**Documento tecnico per l’Applicazione di Computer Vision**

**per rilevare i difetti di produzione**

Ermano Buikis

Systematik s.r.l. anno 2022

****

**Indice**

* Introduzione
* Descrizione dettagliata dell’applicazione
* Descrizione degli oggetti in un immagine
  + contorni
  + colori
  + oggetti
* Possibili difficolta’ per lo sviluppo di Computer Vision
* Cosa e’ una rete neurale
* Workflow dell’applicazione
* Generare un processo in background nel server
* Esempi di comandi
  + prendere le immagini con le videocamere
  + lanciare l’applicazione
* Hardware
  + Videocamere
  + Raspberry PI
  + TPU e Coral AI
  + GPU & CUDA (NVIDIA)
* Software utilizzati
  + python3
  + opencv
  + tensorflow
  + keras
  + pixellib
  + labelImg
* Problematiche tecniche & possibili soluzioni
* Esperienze di acquisizione & risultati
* Costruire una rete neurale
* Architetture di reti neurali per la Object Detection
  + rcnn
  + faster rcnn
  + mask rcnn
* Ottimizzare il modello di rete neurale
  + Transfer learning
  + Neuroevoluzione
    - NEAT
    - TPOT
* MLOps e riproducibilita’ del modello
* Risultati sperimentali
  + Visualizzazione degli errori di produzione
  + Scambio di messaggi con PLC

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Introduzione**

Il Sistema di Computer Vision di Systematik s.r.l. è un sistema composto da hardware e software con l’obbiettivo di migliorare il lavoro giornaliero ddi rilevare errori durante il processo industriale in maniera automatica, andando così ad ottimizzare il processo stesso. Attraverso l’introduzione di una o più videocamere all’ interno del processo industriale, è possibile connette queste videocamere ad un computer apposito, il quale visualizza le immagini, ed attraverso un sistema di algoritmi informatici, è in grado di rilevare errori nel prodotto con un’efficienza pari ad un lavoratore medio.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Descrizione del problema**

**Lista dei difetti di produzione**

* Fori
* Sale e pepe
* Scccarrellature
* Occhio di pesce
* Lama raica
* Rigate (accumulo di colore)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Descrizione dell’applicazione**

Questa Applicazione di Computer Vision permette di:

1. Comunicare attraverso socket con il microcontrollore PLC connesso ai macchinari industriali.
2. Quando viene attivato un evento da remoto tramite PLC, permette di scattare una foto ad una o più videocamere connesse in locale.
3. Processare la foto per verificarne la presenza di errori nel prodotto finito.
4. Inviare il risultato dell’elaborazione ad un’interfaccia ed alla PLC connessa al macchinario.

Quest’app non è niente che di meno di un processo sul server lanciato in background. Questo processo deve mantenere attiva la connessione con la videocamera e con la PLC.

##### **List of the Most Popular Computer Vision Tools in 2022**

* Tool #1: OpenCV
* Tool #2: Viso Suite
* Tool #3: TensorFlow
* Tool #4: CUDA
* Tool #5: MATLAB
* Tool #6: Keras
* Tool #7: SimpleCV
* Tool #8: BoofCV
* Tool #9: CAFFE
* Tool #10: OpenVINO
* Tool #11: DeepFace
* Tool #12: YOLO

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### **COSA e’ l’ Object Detection**

**Questo articolo fornirà un'introduzione al rilevamento degli oggetti e fornirà una panoramica degli algoritmi di rilevamento degli oggetti di visione artificiale all'avanguardia. Il rilevamento di oggetti è un campo chiave nell'intelligenza artificiale, poiché consente ai sistemi informatici di "vedere" i loro ambienti rilevando oggetti in immagini o video visivi.**

In particolare imparerai:

* Che cos'è il rilevamento degli oggetti e come si è evoluto negli ultimi 20 anni
* Tipi di metodi di rilevamento degli oggetti di visione artificiale
* Elenchiamo esempi, casi d'uso e applicazioni di rilevamento di oggetti
* Gli algoritmi di rilevamento degli oggetti più popolari oggi
* Nuovi algoritmi di riconoscimento degli oggetti introdotti nel 2022

Informazioni: su viso.ai, forniamo la principale piattaforma di visione artificiale senza codice [Viso Suite](https://viso-ai.translate.goog/features/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) . La soluzione integrata aiuta le organizzazioni di tutto il mondo a creare, distribuire, scalare e proteggere le proprie applicazioni di visione artificiale.

### Che cos'è il rilevamento di oggetti?

Il rilevamento di oggetti è un'importante attività di visione artificiale utilizzata per rilevare istanze di oggetti visivi di determinate classi (ad esempio esseri umani, animali, automobili o edifici) in immagini digitali come foto o fotogrammi video. L'obiettivo del rilevamento degli oggetti è quello di sviluppare modelli computazionali che forniscano le informazioni più fondamentali necessarie alle [applicazioni di visione artificiale](https://viso-ai.translate.goog/applications/computer-vision-applications/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) : " Quali oggetti sono dove ?".

##### **Perché il rilevamento degli oggetti è importante?**

Il rilevamento di oggetti è uno dei problemi fondamentali della visione artificiale. Costituisce la base di molte altre attività di visione artificiale a valle, ad esempio segmentazione di istanze, sottotitoli di immagini, tracciamento di oggetti e altro ancora. Applicazioni specifiche per il rilevamento di oggetti includono il rilevamento dei pedoni, il [conteggio delle persone](https://viso-ai.translate.goog/applications/people-counting-system/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) , il rilevamento del [volto, il rilevamento](https://viso-ai.translate.goog/deep-learning/face-detection-overview/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) del testo, il rilevamento della [posa](https://viso-ai.translate.goog/deep-learning/pose-estimation-ultimate-overview/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) o [il riconoscimento della targa](https://viso-ai.translate.goog/computer-vision/automatic-number-plate-recognition-anpr/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) .

##### **Rilevamento di oggetti e Deep Learning**

Negli ultimi anni, i rapidi progressi delle tecniche di deep learning hanno notevolmente accelerato lo slancio del rilevamento degli oggetti. Con le reti di deep learning e la potenza di calcolo delle GPU, le prestazioni dei rilevatori di oggetti e dei tracker sono notevolmente migliorate, ottenendo significativi progressi nel rilevamento degli oggetti.

L'apprendimento automatico (ML) è una branca dell'intelligenza artificiale (AI) e implica essenzialmente modelli di apprendimento da esempi o dati di esempio mentre la macchina accede ai dati e ha la capacità di imparare da essi (apprendimento supervisionato su [immagini annotate](https://viso-ai.translate.goog/computer-vision/image-annotation/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) ). Il deep learning è una forma specializzata di machine learning che prevede l'apprendimento in diverse fasi.

##### **Gli ultimi progressi tecnologici**

È diventata disponibile un'ampia gamma di [applicazioni di visione artificiale per il rilevamento e il tracciamento di oggetti.](https://viso-ai.translate.goog/applications/computer-vision-applications/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) Di conseguenza, numerose applicazioni del mondo reale, come il monitoraggio sanitario, la guida autonoma, la videosorveglianza, il rilevamento di anomalie o la visione robotica, si basano sul rilevamento di oggetti di deep learning.

La tecnologia di imaging è notevolmente progredita negli ultimi anni. Le fotocamere sono più piccole, più economiche e di qualità superiore che mai. Nel frattempo, la potenza di calcolo è aumentata notevolmente ed è diventata molto più efficiente. Negli anni passati, le piattaforme di elaborazione si sono spostate verso la parallelizzazione attraverso l'elaborazione multi-core, l'unità di elaborazione grafica (GPU) e gli acceleratori di intelligenza artificiale come le unità di elaborazione tensoriale (TPU)

Tale hardware consente di eseguire la visione artificiale per il rilevamento e il tracciamento di oggetti in implementazioni quasi in tempo reale. Pertanto, il rapido sviluppo delle reti neurali convoluzionali profonde (CNN) e la maggiore potenza di calcolo della GPU sono i principali fattori alla base del grande progresso del rilevamento di oggetti basato sulla visione artificiale.

### **Come funziona il rilevamento degli oggetti**

Il rilevamento degli oggetti può essere eseguito utilizzando le tradizionali (1) tecniche di elaborazione delle immagini o le moderne (2) reti di deep learning.

1. Le tecniche di elaborazione delle immagini generalmente non richiedono dati storici per l'addestramento e sono di natura non supervisionata.
   * Pro: quindi, queste attività non richiedono [immagini annotate](https://viso-ai.translate.goog/computer-vision/image-annotation/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) , in cui gli esseri umani hanno etichettato i dati manualmente (per l'addestramento supervisionato).
   * Contro: queste tecniche sono limitate a molteplici fattori, come scenari complessi (senza sfondo unicolor), occlusione (oggetti parzialmente nascosti), illuminazione e ombre ed effetto disordine.
2. I metodi di Deep Learning generalmente dipendono dalla formazione supervisionata. Le prestazioni sono limitate dalla potenza di calcolo delle GPU che aumenta rapidamente di anno in anno.
   * Pro: il rilevamento di oggetti di apprendimento profondo è significativamente più robusto per l'occlusione, le scene complesse e l'illuminazione difficile.
   * Contro: è richiesta un'enorme quantità di dati di allenamento; il processo di [annotazione dell'immagine](https://viso-ai.translate.goog/computer-vision/image-annotation/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) è laborioso e costoso. Ad esempio, l'etichettatura di 500.000 immagini per addestrare un algoritmo di rilevamento di oggetti DL personalizzato è considerato un piccolo set di dati. Tuttavia, molti set di dati di riferimento ( [MS COCO](https://viso-ai.translate.goog/computer-vision/coco-dataset/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) , Caltech, KITTI, PASCAL VOC, V5) forniscono la disponibilità di dati etichettati.

### **Pietre miliari nel rilevamento di oggetti all'avanguardia**

Il campo del rilevamento degli oggetti non è così nuovo come potrebbe sembrare. In effetti, il rilevamento degli oggetti si è evoluto negli ultimi 20 anni. L'avanzamento del rilevamento degli oggetti è solitamente suddiviso in due periodi storici separati (prima e dopo l'introduzione del Deep Learning):

Prima del 2014 – Periodo di rilevamento degli oggetti tradizionale

1. Viola-Jones Detector (2001), il lavoro pionieristico che ha avviato lo sviluppo dei metodi tradizionali di rilevamento degli oggetti
2. HOG Detector (2006), un popolare descrittore di funzionalità per il rilevamento di oggetti nella visione artificiale e nell'elaborazione delle immagini
3. DPM (2008) con la prima introduzione della regressione del riquadro di delimitazione

Dopo il 2014 – Periodo di rilevamento del deep learning

I più importanti algoritmi di rilevamento di oggetti a due stadi

1. RCNN e SPPNet (2014)
2. Fast RCNN e Faster RCNN (2015)
3. Mask R-CNN (2017)
4. Reti piramidali/FPN (2017)
5. G-RCNN (2021)

I più importanti  algoritmi di rilevamento di oggetti a uno stadio

1. YOLO (2016)
2. SSD (2016)
3. Rete Retina (2017)
4. YOLOv3 (2018)
5. YOLOv4 (2020)
6. YOLOR (2021)

##### **Rilevatori di oggetti di deep learning a uno stadio e a due stadi**

Come puoi vedere nell'elenco sopra, i metodi di rilevamento di oggetti all'avanguardia possono essere classificati in due tipi principali: rilevatori di oggetti a uno stadio o a due stadi.

In generale, i rilevatori di oggetti basati sul deep learning estraggono le caratteristiche dall'immagine di input o dal frame video. Un rilevatore di oggetti risolve due compiti successivi:

* Attività n. 1: Trova un numero arbitrario di oggetti (possibilmente anche zero) e
* Compito #2: classifica ogni singolo oggetto e stima la sua dimensione con un rettangolo di selezione.

Per semplificare il processo, puoi separare queste attività in due fasi. Altri metodi combinano entrambe le attività in un unico passaggio (rilevatori a stadio singolo) per ottenere prestazioni più elevate a scapito della precisione.

**Rilevatori a due stadi:** nei rilevatori di oggetti a due stadi, le regioni dell'oggetto approssimative vengono proposte utilizzando funzionalità profonde prima che queste funzionalità vengano utilizzate per la classificazione, nonché la regressione del riquadro di delimitazione per l'oggetto candidato.

L'architettura a due stadi prevede (1) una proposta di regione dell'oggetto con metodi convenzionali di visione artificiale o reti profonde, seguita da (2) classificazione di oggetti basata su caratteristiche estratte dalla regione proposta con regressione del riquadro di delimitazione.

* I metodi a due stadi ottengono la massima precisione di rilevamento ma sono in genere più lenti. A causa dei numerosi passaggi di inferenza per immagine, le prestazioni (fotogrammi al secondo) non sono buone come i rilevatori a uno stadio.
* Vari rilevatori a due stadi includono la rete neurale convoluzionale regionale (RCNN), con evoluzioni R-CNN più veloci o Mask R-CNN. L'ultima evoluzione è il granulato RCNN (G-RCNN).
* I rilevatori di oggetti a due stadi trovano innanzitutto una regione di interesse e utilizzano questa regione ritagliata per la classificazione. Tuttavia, tali rilevatori multistadio di solito non sono addestrabili end-to-end perché il ritaglio è un'operazione non differenziabile.

Rilevatori a uno stadio: i rilevatori a uno stadio prevedono riquadri di delimitazione sulle immagini senza la fase di proposta della regione. Questo processo richiede meno tempo e può quindi essere utilizzato in applicazioni in tempo reale.

* I rilevatori di oggetti a uno stadio danno la priorità alla velocità di inferenza e sono super veloci ma non così bravi nel riconoscere oggetti di forma irregolare o un gruppo di piccoli oggetti.
* I rilevatori a uno stadio più popolari includono YOLO, SSD e RetinaNet. Gli ultimi rilevatori in tempo reale sono YOLOv4-Scaled (2020) e YOLOR (2021). Visualizza i confronti dei benchmark di seguito.
* Il vantaggio principale del single-stage è che questi algoritmi sono generalmente più veloci dei rivelatori multistadio e strutturalmente più semplici.

##### R-CNN - Reti neurali convoluzionali regionali

Le reti neurali convoluzionali basate sulla regione o le regioni con funzionalità CNN (R-CNN) sono approcci [pionieri](https://translate.google.com/website?sl=en&tl=it&hl=en-US&client=webapp&u=https://d2l.ai/chapter_references/zreferences.html%23girshick-donahue-darrell-ea-2014) che applicano modelli profondi al rilevamento di oggetti. I modelli R-CNN selezionano prima diverse regioni proposte da un'immagine (ad esempio, i riquadri di ancoraggio sono un tipo di metodo di selezione) e quindi etichettano le loro categorie e riquadri di delimitazione (ad esempio, gli offset). Queste etichette vengono create in base a classi predefinite fornite al programma. Quindi utilizzano una rete neurale convoluzionale per eseguire il calcolo in avanti per estrarre le caratteristiche da ciascuna area proposta.

In R-CNN, l'immagine immessa viene prima divisa in quasi duemila sezioni di regione, quindi viene applicata rispettivamente una rete neurale convoluzionale per ciascuna regione. Viene calcolata la dimensione delle regioni e la regione corretta viene inserita nella rete neurale. Si può dedurre che un metodo dettagliato come quello può produrre vincoli di tempo. Il tempo di addestramento è significativamente maggiore rispetto a YOLO perché classifica e crea riquadri di delimitazione individualmente e una rete neurale viene applicata a una regione alla volta.

Nel 2015 Fast R-CNN è stato sviluppato con l'intenzione di ridurre significativamente il tempo di treno. Mentre la R-CNN originale ha calcolato in modo indipendente le caratteristiche della rete neurale su ciascuna delle duemila regioni di interesse, Fast R-CNN gestisce la rete neurale una volta sull'intera immagine. Questo è molto paragonabile all'architettura di YOLO, ma YOLO rimane un'alternativa più veloce a Fast R-CNN a causa della semplicità del codice.

Alla fine della rete c'è un nuovo metodo noto come Region of Interest (ROI) Pooling, che taglia ogni regione di interesse dal tensore di output della rete, la rimodella e la classifica. Questo rende Fast R-CNN più preciso dell'originale R-CNN. Tuttavia, a causa di questa tecnica di riconoscimento, sono necessari meno input di dati per addestrare i rilevatori Fast R-CNN e R-CNN.

##### Maschera R-CNN

[Mask R-CNN](https://viso-ai.translate.goog/deep-learning/mask-r-cnn/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) è un avanzamento di Fast R-CNN. La differenza tra i due è che Mask R-CNN ha aggiunto un ramo per prevedere una maschera oggetto in parallelo con il ramo esistente per il riconoscimento del riquadro di delimitazione. Mask R-CNN è semplice da addestrare e aggiunge solo un piccolo sovraccarico a Faster R-CNN; può funzionare a 5 fps.

##### YOLO – Guardi solo una volta

Come sistema di rilevamento di oggetti in tempo reale, il rilevamento di oggetti [YOLO](https://viso-ai.translate.goog/deep-learning/yolov3-overview/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) utilizza una singola rete neurale. L'ultima versione di ImageAI v2.1.0 ora supporta l'addestramento di un modello YOLO personalizzato per rilevare qualsiasi tipo e numero di oggetti. Le reti neurali convoluzionali sono istanze di sistemi basati su classificatori in cui il sistema riutilizza classificatori o localizzatori per eseguire il rilevamento e applica il [modello](https://viso-ai.translate.goog/deep-learning/ml-ai-models/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) di rilevamento a un'immagine in più posizioni e scale. Utilizzando questo processo, le regioni dell'immagine con "punteggio elevato" vengono considerate rilevamenti. In poche parole, le regioni che assomigliano di più alle immagini di formazione fornite vengono identificate positivamente.

Come rilevatore a stadio singolo, YOLO esegue la classificazione e la regressione del riquadro di delimitazione in un unico passaggio, rendendolo molto più veloce della maggior parte delle reti neurali convoluzionali. Ad esempio, il rilevamento di oggetti YOLO è oltre 1000 volte [più veloce di R-CNN e 100 volte più veloce di Fast R-CNN](https://translate.google.com/website?sl=en&tl=it&hl=en-US&client=webapp&u=https://pjreddie.com/media/files/papers/YOLOv3.pdf) .

YOLOv3 ottiene una mappa del 57,9% sul set di [dati MS COCO](https://viso-ai.translate.goog/computer-vision/coco-dataset/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) rispetto a DSSD513 del 53,3% e RetinaNet del 61,1%. YOLOv3 utilizza la classificazione multi-etichetta con schemi sovrapposti per l'allenamento. Quindi può essere utilizzato in scenari complessi per il rilevamento di oggetti. A causa delle sue capacità di previsione multi-classe, YOLOv3 può essere utilizzato per la classificazione di piccoli oggetti mentre mostra prestazioni peggiori per il rilevamento di oggetti di grandi o medie dimensioni. Leggi di più su [YOLOv3 qui](https://viso-ai.translate.goog/deep-learning/yolov3-overview/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) .

YOLOv4 è una versione migliorata di YOLOv3. Le principali innovazioni sono il miglioramento dei dati a mosaico, la formazione auto-contradattiva e la normalizzazione incrociata di mini-batch.

##### YOLOR

[YOLOR](https://viso-ai.translate.goog/deep-learning/yolor/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) è un nuovo rilevatore di oggetti introdotto nel 2021. L'algoritmo applica la conoscenza implicita ed esplicita all'addestramento del modello allo stesso tempo. Pertanto, YOLOR può apprendere una rappresentazione generale e completare più attività attraverso questa rappresentazione generale.

La conoscenza implicita è integrata nella conoscenza esplicita attraverso l'allineamento dello spazio del kernel, il perfezionamento della previsione e l'apprendimento multi-task. Grazie a questo metodo, YOLOR ottiene risultati in termini di prestazioni di rilevamento degli oggetti notevolmente migliorati.

Rispetto ad altri metodi di rilevamento degli oggetti sul benchmark del set di dati COCO, la MAP di YOLOR è superiore del 3,8% rispetto a PP-YOLOv2 alla stessa velocità di inferenza. Rispetto a Scaled-YOLOv4, la velocità di inferenza è stata aumentata dell'88%, rendendolo il rilevatore di oggetti in tempo reale più veloce disponibile oggi. Leggi di più nel nostro articolo dedicato su [YOLOR – You Only One Representation](https://viso-ai.translate.goog/deep-learning/yolor/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=en-US&_x_tr_pto=wapp) .

### **Challenges in Object Detection**

In case of image classification we have single output for every image, but here we need to output whole set of detected objects where each image might have many different objects in it. So, we need to build model that can output variably sized number of detections.

1. Multiple types of output. We have 2 different types of output:  
   a. Category Label  
   b. Bounding Box
2. Computational Problem: For object detection it typically requires to work on high resolution images. As we want to identify lot of different objects in image, we want enough spatial resolution on each of the objects so overall resolution of image needs to be quite higher.

**MLOps**

Le operazioni di machine learning, note anche come MLOps o ML Ops, sono un insieme di pratiche che forniscono modelli di machine learning (ML) utilizzando flussi di lavoro ripetitivi ed efficienti. MLOps o ML Ops è un insieme di metodi che mirano a distribuire e mantenere i modelli di machine learning (ML) in produzione in modo efficiente e affidabile. Pertanto, MLOps è essenziale per la gestione del ciclo di vita di servizi e applicazioni di machine learning end-to-end. Cerca di automatizzare e migliorare la qualità dei modelli ML di produzione collegando i requisiti tecnici e aziendali.

Come il modo in cui DevOps è necessario per il ciclo di vita dello sviluppo del software (SDLC), MLOps è essenziale per la distribuzione continua di app basate su ML ad alte prestazioni. Considera le esigenze uniche del ML per definire un nuovo ciclo di vita che funzioni insieme ai processi SDLC e CI/CD, producendo un flusso di lavoro più efficace e risultati più efficienti per il ML.

### **Perché sono necessari MLOps?**

ML mostra il suo vero potenziale una volta che i modelli raggiungono la produzione. In ogni caso, le aziende spesso sottovalutano la complessità e le problematiche relative allo spostamento del ML nella produzione, dedicando la maggior parte delle proprie risorse allo sviluppo del ML considerando il ML uno strumento software standard.

Il risultato? Le aziende non riescono a vedere risultati ottimali dai loro servizi e prodotti di machine learning, con conseguenti perdite di entrate, spreco di risorse e difficoltà a trattenere i dipendenti.

Circa l'87% delle aziende e delle aziende ha difficoltà a gestire lunghe tempistiche di implementazione dei modelli. Inoltre, circa il 64% delle aziende impiega più di un mese per implementare un singolo modello. Questi problemi esistono anche se l'86% delle aziende ha aumentato i propri sforzi e budget per il ML per il 2021. Trova il rapporto pubblicato da DataRobot [qui](https://translate.google.com/website?sl=en&tl=it&hl=en-US&client=webapp&u=https://www.datarobot.com/resources/5-latest-trends-in-enterprise-machine-learning-2021/) .

La verità è che ML non è uno strumento software standard. Ha bisogno di un approccio speciale e il codice è solo un aspetto secondario di ciò che rende un'applicazione di intelligenza artificiale efficiente e di successo. Inoltre, dipende da un insieme di strumenti software a ciclo aperto in continua evoluzione. Una volta implementati i modelli ML, il processo è appena iniziato. I modelli in fase di produzione devono essere continuamente gestiti, monitorati e ridistribuiti in relazione ai segnali di dati in evoluzione per garantire le massime prestazioni.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Descrizione degli oggetti in un immagine secondo un computer**

**Cosa sono i contorni**

I contorni possono essere spiegati semplicemente come una curva che unisce tutti i punti continui (lungo il confine), aventi lo stesso colore o intensità. I contorni sono uno strumento utile per l'analisi della forma e il rilevamento e il riconoscimento di oggetti.

Per una migliore precisione, utilizzare immagini binarie. Quindi, prima di trovare i contorni, applica il rilevamento della soglia o del bordo intelligente. Da OpenCV, \_findContours()\_ non modifica più l'immagine di origine. In OpenCV, trovare i contorni è come trovare un oggetto bianco da uno sfondo nero. Quindi ricorda, l'oggetto da trovare dovrebbe essere bianco e lo sfondo dovrebbe essere nero. Vedi, ci sono tre argomenti nella funzione cv.findContours(), il primo è l'immagine sorgente, il secondo è la modalità di recupero del contorno, il terzo è il metodo di approssimazione del contorno. E restituisce i contorni e la gerarchia. Contours è un elenco Python di tutti i contorni nell'immagine. Ogni singolo contorno è un array Numpy di coordinate (x,y) di punti limite dell'oggetto.

**Come disegnare i contorni?**

Per disegnare i contorni, viene utilizzata la funzione cv.drawContours. Può anche essere usato per disegnare qualsiasi forma a condizione che tu abbia i suoi punti limite. Il suo primo argomento è l'immagine sorgente, il secondo argomento sono i contorni che dovrebbero essere passati come un elenco Python, il terzo argomento è l'indice dei contorni (utile quando si disegna un contorno individuale. Per disegnare tutti i contorni, passa -1) e gli argomenti rimanenti sono colore, spessore ecc.

**Metodo di approssimazione del contorno**

Questo è il terzo argomento nella funzione cv.findContours. Sopra, abbiamo detto che i contorni sono i confini di una forma con la stessa intensità. Memorizza le coordinate (x,y) del confine di una forma.

Se si inserisce cv.CHAIN\_APPROX\_NONE, tutti i punti limite vengono memorizzati.

Ma in realtà abbiamo bisogno di tutti i punti? Ad esempio, hai trovato il contorno di una linea retta. Hai bisogno di tutti i punti sulla linea per rappresentare quella linea? No, abbiamo bisogno solo di due estremi di quella linea. Questo è ciò che fa cv.CHAIN\_APPROX\_SIMPLE. Rimuove tutti i punti ridondanti e comprime il contorno, risparmiando così memoria.

Disegna semplicemente un cerchio su tutte le coordinate nell'array del contorno (disegnato in colore blu). La prima immagine mostra i punti che ho ottenuto con cv.CHAIN\_APPROX\_NONE (734 punti) e la seconda immagine mostra quella con cv.CHAIN\_APPROX\_SIMPLE (solo 4 punti). Vedi, quanta memoria risparmia!!!

**Trova i contorni in un'immagine binaria**

cv.findContours( immagine, modalità, metodo[, contorni[, gerarchia[, offset]]] ) -> immagine, contorni, gerarchia

**Parametri**

**Immagine** Sorgente, un'immagine a canale singolo a 8 bit. I pixel diversi da zero vengono trattati come 1. Zero pixel rimangono 0, quindi l'immagine viene trattata come

binario. È possibile utilizzare compare, inRange, threshold , adaptiveThreshold, Canny e altri per creare un'immagine binaria da un'immagine in scala di grigi o a colori. Se mode è uguale a RETR\_CCOMP o RETR\_FLOODFILL, l'input può anche essere un'immagine intera a 32 bit di etichette (CV\_32SC1).

- **contorni**

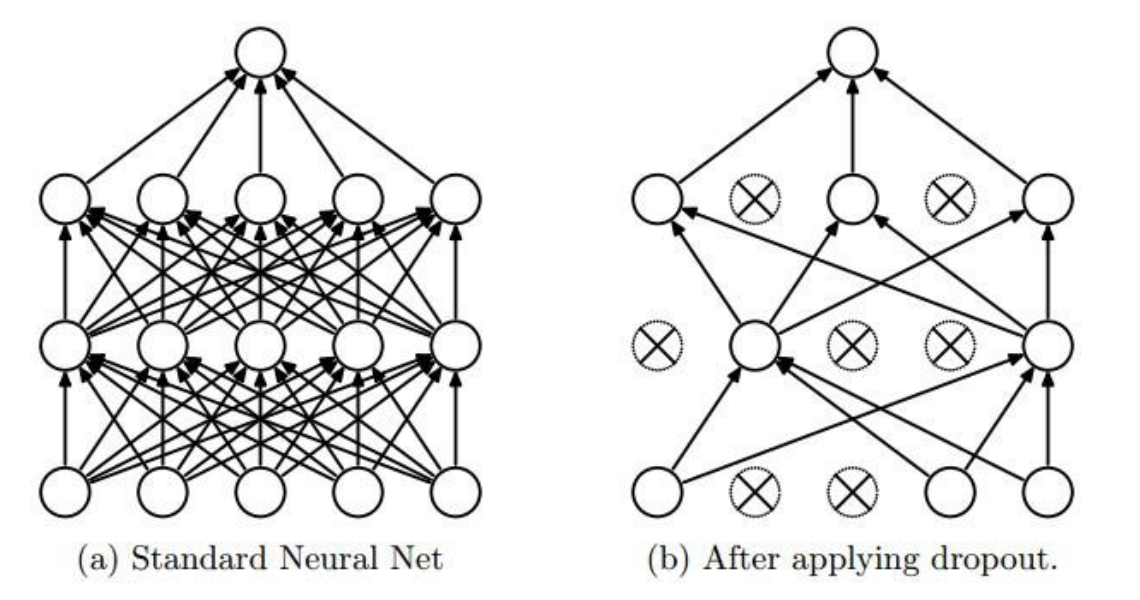
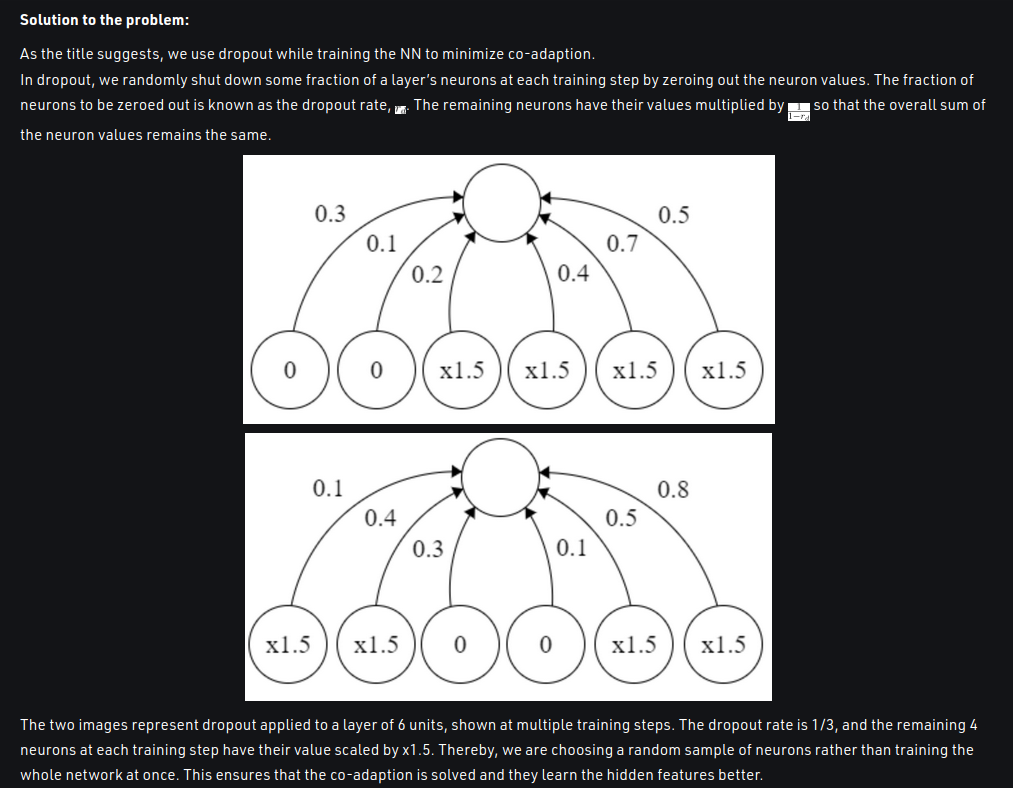
Contorni rilevati. Ogni contorno è memorizzato come vettore di punti.

- **gerarchia** Vettore di output opzionale, contenente informazioni sulla topologia dell'immagine. Ha tanti elementi quanti sono i contorni. Per ogni i-esimo profilo di contorno[i], gli elementi gerarchia[i][0] , gerarchia[i][1] , gerarchia[i][2] e gerarchia[i][3] sono impostati su 0- indici basati nei contorni dei contorni successivi e precedenti allo stesso livello gerarchico, rispettivamente del primo contorno figlio e del contorno padre. Se per il contorno i non ci sono contorni successivi, precedenti, principali o annidati, gli elementi corrispondenti della gerarchia[i] saranno negativi.

In python, la gerarchia è nidificata all'interno di un array di livello superiore. Usa la gerarchia[0][i] per accedere agli elementi gerarchici dell'i-esimo contorno.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Cosa e’ una rete neurale**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

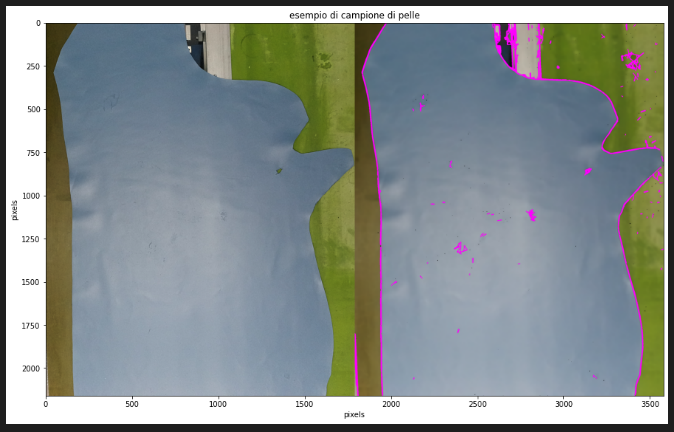
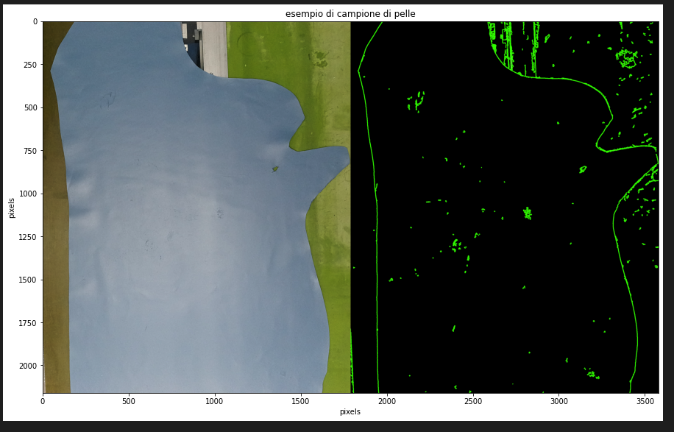
**Workflow dell’applicazione**

L’applicazione è un file scritto in Python denominato *app.py* un processo. Questo file contiene l’intero codice sorgente sistema di computer vision.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Visualizzazione degli errori di produzione**

Fra gli scopi dell’applicazione vi è quello di individuare gli errori nel pezzo di fabbricazione. Questo processo porta a risultati simili ai sottostanti. In entrambe le figure si vede il prodotto di fabbricazione a sinistra ed a destra la stessa immagine rielaborata al fine di evidenziare gli errori di produzione.



Questo sistema avviene attraverso un flusso di funzioni specifiche per rielaborare l’immagine. Molte di queste funzioni appartengono alla libreria open source opencv.

Flusso di lavoro per generare immagini con contorni e imperfezioni disegnate:

1. Caricare l’immagine
2. Trasformare in scala di grigi
3. Sfocare l’immagine
4. Generare una threshold di intensità
5. Utilizzare la funzione *Canny* per il riconoscimento di contorni
6. Utilizzare le funzioni *FindContours* e *DrawContours* per generare l’immagine con i contorni trovati
7. Visualizzare l’immagine

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Object detection**



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Generare un processo in background**

Come trasformare il file *app.py* in un processo in background nel server?

Questo file viene trasformato in un processo in background (ovvero un processo demone) attraverso il software da linea di comando open source chiamato **nohup***.*

Utilizzando questo software è possibile *far girare* continuamente l’applicazione su di un server, senza doversi preoccupare di lasciare aperto il terminale di comunicazione con il server. Attraverso nohup in combinazione con Python è possibile non dover compilare il file prima di restituirlo al server, aumentando così la velocità di sviluppo e manutenzione. Il programma da linea di comando nohupuna volta eseguito, restituisce un valore numerico che equivale al Process ID (PID), ovvero all’id identificativo del processo. Questo PID serve in casi di debugging in quanto permette di arrestare il processo attraverso il programma da linea di comando **kill <PID>***.*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Networking & Sockets**

Un socket, in informatica, è un'astrazione software, gestita dal sistema operativo, che rappresenta un canale di comunicazione di rete tra un processo e una risorsa. Per un programmatore, un socket è un particolare oggetto sul quale leggere e scrivere i dati da trasmettere o ricevere.

Noi useremo i socket per creare connessioni fra il microcontrollore PLC ed il nostro server per la computer vision. I socket vengono generati in rete locale e comunicano attraverso cavo ethernet. Per connettersi ad un macchinario attraverso socket è neccessario il protocollo di comunicazione, l’indirizzo ip e la porta della macchina.

IP : porta =====socket TCP IP ====== IP : porta

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Generare un demone dell’applicazione con nohup**

## [input] nohup python3 app.py %1> logs.out %2 > logs.err

[output] 188929

**Uccidere il processo demone dell’applicazione**

**[input] kill -f 188929**

Questo comando permette di generare un processo in background e generare due file prodotti dal processo. Uno chiamato *logs.out* dove vi sono gli standard outputs del programma (esempio *print)*. L’altro file è chiamato *logs.err* e contiene gli standard errors del processo. Così facendo è possibile visionare i file in maniera divisa fra errori ed output. Così facendo inoltre è possibile visionare in un qualsiasi momento questi file anche quando il processo è attivo. Per motivi di sicurezza sarebbe meglio scaricarli dal server prima di leggerli.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hardware**

Le specifiche hardware per questo sistema sono molto flessibili, in quanto è possibile utilizzare diversi microcontrollori PLC, diversi server computer e diverse videocamere per lo stesso sistema. Il sistema è pensato per essere configurato su di un tipo di hardware, ovvero specifiche videocamere, PLC e tipi di server computer, ma cambiando i parametri per l’acquisizione e l’elaborazione delle immagini è possibile un qualsiasi sistema hardware si voglia.

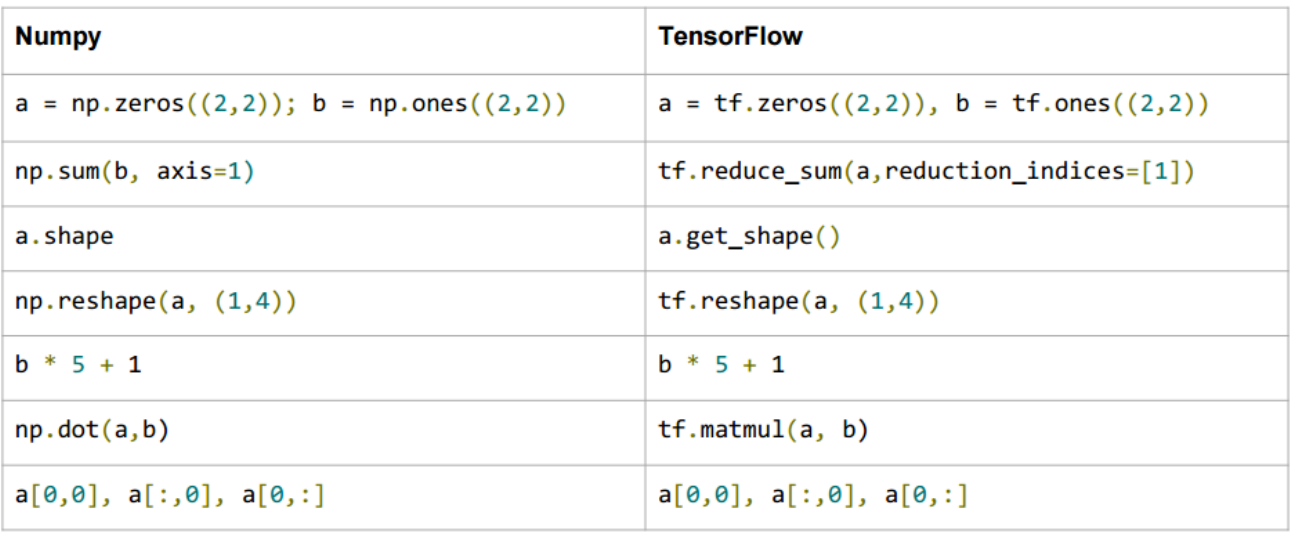
In specifico riporto i nomi tecnici della nostra strumentazione per l’intero sistema di Computer Vision.

* Videocamere
  1. Videocamera stereo a 16 mega pixel
* Computers
  1. Latte Panda

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Software**

* Linguaggi di programmazione
  1. Python 3.7
  2. PLC programmazione a blocchi
* Librerie software
  1. Opencv
  2. Tensorflow
  3. Sockets

****

**CUDA**

**CUDA is Nvidia’s GPGPU computing platform which is made of:**

* hardware (CUDA parallel compute architecture within GPU)
* API (to harness the compute power of Nvidia GPUs)
* libraries (out-of-box scientific computing libraries such as Fourier Transform etc)
* tools (debuggers, profilers, IDEs, etc)

With all these, you can easily accelerate your parallelized algorithms. Most basic part is that you divide work into CUDA threads and they are streamed through CUDA cores.

Each CUDA core is actually a single pipeline in a warp (SIMD) unit which consists of 32 pipelines. Multiple warps make a streaming multiprocessor unit (SM). Multiple SM units make a GPU.

CUDA core is similar to a 1 lane in CPU’s AVX/SSE SIMD hardware. They work together with their neighbor pipelines in same SIMD unit (or warp).

Each CUDA core (pipeline) can calculate integer, floating-point and special functions by getting help from SFU(special function unit) and other compute resources in the SM unit.

Each warp runs its threads lock-step manner so that they eventually re-converge on the branching. When all threads do same operation, warp efficiency is maximum. When every thread does a unique operation at a time, warp efficiency is minimum because it is single-instruction-multiple-thread architecture. Lately Nvidia launched new GPUs with per-thread scheduling to further improve performance on more types algorithms. Per thread scheduling is more freedom than per warp scheduling.

Every CUDA thread or warp is scheduled opportunistically to fill any possible pipeline bubbles(on CUDA pipelines) to improve throughput of pipelines. This is thread-level-parallelism. When there is also data-level-parallelism, it becomes even better at throughput. There is also instruction-level-parallelism effect when other optimizations are saturated. Unrolling loops are one of those ilp optimizations.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problemi tecnici & possibili soluzioni**

riconoscimento del colore

- sensore che legge un multipsretto (rgb altissima risoluzione + infrarosso )

- illuminazione multispettrale ( ) , non si sa se diffussa o diretta

togliere lo sfondo

- procedura di calibrazione iniziale

- neural networks per autofocus

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Esperienze**

Il 3 Giugno 2022 sono state testate le distanze dal rullo per verificare a quale distanza andava messe una o piu’ videocamere OAK con 82’ di apertura su un rullo Barnini da 320 cm. Risoluzione a 3840x2160 pixels.

|  |  |  |  |
| --- | --- | --- | --- |
| **Numero camere** | **Altezza camera** | **Campo rilevabile** | **Gradi camera** |
| 1 videocamera | 265 cm +- 10 cm | 320 +- 10 cm | 82 |
| 2 videocamere | 150 +- 5 cm | 160 +- 10 cm | 82 |
| 3 videocamere | 93 +- 5 cm | 105 +- 10 cm | 82 |

**Esperienza Barnini venerdi 15 Luglio**

videocamera con 120 gradi di apertura

rullo Barnini da 320 cm

Risoluzione a 3840x2160 pixels. 12.3 Mp = 4295 x 2864

|  |  |  |  |
| --- | --- | --- | --- |
| **Numero camere** | **Altezza camera** | **Campo rilevabile** | **Gradi camera** |
| 1 videocamera | 125 cm +- 5 cm | 320 +- 10 cm | 120 |
|  |  |  |  |
|  |  |  |  |

**Data preparation for Object Detection**

resize **(all the same)**

ritaglia tutte le immagini alla piu alta risoluzione possibile per standardizzarle con il traning per le reti neurali

label (with integers)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Costruire una rete neurale**

1 Identifies relevant data sets and prepares them for analysis.

2 Chooses the type of algorithm to use.

3 Builds an analytical model based on the algorithm used.

4 Trains the model on test data sets, revising it as needed.

5 Runs the model to generate test scores.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Architetture di reti neurali per Object detection**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Image Classification vs Object Detection**

Image Classification: Predict the type or class of an object in an image.

Input: An image with a single object, such as a photograph.

Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Detection: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image with one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Architetture di reti neurali**

**R-CNN family in ordine temporale**

1. R-CNN (meno avanzata)
2. Fast R-CNN
3. Faster R-CNN
4. Mask R-CNN (piu avanzata)

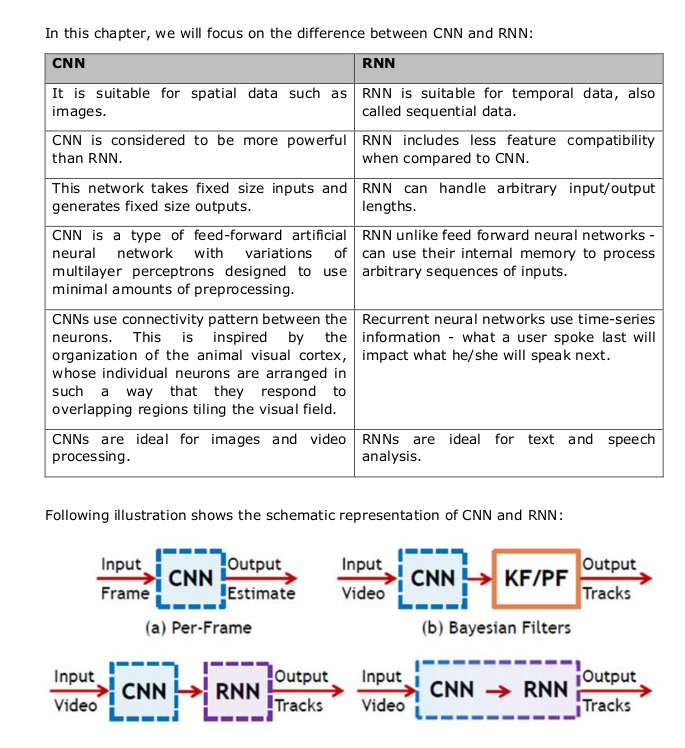
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**RCNN**

Cons of R-CNN:

1. It takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
2. It cannot be implemented real time as it takes around 47 seconds for each test image.
3. The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**fast RCNN**

Step1: Take input image and process whole image with single CNN (without fully connected layers). So the output will be convolutional feature map giving us convolutional features. And this ConvNet we run is often called as backbone network (can be AlexNet, VGG, ResNet, etc.)

Step2: Run region proposal methods and crop & resize features

Step3: Run light CNN (meaning shallow network) per region

**Faster R-CNN**

Faster R-CNN can be generally divided into two parts, RPN part and R-CNN part, each part is an independent neural network and can be trained jointly or separately.

**How build RPN model**

1. RPN model architecture.
2. How to prepare data to train RPN.
3. Loss function.
4. Use trained RPN to predict proposals.

La rete di proposta regionale (RPN) nella rete neurale convoluzionale basata sulla regione più veloce (Faster R-CNN) viene utilizzata per decidere "dove" cercare al fine di ridurre i requisiti di calcolo del processo di inferenza generale.

L'RPN scansiona in modo rapido ed efficiente ogni posizione per valutare se è necessario eseguire ulteriori elaborazioni in una determinata regione. Lo fa emettendo k proposte di riquadro di delimitazione ciascuna con 2 punteggi che rappresentano la probabilità dell'oggetto o meno in ciascuna posizione. I riquadri di ancoraggio sono solo riferimenti, sono selezionati per avere proporzioni e scale diverse in modo da ospitare diversi tipi di oggetti, oggetti allungati come gli autobus, ad esempio, non possono essere rappresentati correttamente da un riquadro di delimitazione quadrato. In Faster R-CNN hanno usato k = 9 che rappresenta 3 scale e 3 proporzioni. **Ciascun regressore nell'RPN calcola solo 4 valori di offset (w, h, x, y) alla casella di ancoraggio** di riferimento corrispondente. dove w = larghezza, h = altezza, (x,y) = centro. L'RPN utilizza una finestra 3x3 che scorre su una mappa di funzionalità di conv di alto livello, la dimensione effettiva di quella piccola finestra è in realtà 177x177 quando viene riproiettata sul livello di input, quindi l'RPN utilizza effettivamente molto contesto durante la presentazione delle proposte. Questa finestra 3x3 viene ricampionata in un vettore dimensionale 256 prima di essere inserita in due livelli completamente connessi, un livello di regressione box (reg), che calcola gli offset del box, e il livello di classificazione box (cls) che calcola i punteggi di confidenza relativi alla probabilità di oggettività. Il livello reg ha uscite 4k mentre il livello cls ha uscite 2k, rendendo l'output RPN totale per posizione a 4k + 2k. Pertanto, in ciascuna posizione del livello di conv, le testine di regressione del riquadro di delimitazione generano gli offset del riquadro di delimitazione per ciascun riquadro di ancoraggio mentre il livello di classificazione restituisce i punteggi di confidenza che rappresentano se un oggetto è presente o meno all'interno di ciascun riquadro di ancoraggio. Solo le scatole con una corrispondente alta probabilità di presenza di oggetti vengono ulteriormente elaborate. Pertanto le proposte finali in ogni posizione sono le scatole di ancoraggio + gli offset delle scatole con un'alta probabilità di contenere un oggetto.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Ottimizzare il modello di rete neurale**

**Transfer Learning**

Let's say we have merely 1,000 images of an object we want to classify. If we take a pre-trained CNN such as ResNet-50 which was trained on millions of images, we can re-train the model on our small dataset and build a state-of-the-art model with minimal effort. In neural networks, this is achieved by removing the final layer in the existing model (called the "loss output" layer) and replacing it with a new layer for the intended prediction.

**Neuroevoluzione**

**Utilizzo**: complex reinforcement learning tasks. In addition, memory is easily represented through recurrent connections in neural networks, making NE a natural choice for learning non-Markovian tasks.

A fully connected network can in principle approximate any continuous function.

**Genetic algorithm challenges**

1. Is there a genetic representation that allows disparate topologies to cross over in a meaningful way?

2. How can topological innovation that needs a few generations to be optimized be protected so that it does not disappear from the population prematurely?

3. How can topologies be minimized throughout evolution without the need for a specially con- trived fitness function that measures complexity?

**cross over**

**mutazione**

spazio di ricerca

Utilizzo: complex reinforcement learning tasks. In addition, memory is easily represented through recurrent connections in neural networks, making NE a natural choice for learning non-Markovian tasks.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**MLOps e Riproducibilita’ del modello**

Determinism

Retraining an existing model with the same hyperparameters and dataset does not always reproduce the same result. This seems counterintuitive -- why is this? There are several factors that contribute to non-determinism:

1. Initialization of layer weights: It is common practice to set initial weights to a non-zero value. These need to be captured to reproduce the same results.
2. Dataset shuffling: Datasets are often randomly shuffled at initialization so this is a fairly obvious root cause of non-determinism. However even if the model is set to use a fixed range of the dataset (e.g. the last 20%), the contents of this set will not be consistent across training runs. Shuffling within the training dataset affects the order of samples and therefore the way the model learns as it iterates over these samples.
3. Randomness in hidden layers: Many neural network-based architectures include layers with deliberate randomness. Dropout is a common example used to prevent overfitting.
4. Updates to ML frameworks, libraries, & drivers: Updates to ML libraries and even GPU drivers can lead to subtly different behavior across iterations.
5. Hardware used during the training process: The specific combination of GPU and CPU can produce different results. This is the result of several factors including the way GPUs handle floating-point calculations and CPUs handle multi-threading.

## Making Machine Learning Reproducible

At a high-level, the first step in making machine learning more reproducible involves capturing all of the core primitives (hyperparameters, code commit, and dataset) and metadata (outlined above) associated with the training process.

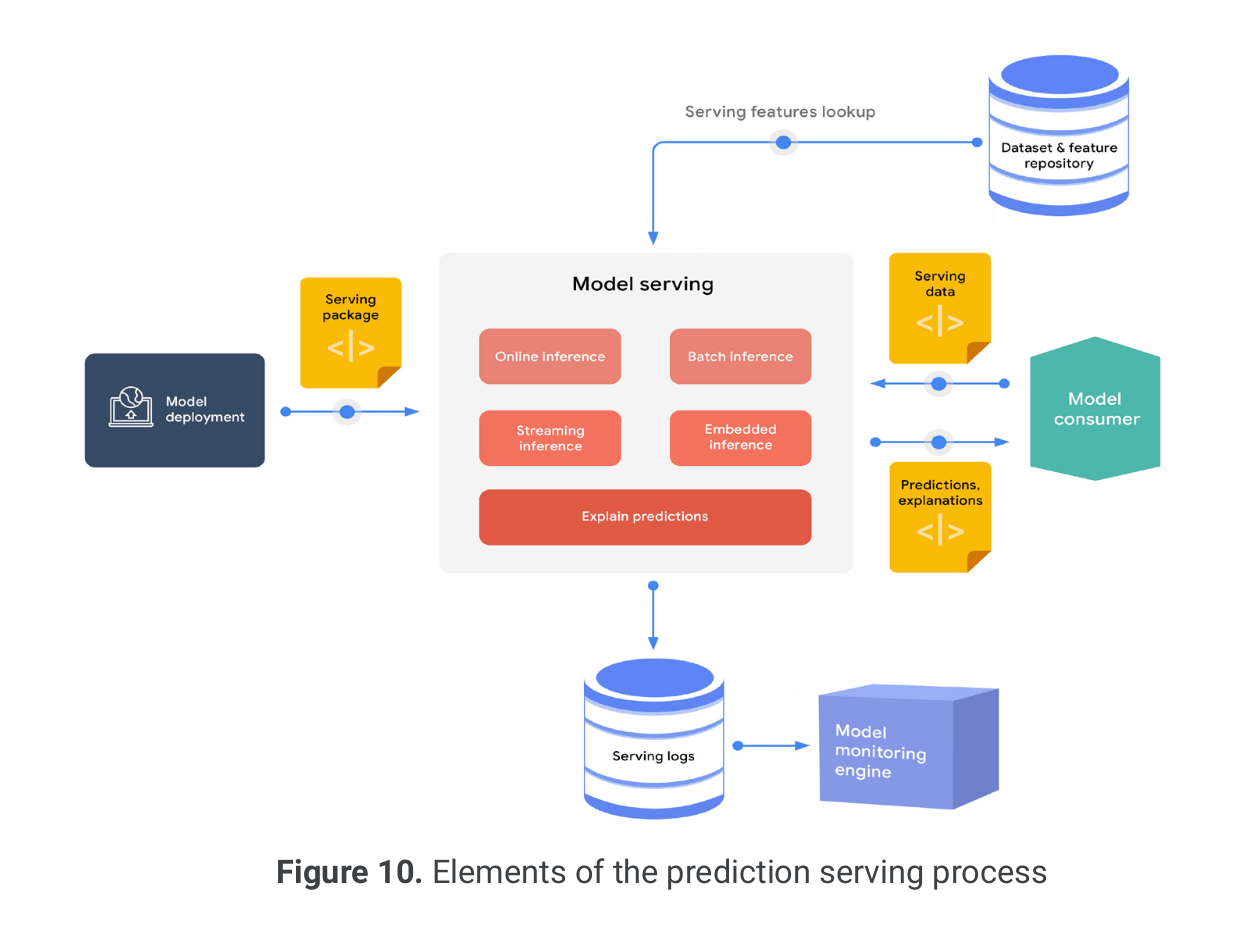
Beyond capturing the basic components in the machine learning system, the path towards reproducible ML can be thought of as a philosophical shift away from ad-hoc methodologies to a more deterministic way of working.

By adopting mature practices found in software engineering and DevOps, machine learning can evolve to achieve improved resiliency and predictability. Versioning and continuous integration are an integral part many software workflows, and transposing these concepts to machine learning will streamline many processes.

Finally, organization-wide visibility and collaboration are also essential. Due to a lack of available tools, data scientists often work in siloed environments and don't have access to shared notebooks, code, and parameters.

**Risultati - Indice**

1. Ritaglio dello sfondo
2. Individuazione degli errori
3. Accuratezza nell’individuazione di errori
4. riflesso della luce provoca cambiamenti di colore sulla pelle

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Risultati tecnici**