Sistemi Distribuiti

Impiego di AI2GO per lo sviluppo di applicazioni basate su Edge Intelligence

Federico Sabbatini

836641

Edge Intelligence ed *Edge Computing.*

Si definisce con il termine *Edge Intelligence* l'impiego di tecniche di *Edge Computing* e di *Machine Learning* all'interno di una rete con buone prestazioni.

Con *Edge Computing* si identifica un paradigma di programmazione distribuito che mira a spostare computazioni e memorizzazione dei dati vicino al luogo in cui sono necessari, decentralizzando tali compiti dal *cloud* ai singoli nodi della rete.

L'esigenza di modificare il modello del *cloud* classico deriva dal fatto che sempre più spesso i dati vengono generati fuori dal *cloud*, ad esempio da sensori, oggetti *smart*, telecamere o misuratori ambientali. Inoltre, l'utilizzo di dispositivi che sfruttano la tecnologia *Internet of Things* porta ad avere grandi moli di dati esterne al *cloud*. Considerando che l'unico modo di processare istantaneamente tali moli di dati consiste nel ricorrere ad un *processing* locale, occorre decentralizzare il *cloud*.

Così facendo i dati vengono analizzati ed elaborati nello stesso luogo in cui vengono generati. Questo si riassume in vantaggi riguardanti quattro diversi aspetti:

- Riduzione della latenza. Si ottengono risultati immediati senza dover inviare dati grezzi ad un nodo centrale o a terze parti, aspettando la risposta;
- Riduzione dei costi. I dati non devono transitare all'interno della rete;
- Maggiore sicurezza. Riduzione del rischio di intercettazione e di violazione dei dati;
- Maggiore autonomia. Possibilità di svolgere localmente le elaborazioni necessarie, con conseguente scelta di azioni immediate da eseguire.

Edge Computing Nodes.

L'unità di elaborazione principale della *Edge Computing* è l'*Edge Computing Node*, un micro *data center* capace di immagazzinare ed elaborare dati e di prendere decisioni locali basate su informazioni avvertite tramite sensori oppure ricevute da altri nodi o basate su algoritmi dei *servers* centrali.

Le decisioni possono essere applicate localmente e/o comunicate ai *servers* centrali per consentire processi di aggregazione e decisione finale.

I vantaggi dell'impiego di *Edge Computing Nodes* si riassumono nel seguente elenco:

- È possibile avviare procedure locali in attesa della risposta globale dei *servers* centrali;
- C'è un unico dispositivo *hardware* da installare, configurare e aggiornare;
- Sono dispositivi flessibili e adattabili in base alle esigenze;
- Riduzione del traffico di rete (vengono inviati ai *servers* centrali solo le informazioni importanti, i dati su cui eseguire backup o la richiesta di elaborazioni complesse);
- *Performance* elevate in termini di latenza della risposta, impego delle risorse di rete e utilizzo della memoria dei dispositivi.

Inoltre, i nodi hanno la possibilità di accumulare dati prima di inviarli al *server* centrale. Le altre informazioni utilizzate per *processing* locale rapido vengono mantenute temporaneamente in memoria e poi eliminate. In caso di connettività intermittente utilizzano algoritmi predittivi e *buffering.* Alcune funzioni rimangono di competenza del *cloud*, ad esempio il *training* degli algoritmi predittivi: solo il *cloud*, infatti, possiede tutte le informazioni necessarie per eseguire al meglio questo processo.

Tra le funzioni principali dei nodi si evidenziano, di conseguenza:

- *Processing* (ad esempio aggregazione, controllare se un valore rientra in un *range* o è inferiore a una certa soglia, etc…);
- *Caching* e r*outing;*
- Prendere decisioni;
- Autoadattarsi ai valori e fare adeguare i nodi adiacenti.

In caso di dati da segnalare (ad esempio valori fuori *range*), si utilizza una rete *mesh* per diramare il *report.*

Esempi di applicazioni che utilizzano *Edge Computing*.

Un esempio applicativo in cui risulta utile l'impiego di *Edge Computing* potrebbe essere il monitoraggio del traffico. Infatti, in una applicazione di questo genere, inviare i dati a un

server centralizzato comporterebbe un ritardo nella ricezione della risposta tale per cui questa arriverebbe in un momento in cui la situazione del traffico è cambiata, pertanto risulterebbe inutile.

Un altro esempio potrebbe riguardare l'ambito della videosorveglianza. Una videocamera produce un flusso continuo di dati che mette in seria difficoltà sia le capacità della rete che quelle del *cloud*, infatti occorre una analisi *real time* per far fronte ad eventuali situazioni pericolose che richiedono un intervento tempestivo.

Inoltre esistono applicazioni *time sensitive*, in cui il fattore tempo è determinante. Si pensi, ad esempio, alle transazioni bancarie. Occorre una analisi in tempo reale per annullare tempestivamente transazioni malevole o acquisti con carte di credito rubate (ad esempio transazioni multiple dallo stesso POS).

La *Edge Computing* risulta utile anche nel campo delle telecomunicazioni: duplicare dati su *edge servers* regionali permette di abbattere i problemi di latenza per gli utenti finali, che non devono attendere il transito di richiesta e risposta fino al *server* centrale.

Infine si possono citare i veicoli a guida autonoma. Si pensi a un veicolo che fa affidamento unicamente sul *cloud.* Cosa succede se il veicolo perde la connessione? Oppure mentre attende l'elaborazione dei dati? Ad esempio, un sensore rileva la presenza di un ostacolo a poca distanza dalla macchina, come un pedone che all'improvviso attraversa la strada. Serve un'elaborazione *real time* che un sistema *cloud* classico non è in grado di fornire.

AI2GO.

AI2GO è una piattaforma che agevola la programmazione basata su *Edge Computing* fornendo diversi modelli *pre-tuned* e permettendo di eseguirli su dispositivi anche senza connessione internet. I modelli forniti sono energeticamente efficienti, hanno un basso impatto sulla memoria ed essendo eseguiti localmente riducono la latenza nella produzione dei risultati e al contempo aumentano il livello di sicurezza.

AI2GO è uno strumento sviluppato dall'organizzazione Xnor.ai, che si occupa principalmente di applicazioni nel campo della *Edge Intelligence*. Più precisamente, si concentra sull'esecuzione di complessi algoritmi di *deep learning* su una vasta gamma di dispositivi, come *smartphones*, droni o altri oggetti dotati di tecnologia IoT. Si tratta di algoritmi altrimenti riservati al *cloud*. L'obiettivo di Xnor.ai è rendere l'intelligenza

artificiale disponibile per tutti, anche in termini di usabilità senza particolari competenze nel settore.

AI2GO, infatti, permette anche a utenti con pochissima esperienza di creare applicazioni basate su intelligenza artificiale ed *Edge Intelligence*. È sufficiente scaricare come prima cosa l'SDK di Xnor, che include quattro *Bundles* funzionanti. Successivamente si passa al download del *Bundle* Xnor più opportuno: l'utente deve scegliere la piattaforma *hardware*, il caso d'uso e i parametri di latenza e impronta in memoria che preferisce (tra quelli disponibili). Sul sito ufficiale della piattaforma è presente un *tutorial* che aiuta gli utenti nell'installazione dei pacchetti necessari al corretto funzionamento del sistema. Il *tutorial*, inoltre, mostra come rendere operativo un *Bundle* e come passare da un *Bundle* all'altro. Infine, spiega come utilizzare i *Bundles* tramite applicazioni scritte in diversi linguaggi di programmazione, ad esempio C o Python.

Bundle Xnor.

Un *Bundle* è un modulo che contiene un modello AI e un motore di inferenza che consente al modello di essere eseguito in un dispositivo con poche righe di codice. I modelli inclusi nei *Bundles* sono stati pre-addestrati, mentre il motore di inferenza è ottimizzato per *Edge Computing*, cioè è più veloce e accurato nelle predizioni *real time*. Attualmente i modelli non possono essere modificati, né modelli multipli possono essere combinati in una stessa applicazione.

I *Bundles* sono disponibili per molte delle principali piattaforme *hardware*, ma non per *Windows* né iOS. È possibile interagire con essi tramite una vasta gamma di differenti linguaggi di programmazione.

I *Bundles* sono divisi in cinque settori principali (media e intrattenimento, automobilistico, fotografia e video, commerciale oppure *smart home*) e coprono i seguenti casi d'uso:

- Rilevazione e classificazione di oggetti da cucina, oggetti di casa, oggetti da esterno, animali domestici, oggetti di realtà virtuale o aumentata, oggetti da ufficio, oggetti dell'abitacolo delle auto;
- Classificazione di cibo, espressioni facciali, sport, azioni, marchi di auto;
- Rilevazione di volti, persone, oggetti sportivi, animali e segmentazione di persone.

SDK.

L'SDK scaricabile dal sito ufficiale di AI2GO mette a disposizione quattro *Bundles* predefiniti e alcuni *files* Python e C per interagire con essi. Uno dei programmi Python analizza un'immagine e la restituisce con disegnati dei riquadri rossi attorno agli oggetti di interesse e funziona con *Bundles* di tipo *detector*. Un altro analizza una cartella e copia o sposta le immagini in un'altra cartella, raggruppandole in sottocartelle in base alla classificazione delle immagini e funziona con *Bundles* classificatori o *detector*. Personalizzando questi *files* si possono ottenere comportamenti più complessi.

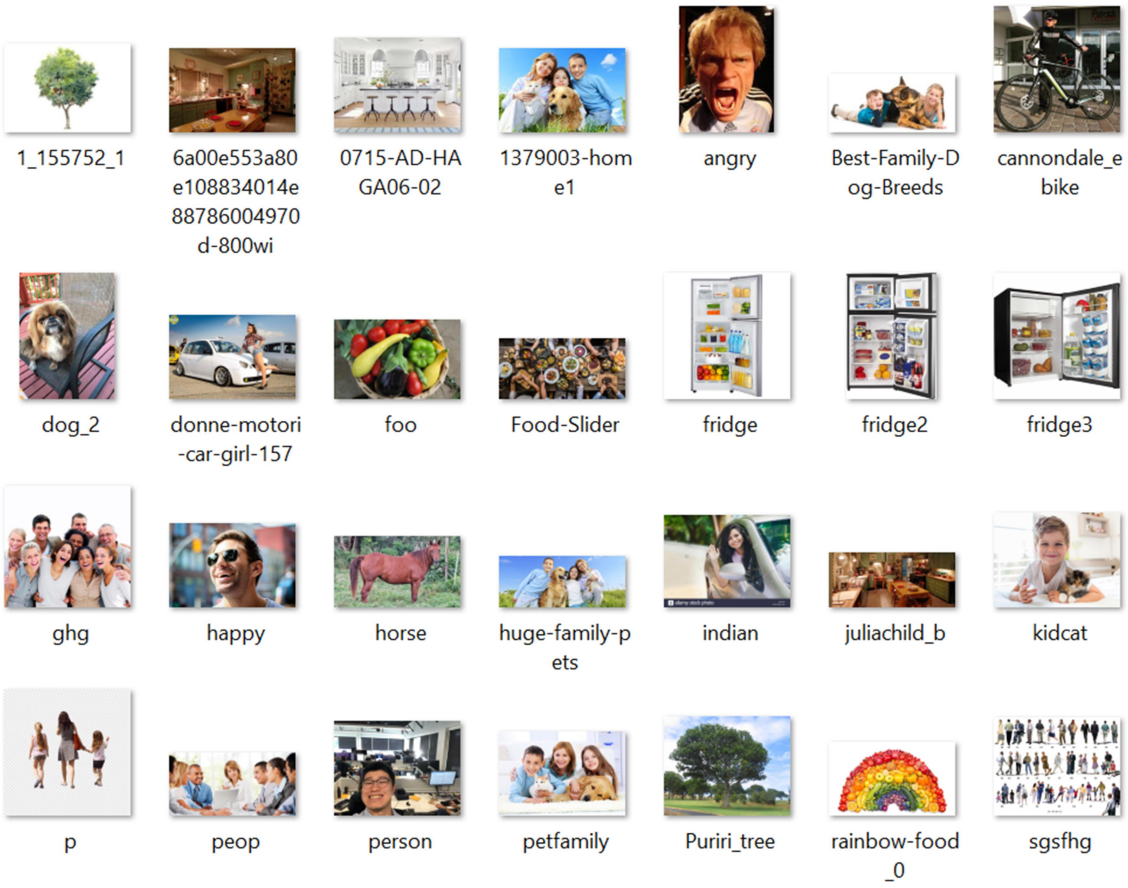
Applicazioni basate su AI2GO.

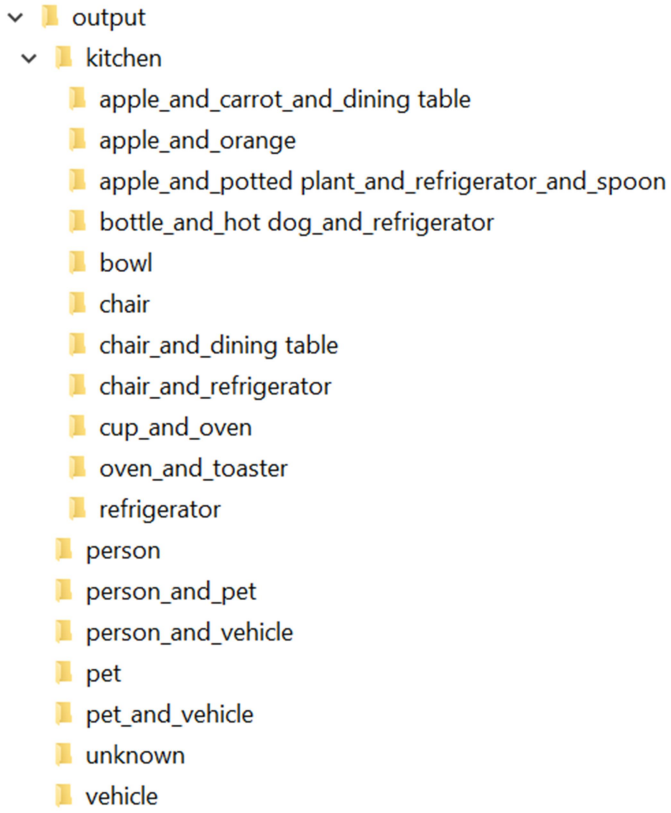
Di seguito vengono proposte diverse applicazioni sviluppate tramite gli strumenti messi a disposizione da AI2GO. Il linguaggio utilizzato per l'implementazione è Python, versione 3, e le applicazioni sono state testate su una macchina virtuale che simula il sistema operativo Debian.

Classificazione di fotografie.

Utilizzando i *Bundles* non è difficile costruire una applicazione che, presa in *input* una cartella contenente *files* in formato jpg (ad esempio fotografie), restituisca in *output* un gruppo di sottocartelle in cui ognuna contiene una diversa categoria di immagini. Ad esempio, avviando il *Bundle "person pet vehicle detector"* verranno create sottocartelle relative a persone, animali domestici, veicoli e combinazioni di essi (persone e animali, animali e veicoli, etc…). Inoltre verrà creata una sottocartella che raccoglierà tutte le immagini che non hanno trovato un riscontro. Cambiando il *Bundle* attivo, è possibile effettuare una classificazione ulteriore delle immagini, ad esempio andando ad analizzare tutte quelle inserite nella cartella "*unknown*". L'applicazione sviluppata fa esattamente questo: prima utilizza il *Bundle "person pet vehicle detector"* sulla cartella di *input,* poi il *Bundle "kitchen object detector"* sulla cartella di *output* "*unknown*".

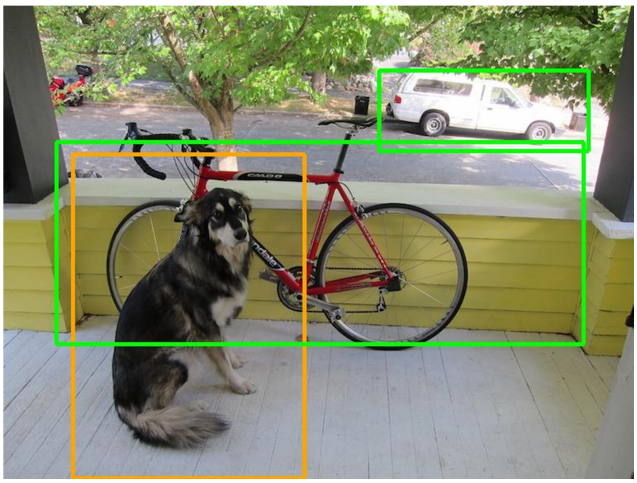
Una porzione dell'insieme delle immagini di partenza e il risultato dell'elaborazione sono mostrati nelle immagini seguenti.

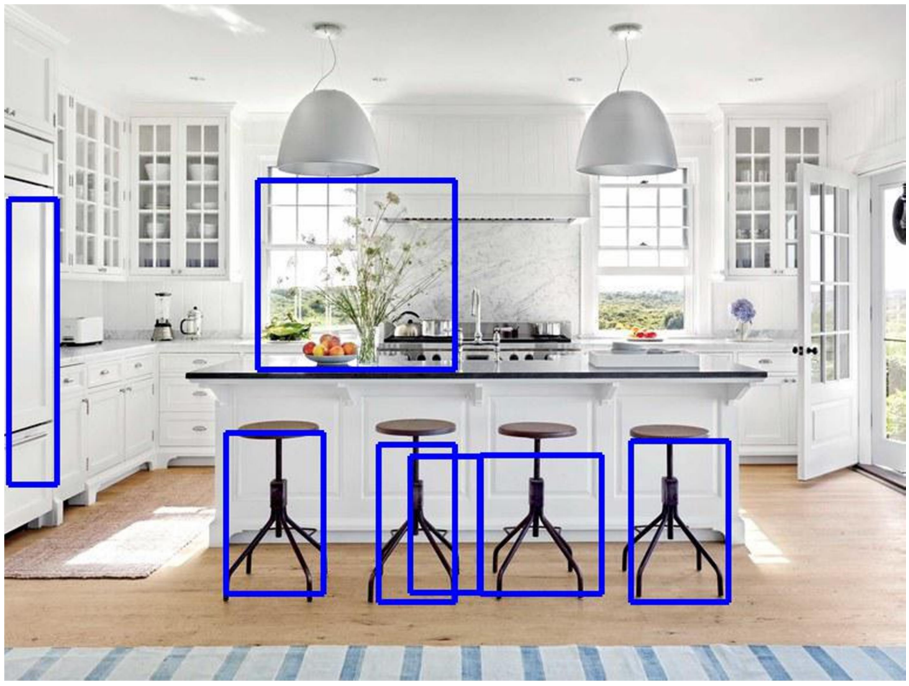




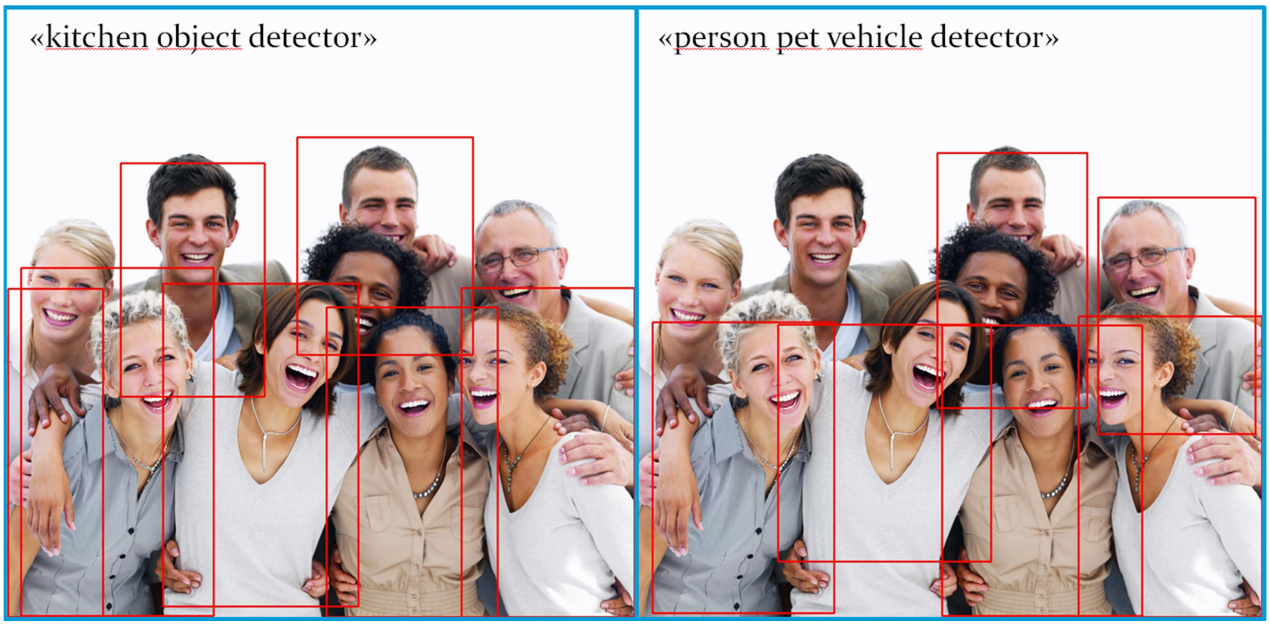
Evidenziare oggetti di interesse.

I *Bundles* utilizzati nell'applicazione precedente possono essere utilizzati anche per un altro scopo, cioè tracciare dei riquadri attorno a oggetti di interesse raffigurati nelle immagini. Nell'SDK è presente un *file* che disegna riquadri rossi su qualsiasi oggetto rilevato, ma è facile modificarlo per far sì che ogni categoria di oggetto abbia un colore differente. L'applicazione sviluppata utilizza il colore rosso per le persone, l'arancione per gli animali domestici, il verde per i veicoli e il blu per gli altri oggetti rilevati dal *Bundle "kitchen object detector"*. L'insieme di *input* è lo stesso dell'applicazione precedente (e sarà il medesimo anche nelle prossime applicazioni). Il risultato è rappresentato nelle immagini seguenti.





Si noti che entrambi i *Bundles* sono in grado di riconoscere le persone, tuttavia mostrano due comportamenti distinti, come evidenziato nelle immagini seguenti.



Il *Bundle "kitchen object detector"*, infatti, sembra rilevare più persone del *Bundle "person pet vehicle detector"*. In realtà, osservando i risultati di entrambi su una quantità maggiore di immagini, si nota che il primo funziona meglio in immagini affollate, mentre il secondo funziona meglio nella rilevazione di persone non in primo piano, fuori fuoco o non completamente visibili.

Combinazione delle precedenti.

È possibile combinare le due applicazioni precedentemente descritte in una nuova applicazione che divide le immagini in sottocartelle ma, invece di spostare le immagini di *input*, provvede a copiare tali immagini e ad aggiungere riquadri colorati attorno agli oggetti di interesse rilevati dai *Bundles*.

L'algoritmo è il seguente: le immagini di input vengono analizzare con il *Bundle "person pet vehicle detector"* e vengono tracciati i riquadri di conseguenza. Le immagini modificate vengono suddivise nelle varie sottocartelle e quelle classificate come *"unknown"* vengono rielaborate utilizzando il *Bundle "kitchen object detector"*. Viene eseguita la stessa coppia di operazioni, cioè tracciamento dei riquadri attorno agli oggetti interessanti e classificazione in sottocartelle. Così facendo si evita il riquadro doppio attorno alle persone dovuto all'utilizzo dei due *Bundles* in sequenza.

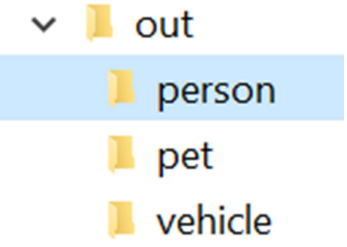
Cosa comprare al supermercato?

Una applicazione in ambito domestico potrebbe essere la seguente: analizzare fotografie dell'interno di frigoriferi per capire cosa è stato acquistato e, di conseguenza, cosa manca e andrebbe comperato. Il problema è dato dai pochi cibi riconosciuti dal *Bundle "kitchen object detector"* e dalla scarsa accuratezza dello stesso. Infatti, è in grado di riconoscere solo mele, arance, banane, broccoli, carote, pizza, *hot dog* e poco altro, e inoltre ha una buona percentuale di fallimento, come evidenziato nelle immagini seguenti.

Estrazione di oggetti.

Sempre utilizzando gli stessi *Bundles* è possibile realizzare una applicazione che estrae dalle fotografie gli oggetti rilevati, salvandoli come nuove immagini in sottocartelle diverse in base alla categoria dell'oggetto.

Nel caso specifico, è stato utilizzato solo il *Bundle "person pet vehicle detector"*. Le immagini seguenti mostrano il risultato dell'applicazione, nell'ordine: suddivisione in sottocartelle e contenuto delle sottocartelle relative a persone, animali domestici e veicoli.




1572653302.95547


1572653302.95935


1572653303.40628


1572653303.62147


1572653303.62518


1572653304.16348


1572653304.41763


1572653305.43419


1572653305.68501


1572653307.95791


1572653307.96589


1572653307.97382


1572653307.98294


1572653307.98896

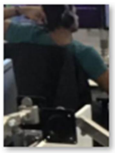

1572653309.45618


1572653310.66113


1572653310.66255


1572653310.66366


1572653310.90405


1572653310.91622


1572653310.91740


1572653311.36665


1572653311.36712


1572653311.36766


1572653312.08604


1572653312.08688


1572653312.08797


1572653312.08916


1572653303.89828


1572653304.41134


1572653305.19541


1572653305.19627


1572653302.95764


1572653303.62280


1572653304.65343


1572653304.93531


1572653305.42670


1572653305.43452


1572653305.43503

1572653307.45870

1572653305.19250

1572653307.69971

1572653308.49619

1572653309.46570

1572653307.46295

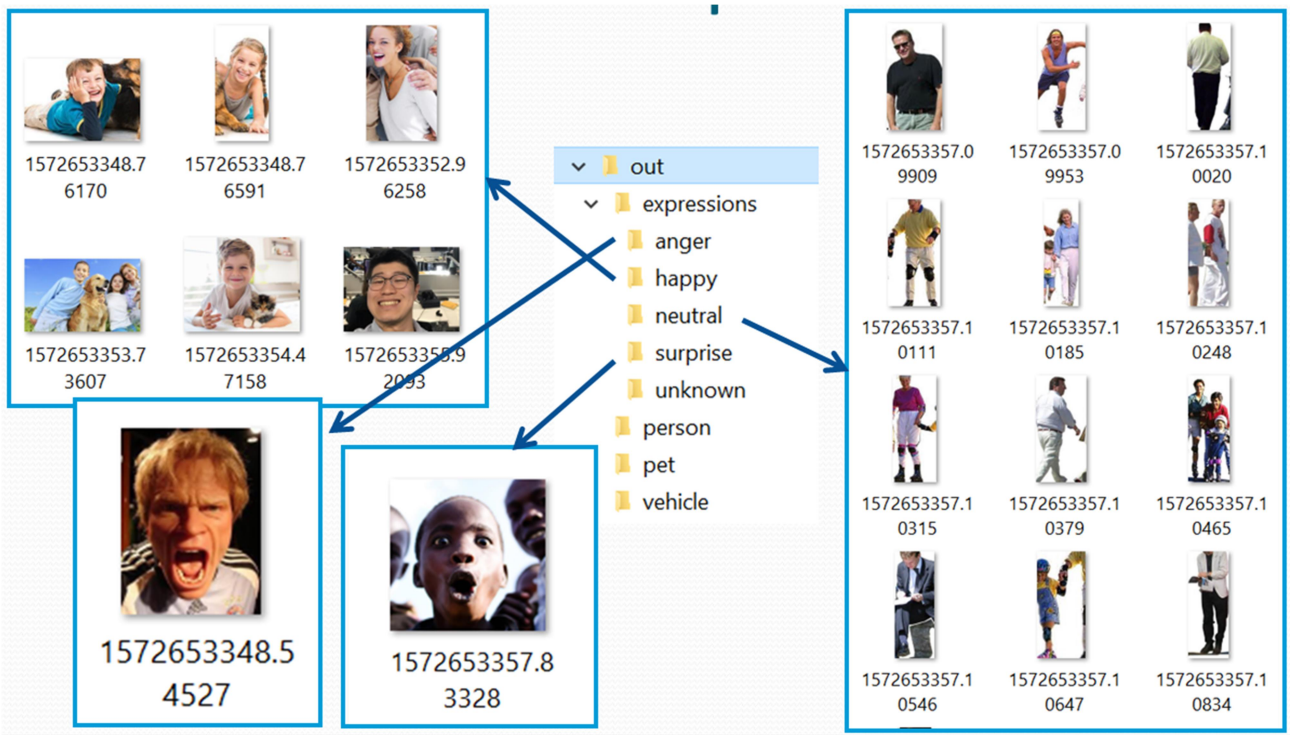
1572653307.46674

1572653309.94261

1572653313.32837

Classificazione delle fotografie in base alle espressioni.

L'applicazione appena descritta può essere estesa effettuando una elaborazione successiva delle immagini salvate nella sottocartella relativa alle persone. Infatti è possibile analizzare le espressioni delle persone raffigurate per effettuare una ulteriore classificazione delle fotografie. Questo è possibile utilizzando un *Bundle* differente, anch'esso incluso nell'SDK. Si tratta del *Bundle "facial expression classifier"*, in grado di distinguere tra espressioni felici, tristi, arrabbiate, sorprese, neutre e non solo. L'immagine seguente mostra il risultato dell'applicazione.



Accuratezza dei *Bundles*.

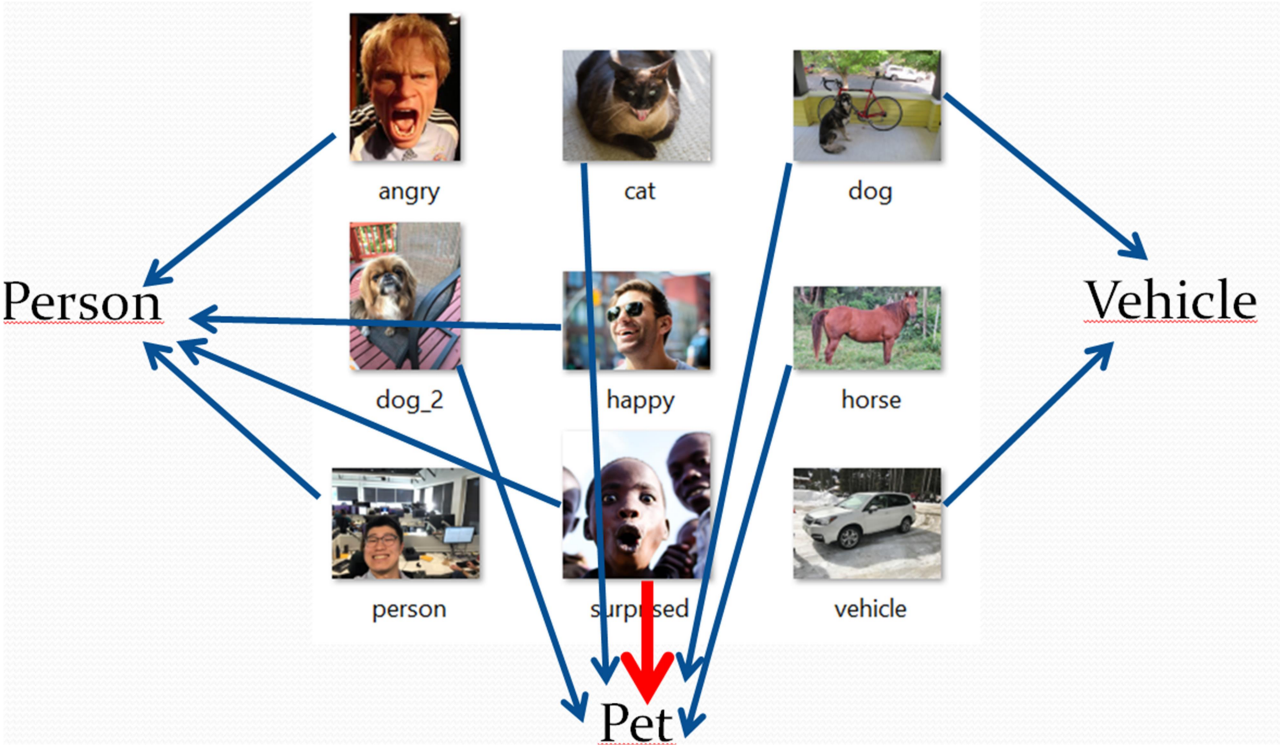
A dispetto di quanto affermato nel sito ufficiale di AI2GO, le semplicissime applicazioni appena descritte evidenziano diverse criticità dei *Bundles*. Si riportano di seguito due esempi abbastanza rappresentativi di tali criticità. Va considerato che questi esempi utilizzano immagini di *test* e *Bundles* inclusi nell'SDK scaricabile dal sito ufficiale di AI2GO, circostanza che rende i problemi ancora più gravi, in quanto denota, per quanto possa

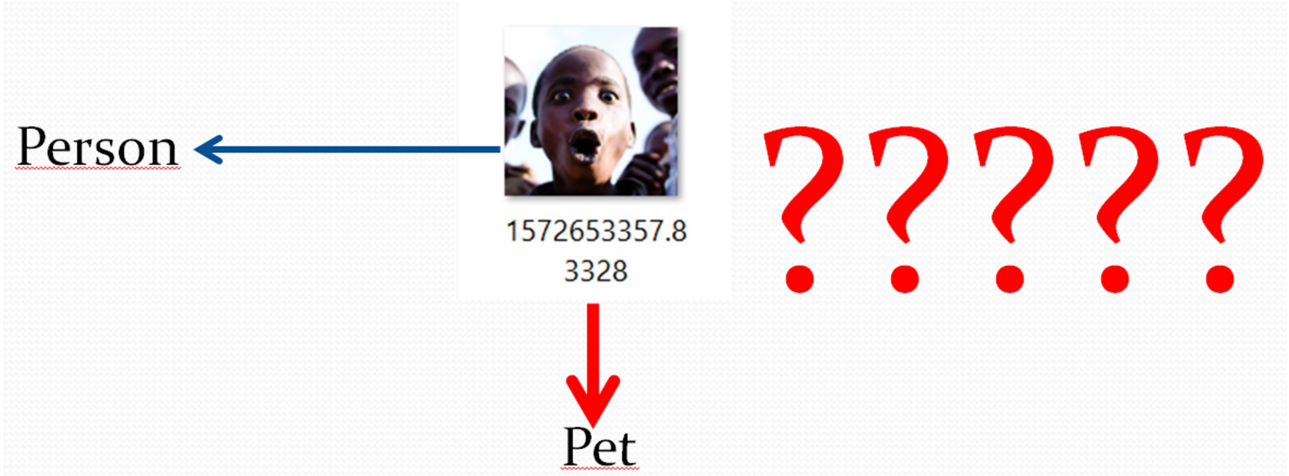
sembrare assurdo in relazione a uno strumento così complesso, un *testing* molto superficiale delle *performance* finali.

Il primo esempio riguarda la classificazione di persone e animali domestici. Infatti, tra le poche immagini di *test* è presente il volto di un bambino di colore che viene classificato in entrambe le categorie.

Il secondo esempio riguarda la classificazione di espressioni. Tra le immagini di *test* è presente l'immagine di un ragazzo felice (e anche il nome del file indica questa classificazione come corretta) la cui espressione viene però riconosciuta come neutra.

Di seguito le immagini relative ai due esempi.





Conclusioni.

L'impiego di AI2GO per lo sviluppo di applicazioni basate su *Edge Computing* presenta notevoli vantaggi ma anche degli svantaggi. Nel dettaglio, i vantaggi sono i seguenti:

- Possibilità di utilizzare *Edge Intelligence* in applicazioni di impiego quotidiano anche senza conoscere tecniche di *Machine Learning*: è sufficiente conoscere un linguaggio di programmazione e come farlo interagire con i *Bundles;*
- Possibilità di utilizzare più *Bundles* in sequenza per eseguire elaborazioni più complesse o accurate;
- *Bundles* eseguibili su quasi tutte le piattaforme *hardware:* basta avere un pc o uno *smartphone*, anche *offline;*
- Velocità di esecuzione.

Tuttavia, ci sono questi svantaggi da tenere in considerazione:

- Impossibilità di modificare i *Bundles*: non si possono estendere (ad esempio, aggiungere una classe a un classificatore oppure eseguire un addestramento con un *training set* personalizzato), né si possono combinare tra loro;

- Impossibilità di eseguire contemporaneamente più *Bundles* nella stessa applicazione:
- Accuratezza dei *Bundles*: non sempre si ottengono i risultati attesi, neanche applicando i *Bundles* di *default* presenti nell'SDK alle immagini di *test* dello stesso SDK;
- Completezza dei casi d'uso: non sempre l'utente trova un caso d'uso adatto alle proprie esigenze e/o aspettative e anche se l'utente trova un caso d'uso adatto, non necessariamente il *Bundle* scaricato è utile (ad esempio, il "*kitchen object detector*" riconosce una scelta così limitata di cibi da renderlo inutile in molte applicazioni.