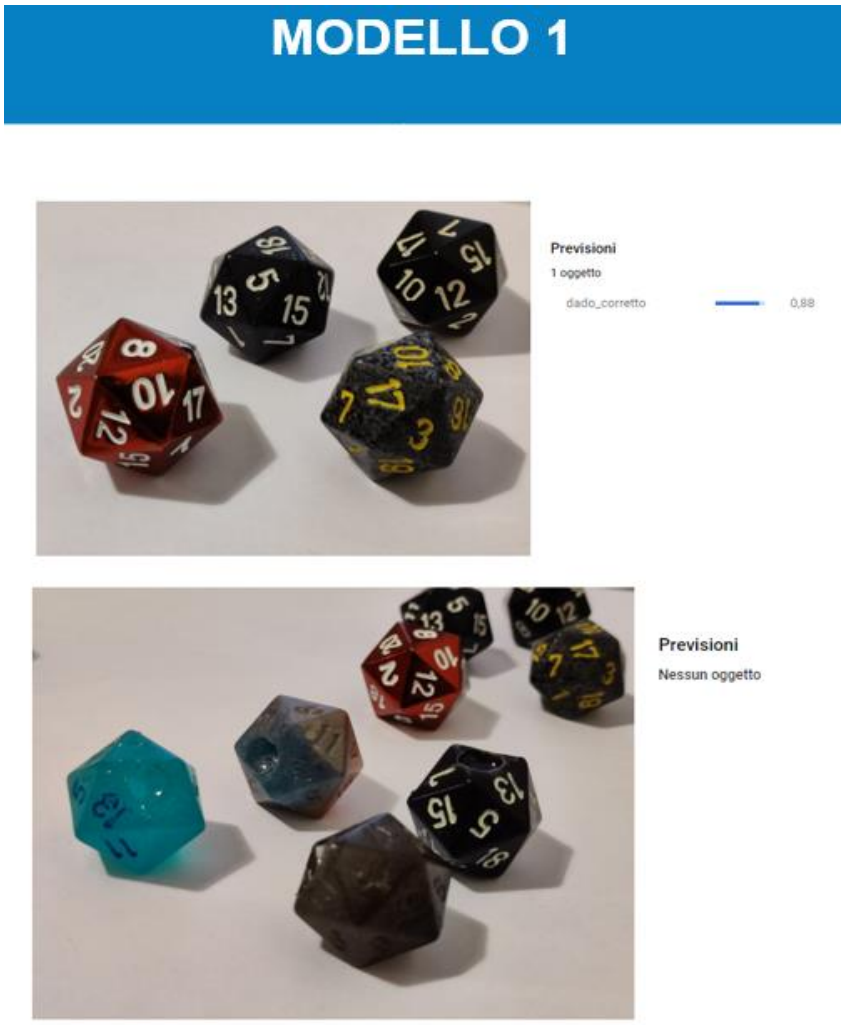


Figura 95 Riconoscimento di gruppi di dadi

Il sistema di Google ricevendo come input un dataset di **qualità molto bassa** è stato in grado di garantire **scarsi** valori di precisione in fase di test. Le difficoltà più evidenti sono state quelle di individuazione di **dadi corretti** molto probabilmente dovuti all'angolazione con la quale sono stati presi i video (dall'alto) rispetto alle fotografie date in input (da davanti).
Altri elementi problematici sono stati la presenza di **glitter** sui dadi (riconosciuta spesso come "graffio") oppure la presenza sia di **graffi** che di **buchi** all'interno del dado, in questo caso le due **classi** sono state spesso **invertite** e **confuse**.
In presenza di alcuni scatti con situazioni di **contesto differenti** (luminosità/ sfondo) il sistema si è **confuso** facilmente e i livelli sono scesi oppure la previsione è risultata **errata.**
In presenza di **gruppi** il sistema è stato spesso in grado di riconoscer una "prevalenza" di dadi corretti, bucati oppure graffiati mentre a volte non è stato in grado di effettuare alcuna previsione.
Alcune immagini di dadi in **contesti differenti**:

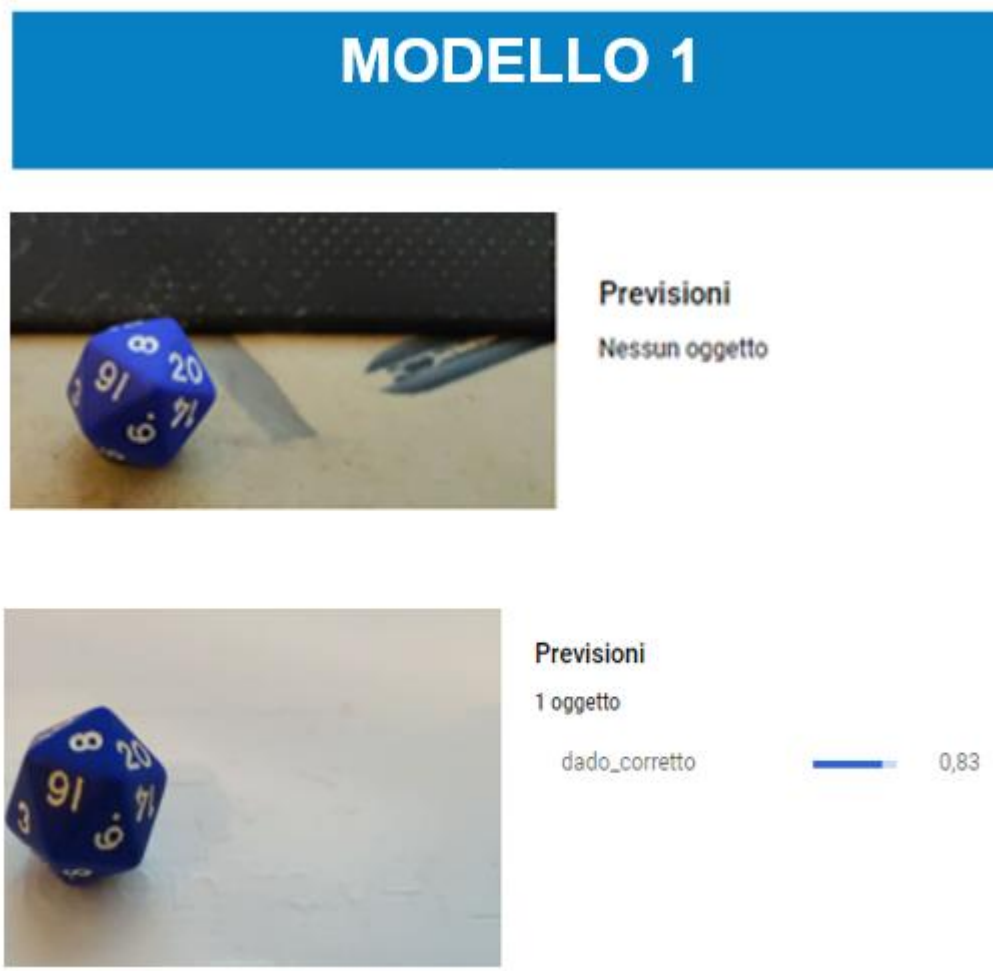


Figura 96 Il sistema tenta di riconoscere dadi in contesti differenti da quelli utilizzati durante il train

Figura 97 Il sistema cerca di riconoscere gruppi di dadi

ESPERIMENTO 2 CLASSIFICAZIONE DI IMMAGINI CON DATASET PULITO

Dataset

Ho preso 4 dadi in **perfette condizioni**, 4 dadi che presentavano un **buco** e 4 dadi che presentavano un **graffio**. A differenza di quanto visto nell'esperimento numero 1, le fotografie sono state scattate a mano.

Per ogni dado **corretto** sono state scattate 20 fotografie: (*2 scartate perché di bassa qualità)

-Una sopra (1 fotografia)

-Una sotto (1 fotografia)

-Una davanti, una ruotando il dado di +30°, una ruotando il dado di -30° per tutte e 6 le possibili posizioni. (18 fotografie)

Per ogni dado **corretto** oppure **graffiato** sono state scattate un certo numero di fotografie corrispondenti alle angolazioni in cui il difetto era **visibile**.

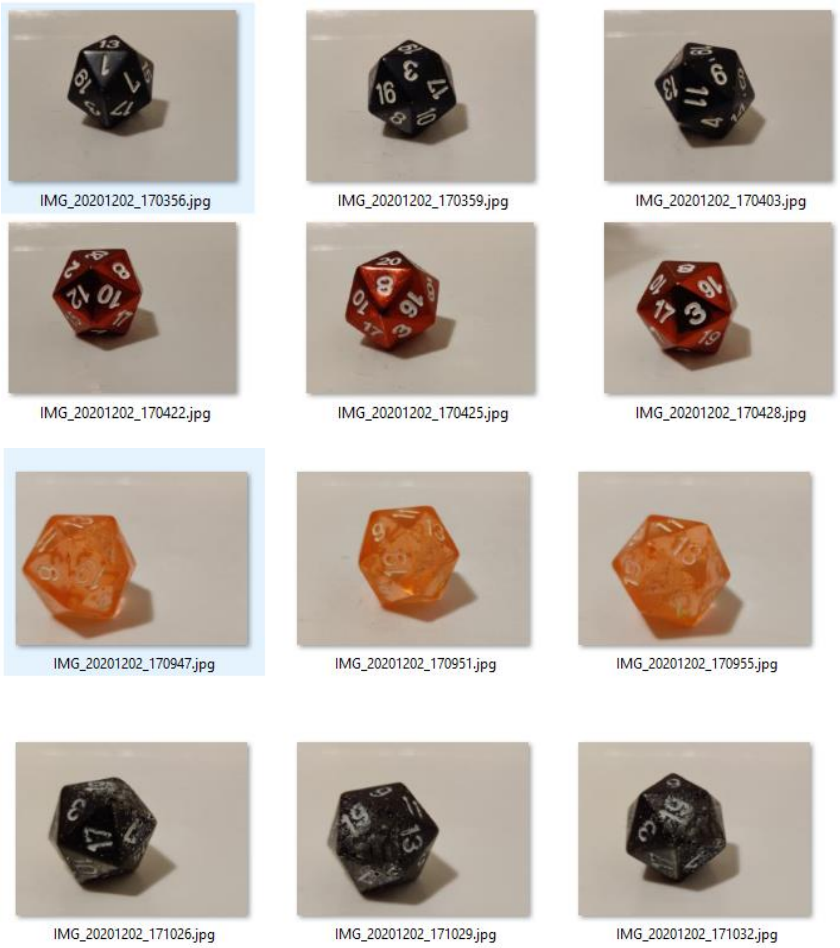


Figura 98 Secondo esperimento: dataset migliorato

Importare le immagini

Una volta preparato il dataset ho creato un nuovo progetto all'interno di Google Cloud Platform Vision e ho caricato queste immagini in un differente **bucket** chiamato **dadi_set_Fotografico_1_etichetta**

Tagging

Una volta terminata l'importazione delle immagini ho effettuato il **tagging** delle stesse sempre con le stesse etichette utilizzate per l'esperimento 1:

dado_corretto

dado_bucato

dado_graffiato

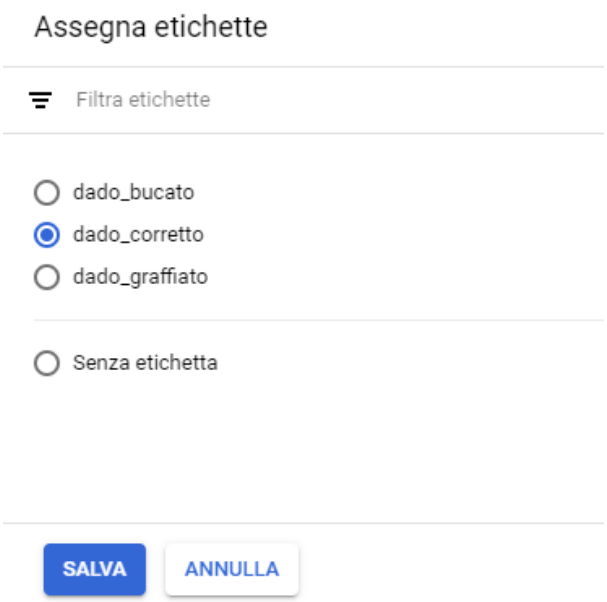


Figura 99 Stesse classi dell'esperimento 1

Una volta terminato l'assegnamento delle etichette ecco la situazione complessiva dei dati relativi al secondo esperimento:

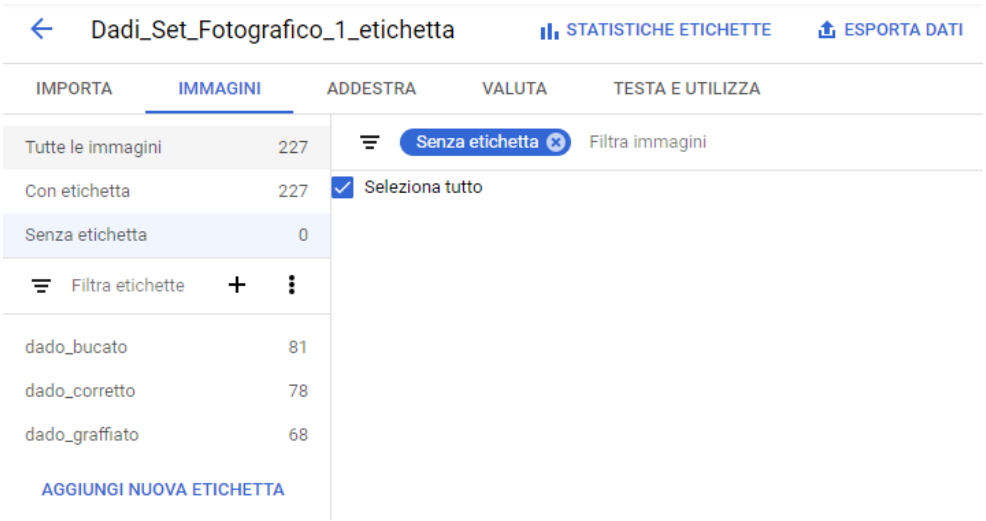


Figura 100 Dati raccolti per l'esperimento 2

Da come si nota nell'immagine 12 dadi complessivi, hanno generato un dataset di **227 immagini**.

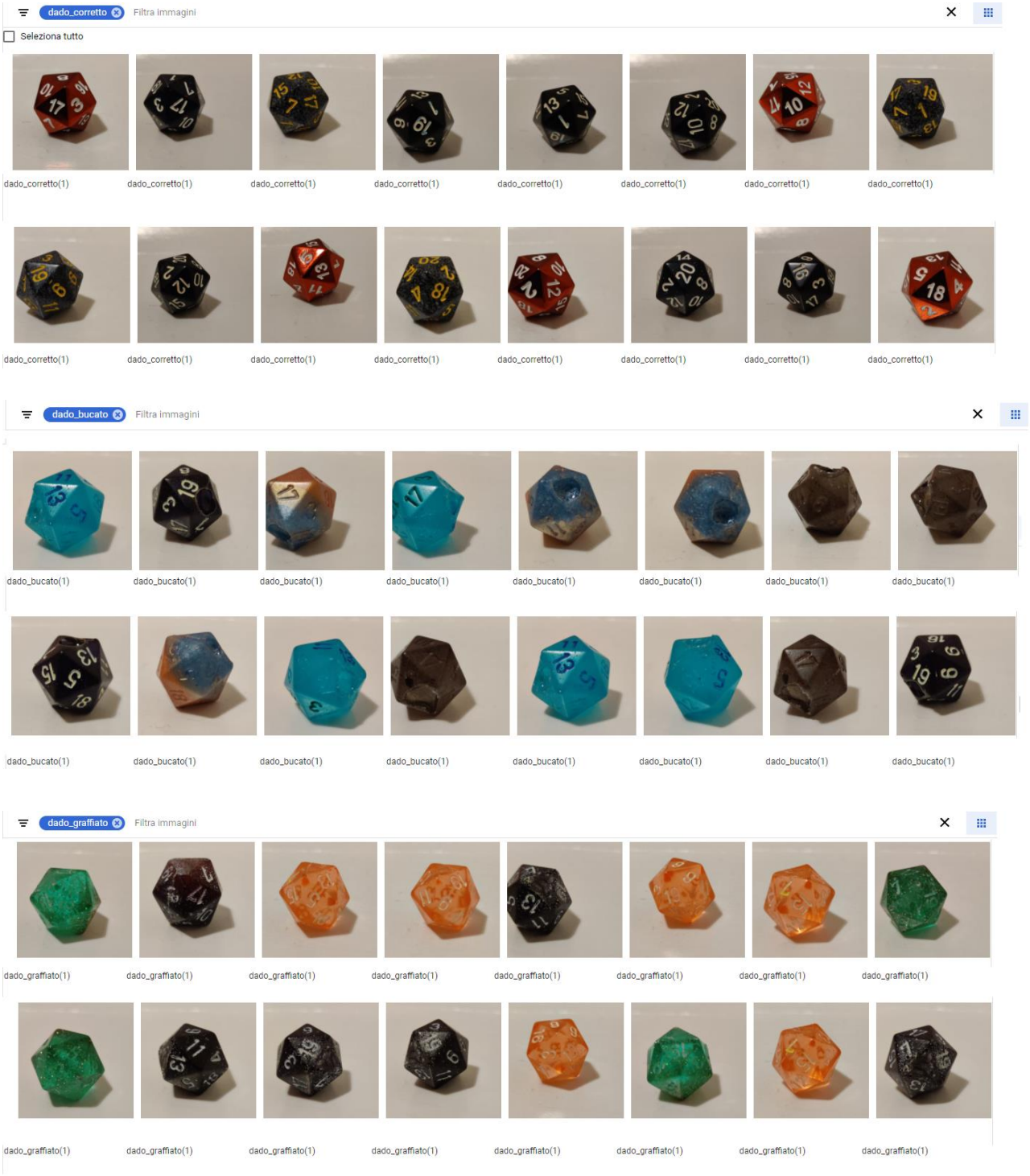


Figura 101 Dataset numero 2 (NO

Addestramento

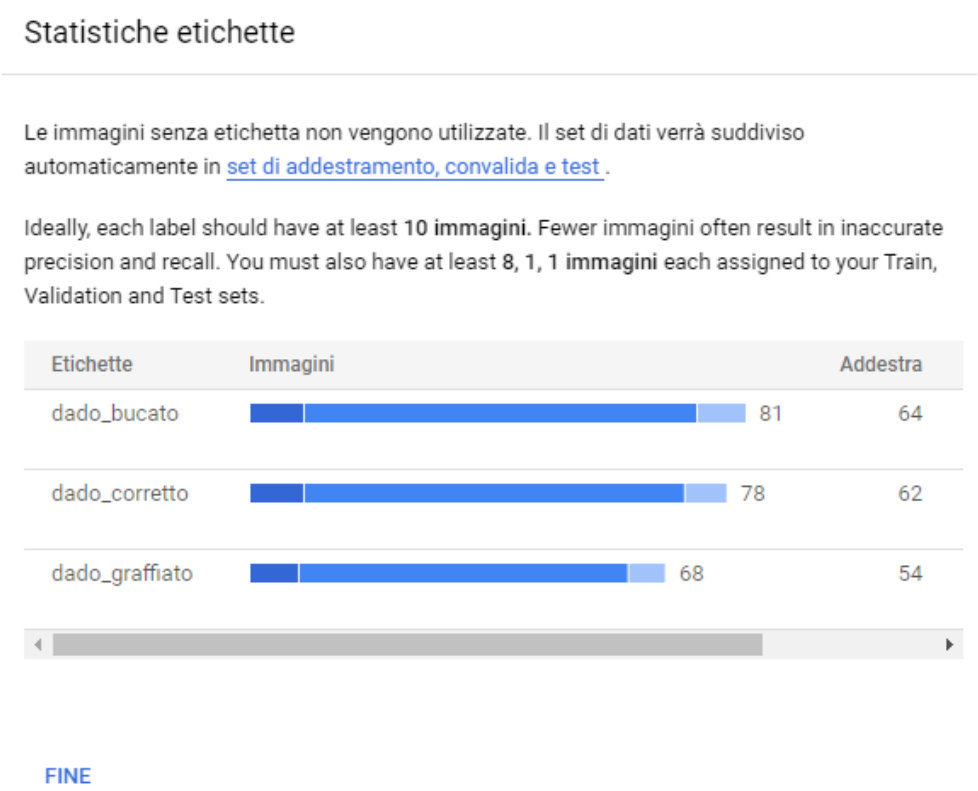


Figura 102 Immagini e suddivisione Train/Test nell'esperimento 2

L'addestramento è durato circa 2 ore ma questa volta il costo è stato di 10 Ore/Nodo quindi **31.5$.**

Test e Risultati

I risultati della rete dopo l'addestramento sono stati i seguenti:

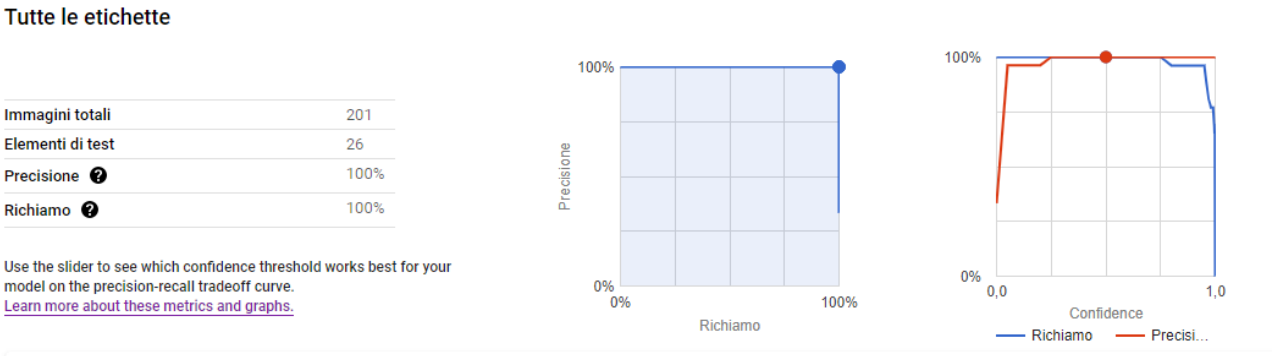


Figura 103 Precisione e richiamo esperimento 2

Il sistema ha preso, delle 201 immagini, un campione di **176** fotografie per effettuare la fase di **training** e un campione di 26 immagini per la fase di **test.**

Figura 104 Matrice di confusione esperimento 2

Da quello che si può notare dalla **Matrice di confusione** e dai valori di **precision e recall,** il sistema sembra performare internamente in maniera **ottimale** e ha risultati del 100% di precisione e richiamo. Ovviamente questi valori sono causati dalla presenza di un numero basso di campioni in input: per il test infatti sono state usate 8/9 immagini per etichetta.

Su **9** immagini di **dadi_corretto**, il sistema non ha mai confuso la classificazione.

Su **8** immagini di **dadi_bucato**, il sistema non ha mai confuso la classificazione.

Su **9** immagini di **dado_graffiato** il sistema non ha mai confuso la classificazione

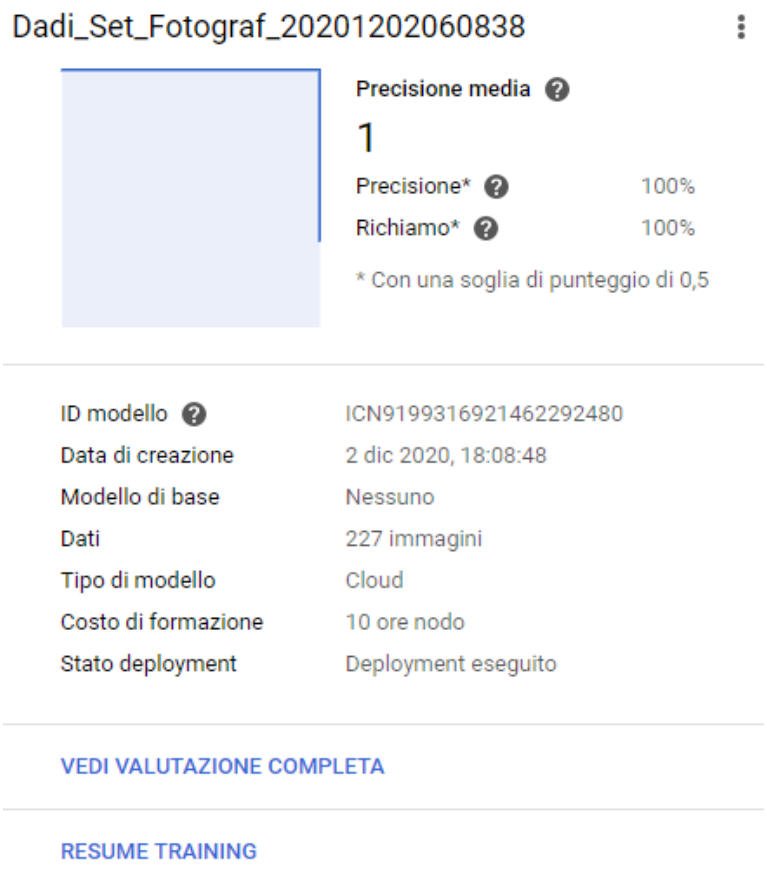


Figura 105 Precisione media e modello secondo esperimento

Alcune immagini di **dadi corretti** interpretati dal sistema:

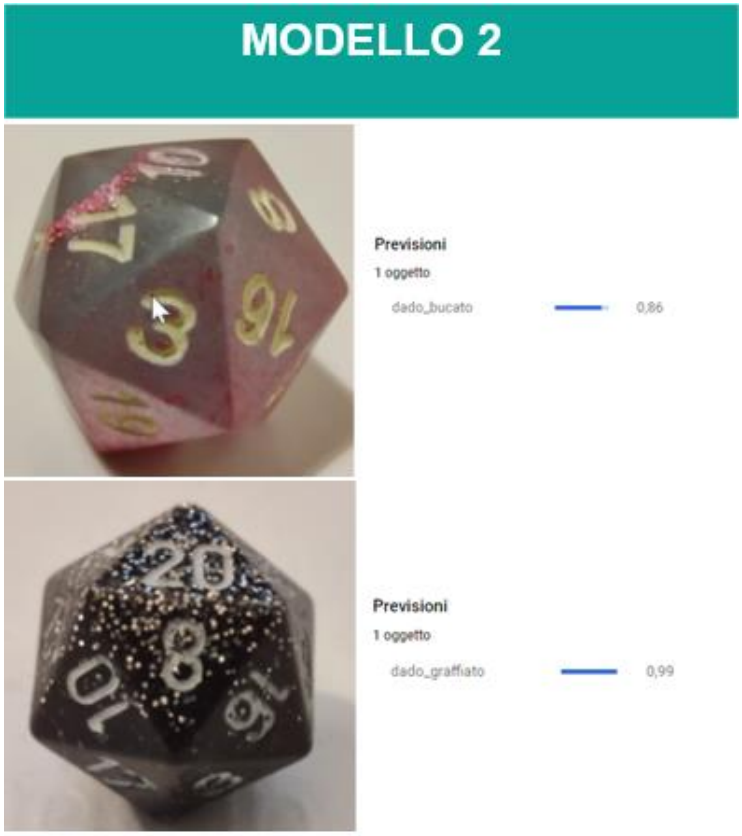


Figura 106 Dadi corretti interpretati dal sistema

Alcune immagini di **dadi graffiati** interpretati dal sistema:

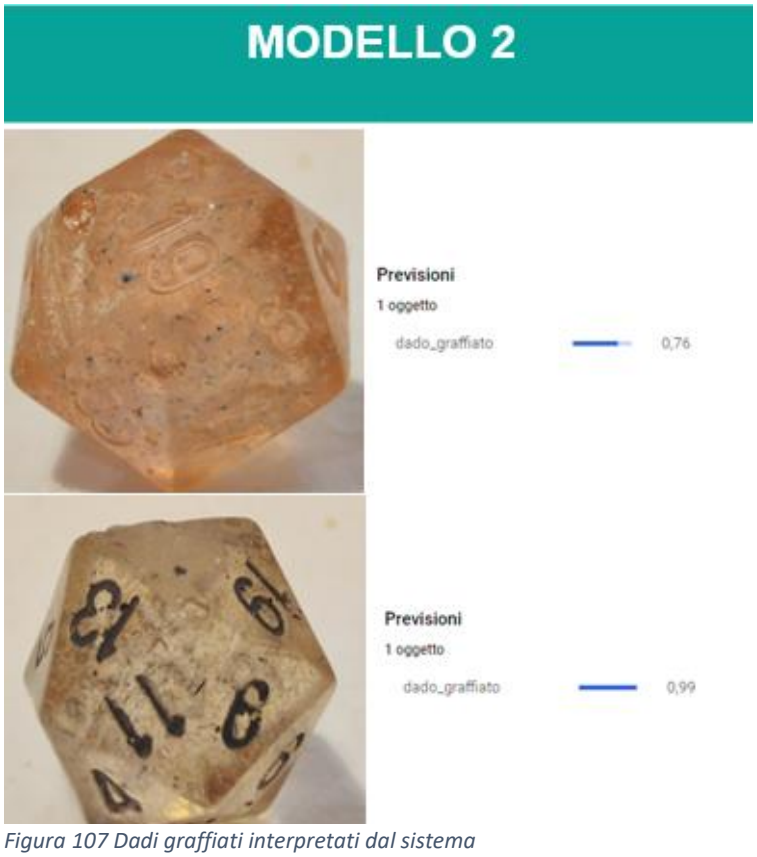


Figura 107 Dadi graffiati interpretati dal sistema

Alcune immagini di **dadi bucati** interpretati dal sistema:

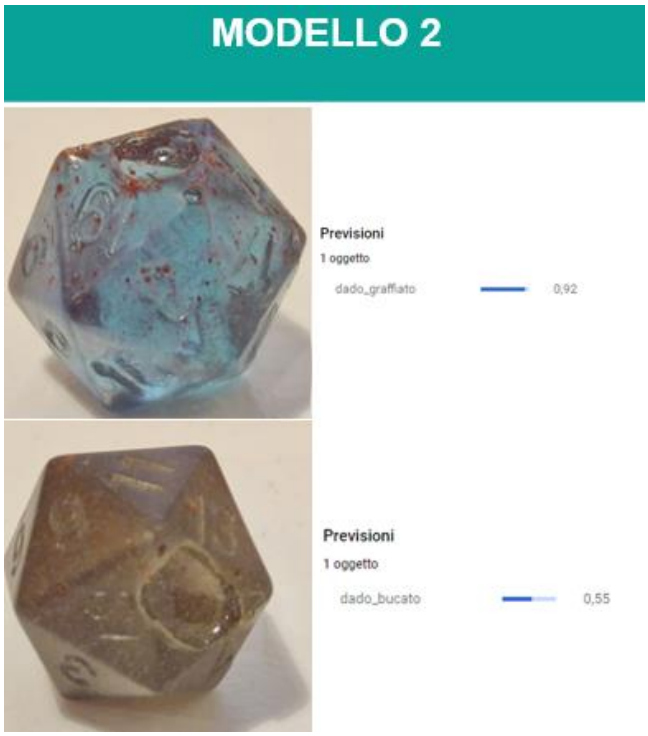


Figura 108 Dadi bucati interpretati dal sistema

Alcune immagini di **gruppi di dadi** interpretati dal sistema:

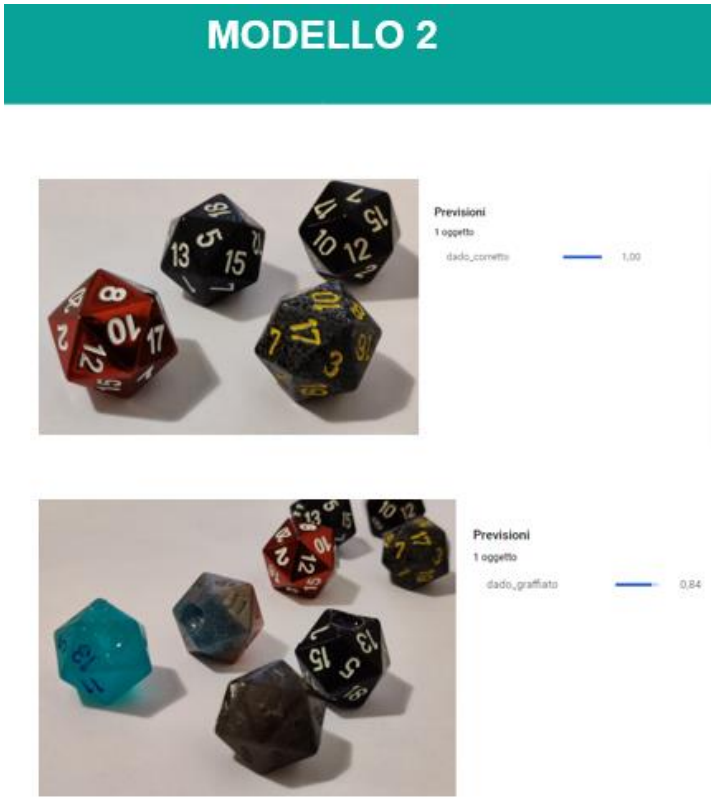


Figura 109 Gruppi di dadi interpretati dal sistema

Alcune immagini di **dadi vari** interpretati dal sistema:

Figura 110 Dadi vari interpretati dal sistema

Figura 111 Dadi vari interpretati dal sistema

Figura 112 Dadi vari interpretati dal sistema

Il sistema di Google, sebbene abbia avuto come input un dataset di **alta qualità** ma di **basso numero di campioni** è stato in grado di garantire buoni valori di precisione.
In presenza di alcuni scatti con situazioni di **contesto differenti** come lo **sfondo** si è confuso facilmente e i livelli sono scesi oppure la previsione è risultata errata.
Il modello numero 2 riconosce meglio i **buchi** del modello numero 1 soprattutto con valori di precisione più elevati. Rispetto al primo modello ha confuso qualche **buco** con qualche **graffio**, ma complessivamente i risultati e i **valori** di **confidenza** nelle previsioni sono **nettamente migliori**.
Quando il modello 1 non riconosceva oggetti, il modello 2 ha sempre dato una risposta (spesso corretta, in diversi casi sbagliata, ma **mai nulla**)

Altri elementi problematici sono stati la presenza di **glitter** sui dadi (riconosciuta spesso come "graffio"). Come vediamo qui è bastato nuovamente cambiare lo sfondo per far confondere la macchina sulla natura del dado. Un semplice cambio di colore sullo sfondo e purtroppo le previsioni risultano errate.
Alcune immagini di **dadi in un contesto differente** rispetto a quello utilizzato nel training set:

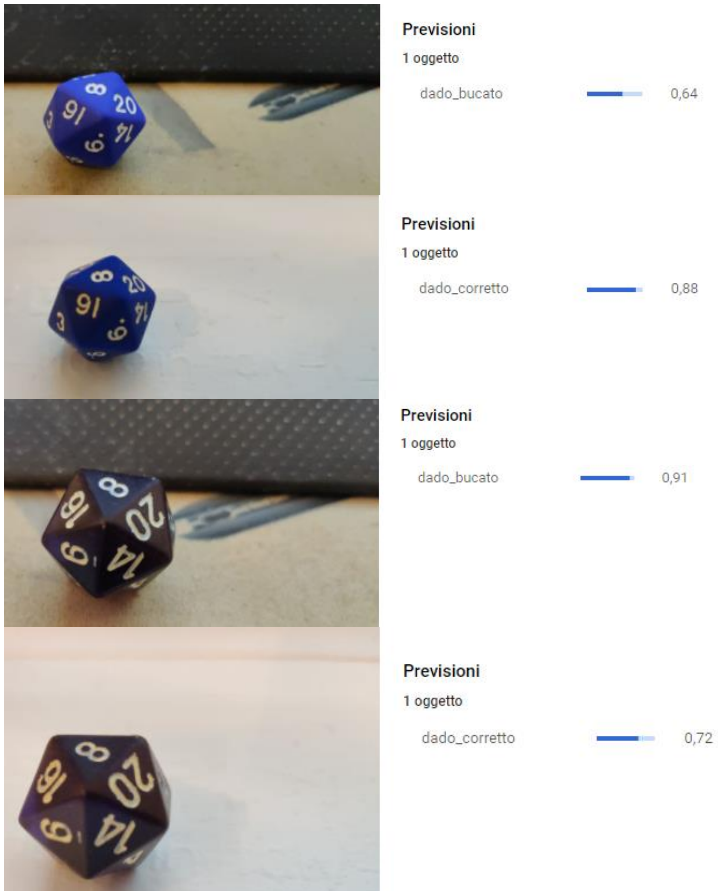


Figura 113 Dadi in contesti differenti dal solito interpretati dal sistema

Figura 114 Gruppi di dadi interpretati dal sistema

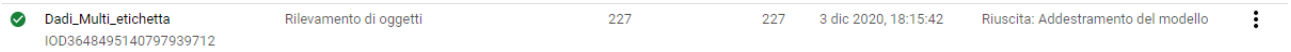
ESPERIMENTO 3 CLASSIFICAZIONE DI IMMAGINI MULTI ETICHETTA CON DATASET PULITO

Dataset

Ho preso 4 dadi in **perfette condizioni**, 4 dadi che presentavano un **buco** e 4 dadi che presentavano un **graffio** sempre dello stesso dataset utilizzato nell'esperimento 2.

Importare le immagini

Una volta preparato il dataset ho creato un nuovo progetto all'interno di Google Cloud Platform Vision e ho caricato queste immagini in un differente **bucket**



Tagging

Una volta terminata l'importazione delle immagini ho effettuato il **tagging** delle stesse, questa volta però sfruttando lo strumento di **selezione** rettangolare offerto da Google. In questo particolare caso, ogni immagine è stata taggata indicando precisamente alla macchina che cosa fosse un **dado** che cosa fosse un **graffio** e cosa fosse un **buco.**

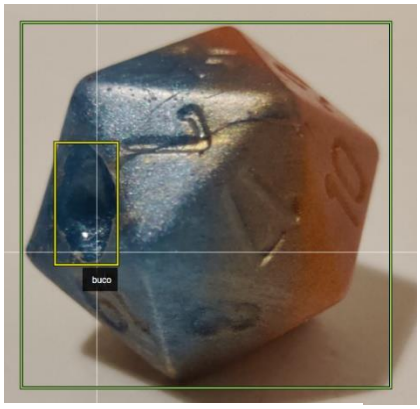


Figura 115 Dado bucato

Figura 116 Dado corretto

Figura 118 Dado graffiato

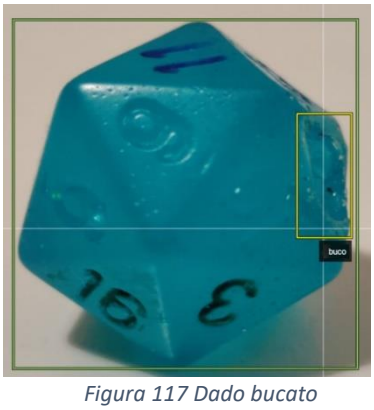


Figura 117 Dado bucato

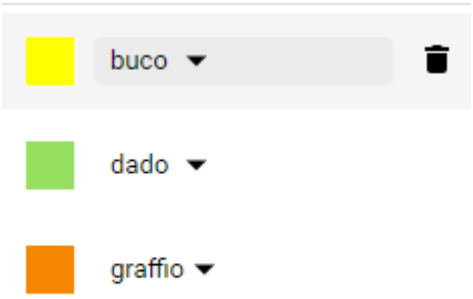


Figura 119 Classi esperimento 3

Una volta terminato l'assegnamento delle etichette ecco la situazione complessiva dei dati relativi al terzo esperimento:

Rispetto agli altri esperimenti si nota che l'etichetta dado è molto più presente rispetto al normale, questo perché tutti gli oggetti che presentano un buco o un graffio sono stati classificati anche come dadi.

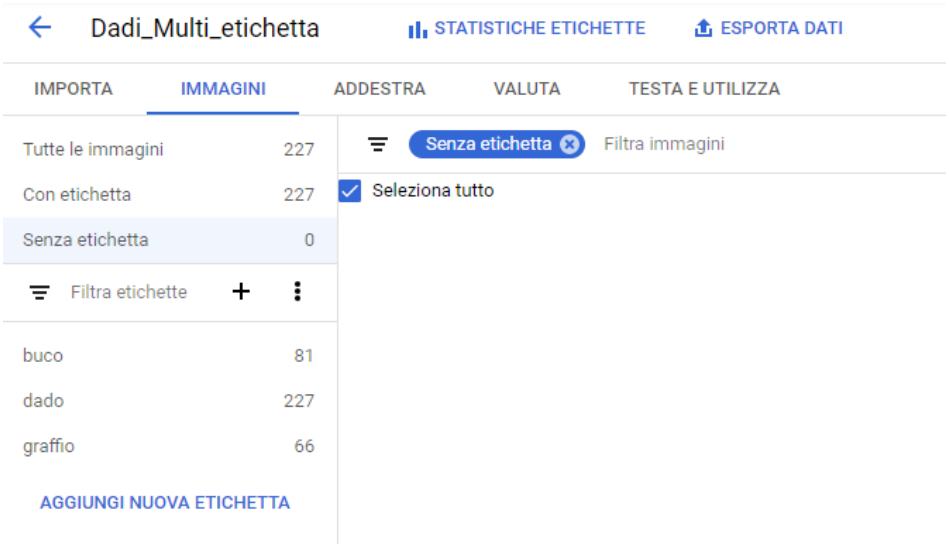


Figura 120 Dati raccolti esperimento 3

Da come si nota nell'immagine 12 dadi complessivi, hanno generato un dataset di **227 immagini**.

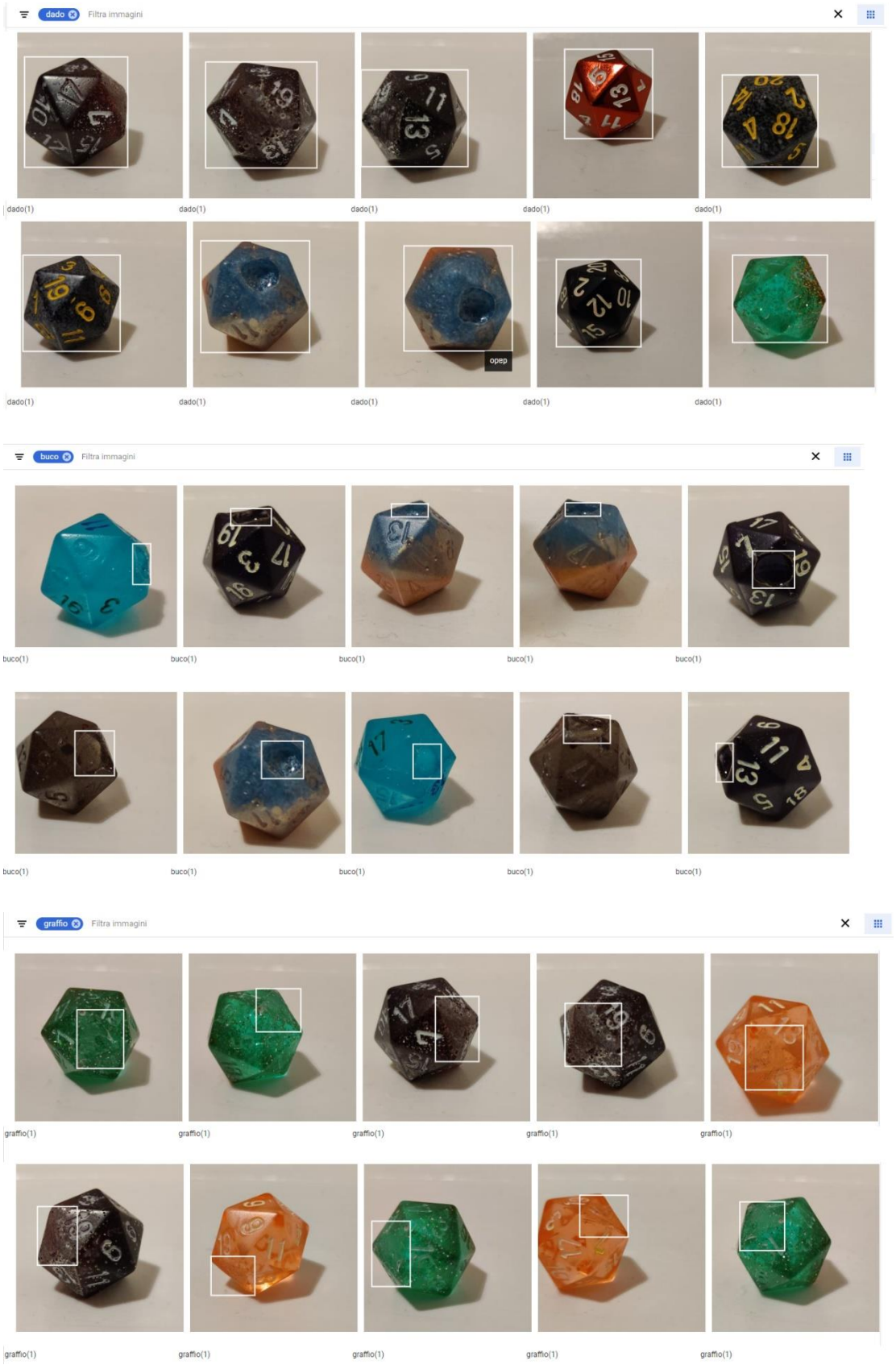


Figura 121 Dataset taggato esperimento 3

Addestramento

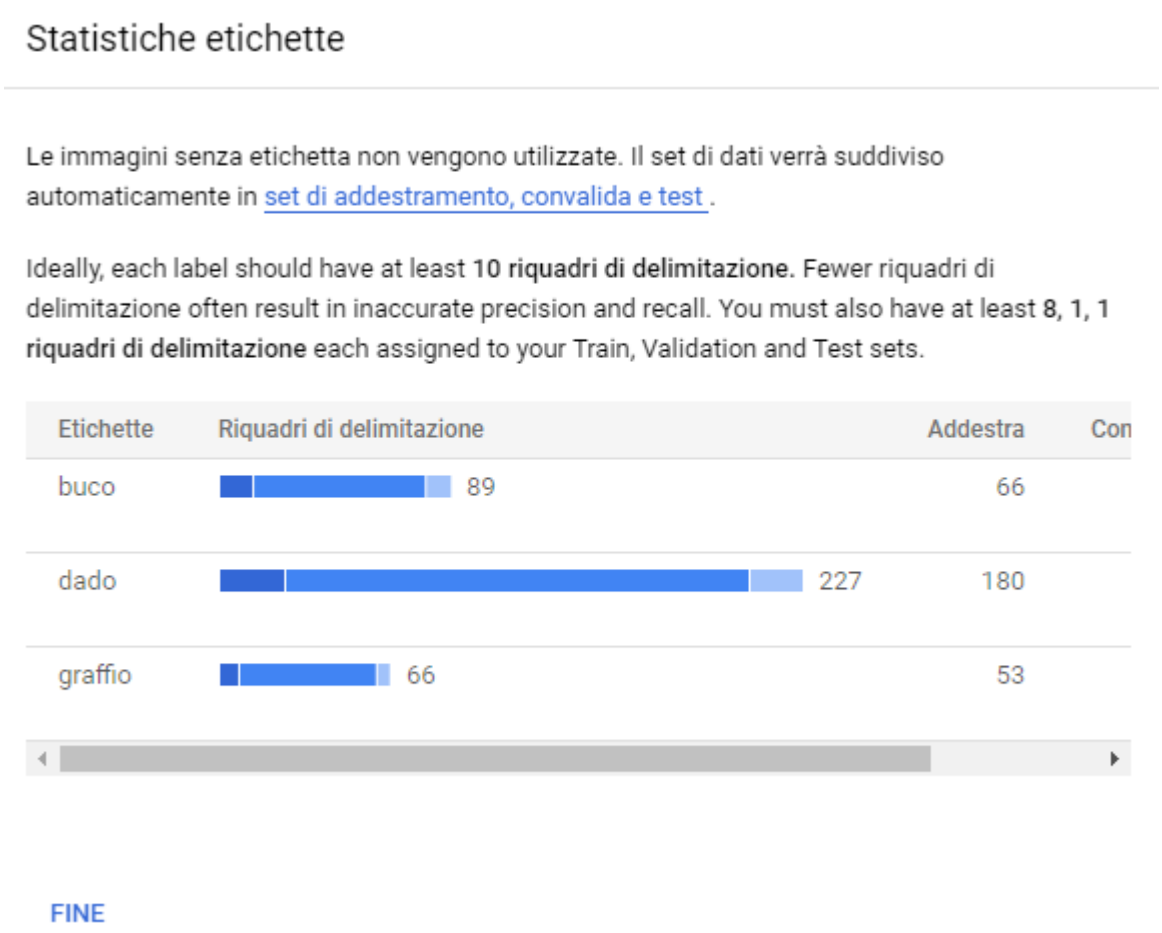


Figura 122 Suddivisione Train/Test esperimento 3

L'addestramento è durato circa 2 ore e 30 minuti ma questa volta il costo è stato di 54 Ore/Nodo quindi **170.1$.**

Test e Risultati

I risultati della rete dopo l'addestramento sono stati i seguenti:

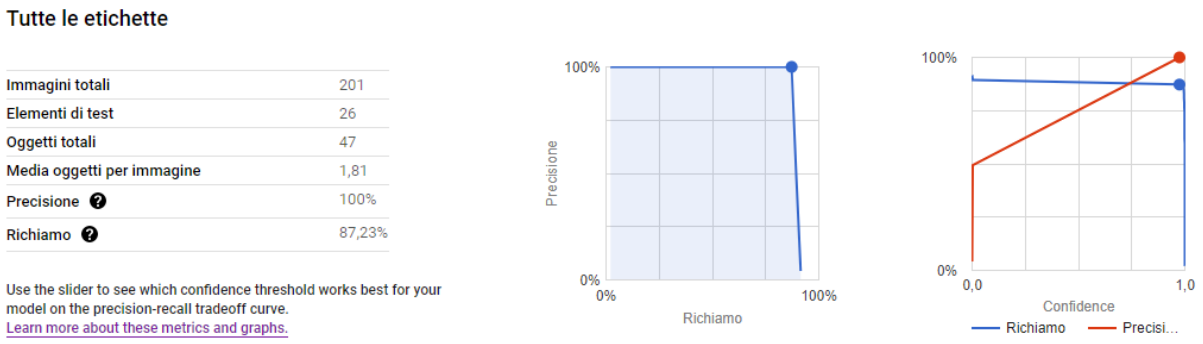


Figura 123 Precisione e richiamo esperimento 3

Il sistema ha preso, delle 201 immagini, un campione di **176** fotografie per effettuare la fase di **training** e un campione di 26 immagini per la fase di **test.**

Da quello che si può notare dai valori di **precision e recall,** il sistema sembra performare internamente in maniera **eccellente** e ha risultati del 100% di accuratezza e 87.23% di richiamo.

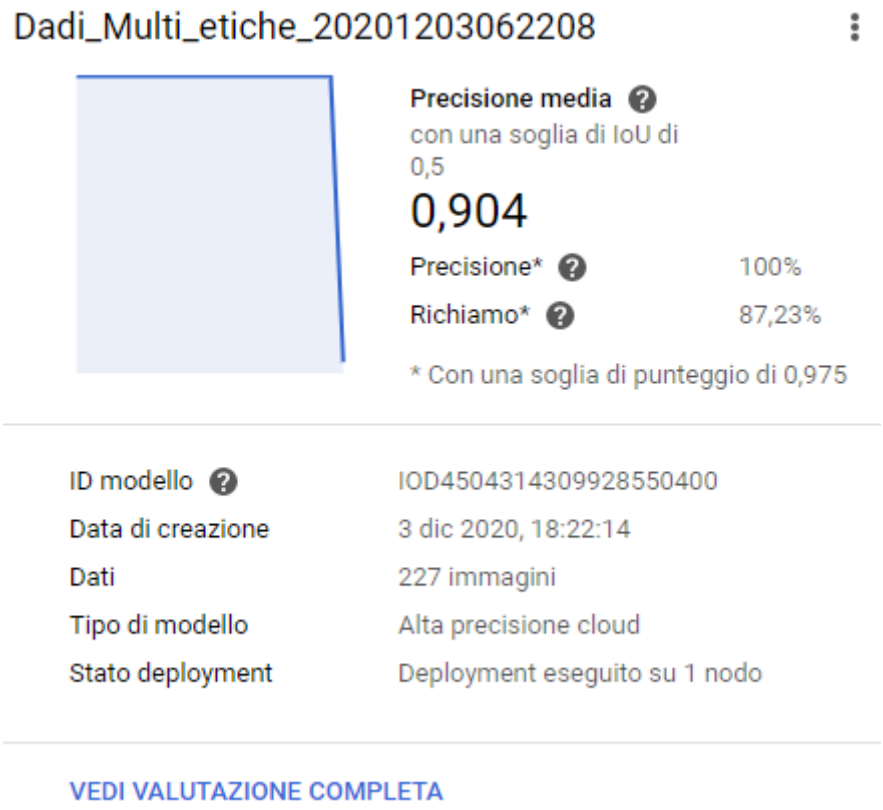


Figura 124 Precisione media e modello esperimento 3

Alcuni risultati ottenti fornendo in input al sistema **dadi mai visti**:

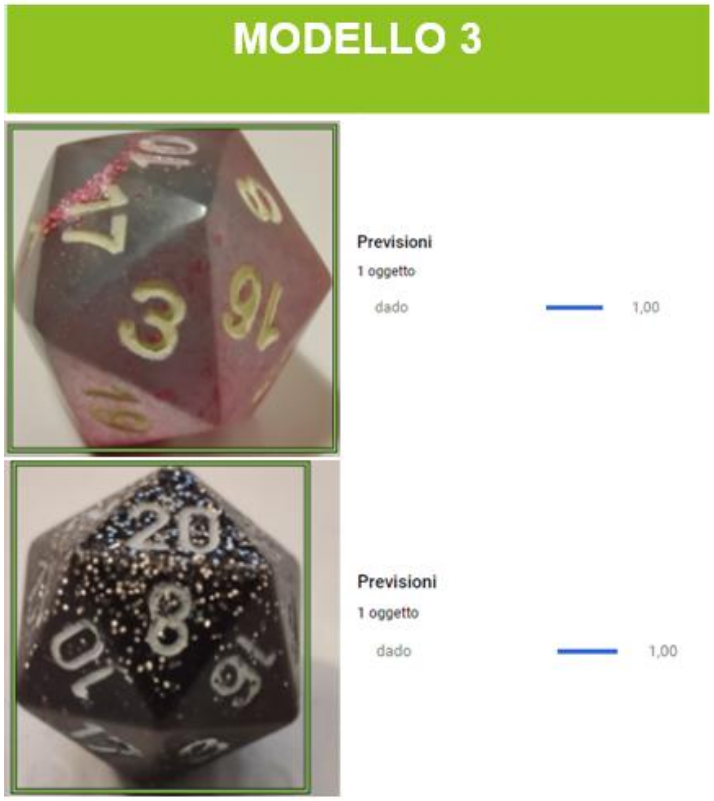


Figura 125 Il sistema tenta di interpretare dadi corretti

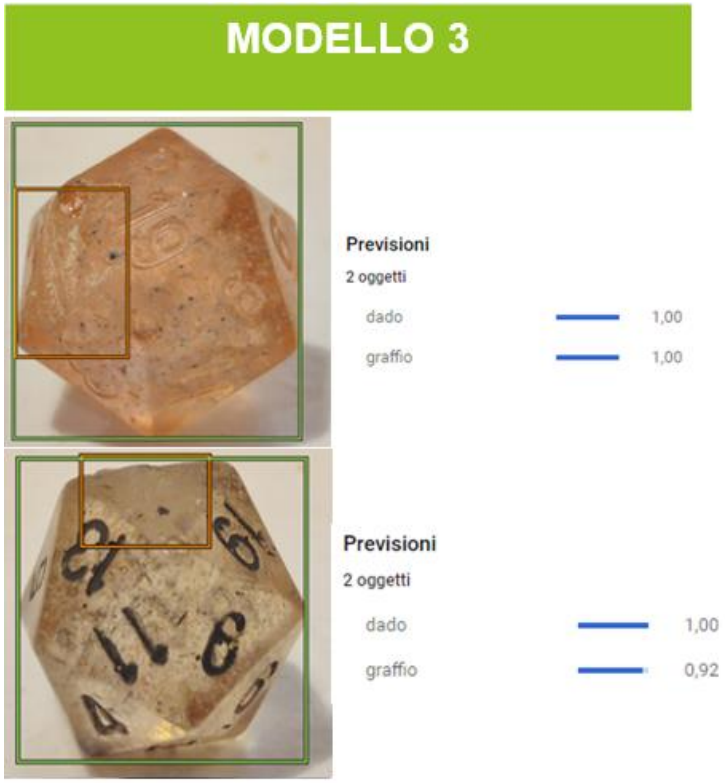


Figura 126 Il sistema tenta di interpretare dadi graffiati

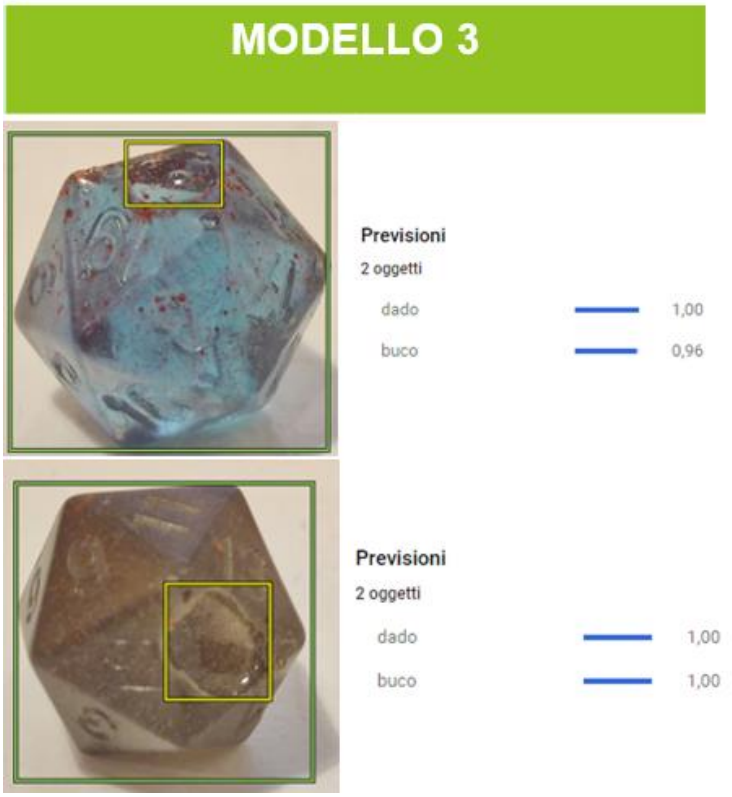


Figura 127 Il sistema tenta di interpretare dadi bucati

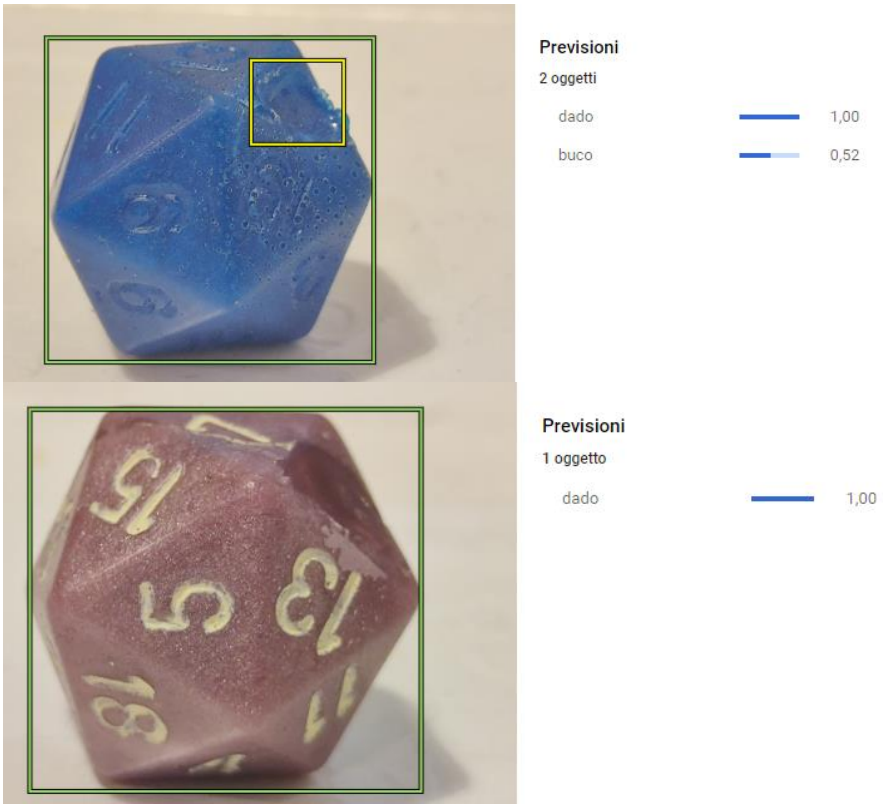
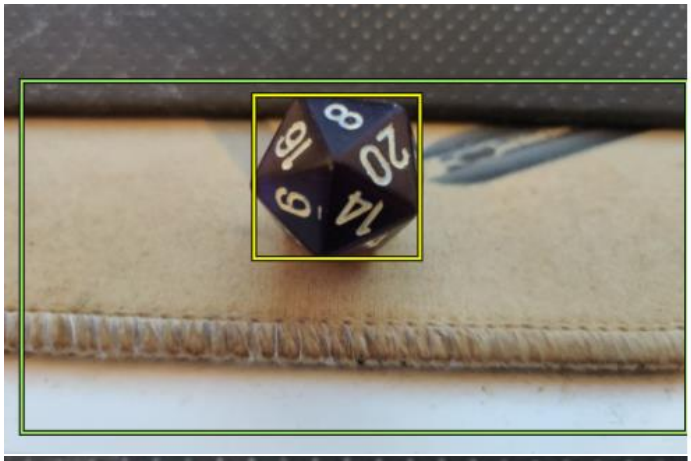


Figura 128 Il sistema tenta di interpretare un dado bucato e uno graffiato

Figura 129 Il sistema tenta di interpretare dadi bucati

Figura 130 Il sistema tenta di interpretare due dadi corretti

Alcune immagini **fuori dal contesto** con il classico vassoio bianco utillizzato per creare il dadaset.



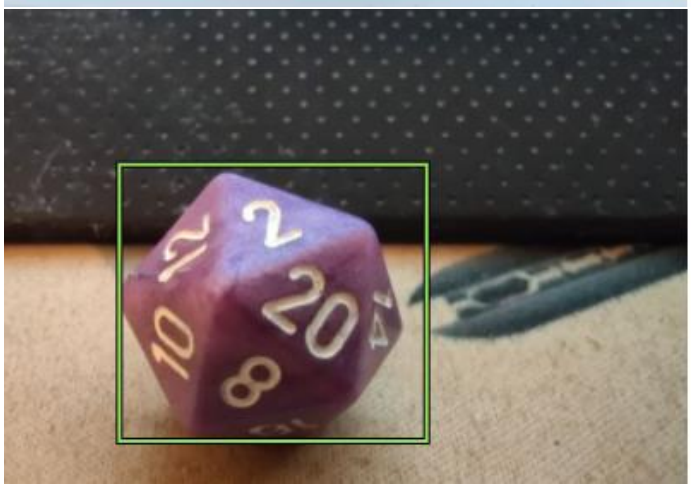


Figura 131 Il sistema tenta di interpretare due dadi in un contesto differente

Alcune immagini di **gruppi di dadi**:

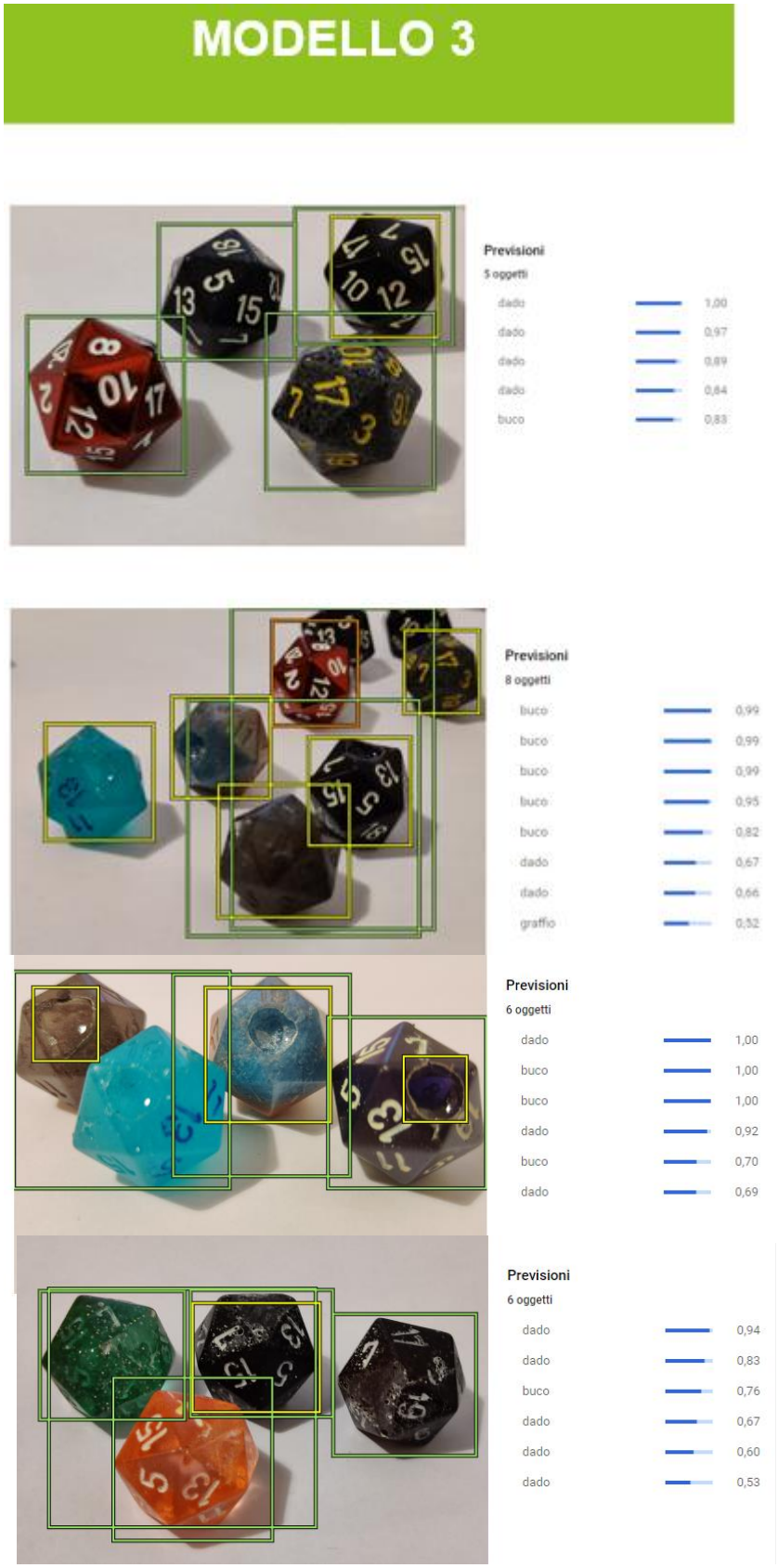


Figura 132 Il sistema tenta di interpretare gruppi di dadi

Analisi e commento finale

Il sistema di Google performa in maniera eccellente in situazioni di riconoscimento standard. Quando però si tenta di effettuare un riconoscimento di diversi oggetti all'interno della stessa immagine modificando allo stesso tempo l'inquadratura della camera (magari scattando la foto dall'alto) viene facilmente confuso. Ancora una volta, la presenza di situazioni di **contesto differenti** come lo **sfondo** hanno abbassato i livelli di precisione o reso la previsione errata.
A differenza degli altri modelli, questo sembra in grado di distinguere un **graffio** dalla presenza di **glitter** applicato sul dado. Il sistema non riconosce più questa **feature** come incisiva dal punto di vista della presenza di graffi sugli oggetti ma fa ancora molta fatica a riconoscere oggetti strutturati in gruppi. Quando si tratta di effettuare il riconoscimento di **più elementi** all'interno di una fotografia, il modello di Google fa molta confusione tra il concetto di **dado** e il concetto di **buco**. Questo sottolinea quanto il contesto **conti** nella creazione di un **dataset** di qualità in maniera da addestrare reti nel modo più **efficace** possibile. Non è stato scelto di utilizzare contesti differenti dal vassoio bianco all'interno di questi esperimenti proprio per sottolineare quanto sia difficile effettuare un riconoscimento corretto di un **difetto** sul corpo di un pezzo senza fornirsi di un dataset molto **ricco** e preciso di **contesti**, **inquadrature** e **luminosità** differenti.

I **tre modelli** dei tre esperimenti **Google a confronto**

Riconoscimento dei **dadi corretti**: Modello 1 vs Modello 2 vs Modello 3.

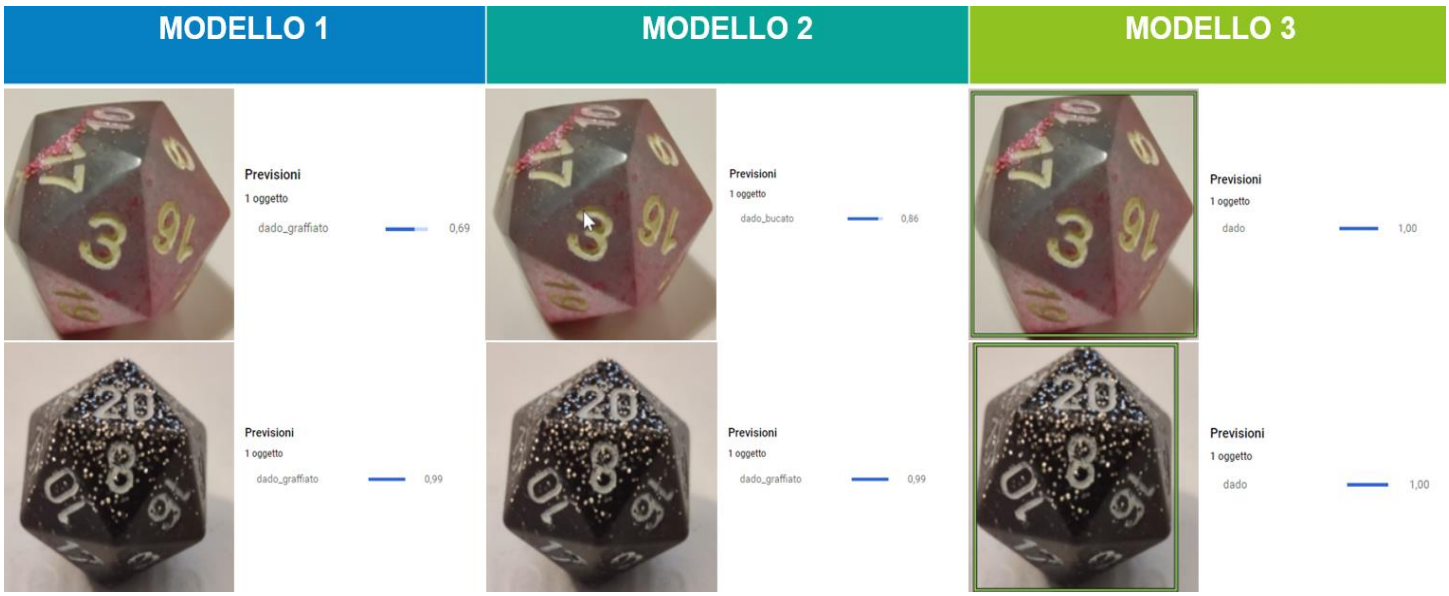


Figura 133 Confronto riconoscimento dadi corretti dei tre esperimenti Google

Riconoscimento dei **dadi graffiati** Modello 1 vs Modello 2 vs Modello 3.

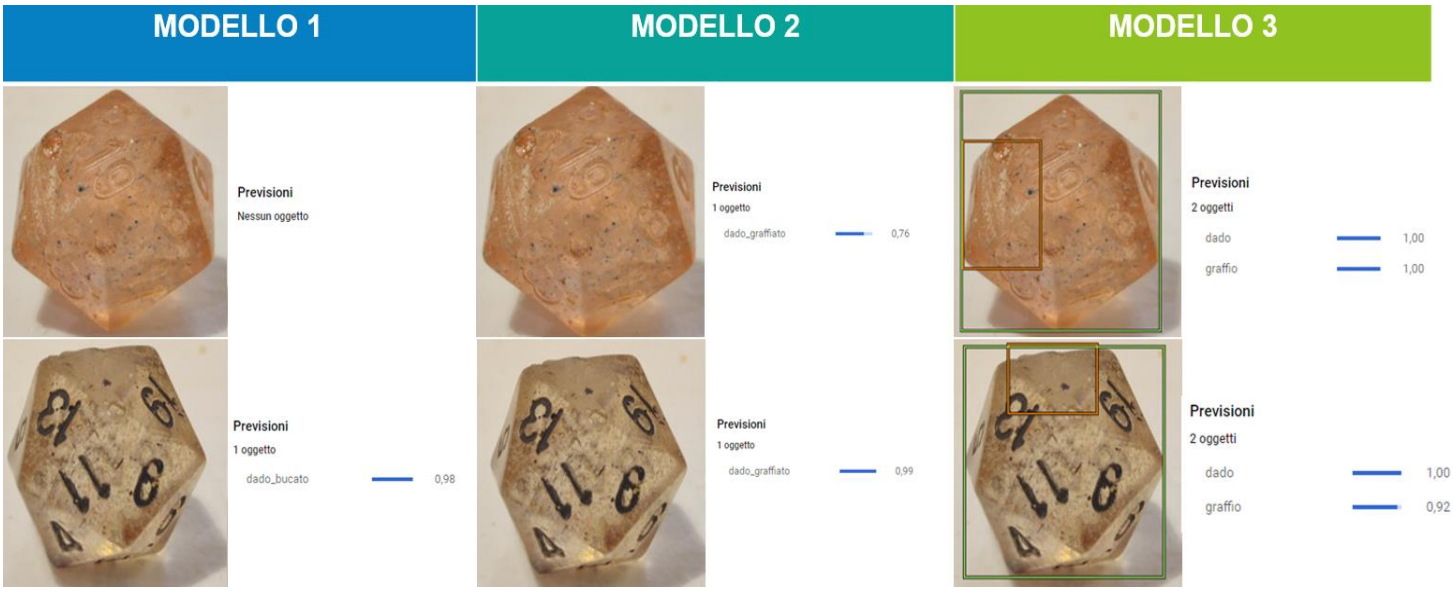
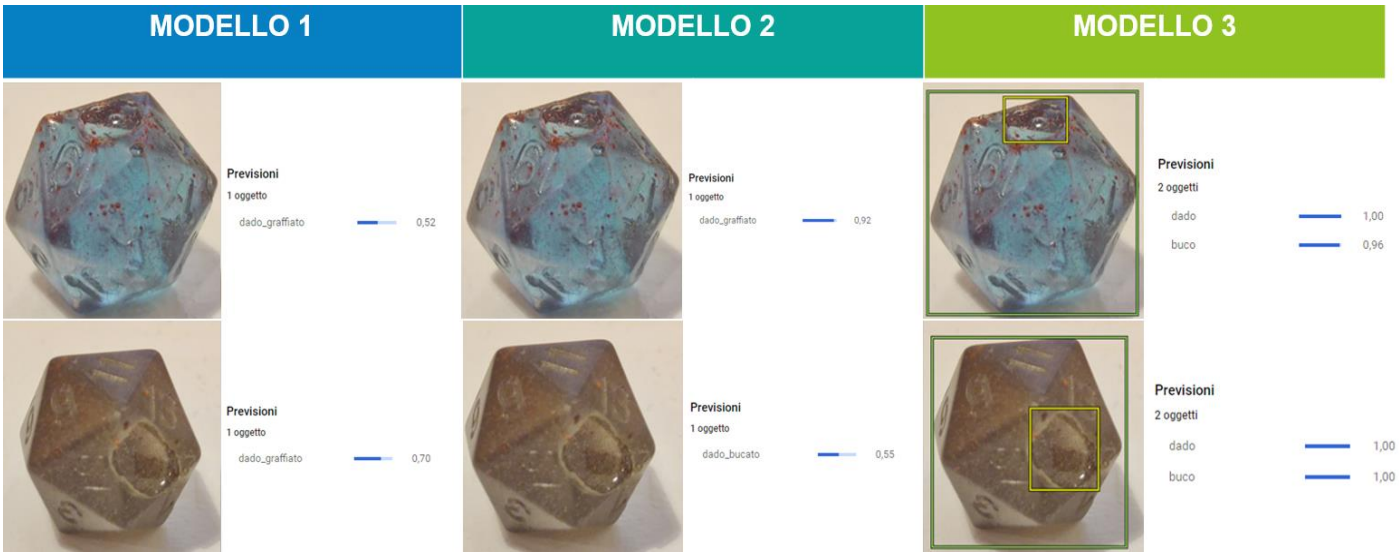


Figura 134 Confronto riconoscimento dadi graffiati dei tre esperimenti Google

Riconoscimento dei **dadi bucati**: Modello 1 vs Modello 2 vs Modello 3.



135 Confronto riconoscimento dadi bucati dei tre esperimenti Google

3.2 Microsoft Vision

Il Sistema **Vision** di Microsoft presenta caratteristiche molto simili a quelle viste con Google.

https://www.customvision.ai/projects

Dataset

Verrà utilizzato lo stesso dataset home-made con un numero di dadi pari a 227. Effettuerò solamente 1 esperimento con lo strumento di **tag rettangolare Microsoft**.

Importare le Immagini

Una volta preparato il data-set è necessario creare un nuovo progetto:

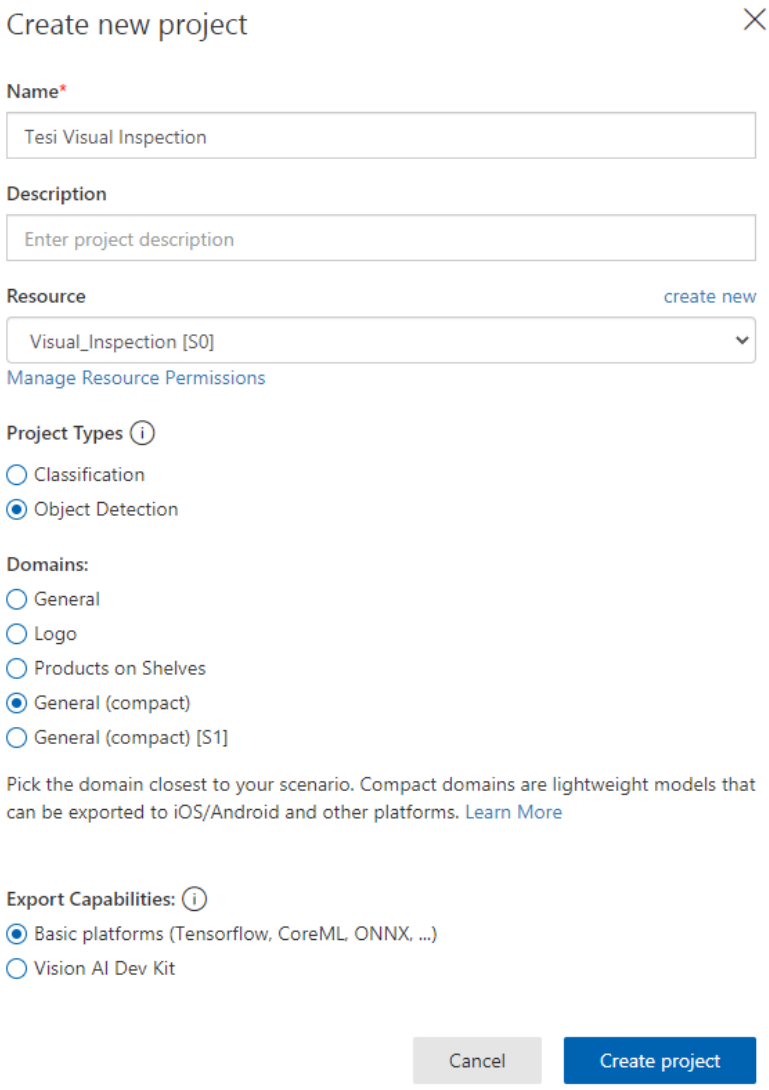


Figura 136 Nuovo progetto Microsoft vision

Microsoft consente all'utente di selezionare tra due tipi di progetto:

- **Classificazione** (per fotografie con singola etichetta, simili agli esperimenti 1 e 2 condotti con Google)
- **Object detection** (per fotografie con etichetta multipla, simile all'esperimento 3 condotto con Google).

In questo caso effettueremo Object detection e tratteremo graffi e buchi come **oggetti** interni alla struttura di un **dado**.

Tagging

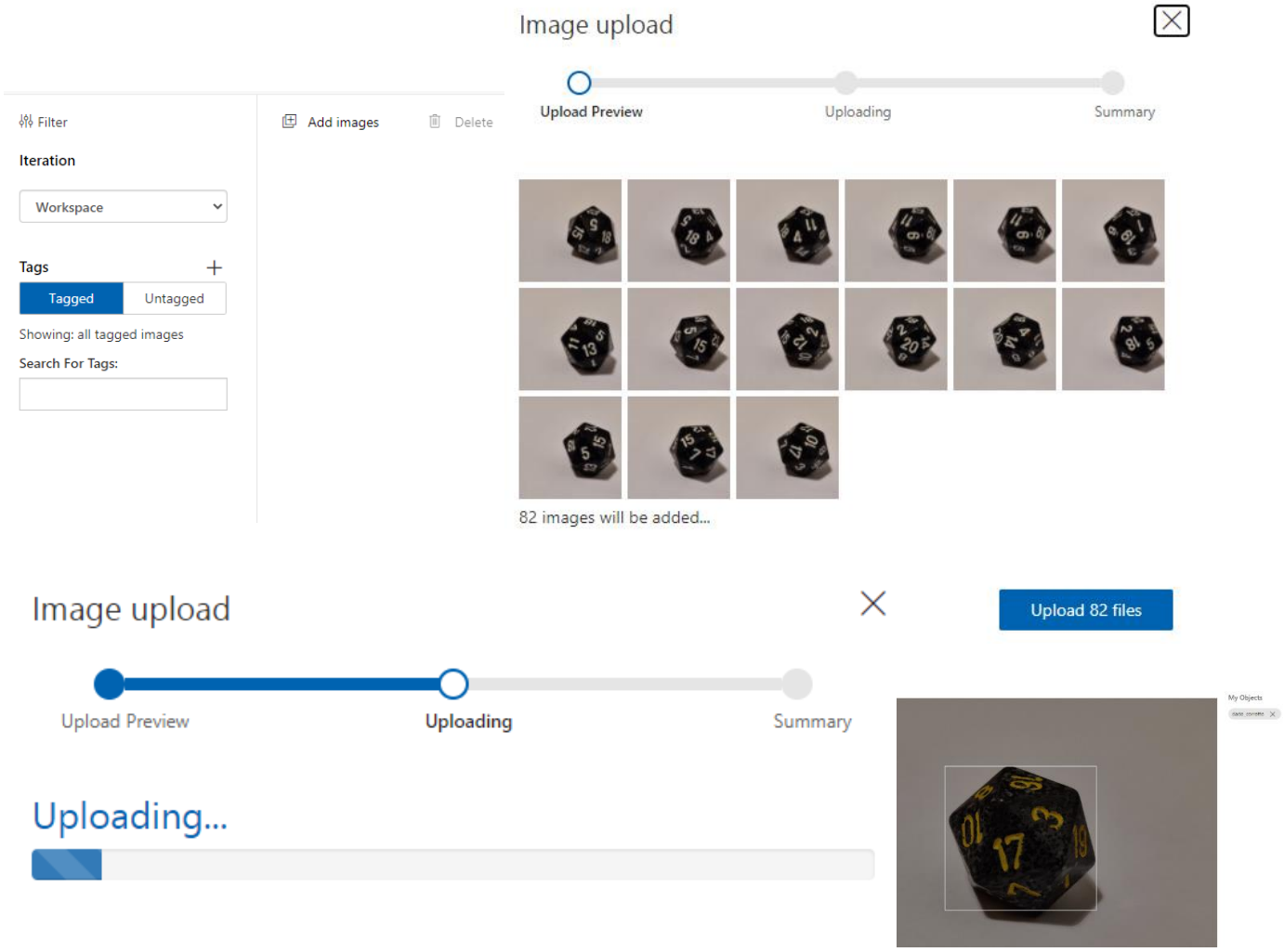


Figura 137 Upload e tagging dei dati Microsoft vision

Le modalità con la quale **Microsoft** consente di taggare le proprie immagini è molto più **pratica** di quella vista con Google.

Nel sistema Google l'utente è costretto a racchiudere con un box rettangolare anche le strutture più ovvie. Microsoft tenta di effettuare un r**iconoscimento automatico** parziale degli oggetti nell'immagine ed è stata **sempre** in grado di riconoscere a primo impatto il **soggetto** di una fotografia, facendomi risparmiare il box rettangolare per le strutture relative ai dadi (ma non a graffi e buchi ovviamente). Altra nota positiva per Microsoft, una volta effettuato il tagging di una foto, il sistema passa alla fotografia successiva molto rapidamente (mentre i tempi di caricamento di Google erano molto più elevati)

Taggare 227 foto con Google: **2 ore**

Taggare 227 foto con Microsoft: **30 minuti**

Addestramento

I tempi di addestramento del sistema Microsoft sono stati veramente rapidi rispetto a quelli di Google. Il modello è stato addestrato in appena **30 minuti** (relativamente alla versione di prova disponibile per gli studenti). L'addestramento è **completamente gratuito** per gli studenti (log-in con l'account universitario) e fornisce anche interessanti statistiche in merito alla precision-recall di tutte le **etichette** utilizzate.

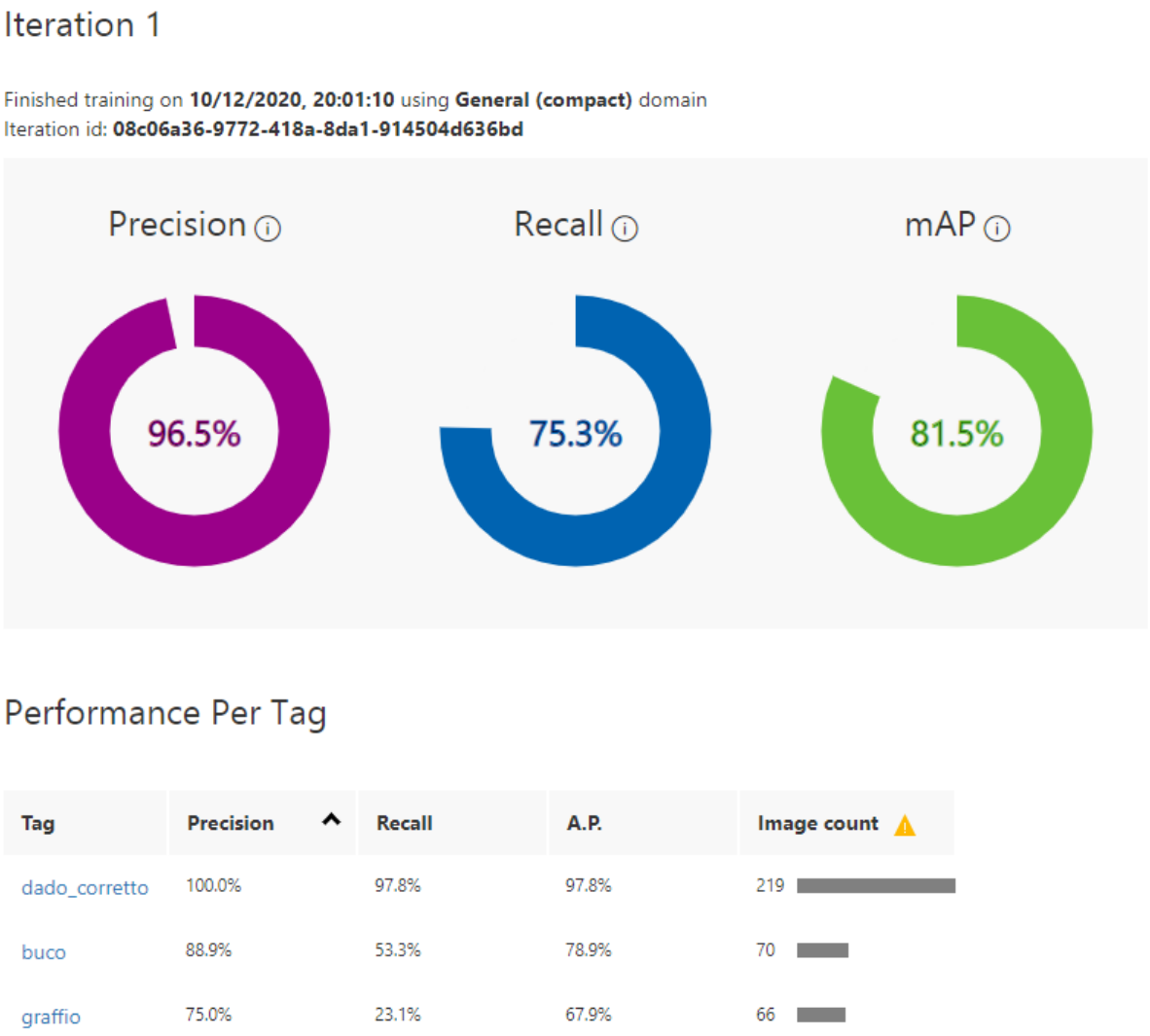


Figura 138 Precisione, richiamo e mAP del modello Microsoft

Alcune immagini di **dadi corretti** interpretati dal sistema:

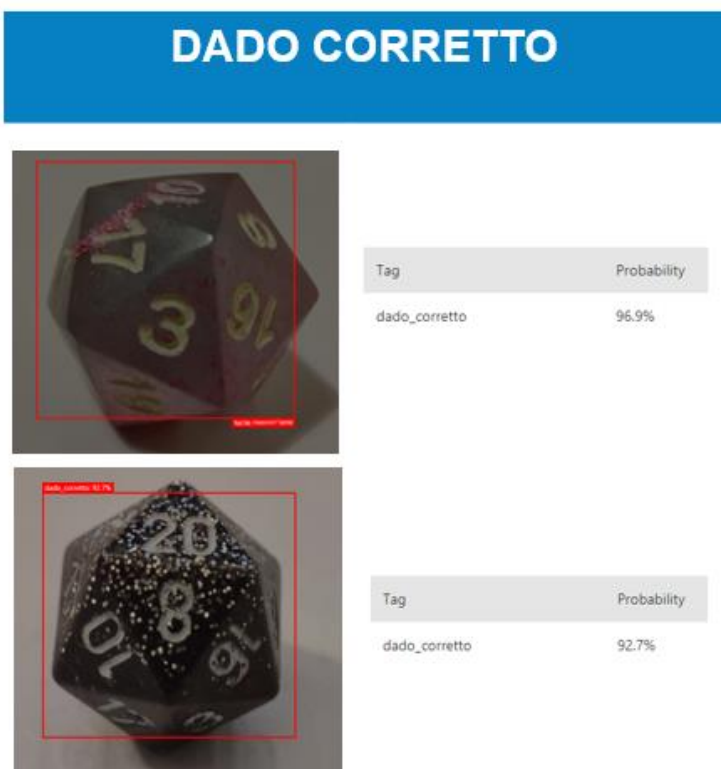


Figura 139 Il sistema tenta di riconoscere due dadi corretti

Alcune immagini di **dadi graffiati** interpretati dal sistema:

Figura 140 Il sistema tenta di riconoscere due dadi graffiati

Alcune immagini di **dadi bucati** interpretati dal sistema:

Figura 141 Il sistema tenta di riconoscere due dadi bucati

Alcune immagini di **gruppi di dadi** interpretati dal sistema:

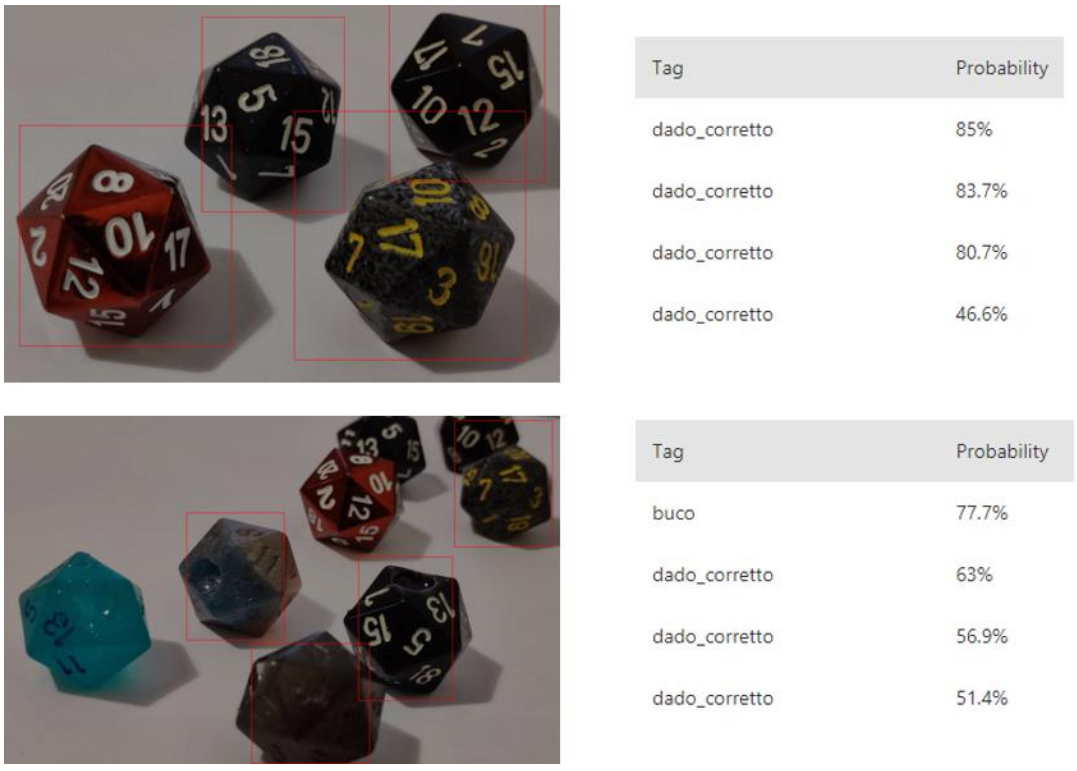
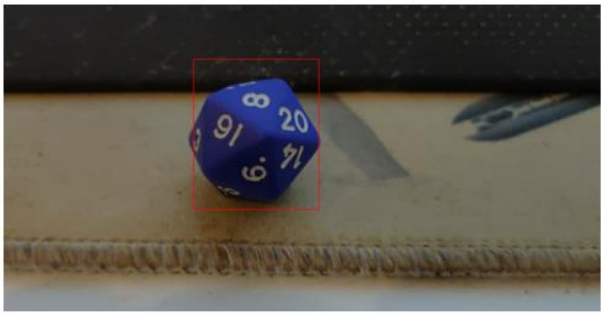


Figura 142 Il sistema tenta di riconoscere gruppi di dadi

Alcune immagini **fuori contesto** rispetto al dataset di training:



Tag	Probability
dado_corretto	72.4%

Tag	Probability
dado_corretto	81.5%

Figura 143 Il sistema tenta di riconoscere dadi in contesti differenti

Analisi e commento finale

Rispetto a Google, Microsoft fa molta più fatica a riconoscere i **graffi** e possiede complessivamente valori di confidenza più **bassi**. Microsoft, ad esempio, **NON** è in grado di riconoscere il graffio sul dado arancione di test né su quello bianco se non abbassando il livello di soglia sotto al **20%.** Il valore di probabilità visto dal tool Microsoft per il graffio del primo dado è del **19.8%** mentre per il secondo è di **8.4%.** I valori di riconoscimento all'interno di un **contesto** sono buoni ma i valori di confidenza non sono troppo elevati.

Per quanto riguarda il riconoscimento di **gruppi** di dadi, il tool di Microsoft effettua delle analisi più **high-level** e fa **molta meno confusione** del modello di Google. Da quello che si vede all'interno dei risultati del primo gruppo di dadi, i dadi corretti sono stati tutti identificati correttamente (con valori bassi di precisione ma comunque non è stato visto nessun buco dal modello Microsoft).

Per il secondo gruppo le considerazioni risultano le stesse: un approccio più ad alto livello fa confondere meno il riconoscitore che non vede **buchi** o **graffi** dove non ci sono ma allo stesso tempo mantiene i propri valori di certezza parecchio bassi.

3.3 IBM Watson Visual Recognition

Il Sistema **Watson vision** di IBM presenta caratteristiche molto simili a quelle viste con Google e Microsoft.

https://www.ibm.com/it-it/cloud/watson-visual-recognition

Dataset

Verrà utilizzato lo stesso dataset home-made con un numero di dadi pari a 227. Effettuerò solamente 1 esperimento con lo strumento di **tag rettangolare IBM**.

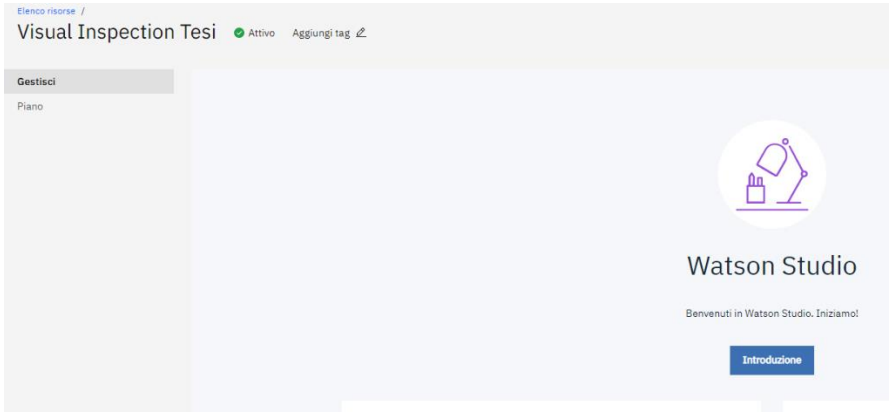


Figura 107 Watson Studio

Importare le Immagini

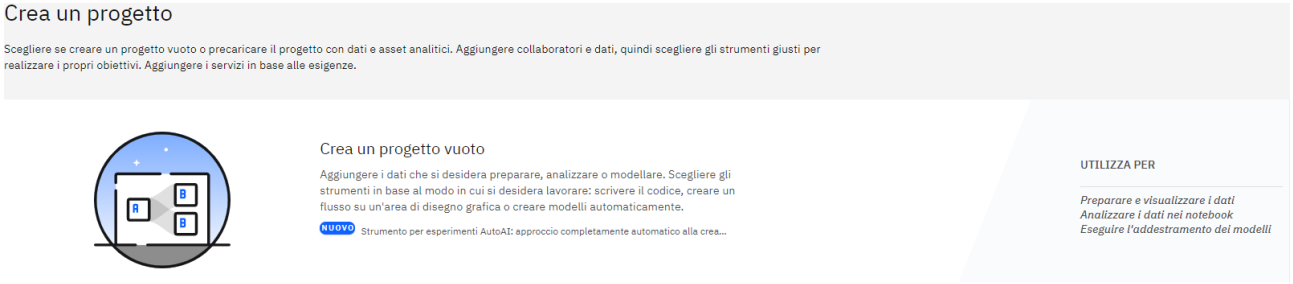
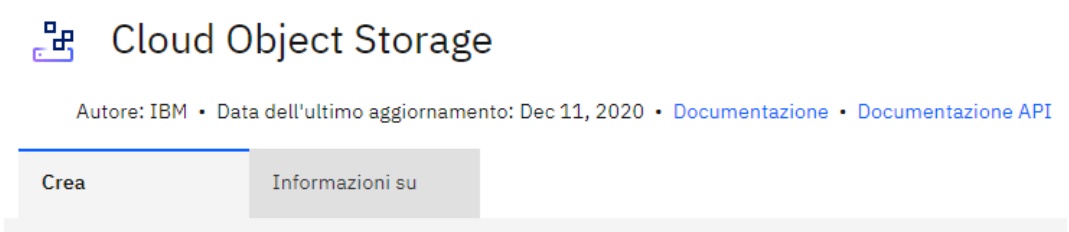


Figura 144 Creazione progetto Watson Studio

Dopo la creazione di un progetto, **IBM** richiede all'utente di impostare un'istanza di archiviazione degli oggetti. Ai fini di questo esperimento è stato selezionato il **Cloud Object Storage** standard di IBM in quanto gratuito e accessibile a tutti.



Una volta preparato il Cloud Storage, si procede ad importare le immagini tramite drag and drop.

Tagging

IBM consente all'utente di selezionare tra due tipi di progetto:

- **Classificazione** (per fotografie con singola etichetta, simili agli esperimenti 1 e 2 condotti con Google e Microsoft)
- **Object detection** (per fotografie con etichetta multipla, simile all'esperimento 3 condotto con Google e Microsoft).

In questo caso effettueremo Object detection e tratteremo graffi e buchi come **oggetti** interni alla struttura di un **dado**.

Le modalità con la quale **IBM** consente di taggare le proprie immagini è leggermente meno **pratica** di quella vista con Google o Microsoft. Il sistema ha bisogno di molto tempo per caricare e

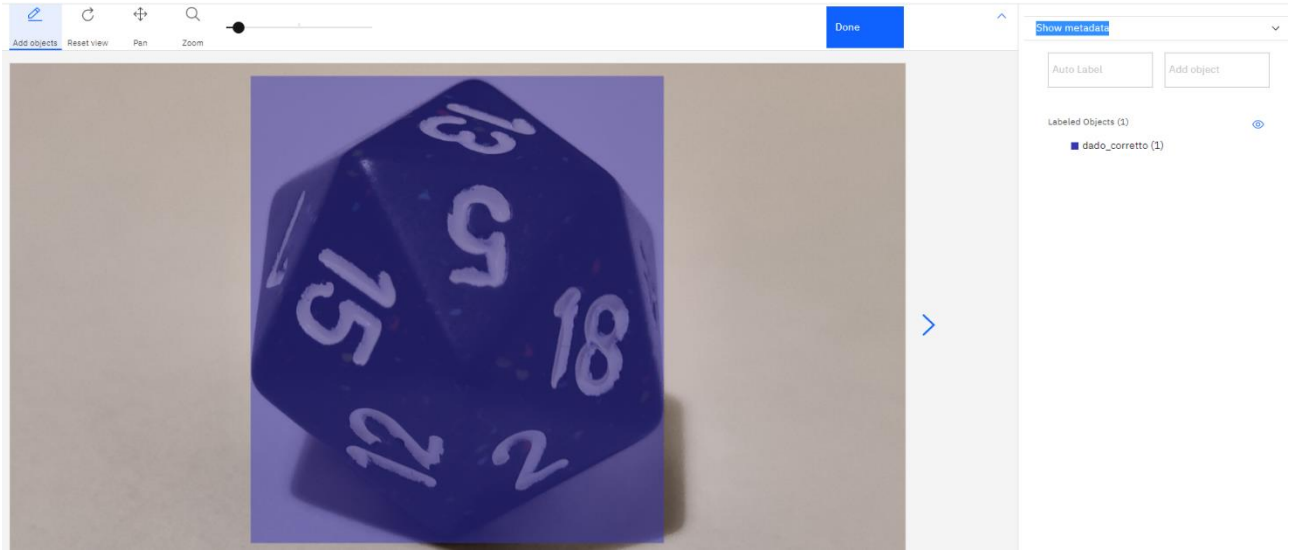


Figura 145 Taggare le immagini con watson studio

importare le immagini all'interno del modello, i tempi di attesa sono molto elevati rispetto a quelli visti con Microsoft e Google. Quando si desidera taggare un immagine, il sistema zoomma in maniera automatica all'interno della stessa, questo richiede al cliente continui zoom-out durante la fase di tagging delle immagini. No shorcut per passare all'immagine successiva (come visto con microsoft), qui occorre cliccare manualmente col mouse.
Nel sistema IBM l'utente racchiude con il classico box rettangolare tutti gli oggetti presenti in un'immagine. Non è presente riconoscimento automatico come con Microsoft, i tempi di **convalida** di un tag sono molto più elevati di quelli analizzati fino ad ora.

Taggare 227 foto con Google: **2 ore**

Taggare 227 foto con Microsoft: **30 minuti**

Taggare 227 foto con IBM: **3 ore**

Addestramento

I tempi di addestramento del sistema IBM sono stati rapidi rispetto a quelli di Google ma leggermente più lenti rispetto a quelli di Microsoft. Il modello è stato addestrato in **1 ora e 15 minuti** (relativamente alla versione di prova disponibile). L'addestramento è **completamente gratuito** ma NON fornisce statistiche in merito alla precision-recall di tutte le **etichette** utilizzate. Per calcolare precision e recall del sistema ho effettuato un controllo di tipo manuale basandomi sullo stesso numero di campioni utilizzati anche dalle altre aziende:

		Truth data				
		Class 1	Class 2	Class 3	Classification overall	Producer Accuracy (Precision)
Classifier results	Class 1	8	0	0	8	100%
	Class 2	0	7	1	8	87.5%
	Class 3	0	1	7	8	87.5%
	Truth overall	8	8	8	24	
	User Accuracy (Recall)	100%	87.5%	87.5%		

Overall accuracy (OA): 91.667%

Figura 146 Precisione e Richiamo IBM

Su 8 immagini di **dado** il sistema non ha **mai confuso** la classificazione (100%PRECISION)

Su 8 immagini di **dado_bucato** il sistema ha confuso il buco con un graffio (87.5% PRECISION)

Su 8 immagini di **dado_graffiato** il sistema ha confuso il graffio con un buco (87.5% PRECISION)

Complessivamente la precisione del sistema risulta **PRECISION = 91.6%**

Alcune immagini di **dadi corretti** interpretati dal sistema:

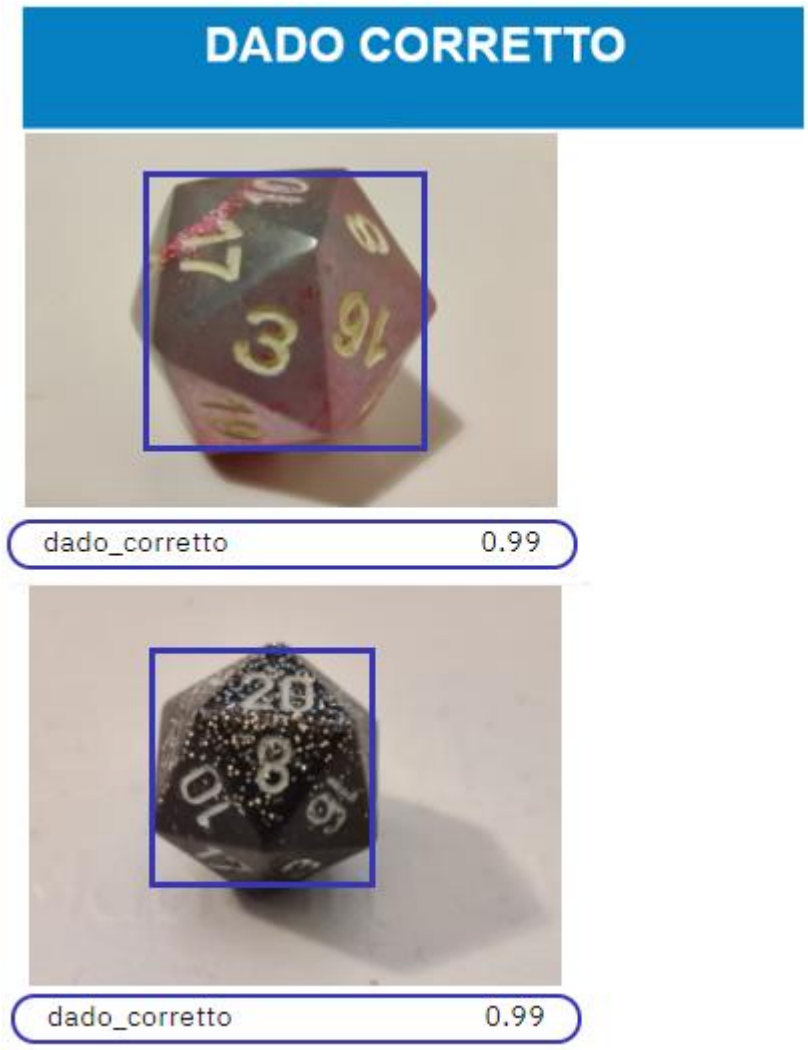


Figura 147 Il sistema tenta di riconoscere dadi corretti

Alcune immagini di **dadi graffiati** interpretati dal sistema:

Figura 148 Il sistema tenta di riconoscere dadi graffiati

Alcune immagini di **dadi bucati** interpretati dal sistema:

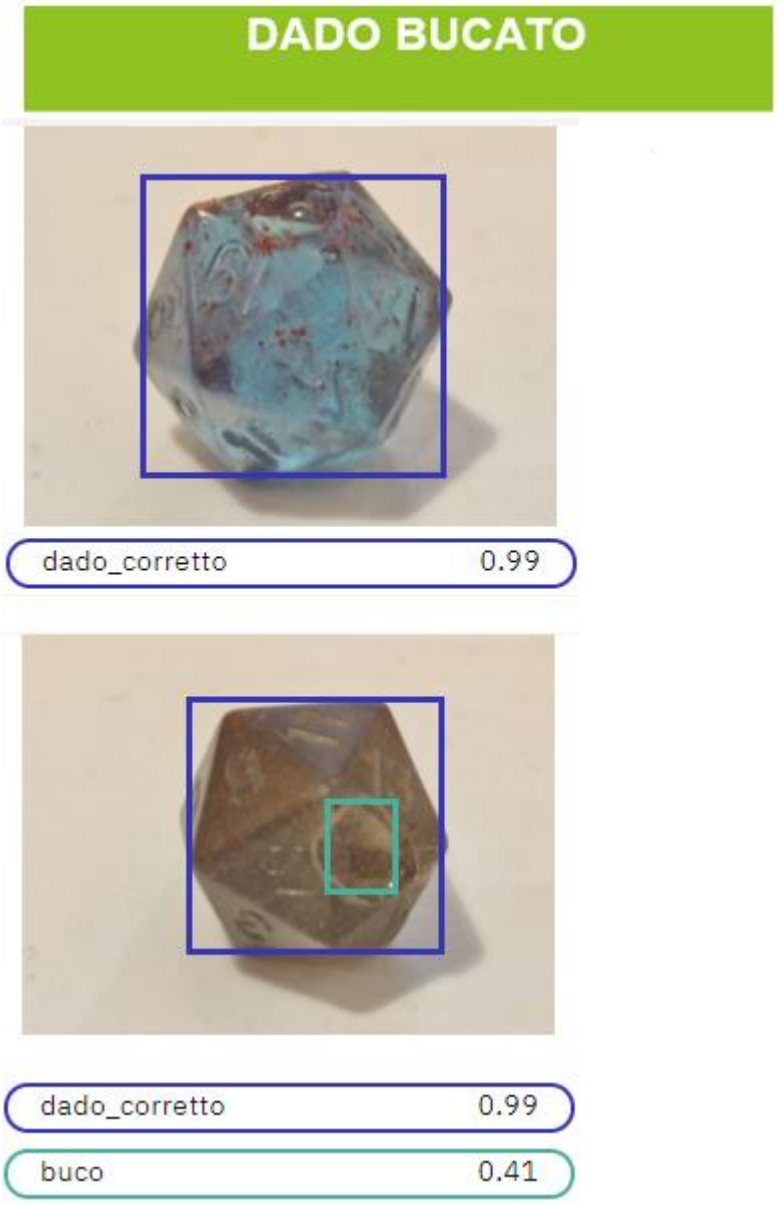


Figura 149 Il sistema tenta di riconoscere dadi bucati

Alcune immagini di **gruppi di dadi** interpretati dal sistema:

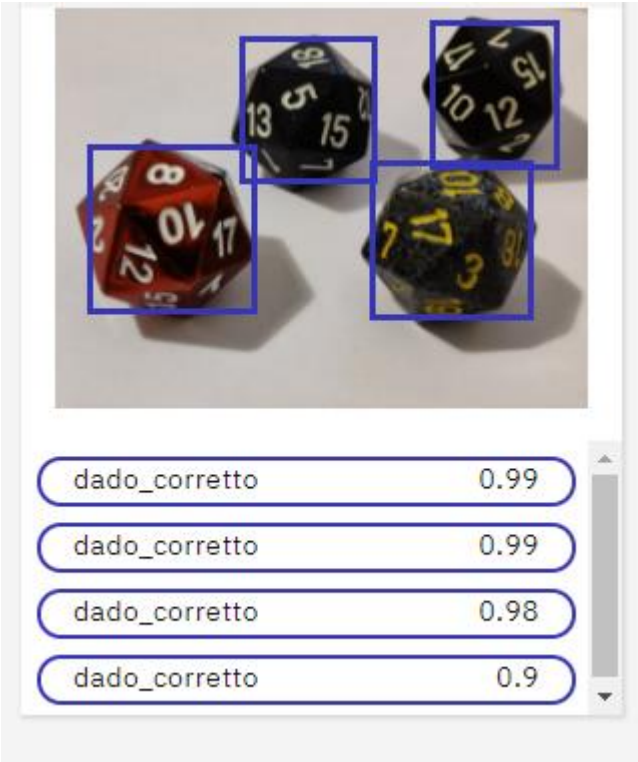


Figura 150 Il sistema tenta di riconoscere gruppi di dadi

Alcune immagini **fuori contesto** rispetto al dataset di training:

Figura 151 Il sistema tenta di riconoscere dadi in contesti differenti

Analisi e commento finale

Rispetto a Google e Microsoft fa molta più fatica ad effettuare riconoscimenti, probabilmente richiede un fine tuning particolare oppure un dataset più rappresentativo. In compenso, i valori di confidenza sul riconoscimento di dadi_corretti sono molto più elevati, mentre quelli di riconoscimento di graffi o buchi sono molto bassi oppure assenti. I valori di riconoscimento all'interno di un **contesto differente** da quello fornito nel train sono molto buoni e i valori di confidenza sono abbastanza elevati. Il tool di IBM effettua delle analisi più **high-level** e fa **molta meno confusione** del modello di Google e Microsoft quando si tratta di riconoscere gruppi di dadi. Da quello che si vede all'interno dei risultati del primo gruppo di dadi, i dadi corretti sono stati tutti identificati correttamente con valori elevatissimi di precisione. Nel secondo gruppo di dadi il sistema ha performato bene, ma non ha riconosciuto graffi o buchi. Nel complesso il sistema IBM sembra quello in grado di astrarre di più i concetti di contesto e gruppo rispetto agli altri.

3.4 **YOLO** OBJECT DETECTION **OPEN SOURCE**

Yolo – You only look once è un algoritmo di **rilevamento di immagini** utilizzato nel campo della object detection a partire dal 2016. Si tratta di una delle tecniche più moderne per la object detection e consente analisi in tempo reale di immagini e video. La differenza rispetto agli altri algoritmi di detection si trova nel modo in cui questo algoritmo funziona: nel passato, gli algoritmi di object detection erano caratterizzati da liste di regole e sliding windows che cercavano di applicare approcci "bruteforce" per capire se un oggetto era presente o meno in una fotografia. L'immagine in input veniva controllata dall'angolo in basso a sinistra all'estremo più basso, cercando l'oggetto da classificare. YOLO adotta un approccio completamente diverso, non si tratta di un classificatore di immagini tradizionale che scorre con una finestra l'immagine, ma di un algoritmo che **in una sola mossa** divide l'immagine in una **griglia 13x13**. Ognuna di queste celle è responsabile di effettuare la previsione di **5 rettangoli adiacenti** che descrivono la presenza di un oggetto e la sua probabile **classe di appartenenza** per un totale di **845 rettangoli per immagine**. In base al valore di confidenza ottenuto e alla probabilità di appartenenza ad una classe, tali rettangoli saranno più **spessi** o più sottili. Definendo uno score anche per ogni rettangolo, YOLO è in grado di combinare questo risultato con le intersezioni dei rettangoli che reputa più interessanti (con un valore di **threshold**) ottenendo un valore di accuratezza finale. Il vantaggio di YOLO è passare sull'immagine di input **una sola volta** ed effettuare le 845 previsioni (rettangoli) allo stesso momento.
Per questo esperimento verrà utilizzato il repository **Darknet** che rappresenta una delle possibili reti di **implementazione** dell'algoritmo YOLO.

Dataset

Per garantire un confronto equo con le aziende di Google, Microsoft e IBM verranno utilizzate le stesse immagini proposte nei precedenti esperimenti. Il dataset in esame è quindi il classico dataset composto da 227 immagini taggate però con un sistema esterno.

Importare le Immagini

In questo specifico caso NON verrà utilizzata nessuna interfaccia grafica, ma si utilizzerà l'interfaccia a riga di comando, codice Python e la libreria YOLO per il riconoscimento di oggetti.

Per taggare le immagini è possibile procedere in due modi differenti:

-Indicare lato coding tutte le coordinate dei rettangoli che rappresentano ogni oggetto o difetto all'interno di un'immagine. (Dispendioso in termini di tempo e poco pratico)

-Sfruttare l'ausilio di un software esterno come **LabelImg** (https://tzutalin.github.io/labelImg/) per annotare tutte le immagini che verranno utilizzate nella fase di train del modello.

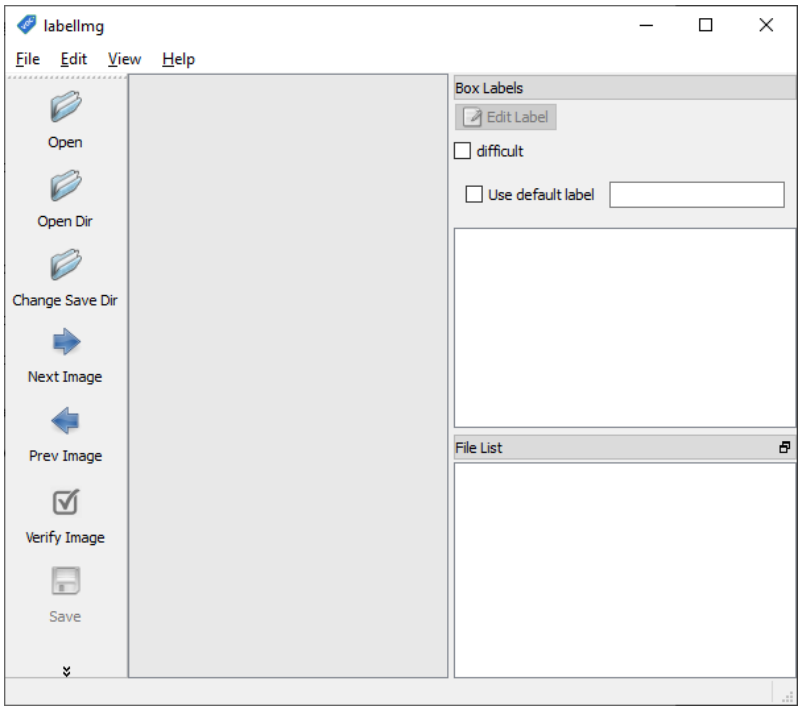


Figura 152 Taggare le immagini con LabelImg

Dopo aver selezionato la cartella contenente le immagini da taggare è necessario selezionare i settaggi per la generazione di immagini per la **libreria Yolo.** Ogni singolo produttore o software open source richiede infatti un particolare formato per le immagini taggate che non è valido solitamente in tutti i casi.

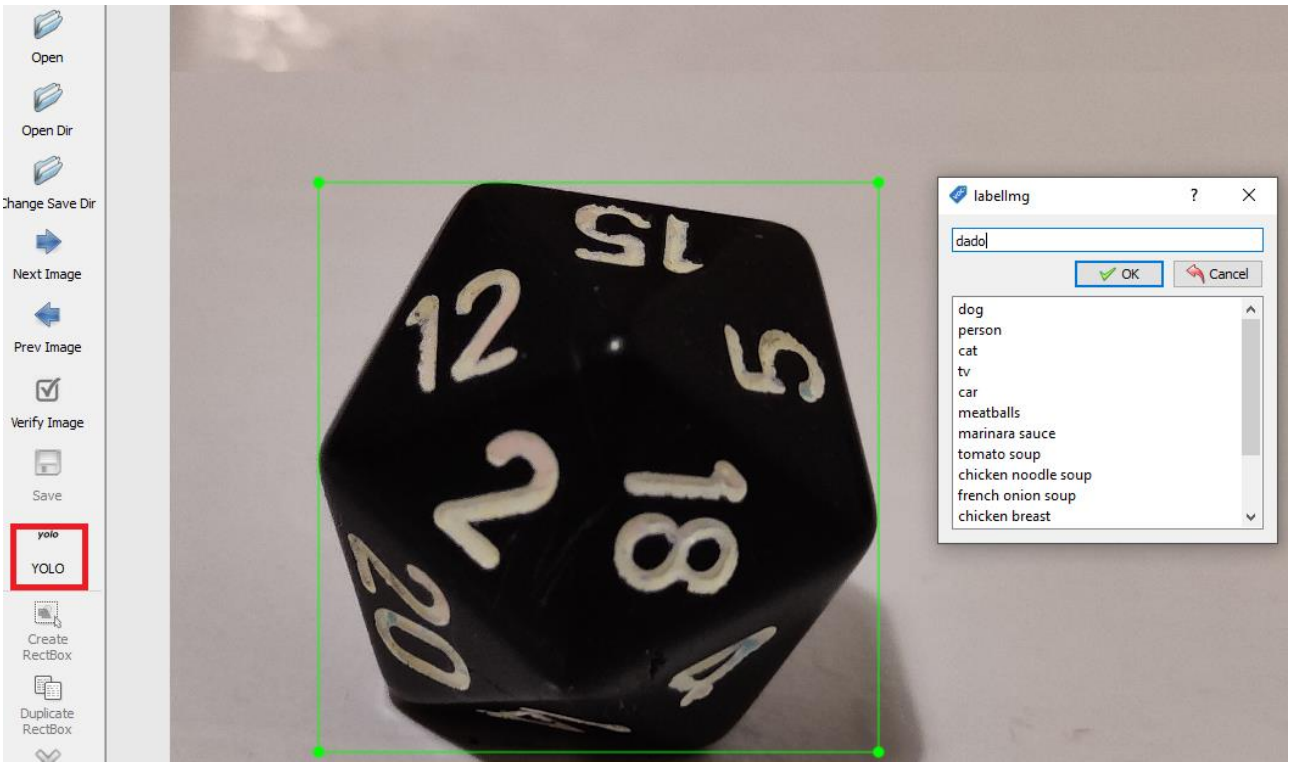


Figura 153 Interfaccia LabelImg e classi

I tempi di attesa per importare le immagini ed efettuare il tagging sono bassissimi grazie al fatto che il tool **labelImg** lavora solamente in locale e non richiede upload di nessuna risorsa nel cloud. Il software si collega direttamente alla directory dove sono presenti i file da taggare e consente all'utente di effettuare lo zoom sulle immagini, aggiungere classi e scorrere tra un immagine e l'altra. Rispetto alle soluzioni proposte dalle grandi aziende, il tagging è stato decisamente rapido anche se poco supportato da strumenti di auto-labeling o riconoscimento parziale delle immagini. Meglio di Google e IBM ma ancora inferiore al comodissimo strumento offerto da Microsoft.

Taggare 227 foto con Google: **2 ore**

Taggare 227 foto con Microsoft: **30 minuti**

Taggare 227 foto con IBM: **3 ore**

Taggare 227 foto con LabelImg : **40 minuti**

Dopo aver terminato la fase di tagging delle immagini sono stati generati 227 file .txt con la seguente struttura:

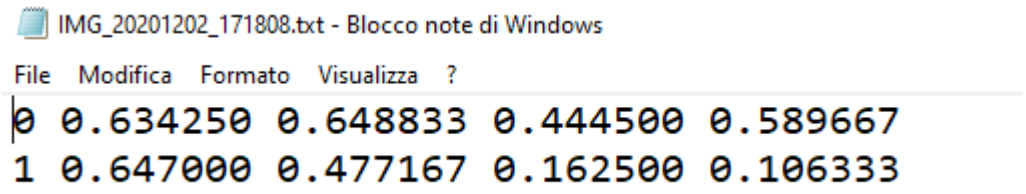


Figura 113 Struttrura del file con le coordinate relative al tagging di una fotografia

Il primo numero rappresenta **l'identificatore** della classe che parte da 0 fino al numero che rappresenta l'indice delle classi definite dall'utente in un apposito file classes.txt

Tutti gli altri numeri rappresentano le **coordinate** del **bounding box** che è stato generato grazie all'applicazione. Le classi presenti nel file.txt sono state generate dall'applicazione; durante il training del modello verranno utilizzate le classi 0,1,2 che rappresentano rispettivamente un dado, un buco e un graffio.

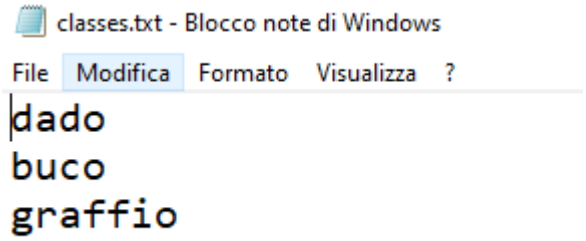


Figura 154 Classi dell'esperimento

Addestramento

Per questo esperimento utilizzerò il servizio gratuito di **Google Collab** (https://colab.research.google.com/notebooks/intro.ipynb#recent=true) che consente di eseguire **script Python** gratuitamente per 12 ore. In questo modo sarà possibile sia scrivere codice Python ed utilizzare le GPU offerte da Google (con lo svantaggio di perdere tutti i progressi eseguiti dopo le prime 12 ore di utilizzo).

Ho affiancato al sistema Google Collab il mio personale account di **Google Drive** per caricare le immagini che verranno elaborate dall'algoritmo:

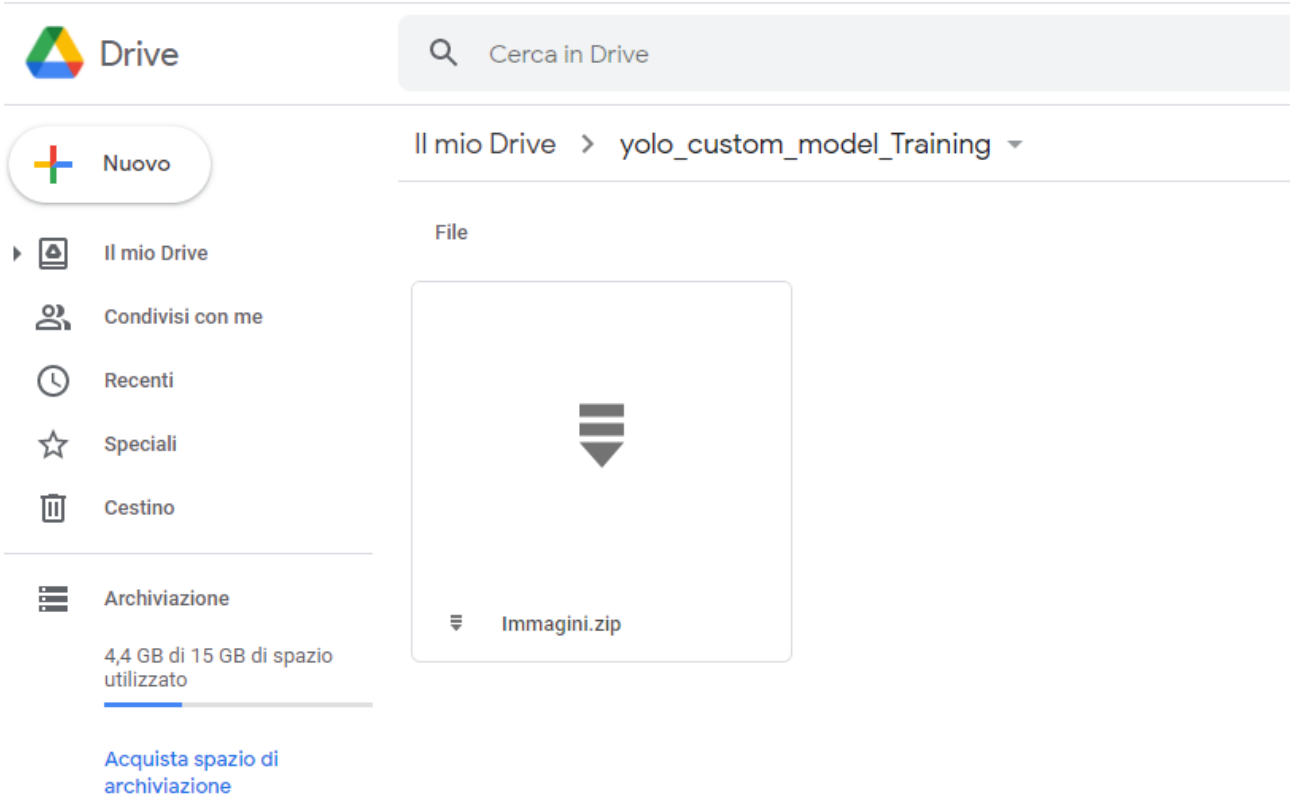


Figura 155 Folder del progetto Google Drive

Nelle impostazioni del NoteBook Google specifico di voler utilizzare l'Accelerazione GPU per rendere la computazione più rapida:

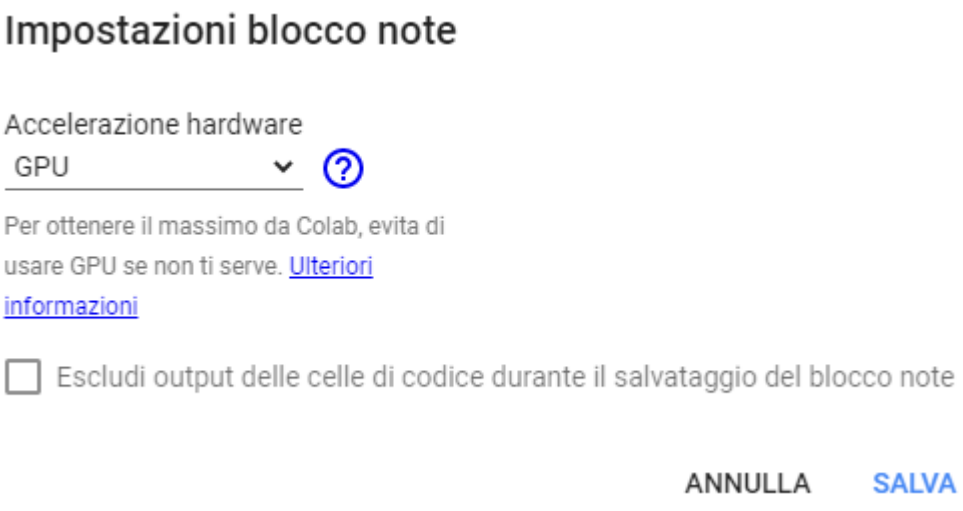


Figura 156 Accelerazione hardware - GPU

Codice e file di configurazione

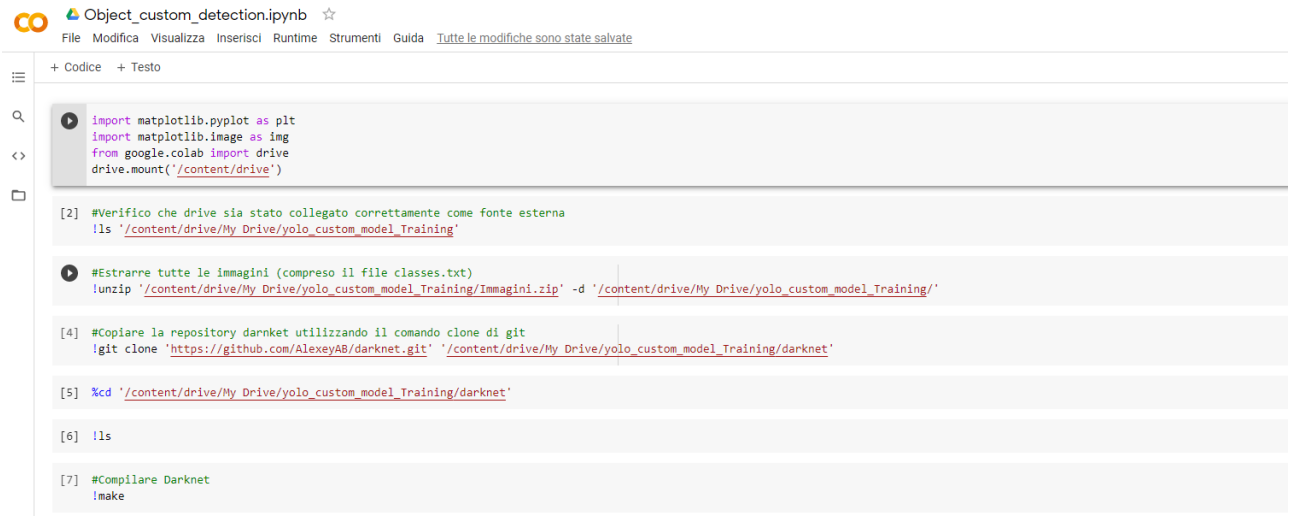


Figura 157 Notebook Python e codice

```
import matplotlib.pyplot as plt
import matplotlib.image as img
from google.colab import drive
drive.mount('/content/drive')
```

```
#Verifico che drive sia stato collegato correttamente come fonte esterna
!ls '/content/drive/My Drive/yolo_custom_model_Training'
```

Come prima cosa occorre collegare un account di Google Drive che servirà come fonte esterna dalla quale il notebook Google Collab recupererà le **immagini** e le rispettive **annotazioni**.

```
#Estrarre tutte le immagini (compreso il file classes.txt)
!unzip '/content/drive/My Drive/yolo_custom_model_Training/Immagini.zip' -
d '/content/drive/My Drive/yolo_custom_model_Training/'
```

Dopo aver collegato il nostro account occorre unzippare tutte le immagini cariate e copiare la repository Darknet direttamente col comando **Git clone** dentro ad una qualsiasi cartella. Darknet rappresenta **un'implementazione** dell'algoritmo YOLO e sarà utilizzata per questo esperimento.

```
#Copiare la repository darnket utilizzando il comando clone di git
!git clone 'https://github.com/AlexeyAB/darknet.git' '/content/drive/My Driv
e/yolo_custom_model_Training/darknet'
```

Una volta ottenuta l'implementazione Darknet occorre **compilarla** con il comando **make**.

```
#Compilare Darknet
%cd '/content/drive/My Drive/yolo_custom_model_Training/darknet'
!make
```

Dopo la compilazione sarà disponibile nel folder **darknet** un **Makefile** da configurare settando ad 1 i flag di **GPU, CUDNN e OPENCV** per istruire il sistema all'utilizzo della GPU (gratuita offerta da Google Collab)

Makefile - Blocco note di Windows

File Modifica Formato Visualizza ?

```
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores)
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)
# set ZED_CAMERA=1 to enable ZED SDK 3.0 and above
# set ZED_CAMERA_v2_8=1 to enable ZED SDK 2.X
```

Figura 158 Modifica al Makefile della rete DARKNET

La fase successiva richiede la divisione tra **train e test** delle immagini fornite in input a **Google Drive.** È possibile ottenere questo risultato con la scrittura di due script Python in grado di suddividere i nostri dati in Train e Test.

Per l'addestramento sono richiesti **5 files differenti** che devono essere creati:

1) **classes.txt:** contiene le 3 classi utilizzate per l'esperimento e corrisponenti agli indici utilizzati nella fase di tagging con il tool **labelImg**.

Nel nostro caso il file classes.txt conterrà 3 righe:

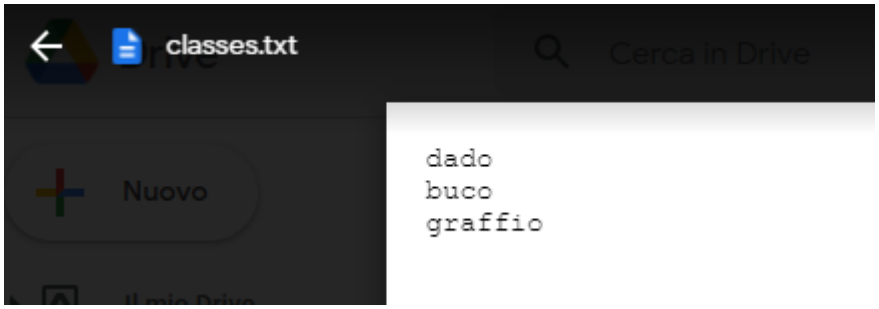


Figura 159 File classes.txt

2) **train.txt e test.txt**: contengono la lista delle immagini utilizzate nella fase di training e di testing.

Per questo esperimento ho tenuto un rapporto 80/20 tra train e test.

```
import os

full_path_to_images = 'Immagini'
os.chdir(full_path_to_images)
p = []

for current_dir, dirs, files in os.walk('.'):
    for f in files:
        if f.endswith('.jpg'):
            path_to_save_into_txt_files = full_path_to_images + '/' + f
            p.append(path_to_save_into_txt_files + '\n')

p_test = p[:int(len(p) * 0.20)]
p = p[int(len(p) * 0.20):]

with open('train.txt', 'w') as train_txt:
    for e in p:
        train_txt.write(e)

with open('test.txt', 'w') as test_txt:
    for e in p_test:
        test_txt.write(e)
```

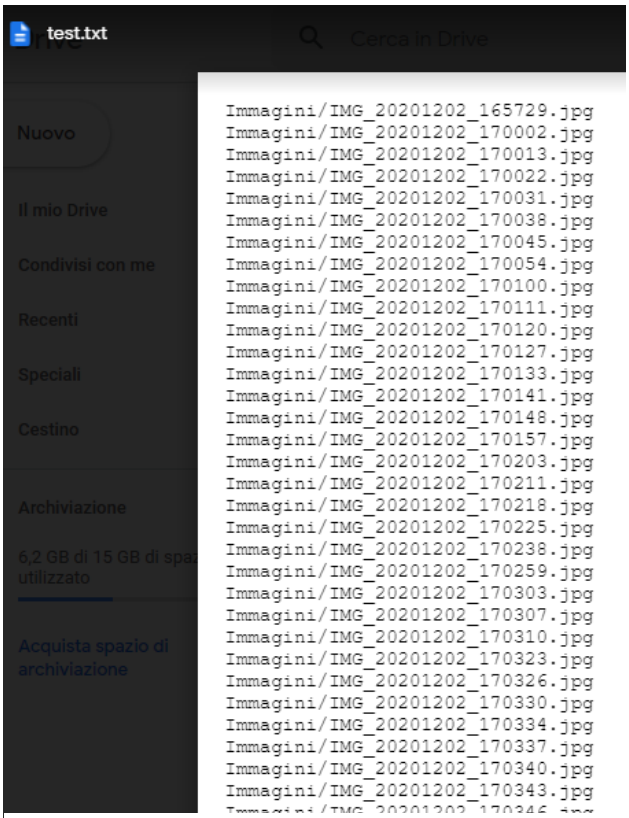


Figura 160 File test.txt e codice per la suddivisione in train e test

3) **labelled_data.data e classes.names:** contengono le informazioni necessarie durante la fase di addestramento per reperire i path delle immagini relative al **train**, al **test** e ai nomi delle classi. **Classes.names** è praticamente lo stesso del file.txt con un estensione diversa. Lo script rende il tutto più scalabile ed adattabile anche a classi più numerose.

```
"""classes.names"""
full_path_to_images = 'Immagini'
c = 0

with open(full_path_to_images + '/' + 'classes.names', 'w') as names, \
     open(full_path_to_images + '/' + 'classes.txt', 'r') as txt:

    for line in txt:
        names.write(line)
        c += 1


"""labelled_data.data"""

with open(full_path_to_images + '/' + 'labelled_data.data', 'w') as data:
    data.write('classes = ' + str(c) + '\n')
    data.write('train = ' + full_path_to_images + '/' + 'train.txt' + '\n')
    data.write('valid = ' + full_path_to_images + '/' + 'test.txt' + '\n')
    data.write('names = ' + full_path_to_images + '/' + 'classes.names' + '\n')
    data.write('backup = backup')
```

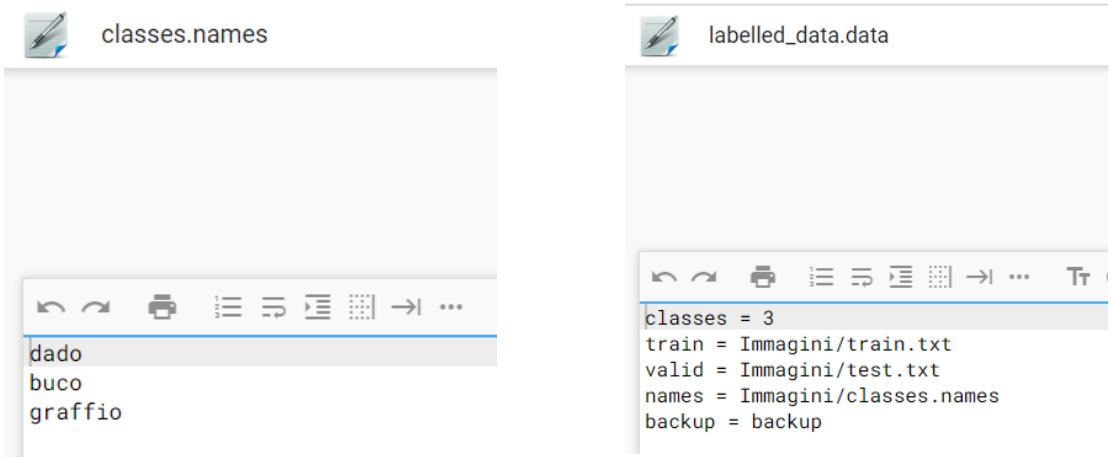


Figura 161 classes.names e labelled_data.data con codice per la creazione dei file

Per il passo successivo è stato scaricato il modello **darknet53.conv.74** pre-addestrato per alcune classi che verrà cambiato per adattarlo al nostro sistema.
Insieme ai file **labelled_data.data**, **train.txt, text.txt** e **classes.names**, YOLOv3 necessita anche di un file di configurazione
`yolov3_custom.cfg`

Tale file, incluso nel codice di base di darknet, deve essere **modificato** (o copiato come in questo caso in un altro file `yolov3_custom.cfg`) e contiene tutti i parametri di allenamento e gli **iper parametri** necessari per il conseguimento dell'operazione.
- **BATCH:** Il parametro **batch** indica la dimensione del batch utilizzata durante l'addestramento. Il processo di addestramento prevede l'aggiornamento **iterativo** dei pesi della rete neurale in base al numero di errori commessi sul set di dati di addestramento. Essendo poco pratico utilizzare contemporaneamente **tutte** le immagini nel training set, per aggiornare i pesi si sceglie un **sottoinsieme** di immagini che viene utilizzato in un'iterazione; tale sottoinsieme è chiamato dimensione del **batch**. Quando la dimensione del batch è impostata su 64, significa che vengono utilizzate 64 immagini in un'unica iterazione per aggiornare i parametri della rete neurale.

- SUBDIVISIONS: Anche scegliendo batch=64 per addestrare la rete neurale si potrebbe non avere una GPU con memoria sufficiente per computare 64 immagini in una sola iterazione. Modificando l'iperparametro subdivision è possibile elaborare una **frazione** della dimensione della batch in una sola volta sulla GPU.

Solitamente si inizia l'addestramento con il numero di suddivisioni = 1 e, se si ottiene un errore di memoria esaurita, si aumenta il parametro delle suddivisioni per multipli di 2 (es.2, 4, 8, 16) finché l'addestramento non procede con successo. E' importante scegliere un numero di **batch** e **suddivisioni** divisibili, in quanto la GPU elaborerà il numero di **batch / suddivisione** di immagini in qualsiasi momento. Durante la fase di test, sia batch che subdivision sono impostati su 1.

Attualmente con il mio numero di immagini (227) ho scelto un **batch** = 8 e **subdivisions** = 2.

- MAX_BATCHES: rappresenta il numero di **iterazioni** che verranno effettuate in fase di training. Solitamente si sceglie un valore pari al numero di **classi usate per l'addestramento * 2000**.
Nel nostro caso quindi sceglieremo 6000 dato che abbiamo solamente 3 classi.

- WIDTH, HEIGHT E CHANNELS: Tre parametri che specificano la dimensione delle immagini in input e il numero di canali da processare (channels = 3 indica che vogliamo processare i 3 canali RGB). Tutte le immagini vengono **ridimensionate** prima di far partire l'addestramento (valori di default 416x416 pixel). Ovviamente aumentando le dimensioni è possibile ottenere risultati migliori ma si aumentano notevolmente i tempi di addestramento.

- **MOMENTUM E DECAY**: Parametri per l'aggiornamento dei pesi. Il **momentum** è una costante moltiplicativa che rappresenta una penalizzazione per grosse variazioni di **peso** durante l'addestramento della rete. Nel caso di **overfitting** è necessario modificare questi parametri.

Nel mio caso i valori sono rimasti costanti a quelli di default (momentum = 0.9 e decay = 0.0005)

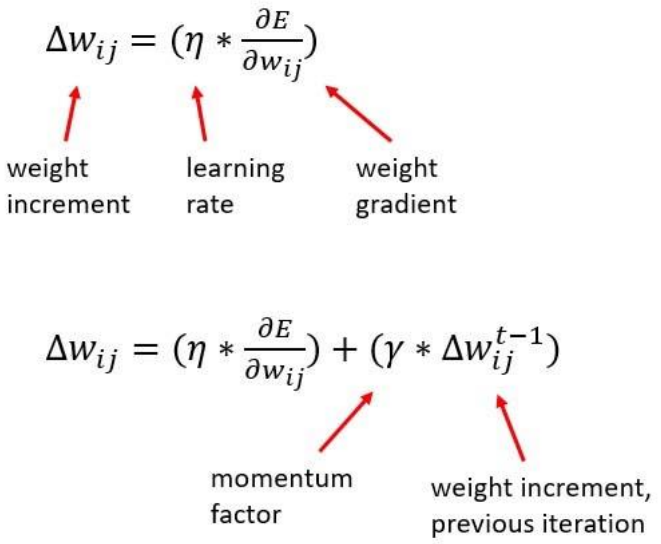


Figura 162 Iperparametri e relazioni con l'incremento dei pesi nelle reti neurali

- LEARNING RATE, STEPS, SCALE, BURN_IN: Parametri che controllano **l'aggressività** dell'apprendimento nel tempo. Il learning rate è un valore solitamente compreso tra 0.01 e 0.0001 (nell'esperimento corrente vale 0.001). All'inizio del processo di learning della rete abbiamo zero informazioni quindi il tasso di apprendimento deve essere **molto elevato**. Vedendo molte immagini in input, tale valore di learning rate va **ridotto nel tempo** oppure avremmo oscillazioni di pesi troppo ampie. Qusta politica di riduzione dell'aggressività è definita dal parametro **steps** che indica **il numero di iterazioni** per la quale la velocità di apprendimento (0.001) resterà costante prima di essere (raggiunto il valore STEPS=X) poi moltiplicata per il valore **scale** per ottenere una nuova velocità di apprendimento.

E’ possibile specificare diversi scale e quindi definire più passaggi per l’apprendimento della rete. Il parametro **BURN_IN** rappresenta un breve periodo all’inizio dell’addestramento in cui si tende a mantenere **basso** il **learning rate** del sistema. È stato provato empiricamente che la presenza di un learning rate basso per un breve periodo di tempo all’inizio del sistema facilita l’apprendimento e la stabilizzazione dei pesi.

- DATA AUGMENTATION: Nello stesso file è possibile specifcare la volontà di effettuare data augmentation ed espandere il set di dati di partenza prima di effettuare l’addestramento della rete. I parametri da modificare in questo caso sarebbero l’**angolo** (che ruota randomicamente l’immagine di un certo valore specificato nel file di configurazione), la **saturazione** (quantità di colore), l’**esposizione** (luminosità) e l’**hue** (colore scelto in tontalità).
- **NUMBER OF ITERATIONS**: Parametro che rappresenta il numero di **iterazioni** che vogliamo eseguire sul nostro modello. Solitamente è consigliabile lanciare un training per una rete per almeno **2000*numeroDiClassi**

Occorre ricordarsi di commentare i parametri di **testing** e de-commentare i parametri di **training** durante le rispettive fasi di allenamento e test dei risultati.

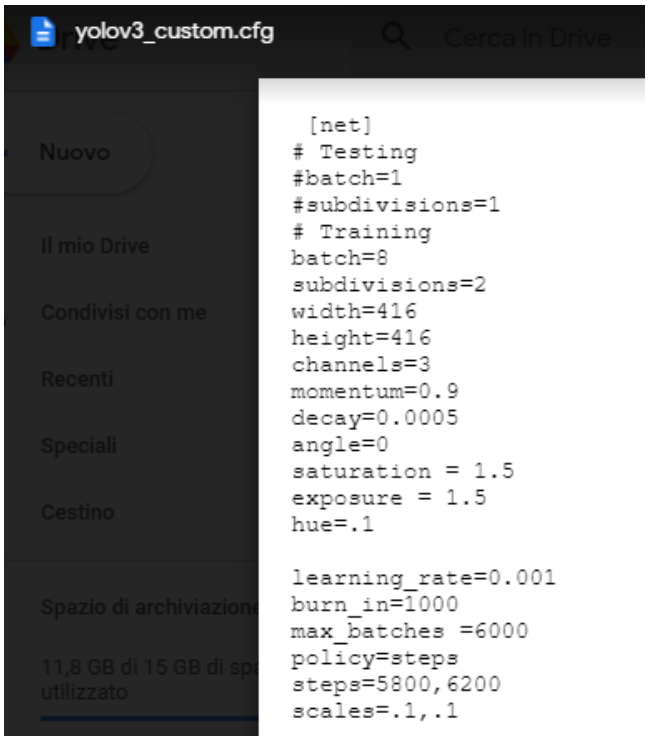


Figura 163 Configurazione del file yolov3_custom.cfg

- LAYER DELLA RETE: Essendo una rete pre-esistente occorre modificare alcuni **layer** della rete per adattarli al nostro specifico caso. La rete di partenza contava infatti 80 classi che nel nostro caso vengono ridotte a 3.

```
[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=3
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Figura 164 Modifica di alcuni nodi della rete pre-esistente

Analogamente si modifica il **numero di filtri** seguendo la formula **(numero di classi +5) *3** che nel nostro caso ci farà ottenere un numero di filtri pari a 24.

```
599 [convolutional]
600 size=1
601 stride=1
602 pad=1
603 filters=24
604 activation=linear
```

Figura 165 Modifica del numero di filtri nella rete

L'operazione va ripetuta **per ogni nodo YOLO** presente nel file di configurazione quindi 3 volte. Si identifica il nodo YOLO, si identifica il rispettivo numero di filtri (capoverso sopra) e si modifica con la formula sopra-elencata.

```
[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear


[yolo]
mask = 3,4,5
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=3
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
```

Figura 166 Modifica del numero di filtri

A questo punto può cominciare il **training** della rete con il comando:

```
#Addestramento

!darknet/darknet detector train Immagini/labelled_data.data darknet/cfg/yolo
v3_custom.cfg custom_weight/darknet53.conv.74 -dont_show
```

Questo comando richiede alla rete darknet di iniziare l'addestramento utilizzando il file labelled_data.data (che contiene tutte le informazioni necessarie alla rete a reperire le classi, il train e il test set) e il file di configurazione yolov3_custom.cfg (modificato a partire dalla versione yolov3.cfg con gli iperparametri corretti)

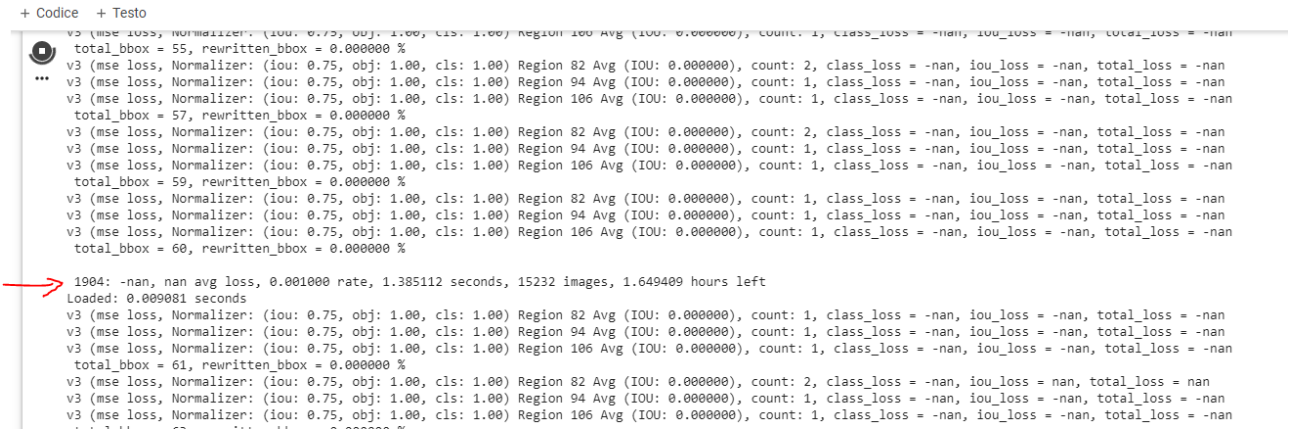


Figura 167 Addestramento in corso e numero di ore rimanenti

Nell'immagine viene mostrato il mio processo di addestramento all'iterazione numero 1984. Avendo selezionato 6000 iterazioni il tempo di training sarà simile a quello visto sui software delle grandi aziende e sarà pari a circa **2 ore e 30 minuti**. Si ricorda che la GPU è messa a disposizione da Google Collab e non si tratta di una mia GPU.

Nel caso in cui Google Collab disconnetta l'utente dalla sessione (o ci siano problemi in fase di addestramento) è possibile recuperare il lavoro in quanto nella cartella backup vengono salvati **nuovi modelli ogni 1000 iterazioni** e tutti i pesi relativi ad ogni **100 iterazioni.** Nel caso si voglia riprendere un addestramento basterà lanciare il comando:

```
#Riprendere l'addestramento con i pesi precedenti

!darknet/darknet detector train Immagini/labelled_data.data darknet/cfg/yolo
v3_custom.cfg backup/yolov3_custom_last.weights -dont_show
```

Tutti I modelli generati sono salvati nella cartella **backup** di Google Drive e possono essere scaricati in locale o utilizzati direttamente dal Notebook per effettuare previsioni con il comando:

```
!darknet/darknet detector test Immagini/labelled_data.data darknet/cfg/yolov3_custom.cfg backup/yolov3_custom_last.weights -dont_show
```

Prima di lanciare questo comando occorre agire sul file di configurazione **yolov3_custom.cfg** modificando opportunamente i settaggi in modo che abbiano commentato il Train e de-commentato il Test.

Nel caso in cui si desideri ri-addestrare un qualsiasi modello occorrerà nuovamente cambiare in modalità **training**.

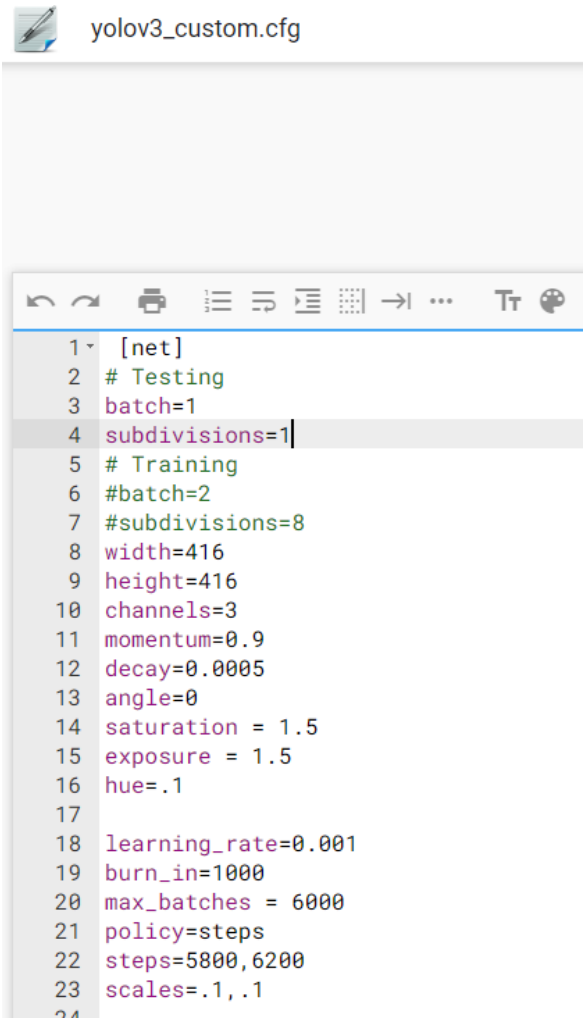


Figura 168 Passare tra la fase di Training e Testing

Ecco i risultati ottenuti :

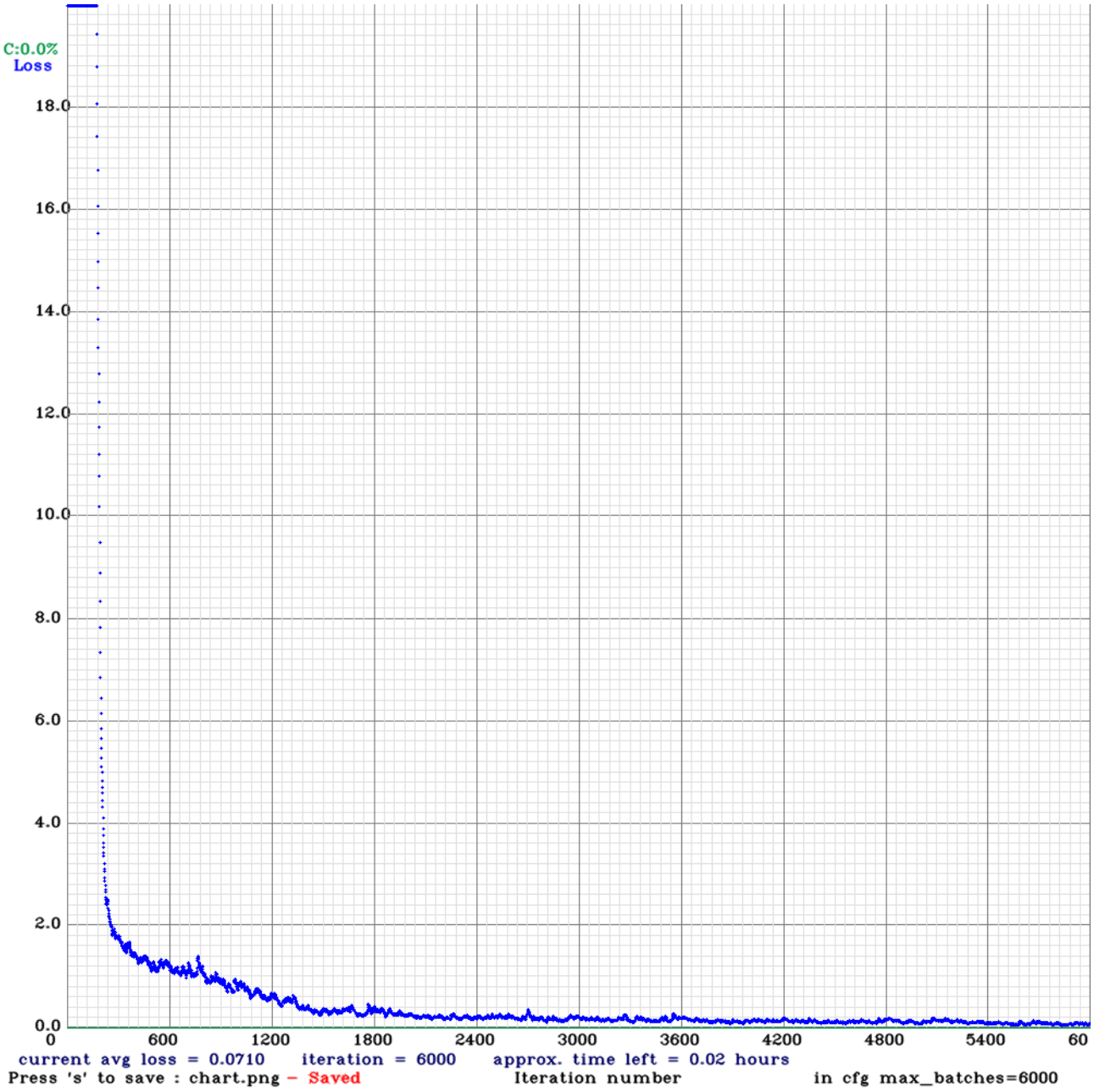
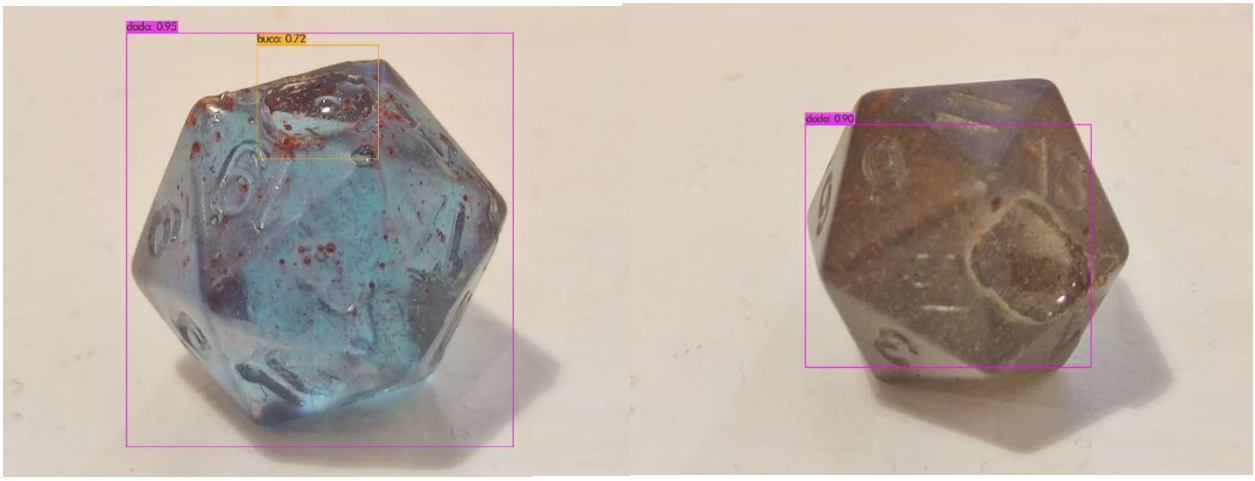


Figura 169 Perdite in relazione al numero di iterazioni

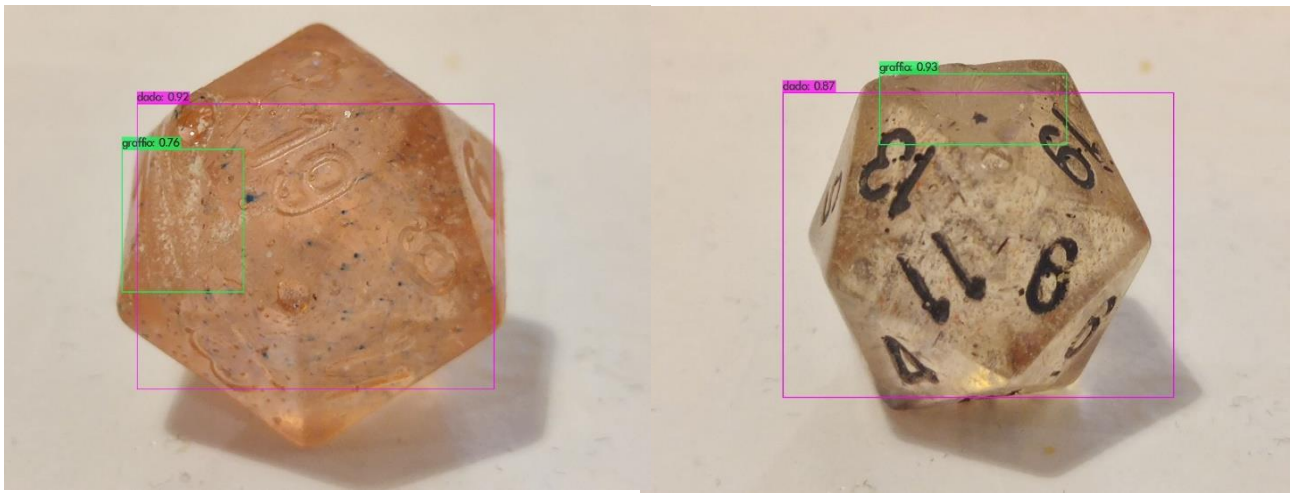
Classificazione di dadi bucati:



Prima immagine: **Dado** 95% **Buco** 72%
Seconda immagine: **Dado** 90%

Figura 170 Il sistema cerca di ricnoscere dadi bucati

Classificazione di dadi graffiati:



Prima immagine: **Dado** 92% **Graffio** 76%
Seconda immagine: **Dado** 87% **Graffio** 93%

Figura 171 Il sistema cerca di riconoscere dadi graffiati

Classificazione di dadi corretti:

Figura 172 Il sistema cerca di ricnoscere dadi corretti

Prima immagine: **Dado** 98%
Seconda immagine: **Dado** 73%

Classificazione di gruppi di dadi:

Prima immagine: Rilevati **4 dadi** con valori di confidenza pari a 97% 56% 78% e 99%

Seconda immagine: Rilevati **7 dadi** ed un buco. Valori di confidenza oscillano tra il 50% e 95%

Figura 173 Il sistema cerca di riconoscere gruppi di dadi

Classificazione di dadi fuori dal contesto usuale:

Figura 174 Il sistema cerca di riconoscere dadi in contesti differenti

Analisi e commento finale

L'implementazione del modello YOLO grazie all'utilizzo della rete Darknet ha ottenuto ottimi risultati soprattutto per i **gruppi** di dadi. La possibilità di configurare manualmente tutti gli **iperparametri** della rete garantisce all'utente una maggiore **libertà** e la possibilità di massimizzare alcuni aspetti della rete non adattabili durante l'utilizzo dei software proposti dalle grandi aziende. Purtroppo, per ottenere questo risultato i tempi sono stati molto lunghi: ho dovuto eseguire 15 addestramenti da 2 ore e 30 e modificare i parametri moltissime volte. In uno scenario industriale adottare questo approccio potrebbe risultare sconveniente a causa dei tempi ristretti entro la quale è richiesto un risultato. È stato comunque un modo molto interessante per apprendere come funziona davvero una rete neurale e come performano questi software di object classification. La rete non ha mostrato risultati eccellenti per quanto riguarda lo scontornamento di immagini: spesso i box rettangolari sono troppo grandi o troppo piccoli.

CAPITOLO 4 – CRISP-DM E IBM MAXIMO VISUAL INSPECTION

Negli esperimenti analizzati fino a questo momento non è stato mai considerato l'aspetto industriale e il valore aggiunto che uno strumento di visual inspection deve essere in grado di portare ad un cliente. I costi che un'azienda deve sostenere nel caso in cui un proprio prodotto venga ritirato dal mercato sono infatti sia di **brand reputation** che di **rielaborazione** del pezzo difettoso. Con l'obiettivo di comprendere più a fondo i **processi** che si racchiudono dietro ad un caso di studio **reale** in ambito visual inspection verrà presa in considerazione il processo **CRISP-DM.**

4.1 Modello **CRISP-DM**

CRISP-DM è un acronimo che si riferisce ai termini "*Cross-Industry Standard Process for Data Mining"* e si tratta di una serie di **metodologie** e **tecniche** attuabili in campo computer science che cercano di **standardizzare** l'approccio con la quale si può affrontare un problema di data mining.

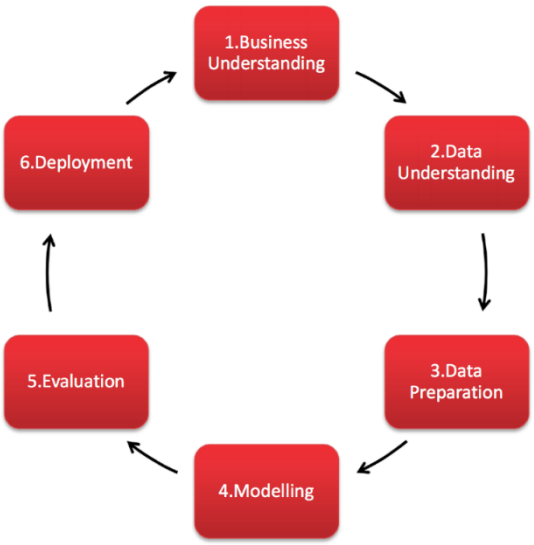


Figura 175 Ciclo di vita del modello CRISP-DM

Il modello del ciclo di vita è costituito da sei fasi la cui sequenza non è rigida ma facilmente adattabile anche per quei progetti per la quale è necessario spostarsi avanti e indietro secondo necessità. Il modello CRISP-DM è flessibile e può essere personalizzato facilmente consentendo di creare un modello di data mining che si adatta alle esigenze particolari del problema in esame.

1. Business understanding

Il primo passo nel processo CRISP-DM è capire cosa si vuole ottenere da una **prospettiva aziendale** e formulare un vero scenario di quello che si andrà a fare. L'obiettivo di questa fase è scoprire fattori importanti che possono influenzare i risultati del progetto. Ignorare questo passaggio può causare un grande dispendio di **risorse** nel futuro, quindi è bene prendere in considerazione tutte le eventualità in questa fase. Nel primo punto potrebbe anche essere necessario rispondere a domande correlate come "*Introdurre questo sistema avrà impatto dal punto di vista economico per l'azienda*?". Dopo questa breve fase iniziale vanno impostati i **criteri** utilizzati per determinare il **successo** del progetto da una prospettiva aziendale. Idealmente, questi dovrebbero essere **specifici** e **misurabili**, come la "riduzione del 5% del tasso delle parti scartate dal sistema", "ottenere un certo valore di precisione" oppure "fornire informazioni utili sulla difettosità o meno di un pezzo". In questo caso, deve essere chiaro chi sta esprimendo il giudizio soggettivo. Per qualsiasi caso d'uso reale bisogna infatti avere ben chiaro qual è il **difetto** che si desidera riconoscere, occorre affidare alcune fasi (come quella di **tagging**) ad esperti di dominio che siano in grado di capire la gravità del problema desiderato e di definire rigorosamente quali sono i problemi che devono essere effettivamente considerati come difetti. Un progetto di visual inspection deve essere approfondito combinando la **sensibilità** di chi espone il problema con chi deve **tradurre** in linguaggio matematico il problema per poi risolverlo. Una volta individuati i criteri di successo del progetto vanno analizzate le **risorse** a disposizione per la buona riuscita dell'attività. In questa parte occorre **verificare** la presenza di personale specializzato (come esperti di business, supporto tecnico o esperti dei dati che saranno analizzati) ed **esaminare** tutti gli elementi che potrebbero compromettere la risoluzione del problema come il **formato** dei dati, le **risorse hardware** disponibili oppure particolari necessità del cliente. Di fondamentale importanza è stilare una **lista di requisiti** che possano sottolineare anche aspetti particolarmente spinosi come l'utilizzo libero dei dati dal punto di vista legale o eventuali **scadenze** da rispettare. È proprio in questa fase che si mostra al cliente l'aspetto **comparativo** tra **costi e benefici** che rappresenta il potenziale di business che il cliente potrebbe ottenere se decidesse di adattare il sistema.
L'output della fase di business understanding è il **project plan**, un piano che descrive efficacemente tutti i passi, la loro **durata,** le **risorse** necessarie per ottenere un certo risultato e le relative **dipendenze**. Grazie al project plan è possibile stilare una lista di **possibili tools** che siano in grado di risolvere il problema valutando allo stesso tempo i costi e i benefici dei prodotti impiegati. Nel mio specifico caso verrà utilizzato il prodotto **IBM MAXIMO VISUAL INSPECTION**.

2. Data understanding

La seconda fase del processo CRISP-DM richiede l'accesso ai dati elencati nelle risorse del progetto. La raccolta iniziale dei dati include la **decisione del formato** e il **caricamento** dei dati nello strumento incaricato di supportare la risoluzione del problema. Se i dati vengono acquisiti da più fonti bisogna considerare la loro **integrazione** in questa fase preliminare per evitare inconsistenze. È di fondamentale importanza anche verificare la **qualità** dei dati che verranno utilizzati nel progetto con l'obiettivo di capire se **tutte le casistiche** sono trattate oppure no. Registrare i problemi che si riscontrano in questa fase e le relative soluzioni raggiunte contribuirà a replicare il sistema in futuro e ad eseguire progetti simili in tempi più rapidi.

3. Data preparation

Questa è la fase del progetto in cui si decidono **quali dati** utilizzare per l'analisi. I criteri che possono essere utilizzati per prendere questa decisione includono la **rilevanza** dei dati rispetto agli obiettivi di data mining, la **qualità** dei dati e limitazioni tecniche, come le restrizioni sulla quantità di dati o sul tipo di dati. Durante la preparazione dei dati è possibile effettuare operazioni di **pre-processamento** delle informazioni attue al miglioramento della qualità dell'input con la quale si svilupperà il sistema. In ambito visual inspection potrebbe essere interessante **rimuovere automaticamente lo sfondo** delle immagini per evitare che questi componenti vadano a pregiudicare l'addestramento della rete e il riconoscimento di difetti.

4. Modelling

Come primo passo nella modellazione, occorre scegliere la **tecnica di modellazione** effettiva da utilizzare. Sebbene il tool da utilizzare sia stato scelto nella fase di "comprensione del business", in questa fase verranno adottate le scelte in merito all'impiego di un albero decisionale oppure la generazione di una rete neurale con retro propagazione. È importante modellare l'intero sistema e definire in maniera opportuna gli **input** e gli **output** di ogni tecnologia per evitare di dover raffinare di continuo la soluzione adottata. Nello specifico caso che esaminerò NON verrà effettuata direttamente una modellazione tecnologica poiché il software IBM MAXIMO VISUAL INSPECTION è stato già pre-configurato per la risoluzione di problemi di visual inspection e non richiede tuning specifici se non degli **iperparametri** in fase di addestramento. Una volta terminata la fase di **modellazione** verrà effettuato il **training** della rete e si otterranno i valori di **precisione** relativi al primo modello generato.

5. Evaluation

In questa fase si valuta in che misura il modello soddisfa gli obiettivi di business e si proverà a determinare se il modello costituisce un potenziale elemento di **guadagno** per il cliente. Un'altra opzione è testare il modello in un'applicazione reale quando i vincoli di tempo e budget lo consentono. Da questo momento in poi le fasi precedenti possono essere **ripetute** con l'obiettivo di **migliorare** le performance del modello e raggiungere i risultati attesi.
Quando il sistema è stabile e i risultati sono soddisfacenti si può passare alla fase di **deployment.**

6. Deployment

Nella fase di deployment, si determina la strategia di distribuzione del prodotto finito e si **installa** il modello finale nel luogo prestabilito.

4.2 Modello CRISP-DM applicato alla VISUAL INSPECTION

Una volta compreso il processo CRISP-DM è possibile **standardizzare** una serie di passi che guidino durante la risoluzione di un problema di visual inspection industriale:

A. Business understanding: Capire il problema dal punto di vista reale contestualizzandolo agli aspetti di **business** richiesti dal cliente.

B. Raccolta di dati: L'applicazione del modello va fatta su un **dataset** generato con la medesima **strumentazione**. Se le immagini del dataset sono prese con la telecamera X e le immagini su cui verrà effettuato il test finale sono prese con la telecamera Y, il modello non potrà funzionare. Il dato su cui si **costruisce il modello** deve provenire dallo stesso luogo e deve essere raccolto nello stesso modo per evitare il problema della **rappresentatività dei dati**.

C. Pre processamento: Si tenta di applicare un processamento delle immagini con l'obiettivo di generare dati puliti e facilmente elaborabili. Una delle operazioni che viene solitamente eseguita è quella di separazione tra il **bordo** e il **soggetto** delle immagini. Per evitare di elaborare troppi pixel contemporaneamente, in presenza di immagini molto grandi è possibile dividere il tutto in sotto-immagini, elaborare la porzione della fotografia (individuando i difetti) e poi, a partire dalle coordinate, ricomporre l'immagine nel suo complesso. Per evitare che il bordo influenzi la buona riuscita del riconoscimento si applica solitamente un primo modello di **object detection** in grado di riconoscere il "template" ad alto livello dell'oggetto ed estrarre la **regione** che lo contiene, per poi passare ad un **secondo modello** solo per il riconoscimento delle **difettosità** che utilizza tali regioni individuate come proprio input.

D. Tagging dei difetti: Riunirsi con il personale tecnico per definire in maniera **formale** la definizione di difetto per il cliente individuando il **numero** di difetti che si desidera riconoscere. In questa fase è fondamentale capire quali sono i **criteri** secondo la quale il cliente **scarterebbe** oppure no un pezzo. Ad esempio, nel caso in cui si dovessero verificare difettosità su una mela si potrebbe definire una **classificazione OK/KO** come la seguente:

Una mela va scartata se una o più combinazioni dei seguenti criteri sono vere:

- **Tutti** i difetti di lunghezza, larghezza o diametro > di 2cm
- **Macchie** di diametro > di 2cm o presenza di almeno due macchine di diametro < 2cm.
- **Superficie inquinata** definita da 8 punti ravvicinati in una superficie di **4cm** quadrati
- **Ammaccature** presenti di dimensione > di 1cm sulla superficie dell'oggetto

Chiedere dei criteri **geometrici** al cliente è spesso una sfida complicata da superare poiché è difficile per l'umano trasmettere sottoforma di **parametri matematici** il proprio processo di pensiero e quella che alla fine è definibile come la propria **esperienza**. Per questo motivo si divide il processo di tagging in due parti:

- Il tecnico esperto effettua il **tagging** consapevole di tutte le difettosità con lo strumento di selezione rettangolare.
-Se il modello di **machine learning** è particolarmente bravo a riconoscere i difetti è possibile ricavare una serie di **proprietà <u>geometriche</u>** proprio dai difetti ed effettuare delle elaborazioni su questi difetti al fine di prendere la decisione fondamentale: lo scarto oppure no?

E. Costruire il modello: Si tenta di evitare l'overfitting creando gli insiemi di **train** e **test** dove ci si deve assicurare che il campione del training sia rappresentativo della popolazione di situazioni in cui mi troverò nella pratica. Se ci sono situazioni NON presenti il modello non potrà ricavarle. Lo spazio di conoscenza del training set deve essere denso nello spazio complessivo delle situazioni di difettosità tenendo però conto delle **proporzioni** tra pezzi difettosi oppure no.

Se ad esempio un'azienda produce 1000 tettucci al **mese** di cui 10 sono difettosi allora il rapporto tra oggetti difettosi e corretti del train set dovrà essere il medesimo! Analogamente anche l'insieme di **test** dovrà avere le stesse proprietà del training e quindi mantenere il rapporto e le **proporzioni** tra pezzi difettosi e corretti in egual modo.

F. Valutazione: Il modello, dopo aver effettuato l'addestramento, possiederà delle **confidenze** e dei valori di precisione. Se però il cliente desidera un SI/NO dobbiamo trasformare questo valore statistico in funzione della **metrica che si vuole massimizzare.** Il valore di cut-off si definisce facendo **esperimenti** e cercando di massimizzare gli obiettivi di **business.** Il valore di cut-off per determinare la presenza di un difetto oppure no avviene in fase di planning e business understanding. Per valutare i risultati del modello si adottano strategie di **intersezione tra le aree** delineate nella fase di tagging e generate dal modello addestrato. Da tali intersezioni è possibile definire i valori di **precisione** del modello. Il valore di **cutoff** del sistema dipende da **ogni singolo difetto** presente sull'immagine di input.

Esistendo a priori davvero tantissimi algoritmi e **reti convolutive** risulta complicato capire a **priori** qual è la rete migliore da adottare per il mio problema. Inizialmente si provano **tante reti e diversi algoritmi** per capire quali sono le risposte del sistema all'input. Alcuni algoritmi possono agire con un **grid search** con l'obiettivo di testare il modello in tutte le famiglie di **configurazioni** diverse possibili per ottenere la risposta migliore del sistema. Una volta scelto l'algoritmo si lavora sulla **parametrizzazione** dello stesso agendo sui parmetri di configurazione e gli **iperparametri** specifici della rete scelta.
Per il caso di studio reale che verrà analizzato in questo capitolo utilizzerò un software commerciale prodotto da IBM chiamato **MAXIMO VISUAL INSPECTION**.

4.3 IBM MAXIMO VISUAL INSPECTION

Maximo Visual Inspection è un software concepito appositamente per operare in ambiente **industriale** per **automatizzare** il processo di ispezione visiva digitalizzandolo. Il programma offre una piattaforma di analisi **video** e di **immagini** in grado di sfruttare **modelli pre-addestrati** ottimizzati per task di **classificazione** e **object detection**. Rispetto ai sistemi **general purpose** di **visual recognition** affrontati durante questa tesi, questo prodotto è stato pensato per lavorare in ambienti challenging anche senza la necessità di mettere mano a dettagli tecnici come le **reti neurali** o le configurazioni degli **iperparametri**.
IBM MAXIMO è in grado di apprendere dal corpus del campione e di ottimizzare automaticamente la rete per ottenere il miglior risultato possibile. Per quanto riguarda il tagging, IBM MAXIMO offre modelli di deep learning addestrati in modo iterativo per **etichettare automaticamente** i set di

dati. Tale miglioramento riduce drasticamente i tempi di preparazione del dataset accelerando al contempo i tempi per le aziende.

Figura 176 Flow operazioni IMB Maximo VI

4.3.1 Classificazione di immagini

Nella classificazione di immagini, il sistema MAXIMO VISUAL INSPECTION non andrà a circoscrivere nulla ma si limiterà a definire un oggetto come **differente** rispetto ad un altro tramite una **heatmap** visualizzabile in output dall'utente. Le zone più rosse rappresentano dove il software si va a **concentrare** per stabilire l'appartenenza ad una determinata classe di una fotografia o un video.

Figura 177 Heatmap output

4.3.2 Come costruire un buon modello di **classificazione** di immagini

Il software si focalizza sulle **entità discriminanti** presenti nell'immagine per definire una certa categoria di appartenenza. Il modello è solitamente sempre portato a riconoscere l'oggetto più grande, è quindi di fondamentale importanza che, nel caso di **classificazione**, l'oggetto preso in esame rivesta **almeno il 50%** dell'immagine. Nei casi in cui questo non sia possibile allora si rientra nel campo della **object detection** che è in grado di circoscrivere elementi più piccoli all'interno delle immagini.

Occorre selezionare un **minimo di 5 immagini** per ogni oggetto da classificare, un numero ottimale di fotografie è dell'ordine delle 25-30 immagini per oggetto.

Il dataset deve essere **bilanciato** in quantità tra le varie categorie (se devo classificare arance e angurie, è meglio avere 25 immagini di arance e 25 immagini di angurie).

Se il modello viene abituato con **sfondi diversi** la sua capacità di riconoscere maggiormente l'oggetto di interesse aumenta! Ovviamente è possibile estrarre l'oggetto dallo sfondo in fase di pre-processing per ovviare direttamente al problema.

Le **viste parziali** dell'ogetto vanno ad incidere sul livello di **confidenza** finale, quindi degradano l'accuratezza del modello finale.

NB: Qualsiasi fotografia viene scalata a 224x224 pixel dal sistema!

4.3.3 Un modello di classificazione che riconosce pneumatici

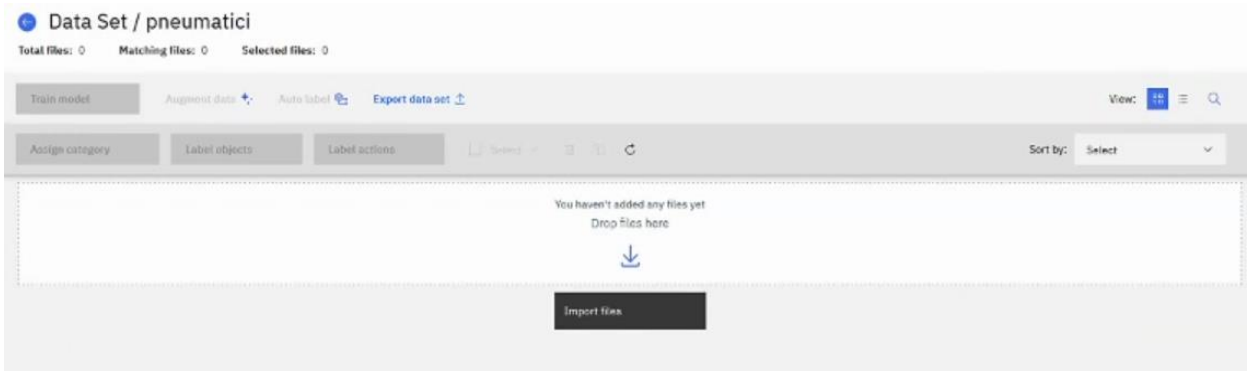


Figura 178 interfaccia di caricamento IMB Maximo

L'interfaccia risulta molto simile a quella già vista per altri software come quello di Microsoft o di Google. È possibile caricare le immagini semplicemente trascinandole sullo schermo, i tempi di upload sono veramente **rapidi:** il sistema elabora 20 immagini in 5 secondi rendendole disponibili per l'utente sull'interfaccia grafica.

Per questa demo sono state utilizzate **20 immagini** di differenti **pneumatici** provenienti da marche distinte:

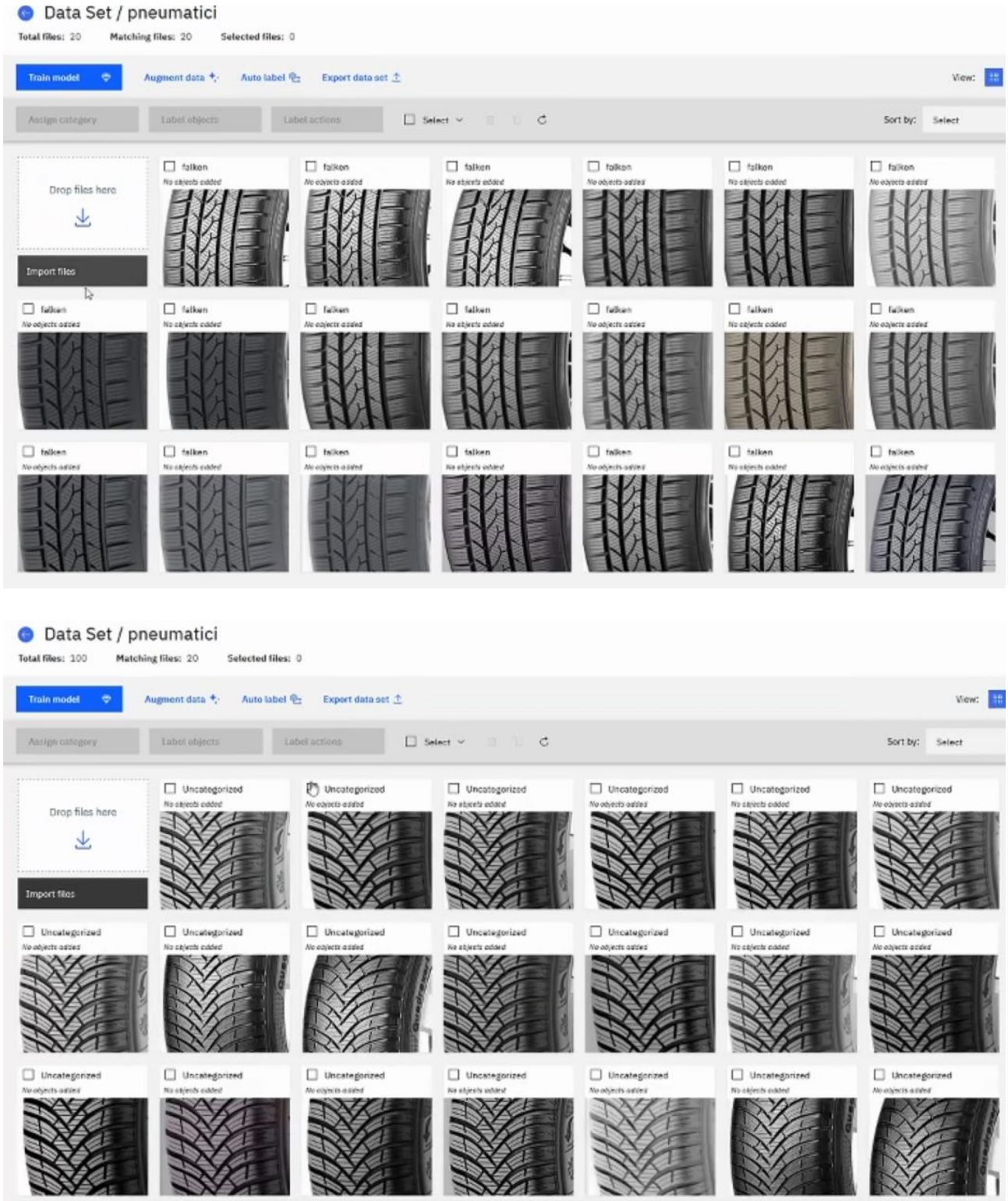


Figura 179 Primo dataset di pneumatici

Una volta caricate le immagini è possibile effettuare una **selezione multipla** e assegnare l'etichetta corrispondente al contenuto delle immagini. È consigliabille caricare prima tutte le fotografie e poi selezionarle tutte con l'apposito pulsante per assegnare la corretta categoria.
Una volta terminata la fase di **tagging** dell'intera immagine è possibile procedere con il **training** che, nel caso della **classificazione**, offre interessanti modelli **pre-addestrati** con dataset già caricati nel sistema.
Nei **settaggi avanzati** del modello è comunque possibile mettere mano agli **iper parametri** del modello preso in esame:

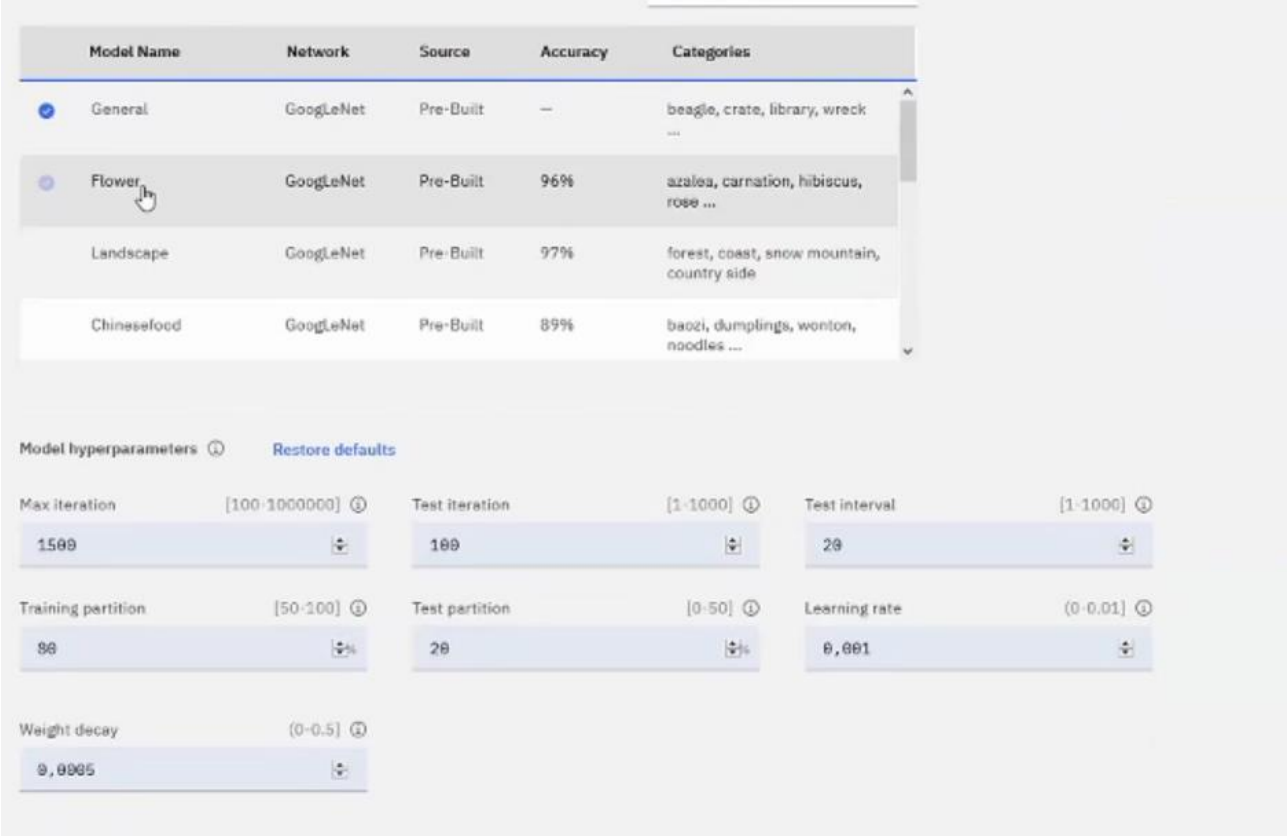


Figura 180 Settaggi avanzati e iper parametri

In questo esempio è stato utilizzato un rapporto tra **train/test** di 80/20 definiti in **training partition** e **test partition**.
La **topologia** della **rete di convoluzione** del software IBM MAXIMO è stata già scelta in fase di implementazione: gli algoritmi che stanno alla base della object detection sono infatti noti e vengono resi disponibili spesso anche con soluzioni open source.
Nel tool si trova la re-implementazione dell'algoritmo con la possibilità di agire **manualmente** sugli iper parametri (senza usare tecniche di **auto-ai** che lanciano configuratori di parametri **ottimali**

per il mio problema). Se si conoscesse esattamente il modo con la quale la rete di **convoluzione** è stata implementata (numero di filtri, layers etc) si potrebbe ottenere lo stesso risultato anche con un open source!

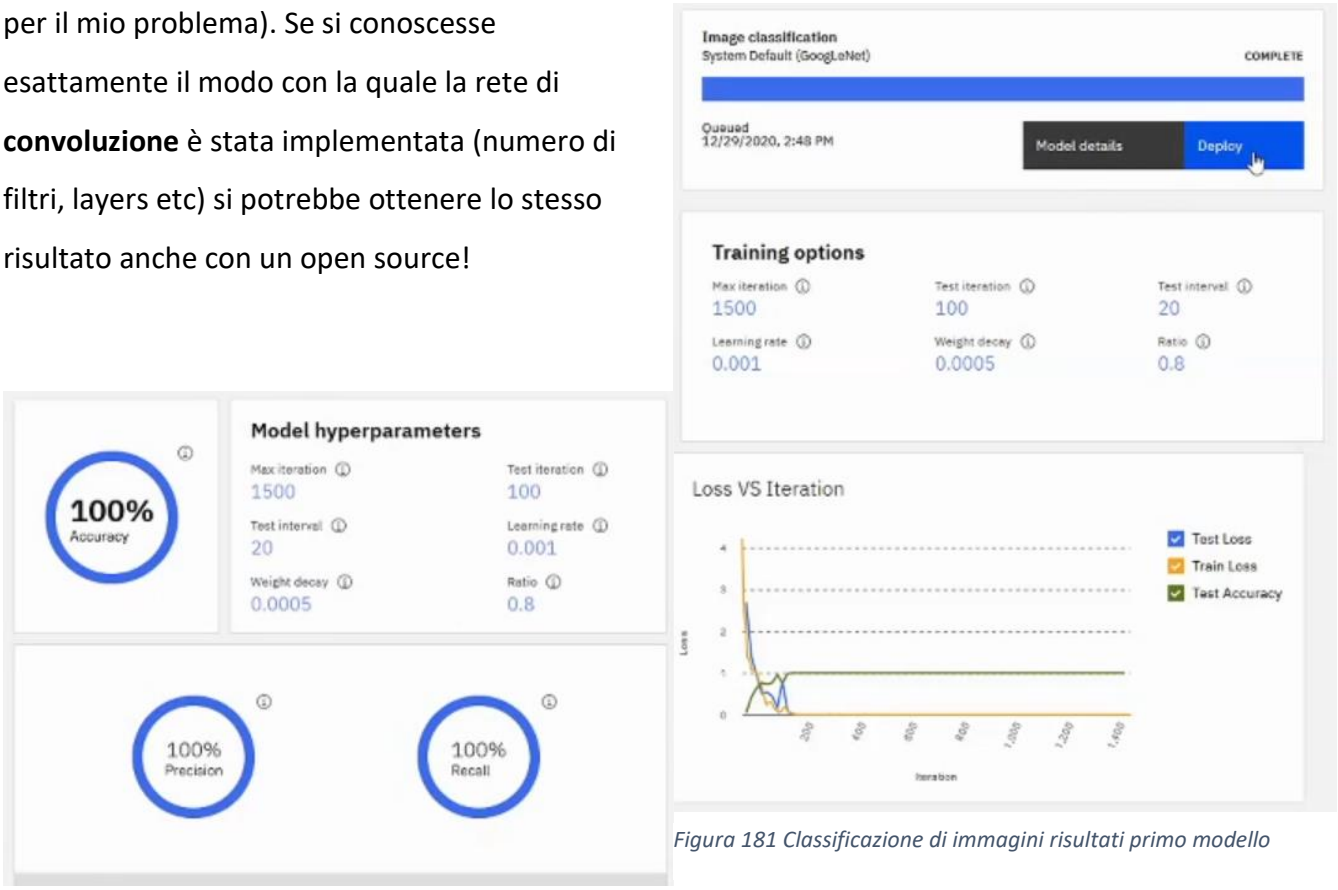


Figura 181 Classificazione di immagini risultati primo modello

Nelle metriche avanzate è possibile visualizzare anche la **matrice di confusione**:

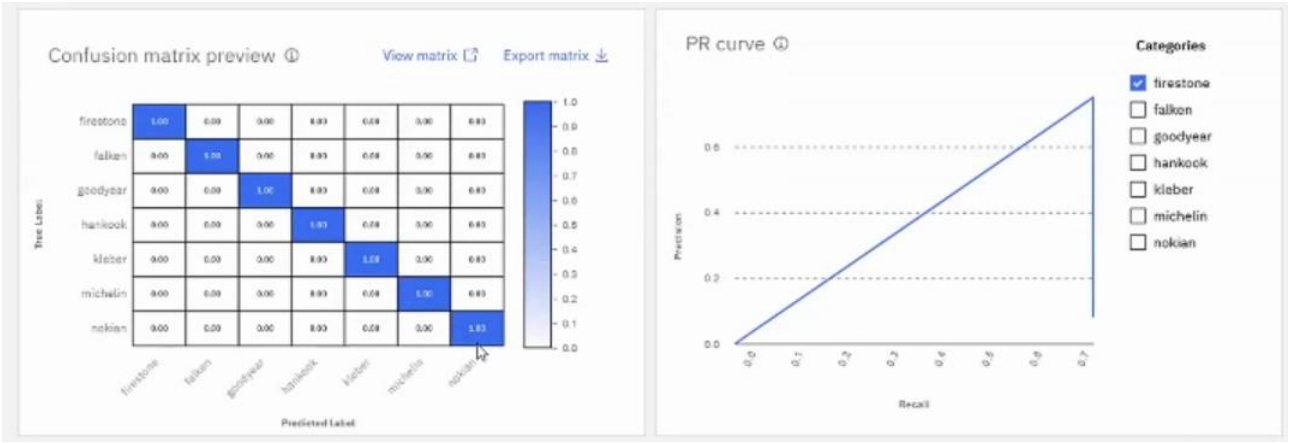


Figura 182 Matrice di confusione primo modello

Una volta terminato il **training** è possibile effettuare il **deploy** del modello che consente anche al cliente di **scaricare il modello** sottoforma di file .zip

Analogamente è possibile esportare anche il **dataset di immagini**

Alcuni **risultati** di **classificazione:**

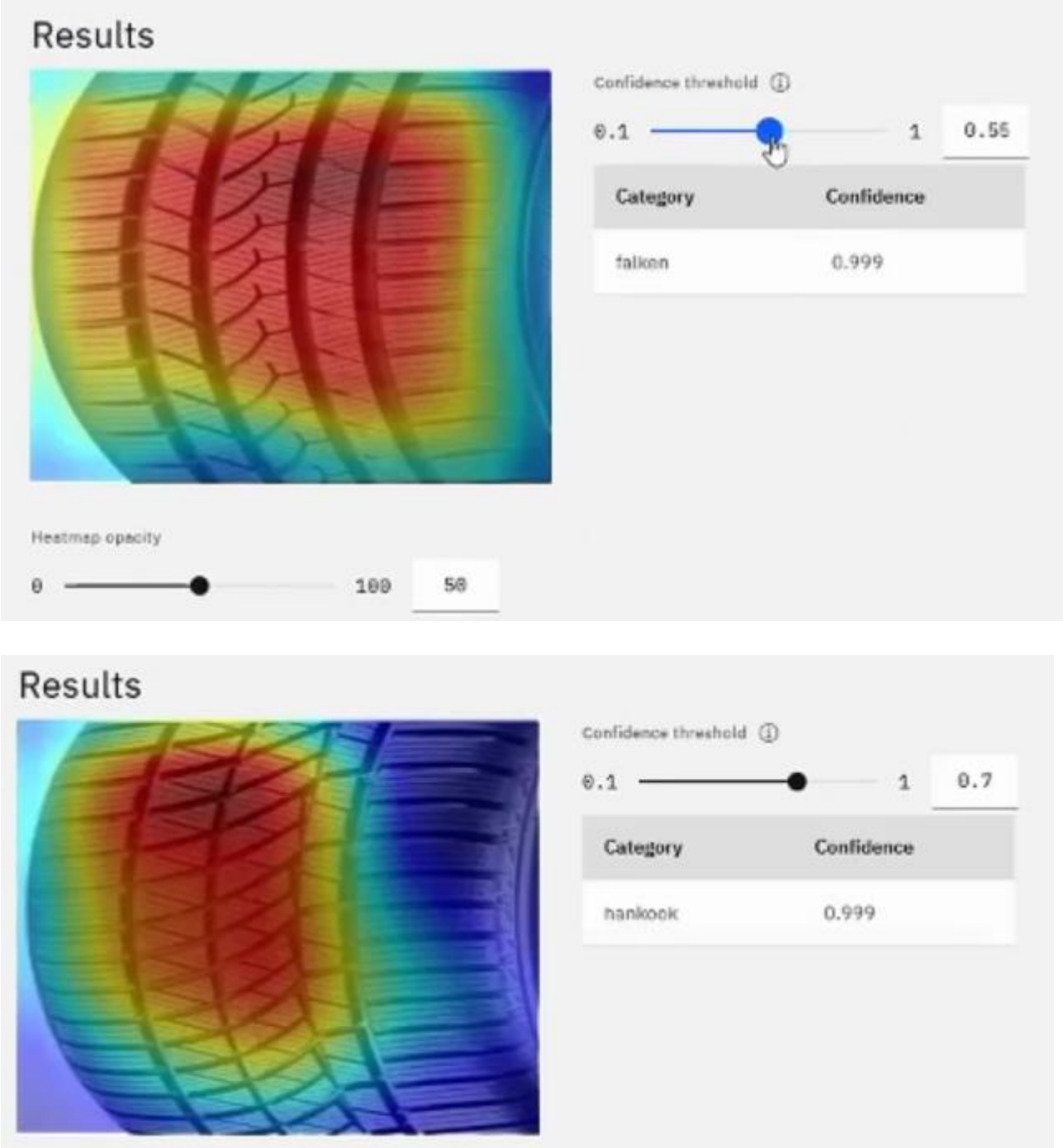


Figura 183 Risultati riconoscimento pneumatico

4.3.4 Come costruire un buon modello di **object detection**

Nel caso della **object detection** siamo interessati a riconoscere uno o più oggetti (e quindi anche difettosità) presenti all'interno di una fotografia o un video. I suggerimenti da applicare per costruire un buon dataset sono fondamentalmente i seguenti:
Cercare di **taggare** l'oggetto con una buona precisione **evitando finestre sovradimensionate** oppure si potrebbe inserire rumore causato dalla presenza dello sfondo. È una buona pratica nominare le label con **"pass"** oppure **"fail"** davanti alle tag, questo faciliterà il lavoro di chi **programmerà** eventuali software a **valle** dell'operazione di detection. Un buon oggetto dovrà essere indicato ad esempio come "good_mela".
Occorre eseguire il **labeling** assolutamente **prima** dell'operazione di **data augmentation** in quanto IBM MAXIMO consente di mantenere **automaticamente le labels** anche sulle nuove immagini generate facendo risparmiare moltissimo tempo. Durante il **deploy** è bene evitare un mix tra tagging "**box**" e "**segmentation**" in quanto il software è specializzato nel trattare immagini con un particolare labelling (DETECTRON per segmentation, R-CNN per box labelling); i modelli R-CNN si confondono facilmente con il **segmentation,** è quindi buona norma cercare di adottare dei box come tecnica principale.
Per evitare l'effetto della **doppia labeling** che può avvenire in fase di riconoscimento è necessario restringere il campo delle variabili in esame come la luce, l'angolazione o le dimensioni dell'oggetto. Non taggare oggetti parzialmente nascosti o che sono esposti meno del 50% della loro figura.

4.3.5 Un modello di **object detection** che riconosce **parti di motore**

Si vogliono identificare 4 oggetti presenti all'interno di un motore : **filtro, bullone, connettore e tappo** di colore rosso.

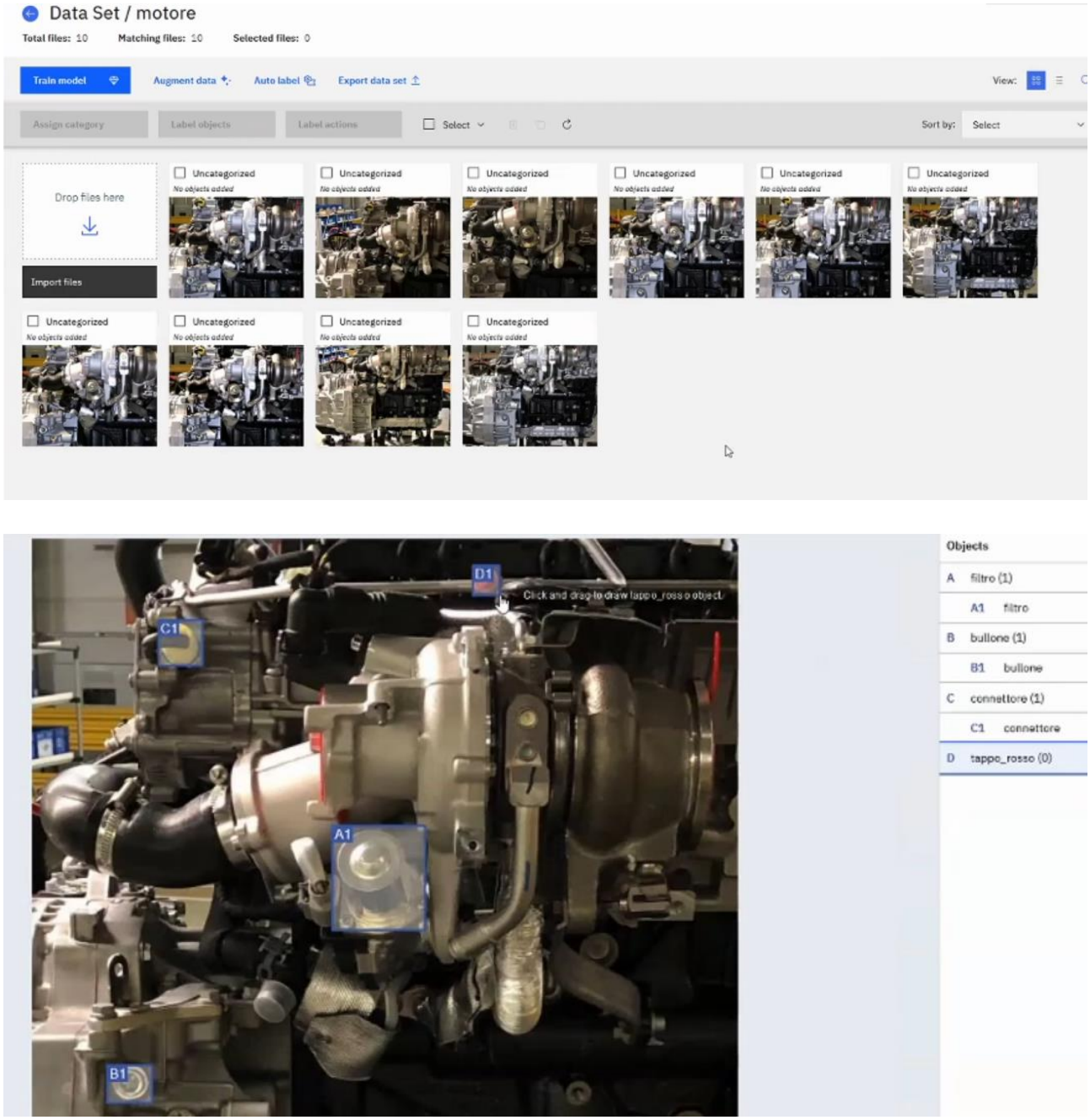


Figura 184 Tagging con bounding box su motore

Per questa demo sono state utilizzate 10 immagini di **motore** generico della quale sono stati individuati i pezzi e taggati utilizzando lo strumento di selezione rettangolare del software. Per garantire un po' di variabilità dell'input, il dataset di partenza è stato alterato con alcuni colori con Photoshop.

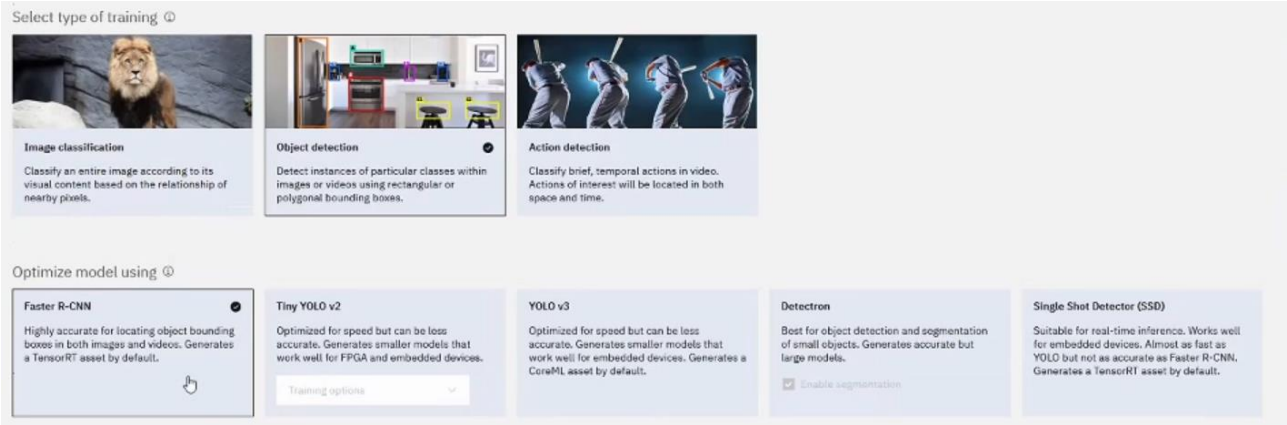


Figura 185 Tipologia di addestramento e di modello

I modelli selezionabili per l'addestramento sono R-CNN, YOLO v2, YOLOv3, Detectron e Single Shot detection. Per questa demo è stata utilizzata una rete R-CNN con **1000** iterazioni, momentum 0.9 e learning rate 0.001. Il rapporto tra train e test è sempre 80/20.

Il modello è stato allenato in **7 minuti** complessivi e mostra il grafico **loss-iteration** in tempo reale durante l'addestramento. È buona norma che le due curve si avvicinino nel tempo e non subito per evitare **overfitting** come già visto con il YOLO **open source**.

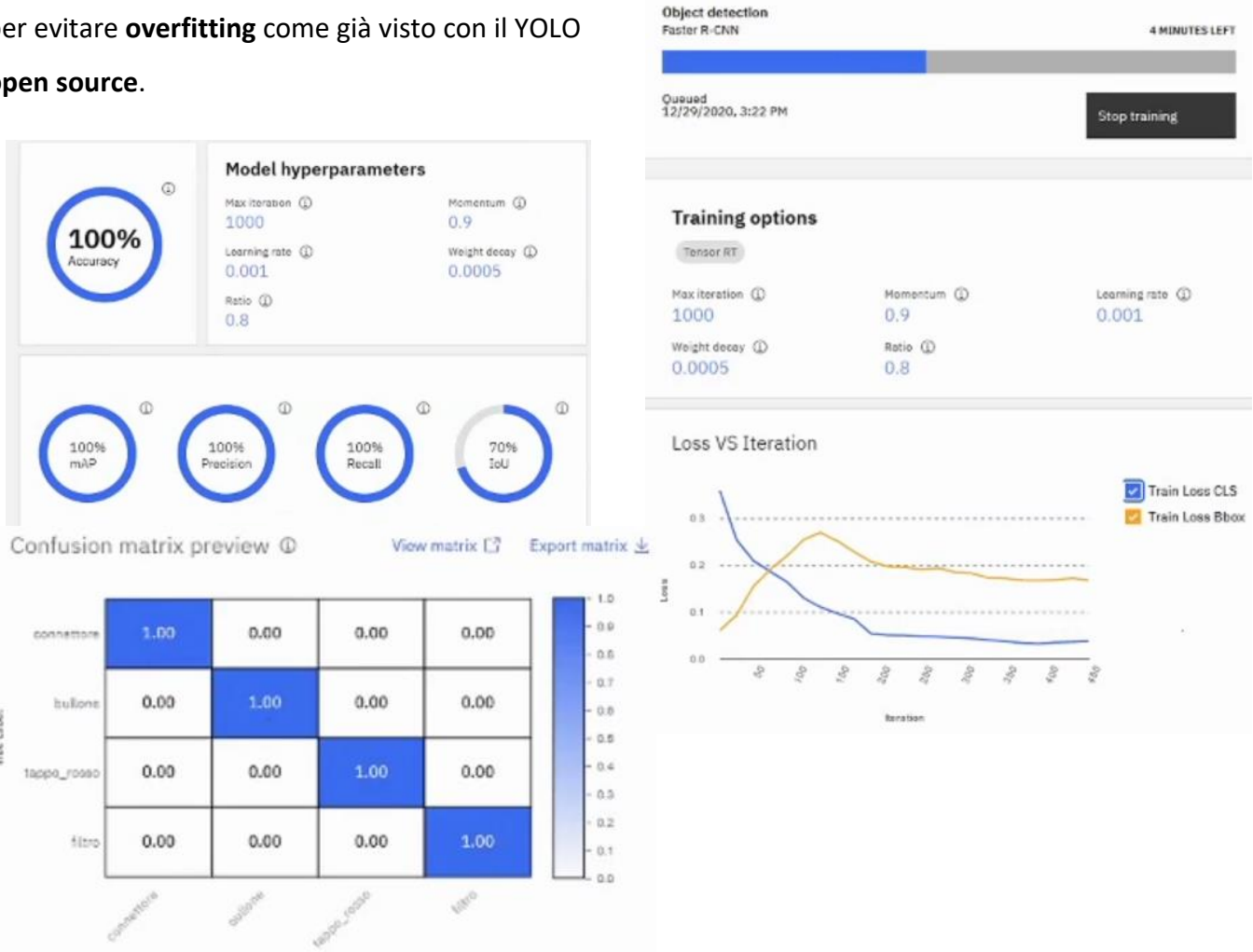


Figura 186 Confusion matrix primo modello

Il valore **IOU** mostra le **precisione delle finestre** dei box rettangolari disegnati dal modello rispetto a quelle indicate dall'utente. Gli altri parametri **mAP, precision e recall** sono stati già trattati nella tesi nei precedenti capitoli.

Ecco alcuni risultati di questo modello addestrato con sole 10 immagini:

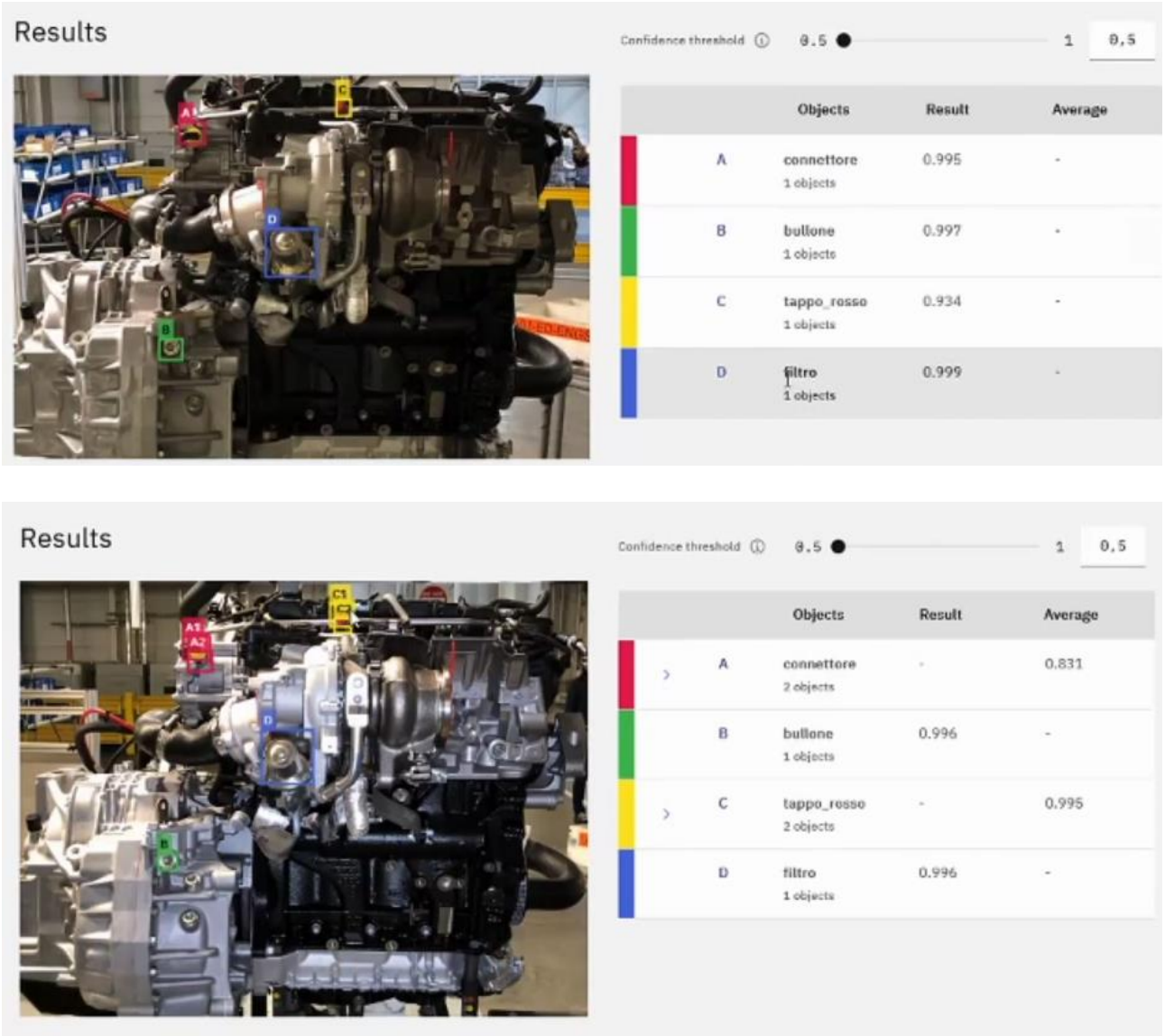


Figura 187 Riconoscimento con 10 immagini

Qui si nota il fenomeno della **doppia labeling**, ciò significa che 10 immagini **non sono sufficienti** e il modello deve essere **raffinato**. Se è stato **già generato un modello** è possibile aggiungere ulteriori immagini che vengono **automaticamente taggate** dal sistema di **auto-labelling**. Selezionando l'ultimo modello eseguito, dopo aver caricato 13 ulteriori fotografie, il sistema riconosce automaticamente i potenziali oggetti presenti nell'immagine.

Figura 188 Tool di auto-labelling segmenta le immagini

Tutte le immagini vanno comunque **ricontrollate** a mano per verificare che il tool abbia fatto un buon lavoro.

Per migliorare ulteriormente il modello è possibile effettuare **data augmentation** che genera un **nuovo dataset** a partire dai dati di partenza.

Per generare un buon dataset con l'augmentation occorre analizzare bene il problema in esame: se per esempio l'oggetto è orientato **sempre allo stesso modo** non ha senso utilizzare **rotation**; se la luce dell'oggetto è controllata analogamente non ha senso usare **color**; se l'oggetto è vicino al **bordo** dell'immagine non ha senso usare il **crop**. Dopo il data augmentation occorre **rivedere le immagini** ed eliminare tutte quelle che non rientrano in un ambito realistico. Nel caso in cui si vogliano identificare **graffi** su una superficie, sono indicate come buone le opzioni di **rotation e flip** che consentono di moltiplicare le **casistiche di anomalie** con il semplice utilizzo di poche immagini.

Gli effetti di **blur** generano immagini **mosse**.

Gli effetti di **sharpen** generano immagini più **nitide.**

Gli effetti di **crop** generano immagini **ritagliate** in più parti.

Gli effetti di **color** modificano **l'intensità** di colori.

Gli effetti di **noise** generano immagini poco **definite**.

Gli effetti **rotate** generano immagini con **rotazioni** nel range scelto.

Gli effetti di **flip** capovolgono l'immagine.

È buona norma chiamare i dataset con i **nomi** degli effetti applicati alle immagini come "motore_color_blur". Per questa demo verranno utilizzati sia l'effetto **blur** che di **color**.
A partire da sole 3 immagini selezionate prima di cliccare su agument data, il sistema è in grado di generare **45 nuove fotografie** che, se sommate alle 23 di partenza andranno a costituire un dataset complessivo di **68 immagini**.

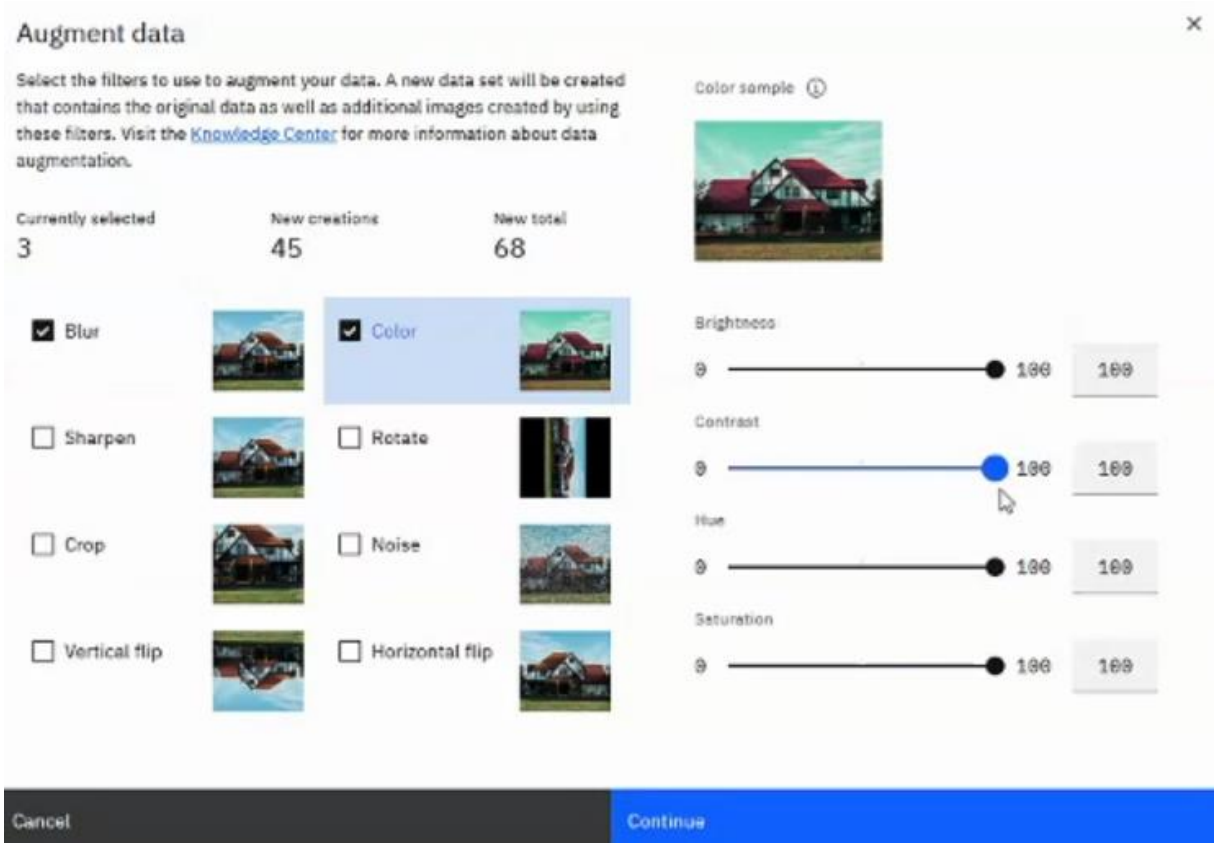


Figura 189 Data agumenting

La comodità risulta nel fatto che il sistema è in grado di **mantenere le label** anche sulle immagini su cui effettua **augmentation**, non richiedendo ulteriori tagging da parte dell'utente.

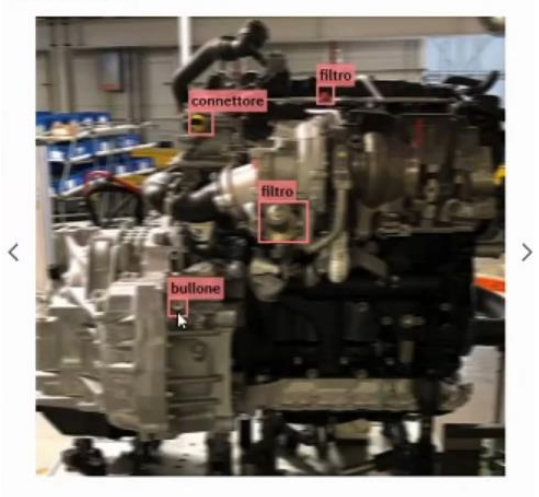


Figura 190 Immagine ottenuta nel data agumetning già taggata dal sistema

Ecco i risultati dell'addestramento con queste 68 immagini:

Figura 191 Risultati addestramento secondo modello

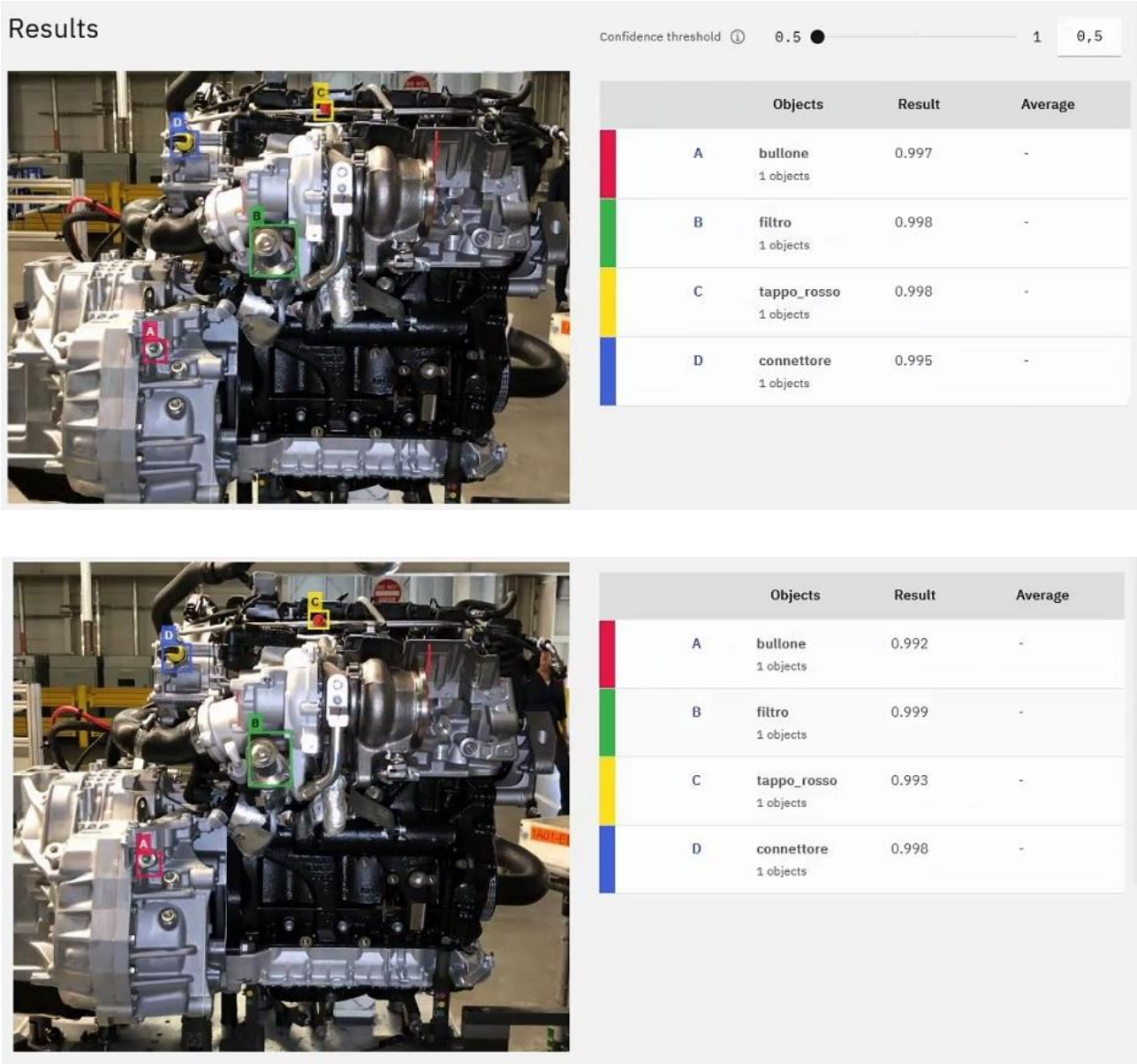


Figura 192 Risultati: doppie label sparite

Come si nota le **doppie label** sono sparite e il modello riconosce correttamente gli oggetti presenti nell'immagine!

4.3.6 Un modello di **object detection** che **graffi** su una ruota

In questa demo verrà presentata una casistica di riconoscimento di **graffi** presenti su pneumatici, un task molto legato al mondo della **visual inspection**.

Si parte da un dataset di partenza di 33 immagini che sono state segmentate con il tool **segmentation**.

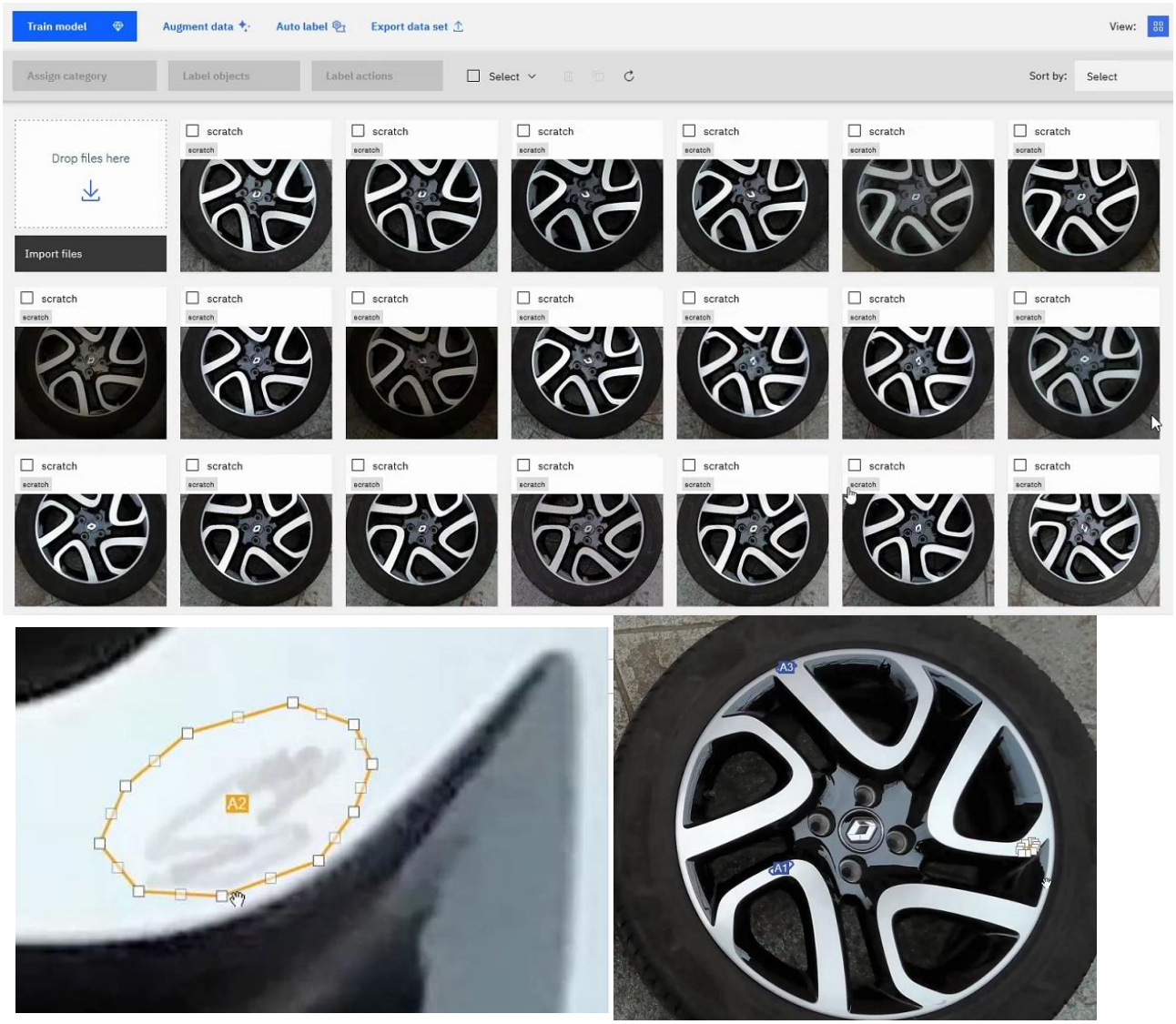


Figura 193 Segmentazione dei difetti

Col segmentation è possibile definire un **poligono** che racchiude il nostro elemento di difettosità. (Ovviamente è anche possibile modificare la label nelle sue proprietà come le **coordinate** o il **nome**).

Ecco i risultati dopo l'addestramento effettuato con 4000 iterazioni:

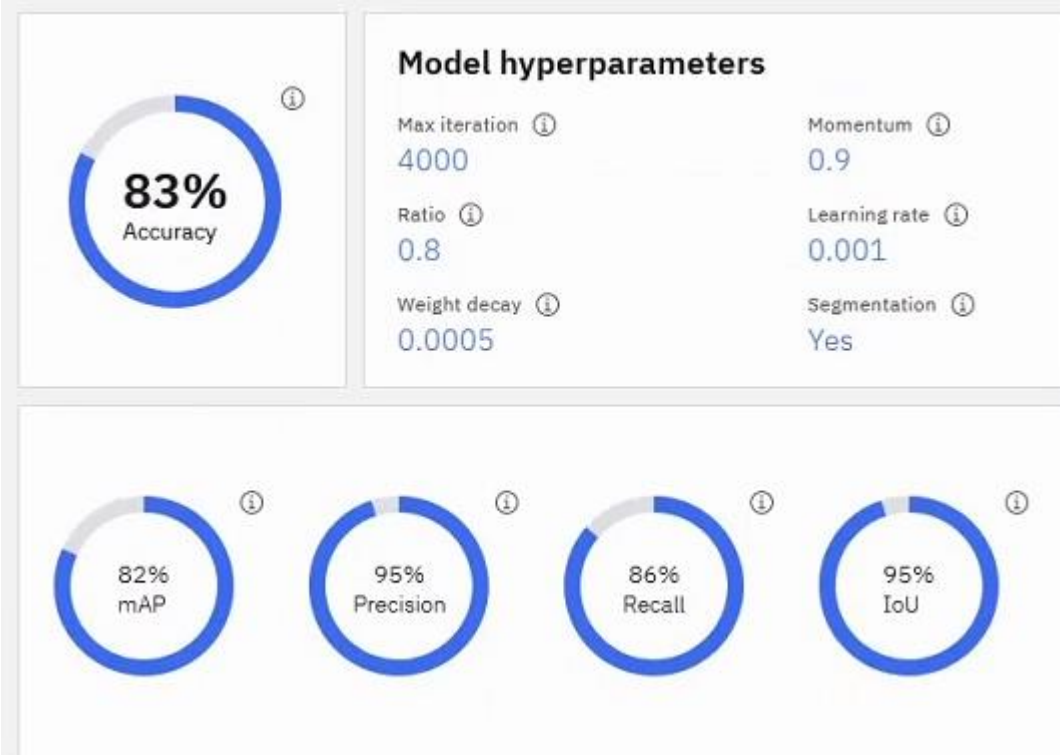


Figura 194 Risultati primo modello addestrato

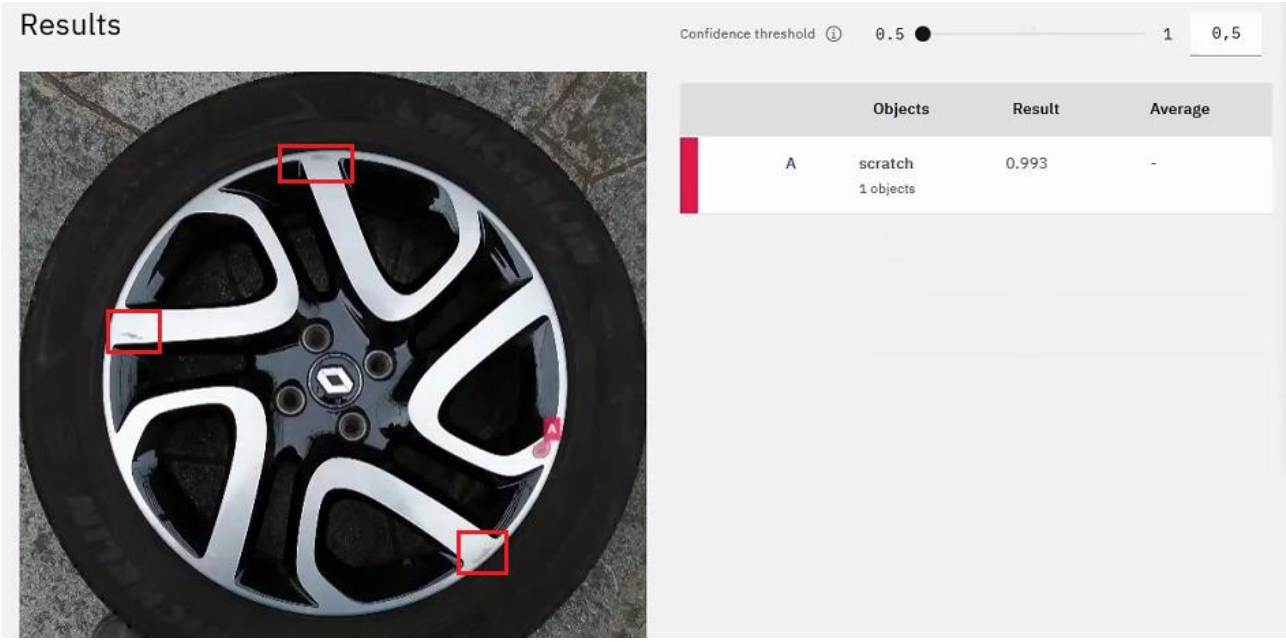


Figura 195 Primi risultati del modello

Come si può notare da questa immagine di prova è stato individuato **solamente un graffio**. Nella fotografia sono state segnate con dei rettangoli le zone della ruota che **presentavano graffi** non individuati dal sistema.

Visti questi risultati il modello è stato raffinato e il dataset è stato migliorato.

Per il secondo dataset sono state utilizzate 81 immagini **ottenute con augmentation** di **rotazione e di colore** con un numero di iterazioni pari a 7000. Il tempo di addestramento è stato di **68 minuti.**

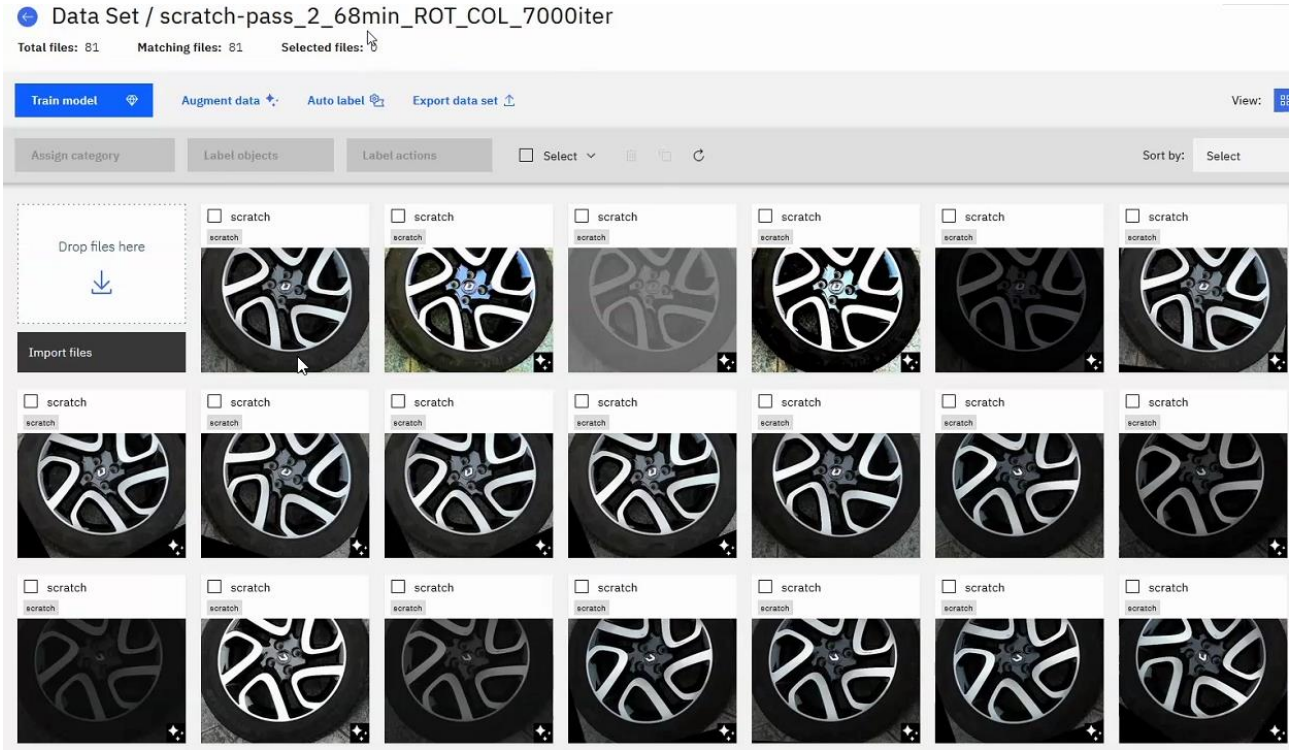


Figura 196 Secondo dataset ruote con graffi e data augmentation

Ecco i risultati ottenuti dal secondo modello:

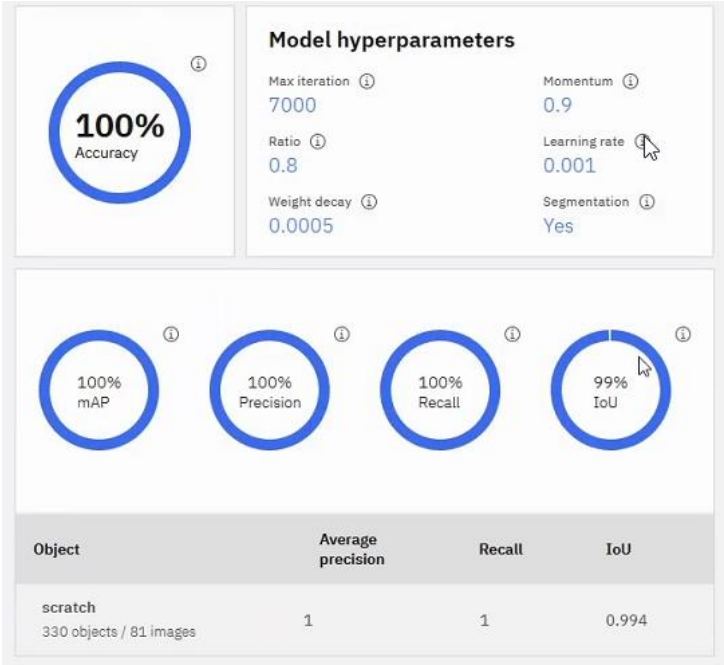


Figura 197 Risultati secondo modello addestrato

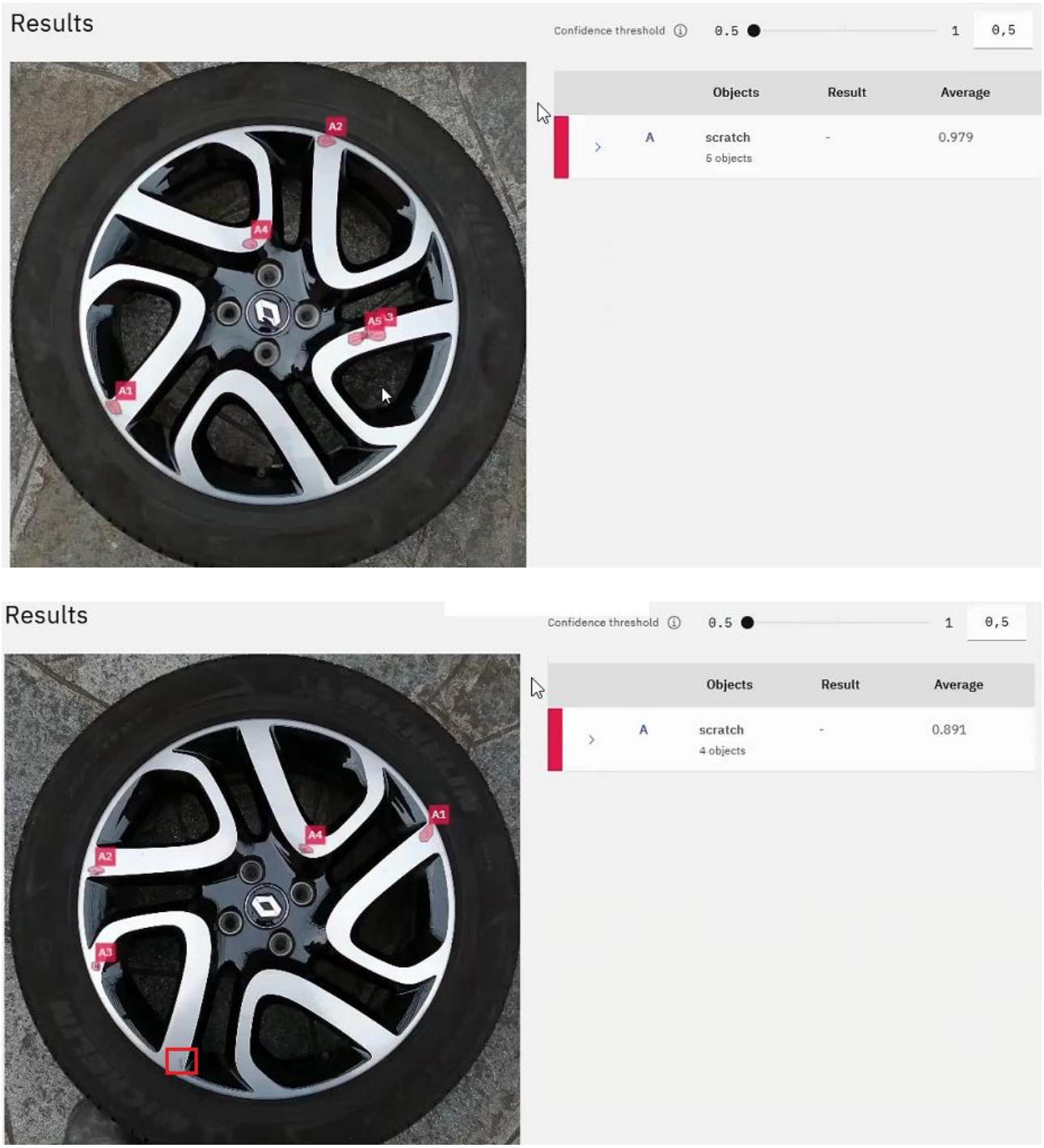


Figura 198 La precisione nel riconoscere i graffi è migliorata nel modello

Sebbene la precisione del sistema sia **molto migliorata** ci sono ancora alcuni graffi che non vengono rilevati dal sistema. Il modello richiede di essere ulteriormente raffinato!

Per il terzo ed ultimo dataset sono state **aggiunte ulteriori 10 immagini** e dopo averle taggate correttamente con lo strumento di segmentation è tato effettuato un data augmentation di **blur e rotazione** che ha portato il dataset ad un totale di **166 immagini** allenate per 7000 iterazioni (**68 minuti**). Da notare che l'aggiunta di immagini **non incide** sui tempi di addestramento del modello!

Ecco i risultati ottenuti:

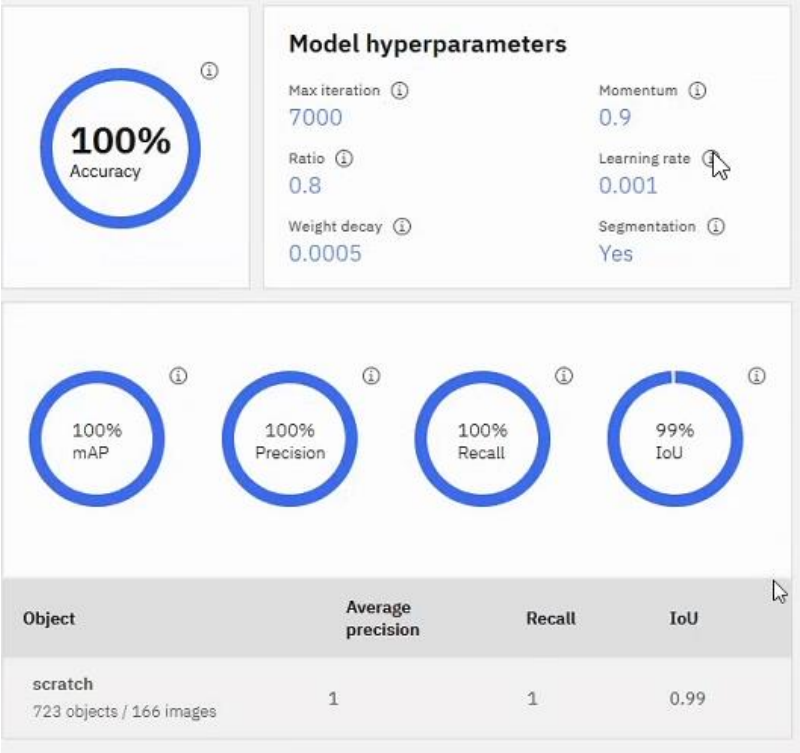
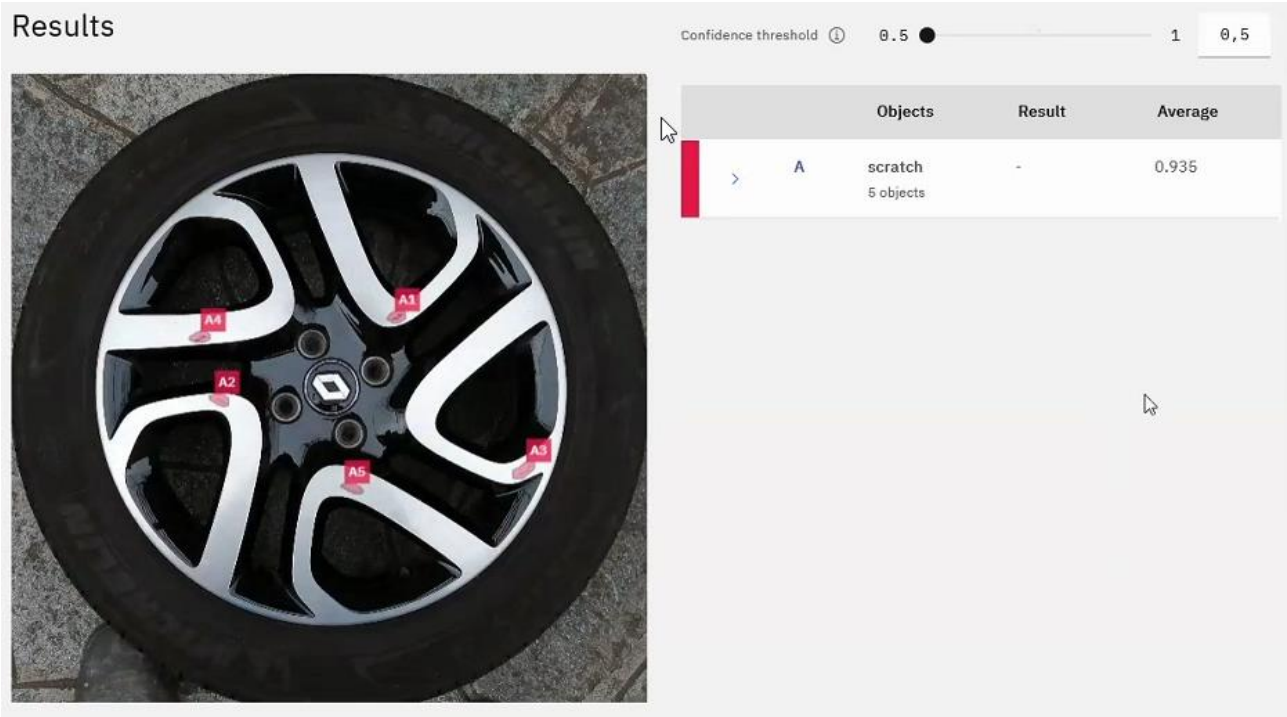


Figura 199 Risultati terzo modello addestrato



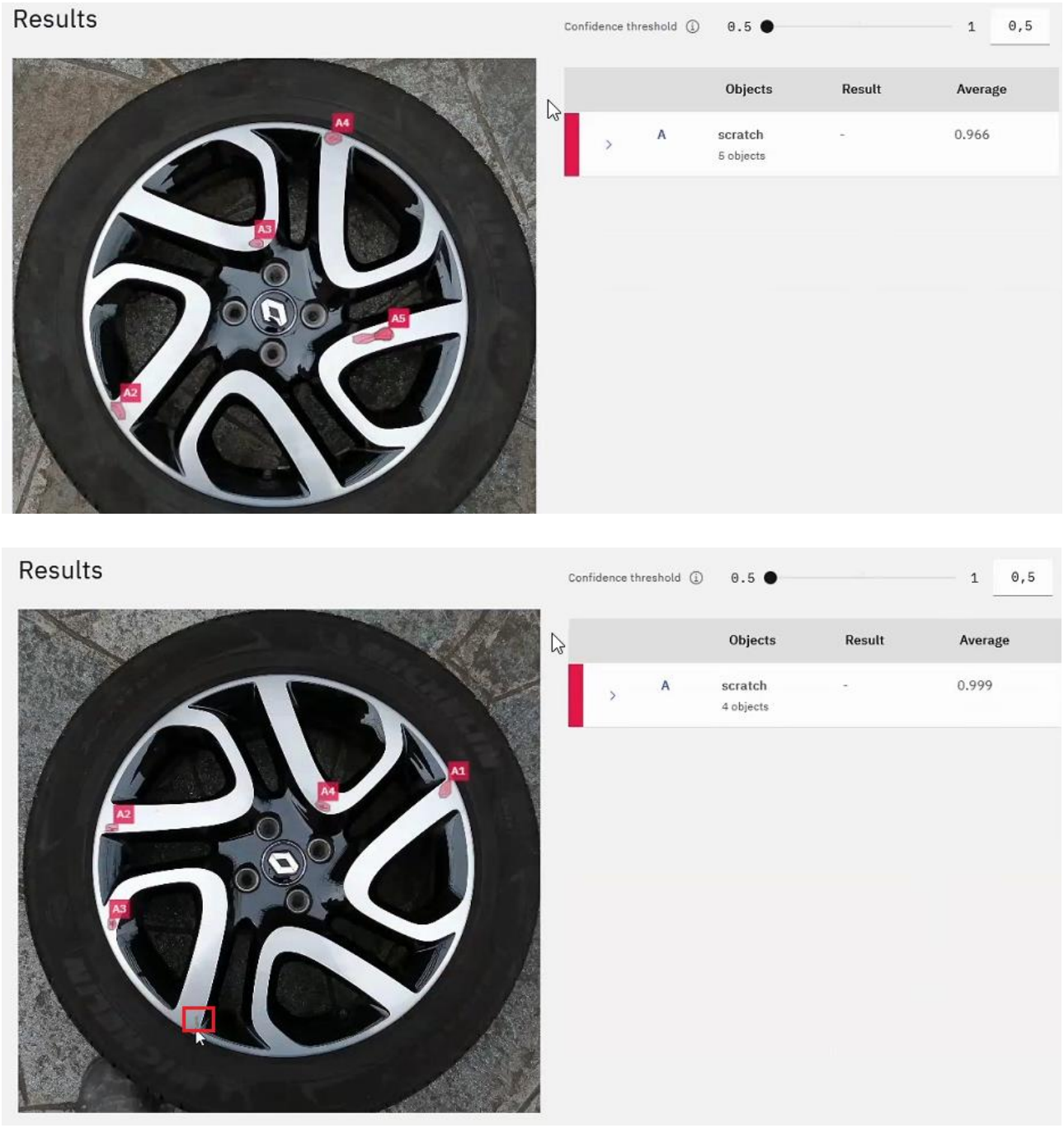


Figura 200 Riconoscimento graffi: Modello e risultati finali

C'è ancora margine di miglioramento ma questo modello sembra già ottimo per riconoscere i difetti sulla ruota in esame. I graffi possono trovarsi **in qualsisi posizione della ruota** e avere un colore/forma/dimensione molto differenti tra loro, questo sottolinea il grado di sfida nell'affrontare questa tipologia di problematiche anche con l'ausilio di software moderni e allenati. Nella capitolo successivo verrà preso in esame un **caso reale** in collaborazione con l'azienda **Lamborghini**.

CAPITOLO 5 – CASO DI STUDIO REALE IN COLLABORAZIONE CON LAMBORGHINI

5.1 Applicazione del modello CRISP-DM ad un caso reale

Durante lo sviluppo di questa tesi è nata una collaborazione con l'azienda **Lamborghini S.P.A** al fine di sperimentare quanto appreso durante la mia esperienza. Con l'obiettivo di ottenere risultati **industrialmente apprezzabili**, le soluzioni proposte si sono basate sul modello CRISP-DM preceduto da un'accurata analisi del problema.

5.2 Il problema: Business understanding

All'interno della linea produttiva di Lamborghini vengono scattate delle **fotografie di autovetture** per mostrare ai clienti il risultato preliminare della fase di assemblaggio del veicolo presso lo stabilimento che racchiude la linea produttiva. Queste immagini servono per far capire al cliente lo stato corrente del proprio prodotto (prossimo alla spedizione) e per tale motivazione devono essere ben curate sotto ogni aspetto. Le fotografie in questione possono essere soggette alla presenza di **difetti** propri dell'immagine (e non del veicolo) a causa di uno o più dei seguenti fattori:

- **Fari** spenti del veicolo
- Presenza di **tracce** e **sporco** sul pavimento
- Presenza di **fogli** o **altro materiale** sul cruscotto
- Auto **non allineata** nel parcheggio (piazzole di gomma sporgenti)

Se l'immagine catturata dall'operatore contiene uno o più difetti egli è costretto a ri-posizionare il veicolo nella medesima posizione e scattare nuovamente le foto facendo perdere tempo e denaro all'azienda.
Si vuole sviluppare un modello in grado di riconoscere la presenza di questi difetti e segnalare prontamente le anomalie in modo da evitare ulteriori posizionamenti di veicoli da parte dell'operatore.

5.3 Raccolta dati -IBM MAXIMO vs OPEN SOURCE

Prima di iniziare a lavorare direttamente con le immagini del cliente ho creato un caso d'uso rappresentativo che contenesse gli **elementi essenziali** del problema al fine di mostrare a Lamborghini le potenzialità del modello e i risultati che si sarebbero potuti ottenere una volta terminato il lavoro. Dopo un incontro preliminare è nata la necessità di analizzare immagini solo con angolazione **TOP-LEFT** in quanto le fotografie scattate sulla linea produttiva erano tutte caratterizzate da questo angolo di esposizione.

5.3.1 Un primo prototipo: IBM MAXIMO

Questo primo prototipo è stato sviluppato utilizzando il software **IBM MAXIMO VISUAL INSPECTION.** Come prima cosa sono state scattate 7 fotografie di un veicolo (Renault Capture) con la **stessa angolazione** proposta dalla telecamera di Lamborghini del caso di studio reale.

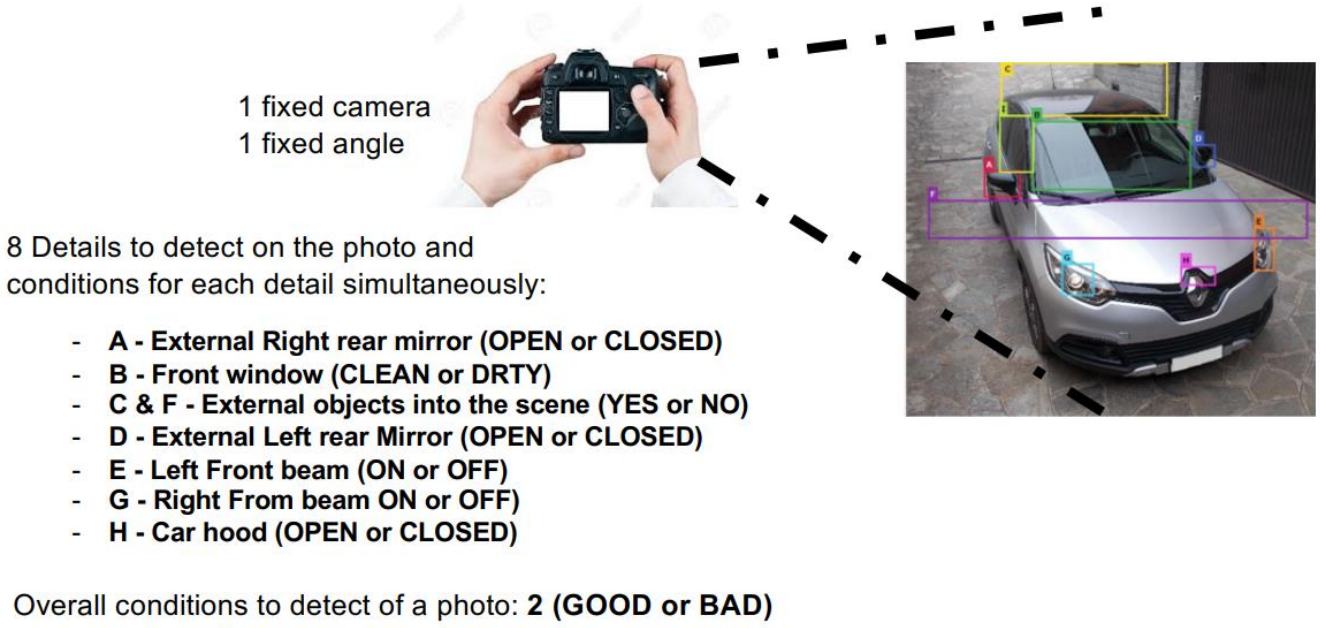


Figura 201 Lo scenario di partenza fittizio

5.3.2 Generazione del dataset e tagging

Sono state raccolte **7 fotografie** del veicolo in **condizioni ottimali** e **10 fotografie** del veicolo in una condizione in grado di **simulare la presenza di difetti** nell'immagine. Il numero di fotografie utilizzate per l'addestramento del sistema è stato volutamente tenuto basso poiché, come già visto durante lo sviluppo di questa tesi, una delle motivazioni che impedisce ad un cliente di adottare un sistema di visual inspection è proprio la richiesta di un **numero elevato** di immagini. Sono state generate sinteticamente **7 ulteriori immagini** per arricchire il training set **GOOD** e altre **12 immagini** per arricchire il set **BAD.**

Per ottenere questi risultati è stata perturbata la **luminosità** e la **saturazione** del colore.

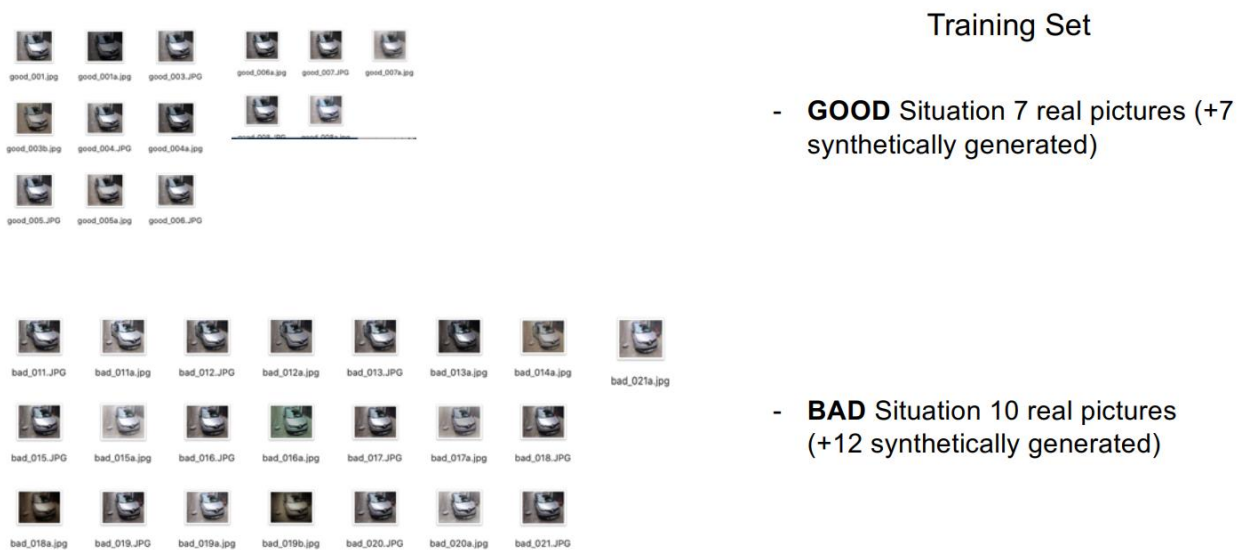


Figura 202 Dataset di partenza con augmentation

Sono state individuate **18 classi** differenti rappresentative delle condizioni **GOOD/BAD** di ogni elemento in esame:

-**Good/Bad** specchietto sinistro e destro (4 classi)

-**Good/bad** cofano aperto/chiuso

-**Good/Bad** fari accesi/spenti

-**Good/Bad** presenza di oggetti sul campo visivo **orizzontale**

-**Good/Bad** presenza di oggetti **dietro** al veicolo

-**Good/Bad** portiera aperta/chiusa

-**Good/Bad** presenza di fogli sul cruscotto

bad_faro_sx

bad_faro_dx

bad_cofano

bad_oggetti

bad_portiera

bad_vetro

bad_specchietto_dx

bad_oggetto_dietro

bad_specchietto_sx

good_specchietto_sx

good_specchietto_dx

good_faro_sx

good_faro_dx

good_cofano

good_oggetti

good_vetro

good_portiera

good_oggetto_dietro

La presenza delle etichette "good_..." oppure "bad_..." facilita il processo di elaborazione dell'immagine nel caso in cui siano necessari successivi passaggi di coding.

Le fotografie sono state tutte taggate utilizzando lo strumento software **IBM VISUAL INSPECTION.**

5.3.3 Generazione del modello (IBM MAXIMO)

Il modello generato è stato addestrato per **2 ore** e ha eseguito **6000 interazioni** ottenendo il 100% di precisione nel riconoscimento di tutte le situazioni.

La rete scelta è stata la R-CNN.

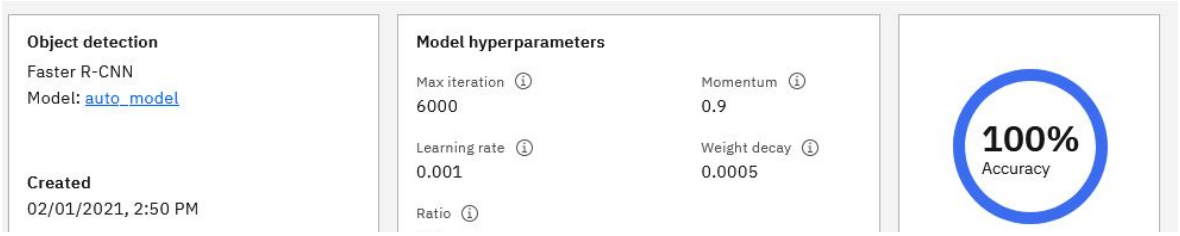


Figura 203 Statistiche e precisione del primo modello

5.3.4 Risultati ottenuti (IBM MAXIMO)

Nonostante la presenza di un limitato numero di immagini, il modello ha ottenuto risultati eccellenti ed è stato in grado di riconoscere la presenza e l'assenza di difetti **con ottima precisione**.

-TEST con immagine **GOOD** in input:

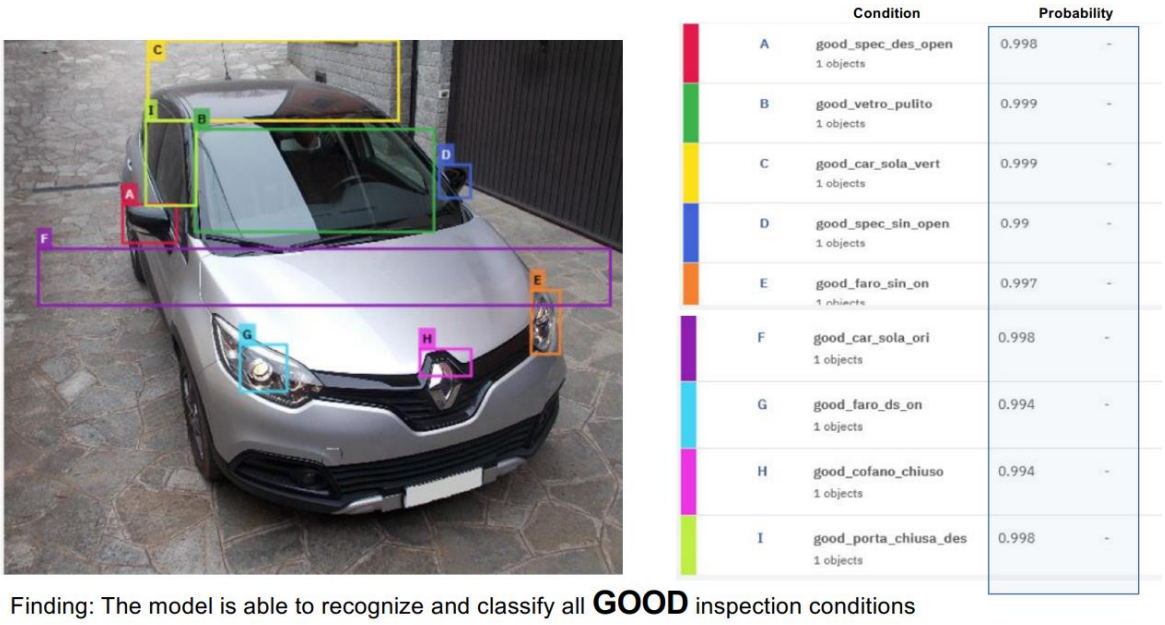


Figura 204 Il sistema riconosce e classifica un'immagine GOOD

-TEST con immagine **BAD** in input:

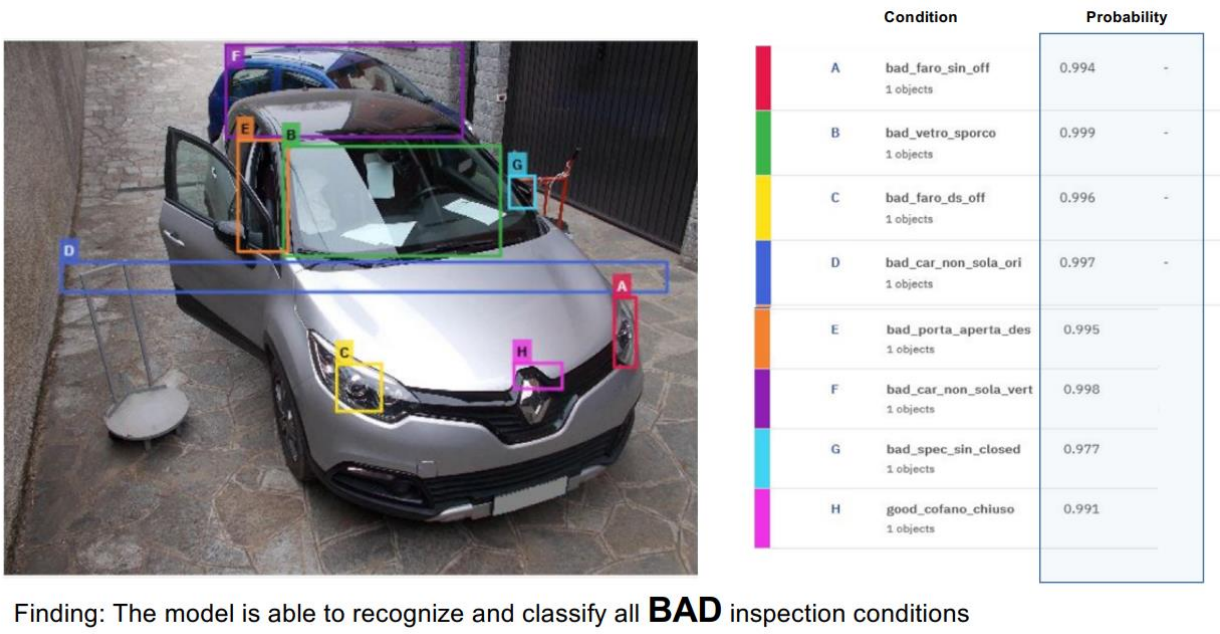


Figura 205 Il sistema riconosce e classifica un'immagine BAD

5.3.5 Un secondo prototipo: YOLO – DARKNET (OPEN SOURCE)

Al fine di valutare le performance del modello proposto dalla soluzione IBM ho eseguito l'addestramento di una rete **Darknet – YOLO** mantenendo **il medesimo dataset** utilizzato nell'esperimento precedente.

Il tagging delle immagini è stato effettuato con **labelImg**

Figura 206 Taggare l'auto con l'open source

Il dataset generato è il medesimo ma è abbinato alla rappresentazione (con coordinate) dei difetti presenti nelle 18 differenti classi del problema.

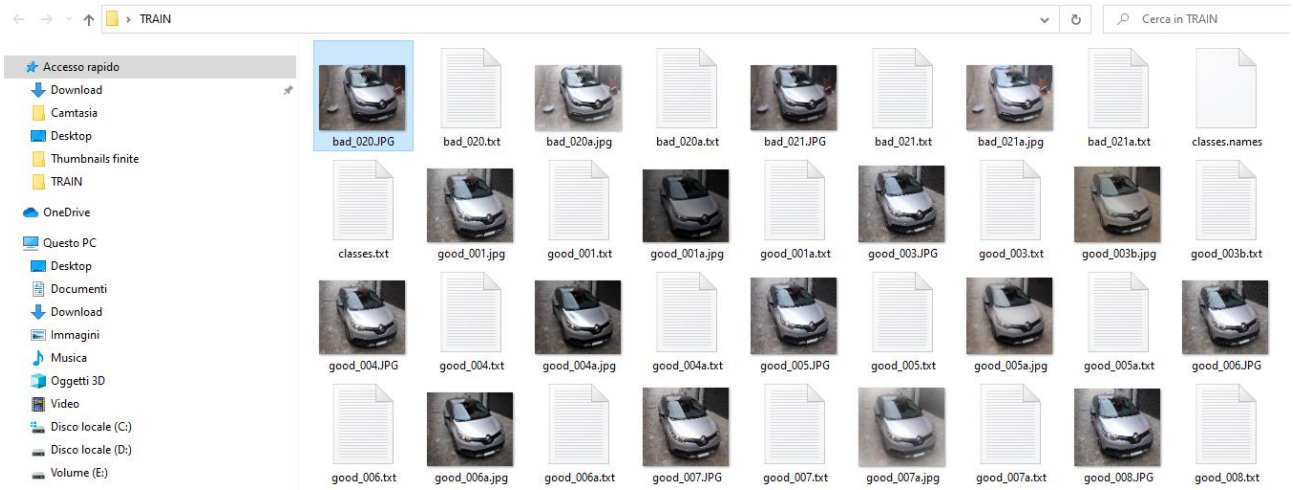


Figura 207 Dataset di training con file txt contenente le coordinate dei difetti

5.3.6 Generazione del modello (YOLO – DARKNET)

Il modello generato è stato addestrato con la stessa configurazione di iper-parametri proposta da IBM. Il corrispondente file di configurazione mostrato nel capitolo 3 è stato modificato al fine di avere la seguente struttura:

Da come si può notare nell'immagine ho mantenuto la suddivisione 8/2 (**batch/subdivision)** presentata nel capitolo 3 e utilizzato lo stesso numero di **iterazioni** (6000) proposte dal modello IBM.

Avendo la possibilità di modificare anche il **Burn-in** ho optato per una scelta di apprendimento non troppo aggressiva (1000) con una policy a step di 1500.

```
[net]
# Testing
batch=1
subdivisions=1
# Training
#batch=8
#subdivisions=2
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=30
saturation = 1.5
exposure = 1.5
hue=.5

learning_rate=0.001
burn_in=1000
max_batches =6000
policy=steps
steps=1500
scales=.1,.1
```

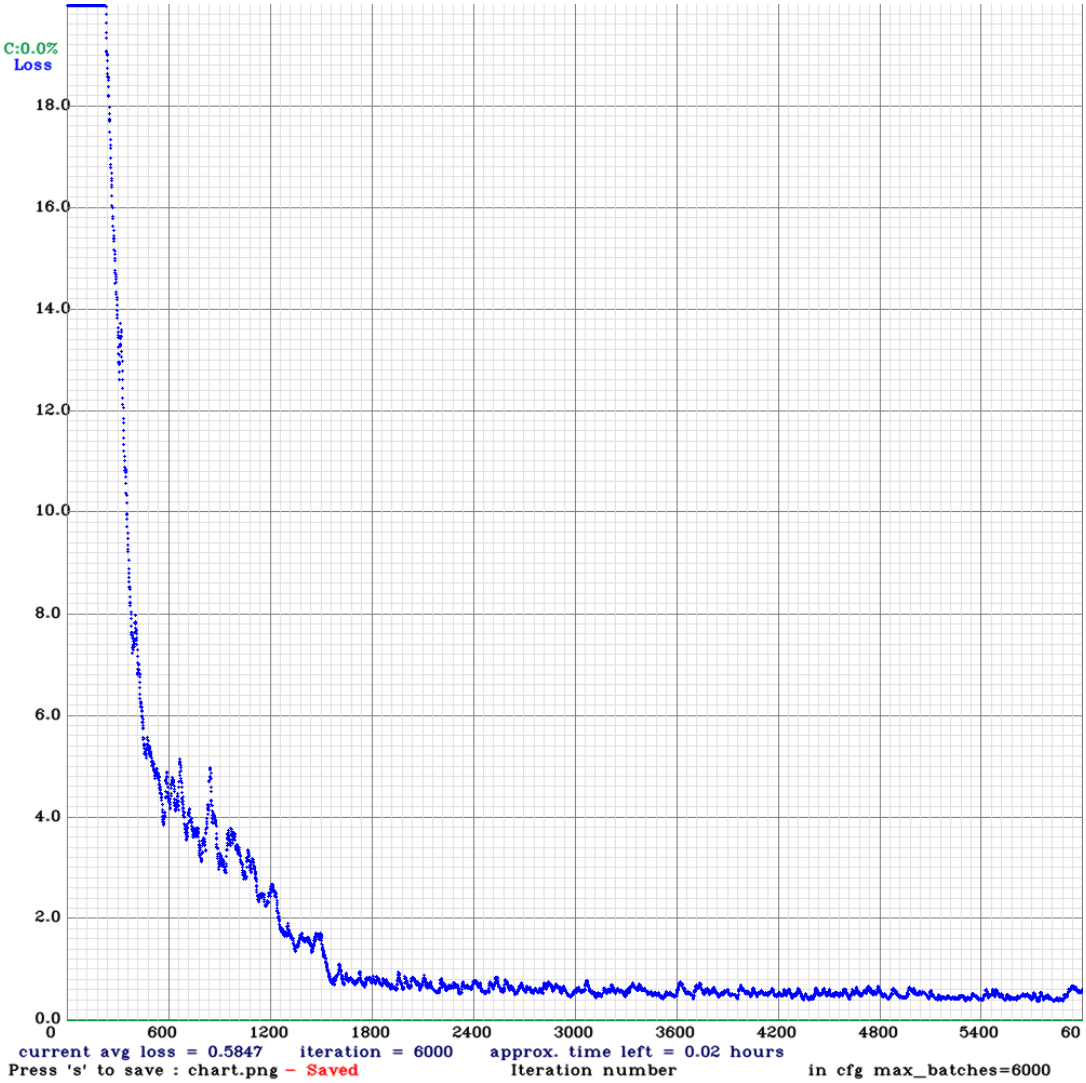


Figura 208 Rapporto iteration / Loss del modello prodotto

5.3.7 Risultati ottenuti (YOLO – DARKNET)

Ecco i risultati ottenuti sottoponendo al modello le due condizioni:

- Test con immagine **GOOD** in input:

Figura 209 Il sistema tenta di riconoscere un'immagine GOOD

- Test con immagine **BAD** in input:

Figura 210 Il sistema tenta di riconoscere un'immagine BAD

5.4 Dataset Lamborghini e definizione delle classi

Dopo alcuni incontri preliminari con Lamborghini e una breve presentazione del prototipo sviluppato per il riconoscimento di difetti sulla Renault Capture l'azienda ha condiviso con noi un primo possibile dataset sulla quale effettuare la sperimentazione.

Il dataset originale ricevuto dall'azienda contava **19 fotografie** complessive di **2** distinti **modelli** di autovettura ovvero Lamborghini Huracàn e Aventador:

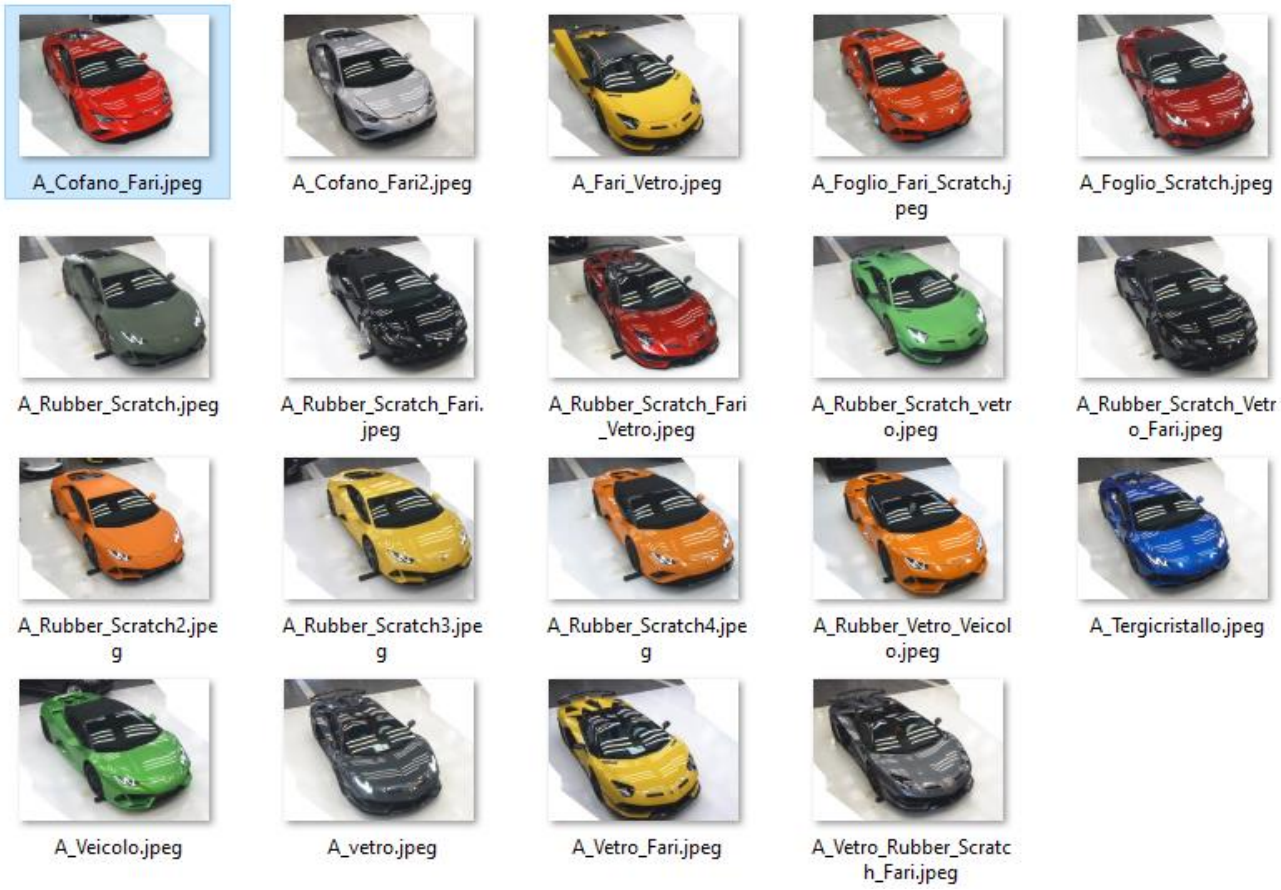


Figura 211 Dataset di partenza Lamborghini Huracàn e Aventador

La prima fase ha richiesto l'individuazione dei singoli difetti all'interno di ciascuna fotografia. Per facilitare il riconoscimento rapido dei vari difetti già a partire dal nome del file ho utilizzato un **codice** in grado di **sintetizzare** lo stato della fotografia. Le difettosità prese in esame sono state:

- **Cofano** (Aperto/Chiuso)
- **Fari** (Accesi/Spenti)
- **Vetro** (Pulito/Sporco)
- **Scratch** presenti sul terreno
- **Rubber** visibili (auto fuori piazzola)

C = COFANO

F = FARI

V = VETRO

S = SCRATCH

R = RUBBER

Ad esempio, un'auto con **tutti i problemi** si chiamerà CFVRS.jpg

Un'auto con solo problemi di cofano aperto e fari spenti si chiamerà CF.jpg

Il dataset di partenza era composto da immagini che comprendevano i seguenti casi GOOD/BAD per ogni possibile classe:

Cofano

2 Bad (CF_01.jpg, CF_02.jpg)

15 Good (FSR_07.jpg, FV_03.jpg, FV_16.jpg, FVS_04.jpg, FVSR_08.jpg, FVSR_10.jpg, FVSR_17.jpg, SR_06.jpg, SR_11.jpg, SR_12.jpg, SR_13.jpg, V_15.jpg, VR_14.jpg, VS_05.jpg, VSR_09.jpg)

Fari

9 Bad (CF_01.jpg, CF_02.jpg, FSR_07.jpg, FV_03.jpg, FV_16.jpg, FVS_04.jpg, FVSR_08.jpg, FVSR_10.jpg, FVSR_17.jpg)

8 Good (SR_06.jpg, SR_11.jpg, SR_12.jpg, SR_13.jpg, V_15.jpg, VR_14.jpg, VS_05.jpg, VSR_09.jpg)

Vetro

10 Bad (FV_03.jpg, FV_16.jpg, FVS_04.jpg, FVSR_08.jpg, FVSR_10.jpg, FVSR_17.jpg, V_15.jpg, VR_14.jpg, VS_05.jpg, VSR_09.jpg)

7 Good (CF_01.jpg, CF_02.jpg, FSR_07.jpg, SR_06.jpg, SR_11.jpg, SR_12.jpg, SR_13.jpg)

Scratch

11 totali (FSR_07.jpg, FVS_04.jpg, FVSR_08.jpg, FVSR_10.jpg, FVSR17.jpg, SR_06.jpg, SR_11.jpg, SR_12.jpg, SR_13.jpg, VS_05.jpg, VSR_09.jpg)

Rubber

10 totali (FSR_07.jpg, FVSR_8.jpg, FVSR_10.jpg, FVSR_17.jpg, SR_06.jpg, SR_11.jpg, SR_12.jpg, SR_13.jpg, RV_14.jpg, VSR_09.jpg)

Le classi individuate per risolvere questo problema sono state **10**:

GOOD_FARO_SX

BAD_FARO_SX

GOOD_FARO_DX

BAD_FARO_DX

GOOD_COFANO

BAD_COFANO

GOOD_VETRO

BAD_VETRO

SCRATCH

RUBBER

5.5 Data augmentation Dataset Lamborghini e suddivisione TRAIN/VALIDATION

Volendo possedere un dataset robusto sulla quale effettuare training è stato effettuato **data augmentation** del dataset di partenza. L'obiettivo costante è stato quello di possedere **almeno 10 fotografie** per ogni caso GOOD/BAD delle classi prese in esame.

Nella generazione sintetica delle immagini sono stati applicati **filtri di colore**, **zoom/traslazione** e **crop**. Essendo le immagini in un ambiente controllato non sono state effettuate modifiche di luminosità poiché essa sarà sempre costante durante la ripresa delle fotografie.

Figura 212 Dataset con data augmentation (33 immagini)

Il risultato dopo il **data augmentation** è stato un dataset di **39 immagini complessive** che sono state suddivise in **TRAIN/VALIDATION** seguendo un rapporto 80% e 20%.

Nel train saranno inserite:

-**33 immagini** (comprendenti situazioni GOOD/BAD)

Nel test saranno inserite:

-**6 immagini** (comprendenti situazioni GOOD/BAD)

5.6 Sviluppo di un modello OPEN SOURCE per il riconoscimento dei difetti (Lamborghini)

Il primo modello sviluppato è stato prodotto con la rete Darknet (algoritmo YOLO). Come già visto diverse volte in questa tesi, la prima fase ha richiesto di individuare le classi e taggare le immagini relative al Train-set preso in esame.

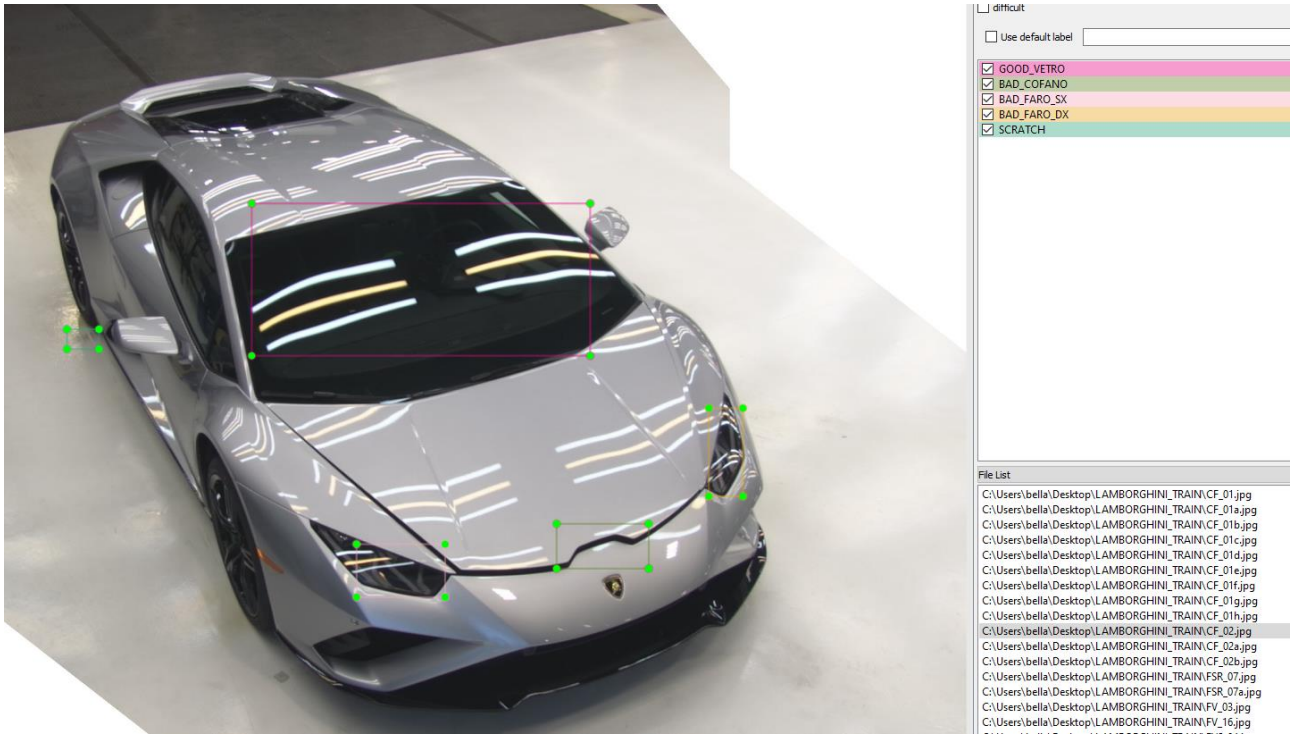


Figura 213 Taggare le auto con l'open source

Una volta terminato di taggare le immagini ho caricato il dataset su Google Drive e ho preparato l'ambiente configurando la rete con il corretto numero di filtri e nodi.

5.6.1 Iperparametri e addestramento della rete

Dovendo trattare 10 classi sono stati utilizzati **49 filtri** nella rete di apprendimento. Ho inoltre deciso di effettuare training per **6000 iterazioni** con suddivisione 8/2 (**batch/subdivision), Burn-in 1000** e policy a **step** di **1500**.

L'addestramento della rete è avvenuto in circa **2 ore** e ha generato i seguenti risultati (matrice di confusione e rapporto LOSS/Iteration):

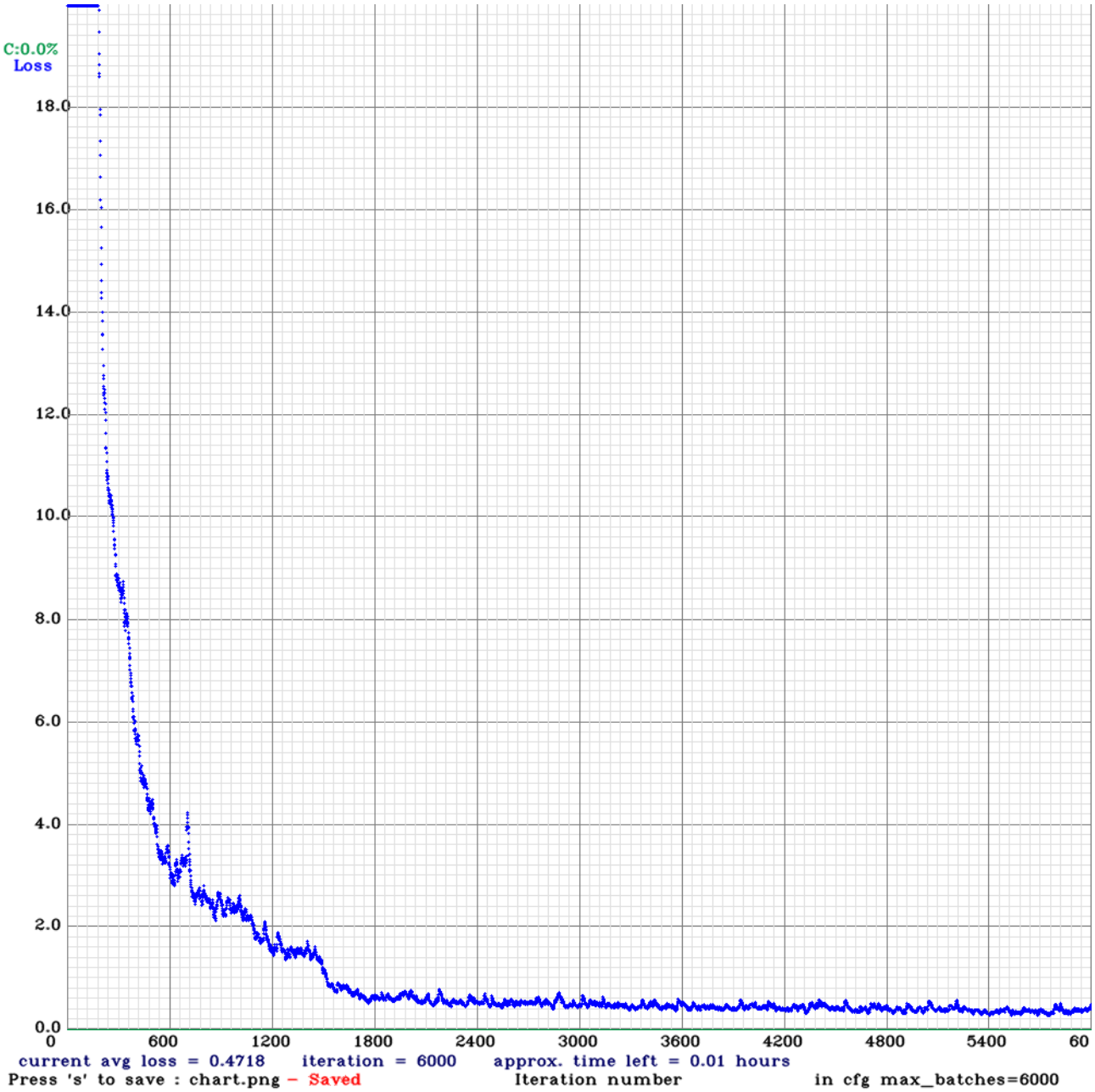


Figura 214 Rapporto Iteration / loss del modello

		Truth data											
		Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	Class 10	Classification overall	User's accuracy (Precision)
Classifier results	Class 1	3	0	0	0	0	0	0	0	0	0	3	100%
	Class 2	0	5	0	0	0	0	0	0	0	0	5	100%
	Class 3	0	0	2	1	0	0	0	0	0	0	3	66.667%
	Class 4	0	0	1	4	0	0	0	0	0	0	5	80%
	Class 5	0	0	0	0	7	0	0	0	0	0	7	100%
	Class 6	0	0	0	0	0	1	0	0	0	0	1	100%
	Class 7	0	0	0	0	0	0	5	0	0	0	5	100%
	Class 8	0	0	0	0	0	0	0	3	0	0	3	100%
	Class 9	0	0	0	0	0	0	0	0	6	0	6	100%
	Class 10	0	0	0	0	0	0	0	0	0	4	4	100%
	Truth overall	3	5	3	5	7	1	5	3	6	4	42	
	Producer's accuracy (Recall)	100%	100%	66.667%	80%	100%	100%	100%	100%	100%	100%		

Overall accuracy (OA):	95.238%
Kappa[1]:	0.946

Figura 215 Confusion matrix modello

Su 3 immagini di **GOOD_FARO_SX** il sistema non ha **mai confuso** la classificazione

Su 5 immagini di **BAD_FARO_SX** il sistema non ha **mai confuso** la classificazione

Su 3 immagini di **GOOD_FARO_DX** il sistema ha confuso 1 classificazione scambiando un **GOOD_FARO_DX** con un **BAD_FARO_DX**

Su 5 immagini di **BAD_FARO_DX** il sistema ha confuso 1 classificazione scambiando un **BAD_FARO_DX** con un **GOOD_FARO_DX**

Su 7 immagini di **GOOD_COFANO** il sistema non ha **mai confuso** la classificazione

Su 1 immagini di **BAD_COFANO** il sistema non ha **mai confuso** la classificazione

Su 5 immagini di **GOOD_VETRO** il sistema non ha **mai confuso** la classificazione

Su 3 immagini di **BAD_VETRO** il sistema non ha **mai confuso** la classificazione

Su 6 immagini contenenti **SCRATCH** il sistema non ha **mai confuso** la classificazione

Su 4 immagini contenenti **RUBBER** il sistema non ha **mai confuso** la classificazione

Complessivamente la precisione del sistema risulta **PRECISION = 95.238%**

5.6.2 Risultati Ottenuti

Le seguenti immagini sono state fornite in input al sistema per effettuare riconoscimento e classificazione di situazioni ed anomalie presenti sui veicoli. Ecco i risultati ottenuti:

- Veicolo con **vetro pulito**, alcuni **scratch** sul pavimento, **fuori piazzola**, **fari spenti** e **cofano chiuso**.

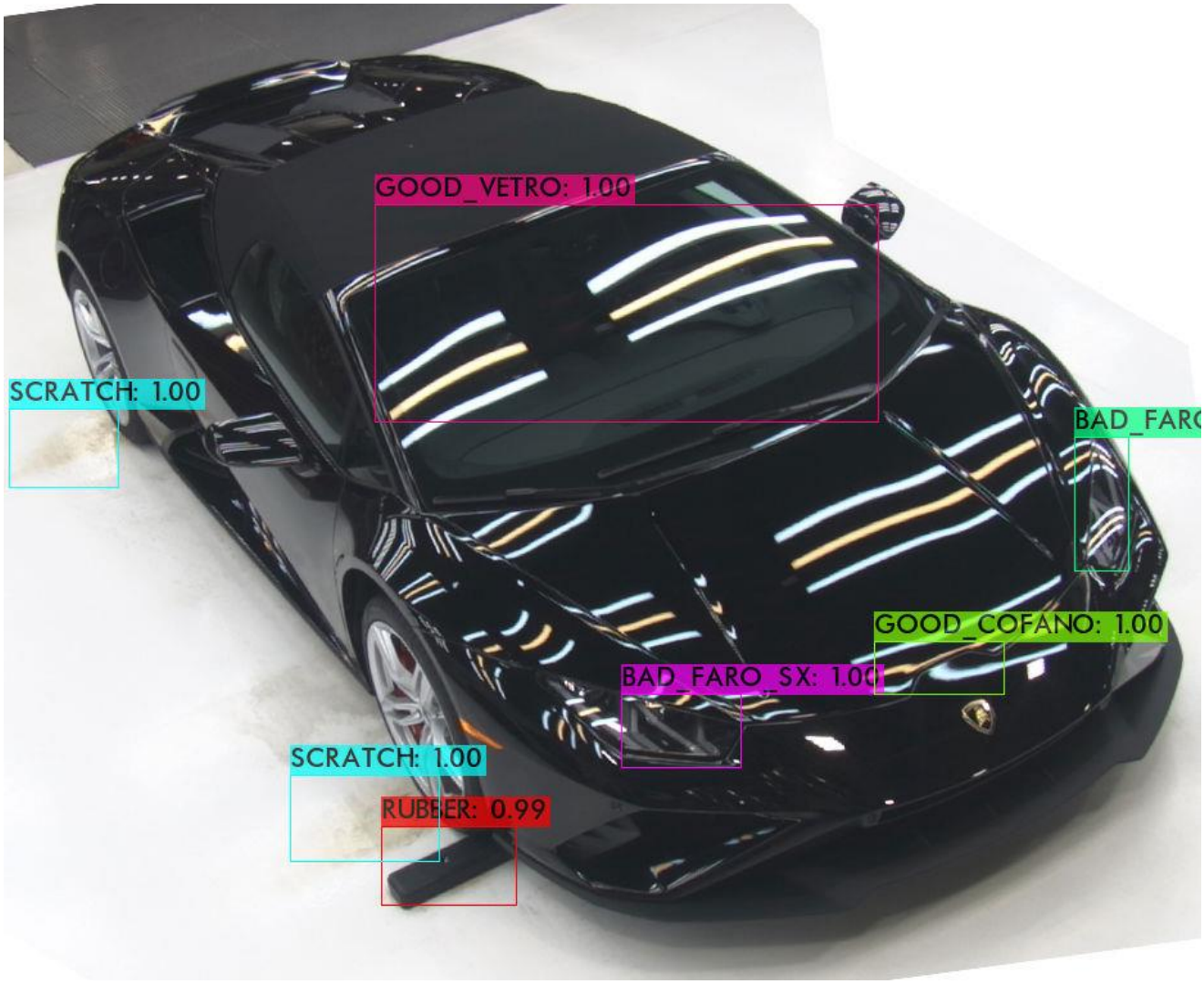


Figura 216 Il sistema riconosce efficacemente tutti i problemi del veicolo

- Veicolo con **foglio sul cruscotto**, 3 **scratches**, **fari spenti** e **cofano chiuso** correttamente.

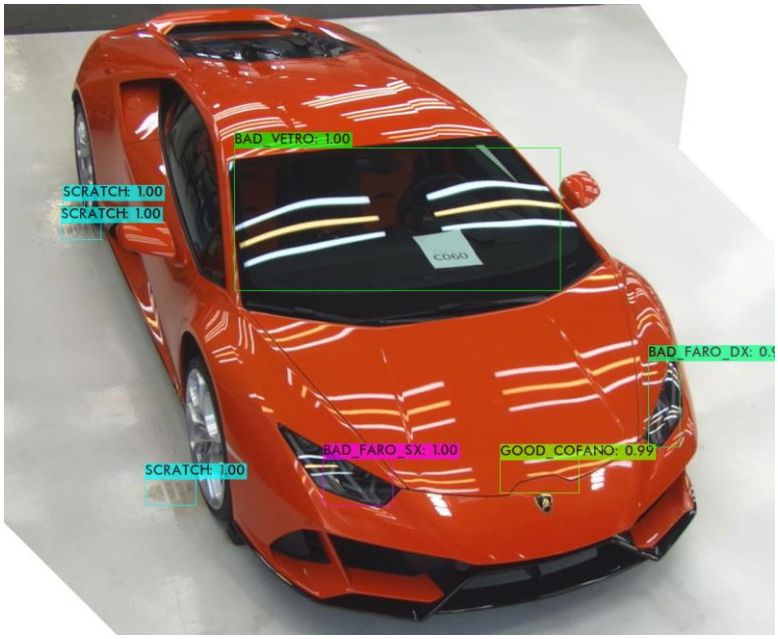


Figura 217 Il sistema riconosce efficacemente tutti i problemi del veicolo 2

- Veicolo con **cruscotto pulito**, 1 **scratch** sul pavimento, **fari accesi** e **cofano** chiuso correttamente.

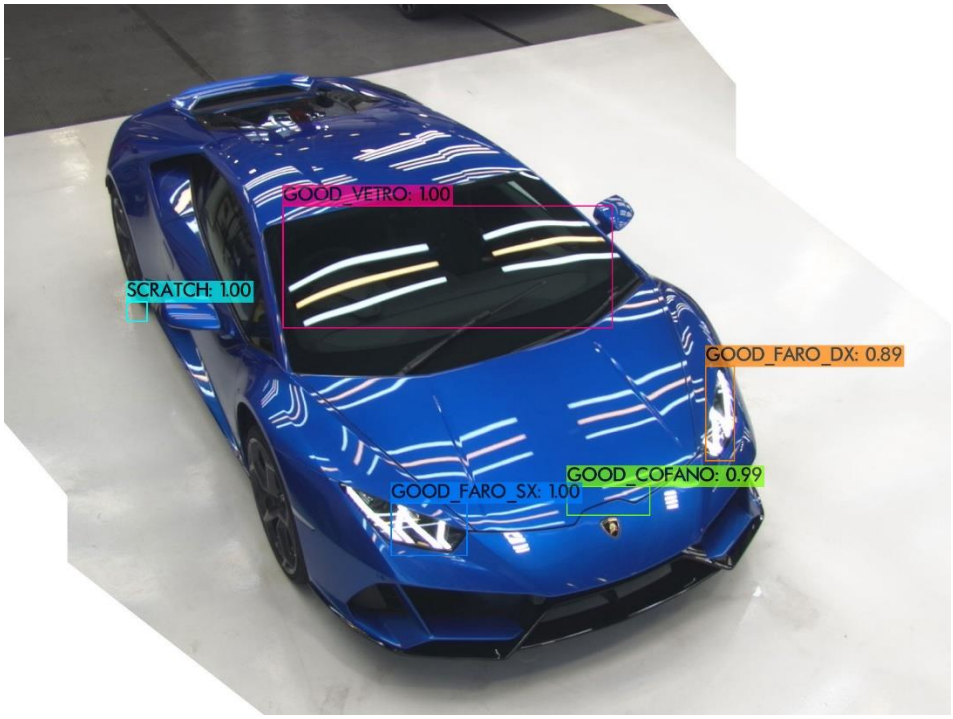


Figura 218 Il sistema riconosce efficacemente tutti i problemi del veicolo 3

- Veicolo con **cruscotto pulito**, nessuno **scratch** a terra ma **disallineata**, **cofano chiuso** e **fari accesi.**

Figura 219 Il sistema riconosce efficacemente tutti i problemi del veicolo 4

- Veicolo **fortemente ruotato** con **cofano aperto, vetro pulito, 1 scratch** sul terreno e **fari spenti**.

Figura 220 Il sistema riconosce efficacemente tutti i problemi del veicolo 5

- Veicolo con **crop** sul **bordo destro**, **fari accesi**, **cofano chiuso**, **scracth** e **vetro pulito**

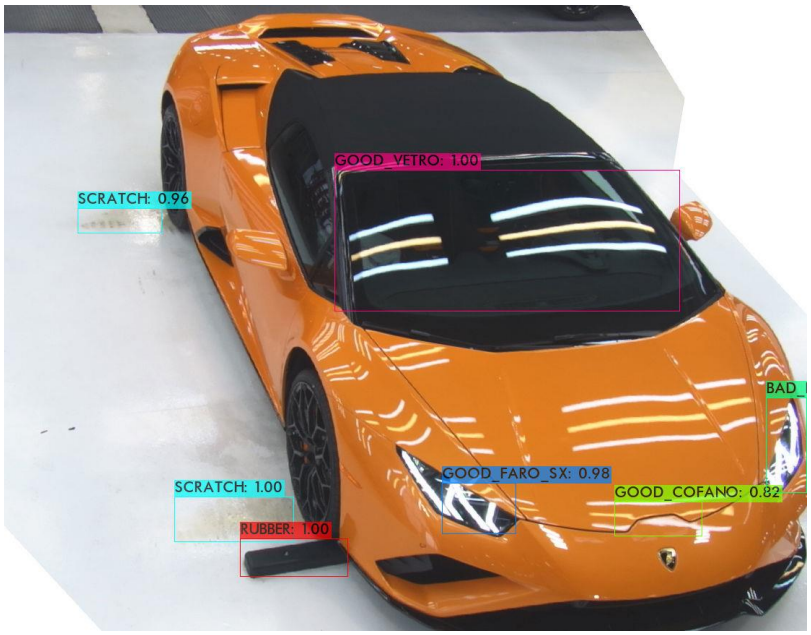


Figura 221 Il sistema riconosce efficacemente tutti i problemi del veicolo 6

- Veicolo con **fari spenti**, **cruscotto sporco, cofano chiuso** e assenza di scratch/rubber.

Figura 222 Il sistema riconosce efficacemente tutti i problemi del veicolo 7

Per valutare la robustezza del sistema ho fatto anche questo esperimento con un veicolo in posizione differente da quelli forniti al sistema durante l'addestramento. I risultati sono ovviamente inferiori a livello qualitativo, ma nel complesso il modello è stato in grado di riconoscere alcuni dettagli interessanti:

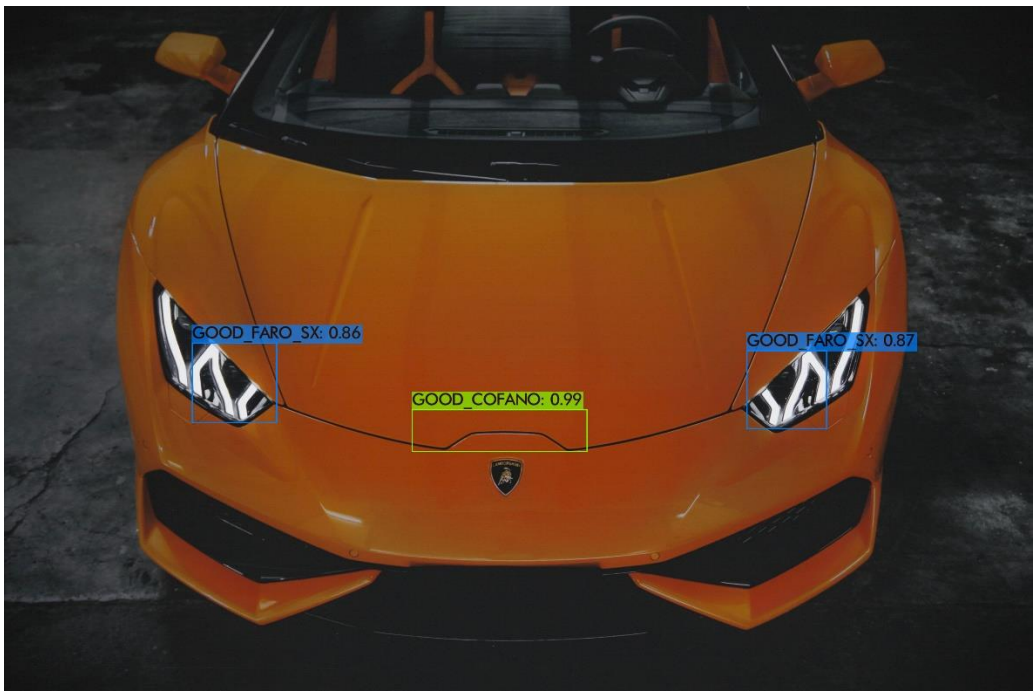


Figura 223 Il sistema riconosce alcuni elementi di un veicolo catturato con angolazione differente rispetto al dataset

ESTENSIONE DEL LAVORO E SVILUPPI FUTURI

L'obiettivo finale di questo progetto di tesi è stato quello di creare una figura esperta sul campo della visual inspection e applicare queste conoscenze su un caso di studio reale. Durante lo svolgimento di questo elaborato non sono state prese in considerazione le integrazioni dei modelli prodotti con open source all'interno di dispositivi differenti dalle interfacce web messe a disposizione da Google Collab.

Sebbene io abbia avuto la possibilità di testare le potenzialità di un modello su dispositivo mobile (prodotto con il software e affiancato dall'applicazione IBM), questa tipologia di ricerca non è stata mai integrata utilizzando sistemi completamente open source.

Come possibile estensione di questo lavoro potrebbe essere interessante progettare un sistema che, facendo uso di un modello prodotto tramite software open source (come la rete Darknet e l'algoritmo YOLOV3), possa essere facilmente trasportato su dispositivo mobile o installato su apparecchiature in grado di effettuare monitoraggio attivo e in tempo reale con i dati elaborati grazie al modello.

La possibilità di utilizzare dei droni potrebbe essere d'aiuto agli operatori che effettuano ispezioni visive giornalmente. L'obiettivo di queste sperimentazioni non sarebbe quello di sostituire l'operatore ma affiancare ad esso uno strumento in grado di fornirgli supporto in situazioni dove il controllo umano diretto può essere particolarmente complicato (si pensi al monitoraggio di piattaforme che si trovano in posizioni molto elevate o alla manutenzione strutturale di ponti e strade). Le potenzialità della tecnologia legata alla Visual Inspection può trovare sbocco in una moltitudine di scenari che vanno dall'industria farmaceutica all'automotive.

Oggi la tecnologia si trova in una fase molto matura e consente a chiunque abbia un minimo di esperienza a livello di programmazione di cimentarsi nello sviluppo di pipeline potenzialmente applicabili anche in ambito industriale.

Riferimenti Bibliografici

[1]BGC, "*Sprinting to value in Industry 4.0*", Dicembre 2016 (versione online, https://www.bcg.com/it-it/publications/2016/lean-manufacturing-technology-digital-sprinting-to-value-industry-40, data ultima consultazione 6 Ottobre 2020).

[2]BMW, "*Agumented reality*" (https://www.press.bmwgroup.com/global/article/detail/T0294345EN/absolutely-real:-virtual-and-augmented-reality-open-new-avenues-in-the-bmw-group-production-system?language=en, data ultima consultazione 12 Ottobre 2020).

[3]UniversalRobots, "*Universal Robots doubles production at Aerospace Manufacturer Despite Labor shortage*", 11 febbraio 2020 (https://www.youtube.com/watch?v=egUv1icA924&ab_channel=UniversalRobots, data ultima consultazione 6 Ottobre 2020).

[4]IoT Analytics, "*Industry 4.0 & smart Manufacturing 2018-2023*", 14 novembre 2018 (https://iot-analytics.com/industry-4-0-and-smart-manufacturing/, data ultima consultazione 12 Ottobre 2020).

[5]Valbruna Group, "*Controlli qualità- test meccanici distruttivi*" (https://www.valbruna-stainless-steel.com/it/quality/controls, data ultima consultazione 7 Ottobre 2020).

[6]Merieux NutriSciences, "*Test di controllo qualità*" (https://www.merieuxnutrisciences.com/it/pharma-e-dispositivi-medici/test-di-controllo-qualita, data ultima consultazione 7 Ottobre 2020).

[7]TecEurolab, "*Magnetoscopia*" (https://www.youtube.com/watch?v=nyP5NMFtaXo, data ultima consultazione 7 Ottobre 2020).

[8]Il sole 24 ore, "*Samsung Galaxy note 7, la colpa delle esplosioni è tutta delle batterie*", Gennaio 2017 (versione online, https://www.ilsole24ore.com/art/samsung-galaxy-note-7la-colpa-esplosioni-e-tutta-batterie-AEvrpeF, data ultima consultazione 8 Ottobre 2020).

[9]Condomett, "*Esame visivo (VT)*" (https://www.condomett.it/controlli-non-distruttivi/esame-visivo/, data ultima consultazione 8 Ottobre 2020).

[10]Shelton Vision "*Automotive fabric producer protects customers, reduces claims and increases throughput*" (https://www.sheltonvision.co.uk/case-studies/automotive-fabric-producer/, data ultima consultazione 16 Ottobre 2020).

[11]FIP International Pharmaceutical Federation, "*Tool for visual inspection of medicines*" (https://www.fip.org/files/fip/counterfeit/VisualInspection/A%20tool%20for%20visual%20inspection%20of%20medicines%20EN.pdf, data ultima consultazione 14 Ottobre 2020).

[12]Stevanto Group, "*Tecnologie di ispezione visiva*" (https://www.stevanatogroup.com/it/offerta/ispezione-visiva/tecnologie-di-ispezione/, data ultima consultazione 17 Ottobre 2020).

[13]Sipotek, "*Vision solutions*" (https://www.sipotek.net/vision-solutions/, data ultima consultazione 18 ottobre 2020).

[14]EyeProSystem, "*Product Inspection*" (https://www.eyeprosystem.com/content/product-inspection, data ultima consultazione 18 Ottobre 2020).

[15]Nanonets, "*Car insurance*" (https://nanonets.com/case-study/car-insurance, data ultima consultazione 8 Ottobre 2020).

[16]Stevanto group (https://www.stevanatogroup.com/it/offerta/ispezione-visiva/innovazione/, data ultima consultazione 22 Ottobre 2020).

[17] "*Studies of Visual Inspection*", luglio 1973 (https://pubmed.ncbi.nlm.nih.gov/28086275/, data ultima consultazione 20 Ottobre 2020).

[18]"*Global Manufacturing Companies Trust LandingLens to Enhance Their Existing Visual Inspection Systems with AI*", Paolo alto, 21 Ottobre 2020 (https://landing.ai/landing-ai-unveils-ai-visual-inspection-platform-to-improve-quality-and-reduce-costs-for-manufacturers-worldwide/, data ultima consultazione 22 Ottobre 2020).

[19]GPT-3 (https://en.wikipedia.org/wiki/GPT-3, data ultima consultazione 30 Ottobre2020).

[20]Transfer Learning (https://en.wikipedia.org/wiki/Transfer_learning, data ultima consultazione 30 Ottobre 2020).

[21]Amazon rekognition(https://aws.amazon.com/it/rekognition/?nc=sn&loc=0&blog-cards.sort-by=item.additionalFields.createdDate&blog-cards.sort-order=desc, data ultima consultazione 31 Ottobre 2020).

[22]Amazon Rekognition Wikipedia (https://en.wikipedia.org/wiki/Amazon_Rekognition#cite_note-HK-16, data ultima consultazione 3 Novembre 2020).

[23]Start Magazine, "*Perché Amazon (dopo Ibm) sospende Rekognition*" (https://www.startmag.it/innovazione/perche-amazon-dopo-ibm-sospende-rekognition/, data ultima consultazione 3 Novembre 2020).

[24]Visione artificiale documentazione Microsoft (https://docs.microsoft.com/it-it/azure/cognitive-services/computer-vision/overview, data ultima consultazione 3 Novembre 2020).

[25]Towards data science, "*Neural Architecture Search (NAS) – The Future of deep learning*" (https://towardsdatascience.com/neural-architecture-search-nas-the-future-of-deep-learning-c99356351136, data ultima consultazione 7 Novembre 2020).

[26]Google AutoML Natural language documentation (https://cloud.google.com/natural-language/automl/docs, data ultima consultazione 7 Novembre 2020).

[27]Google AutoML Tables, (https://www.youtube.com/watch?v=tWbiOuHae0c&t=206s&ab_channel=GoogleCloudPlatform, data ultima consultazione 7 Novembre 2020).

[28]Variations of SSD- Understanding Deconvolutional Single-Shot Detectors (https://medium.com/@amadeusw6/variations-of-ssd-understanding-deconvolutional-single-shot-detectors-c0afb8686d03, data ultima consultazione 18 Novembre 2020).

[29]Mean Average Precision for Object Detection, Jonathan hui (https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173#:~:text=mAP%20(mean%20average%20precision)%20is,difference%20between%20AP%20and%20mAP., data ultima consultazione 18 Novembre 2020).

[30]Google AutoML Vision and Video Intelligence (https://www.youtube.com/watch?v=kgxfdTh9lz0&ab_channel=GoogleCloudPlatform, data ultima consultazione 7 Novembre 2020).

[31]IBM Cloud services (https://cloud.ibm.com/catalog?category=ai#services, data ultima consultazione 9 Novembre 2020)

[32]Luminoth Open source Toolkit(https://luminoth.readthedocs.io/en/latest/index.html, data ultima consultazione 21 Novembre 2020).

[33]Algolux (https://algolux.com/, data ultima consultazione 10 Novembre 2020).

[34]Algolux vs Tesla algorithm (https://www.youtube.com/watch?v=nmWTMxc--n0&feature=emb_title&ab_channel=Algolux, data ultima consultazione 10 Novembre 2020).

[35]Deep Vision AI (https://www.deepvisionai.com/, data ultima consultazione 10 Novembre 2020).

[36]Sighthound (https://www.sighthound.com/, data ultima consultazione 10 Novembre 2020).

[37] ViSenze (https://www.visenze.com/, data ultima consultazione 10 Novembre 2020).

[38]Umbo CV (https://umbocv.ai/, data ultima consultazione 10 Novembre 2020).

[39]Landing Ai(https://landing.ai/, data ultima consultazione 19 Novembre 2020).

[40]Landing AI platform guide (https://landing.ai/wp-content/uploads/2020/10/LandingAI_VI_Platform_Feature_Guide.10-20.pdf, data ultima consultazione 19 Novembre 2020).

[41] Cogniac (https://cogniac.co/)

[42] PEKAT (https://www.pekatvision.com/)

[43]Forrester (https://go.forrester.com/blogs/, data ultima consultazione 11 Novembre 2020).

[44]The Forrester New Wave : Computer Vision Platforms, Q4 2019

(https://www.forrester.com/report/The+Forrester+New+Wave+Computer+Vision+Platforms+Q4+2019/-/E-RES144576, data ultima consultazione 11 Novembre 2020).

[45]Microsoft Azure vision Demo (https://azure.microsoft.com/it-it/services/cognitive-services/computer-vision/#features, data ultima consultazione 25 Novembre 2020).

[46]Google vision API Demo (https://cloud.google.com/vision/docs/drag-and-drop, data ultima consultazione 25 Novembre 2020).

[47]IBM Watson visual recognition Demo(https://visual-recognition-code-pattern.ng.bluemix.net/, data ultima consultazione 25 Novembre 2020).