Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---

Tesi di Laurea Magistrale

Sviluppo di una rete neurale per il riconoscimento automatico di prodotti in ambito *retail*

Development of a neural network for automatic product recognition in the retail environment

Relatore

Prof. Emanuele Frontoni

Candidato

Alessio Gatto

Correlatore

Rocco Pietrini, *PhD*

Sessione di Laurea Ottobre 2021

Anno Accademico 2020/2021

Sommario

Questa tesi intende descrivere e implementare parte di una *pipeline* atta a riconoscere i prodotti presenti su uno scaffale, a livello di singolo EAN, a partire dalla foto tale scaffale; essa è composta da una prima rete neurale che si occupa di effettuare la *detection* dei singoli prodotti presenti nello scaffale (la rete è stata realizzata da [Goldman et al.][1] ed è stata da loro allenata con il *dataset* **SKU110K**) e una seconda rete, progettata e realizzata in collaborazione con Grottini Lab Srl e illustrata in questa tesi, che ha il compito di associare alla singola immagine realizzata dalla prima rete, un vettore di *embedding*, che ne descrive le *feature* distintive. Mediante l'ottenimento di questo vettore dell'immagine di *input*, è possibile misurare la distanza, tramite la *cosine similarity*, tra questo vettore e tutti i vettori di *embedding* presenti nel *dataset* adibito al confronto. Il vettore con la *cosine similarity* più alta è associato ad un'immagine dotata di *label* (EAN) e, dunque, questo EAN sarà quello dell'immagine di *input*. Visto il particolare compito, non esistono attualmente *dataset* in grado di soddisfare i nostri requisiti poiché non presentano un livello di dettaglio così granulare (l'EAN) e, pertanto, è stato necessario realizzarne interamente uno, in collaborazione con l'azienda Grottini Lab e con il VRAI - Vision, Robotics and Artificial Intelligence del dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche. Nella seguente tesi sono descritte, inoltre, le difficoltà incontrate in tutto il processo e i risultati ottenuti, con un occhio di riguardo ad ipotetici sviluppi commerciali.

"By far the greatest danger of Artificial Intelligence is that people conclude too early that they understand it."
Eliezer Yudkowsky

Indice

Elenco delle figure

Capitolo 1

Introduzione al riconoscimento di prodotti nel mondo *retail*

In ambito *retail*, il riconoscimento automatico di prodotti riveste un ruolo fondamentale per migliorare la loro gestione da parte di venditori e produttori, con conseguente miglioramento e soddisfacimento dell'esperienza di acquisto dei clienti.

Ad esempio, grandi colossi del mondo *retail* stanno implementando metodi per automatizzare azioni quotidiane e garantire la miglior esperienza di acquisto possibile. Si citano soluzioni pioneristiche come Amazon Go[1] e Walmart Intelligent Retail Lab[2] che, tramite algoritmi di intelligenza artificiale, *computer vision* e con l'aiuto di sensoristica varia, hanno realizzato punti vendita completamente automatici: in particolare, Amazon Go permette di effettuare la spesa prendendo i prodotti negli scaffali e pagando, all'uscita, eliminando la necessità di dover posizionare i prodotti nelle casse poiché i sistemi intelligenti sparsi nel punto vendita hanno associato al corrispettivo cliente i giusti prodotti presi dagli scaffali.

In questa tesi, si tratterà il riconoscimento dei prodotti non per automatizzare gli acquisti ma per migliorare la gestione del planogramma (o la sua rilevazione automatica), che rappresenta la disposizione dei prodotti in uno scaffale, posizionati all'interno di un punto vendita al fine di aumentarne le *performance* e, conseguentemente, gli acquisti da parte dei clienti.

Solitamente, i planogrammi sono progettati direttamente dalla direzione centrale del *retailer* e vengono inviati ai singoli punti vendita e sono realizzati tenendo conto di diversi fattori come lo spazio disponibile nello scaffale e i requisiti di rifornimento. Infatti, un planogramma può essere progettato, ad esempio, affinché un certo prodotto sia il meno possibile *out-of-stock* al fine di ridurre al minimo i mancati incassi. Purtroppo, però, non sempre i planogrammi sono rispettati dai singoli punti vendita

[1] `https://www.amazon.com/b?node=16008589011`
[2] `https://www.intelligentretaillab.com/`

Household & Beauty Shelf Display

Item	Product
1.	Dish Soap with Bleach
2.	Dish Soap -
3.	Dish Soap - Fruit
4.	Laundry Detergent
5.	Laundry Detergent
6.	Shampoo
7.	Shower Cleaner
8.	Window Cleaner
9.	Easy Scrub
10.	Fine Cleaner
11.	Square Tissue
12.	2 per soda
13.	2 per soda - Diet
14.	Baby wipes
15.	Diapers - Travel Pack
16.	Diapers
17.	Face Wash
18.	Sunscreen
19.	Body Lotion
20.	Toothpaste
21.	Hand Soap
22.	Talcum Powder
23.	Baby Wash
24.	Tolet Paper
25.	Diapers

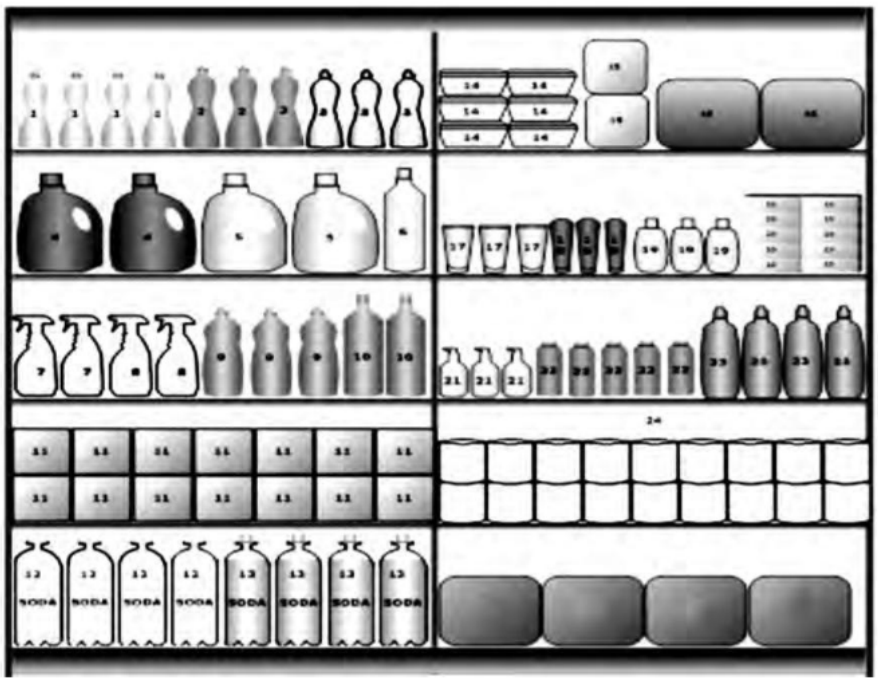


Figura 1.1: Esempio di planogramma di uno scaffale della sezione bellezza. Fonte: [11]

e, per questo motivo, il monitoraggio della conformità dei planogrammi rappresenta una sfida per i negozianti: essi si devono assicurare che gli scaffali dei punti vendita siano conformi ai planogrammi realizzati dalla sede centrale. A volte, non vengono rispettati poiché, ad esempio, i consumatori locali hanno delle esigenze diverse oppure i commessi non si attengono alle direttive. Non seguire le direttive porta ad una bassa conformità del planogramma e alla conseguente incongruenza dei dati eventualmente raccolti in un dato scaffale poiché non si ha più conoscenza della corretta posizione dei prodotti. Infatti, se certi dati si servono del planogramma (ad esempio, analisi sul tempo passato dai clienti davanti un certo scaffale) essi non saranno più utilizzabili e, per questo motivo, avere scaffali non conformi al planogrammi non è accettabile per un negozio.

L'azienda Grottini Lab Srl[3], con la quale ho collaborato per la realizzazione del tirocinio descritto nella seguente tesi, si è posta l'obiettivo di semplificare notevolmente la gestione dei planogrammi nei punti vendita. L'idea è di identificare dalla foto dello scaffale cosa è presente e in che posizione precisa; se si ha un planogramma *master* (che può essere una foto oppure un planogramma testuale fornito dalla catena) viene realizzato il confronto per valutare la conformità, altrimenti si usa tale foto solamente per rilevare i prodotti, ignorando la verifica della conformità con il planogramma. Attualmente, questa attività viene fatta quasi del tutto in maniera manuale, invian-

[3] `https://www.grottinilab.com`

do periodicamente del personale nei punti vendita e, per questo, l'automatizzazione di tutto il processo consentirebbe una velocizzazione del processo, con conseguente riduzione di costi ed errori.

Al momento, l'app fa uso di algoritmi di verifica della similarità di immagini con algoritmi di *Computer Vision* come SIFT [2]; tuttavia, SIFT presenta numerose criticità quali le scarse performance sia da un punto di vista computazionale che di qualità del risultato. Per questo motivo, è nata la necessità di realizzare un nuovo modo di verificare la similarità delle immagini e si è scelto di adottare approcci di *Deep Learning*, un campo di ricerca che fa parte della famiglia del *Machine Learning* che cerca di risolvere dei problemi estrapolando *feature*, in maniera automatica, che ha imparato a riconoscere tramite degli allenamenti. In fase di inferenza, userà questa capacità per svolgere il compito per la quale è stata implementata.

Inoltre, la verifica manuale di ogni singolo prodotto presente in uno scaffale è un'operazione estremamente *time consuming* poiché i codici a barre sono stampati in posizioni differenti (tra *packaging* di diversi prodotti) ed è quindi richiesto del tempo, non trascurabile per grandi numeri, per la loro ricerca e acquisizione. Alcune catene posizionano dei codici a barre nel cartellino dei prezzi ma essi possono riferirsi a codici interni invece che all'EAN e alcune volte capita che i prodotti non sono posizionati correttamente e i cartellini dei prezzi potrebbero riferirsi a prodotti diversi. Infine, delle catene hanno adottato *tag RFID*[4] nelle etichette dei prodotti ma risultano essere costosi e le onde radio possono essere bloccate da altri oggetti o interferire tra di loro [15].

Il lavoro descritto in questa tesi ha l'obiettivo, quindi, di presentare un approccio capace di riconoscere i prodotti presenti su uno scaffale, a livello di singolo EAN, a partire dalla foto tale scaffali, superando gli attuali limiti imposti dagli algoritmi di *Computer Vision* come SIFT e rendendosi indipendente dalla sensoristica. In particolare, si vuole descrivere una possibile *pipeline* composta dall'app Store Audit di Grottini Lab, la rete neurale di [Goldman et al.] (capace di effettuare la *detection* di prodotti a partire dalla foto di uno scaffale) e la rete presentata in questa tesi il cui compito è quello di assegnare un vettore di *embedding* (ad ogni immagine in *output* alla rete di [Goldman et al.]) che sarà poi usato per inferire il giusto EAN valutando la *cosine similarity* con tutti i vettori di *embedding* di un *dataset* adibito al confronto, che sarà l'intero *dataset* realizzato durante il tirocinio. In Figura 1.2 è mostrata una schematizzazione della *pipeline*.

Bisogna considerare, inoltre, che il problema presenta numerose difficoltà e, infatti, una prima problematica può essere l'eccessiva somiglianza di prodotti che hanno EAN diversi oppure problemi relativi all'illuminazione ambientale, al punto di vista di chi

[4]Radio Frequency IDentification

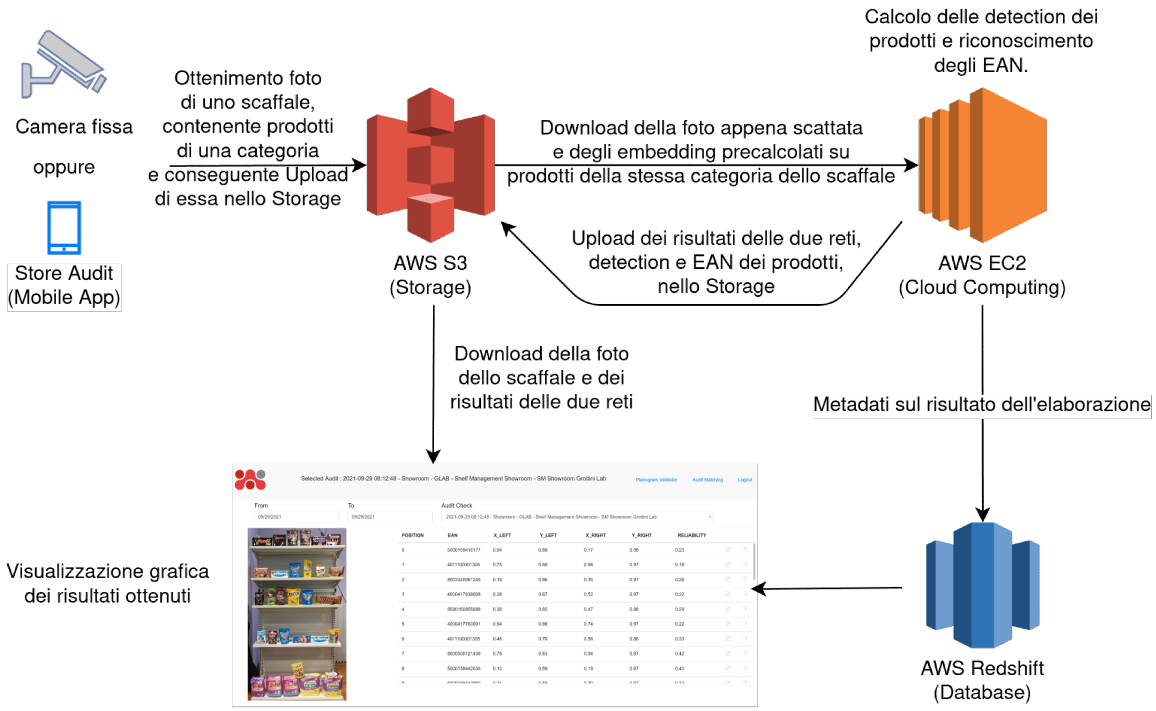


Figura 1.2: Schema di funzionamento della *pipeline*

fa le foto, alle occlusioni, alla qualità delle fotografie scattate e ad un insieme di altri fattori descritti nel Capitolo 3.1.2.

L'approccio utilizzato si pone l'obiettivo di essere il più possibile robusto a queste problematiche.

Nel Capitolo 2 verrà presentata l'analisi dello stato dell'arte per il riconoscimento di immagini simili nell'ambito della *Computer Vision* e del *Deep Learning*, nel Capitolo 3 sono descritte le motivazioni che hanno portato alla realizzazione di un nuovo *dataset*, come è stato costruito e qual è l'architettura della rete neurale presentata, nel Capitolo 4 sono riportati i risultati ottenuti e relativi commenti e nel Capitolo 5 si cercano di individuare possibili sviluppi futuri atti a migliorare i risultati ottenuti.

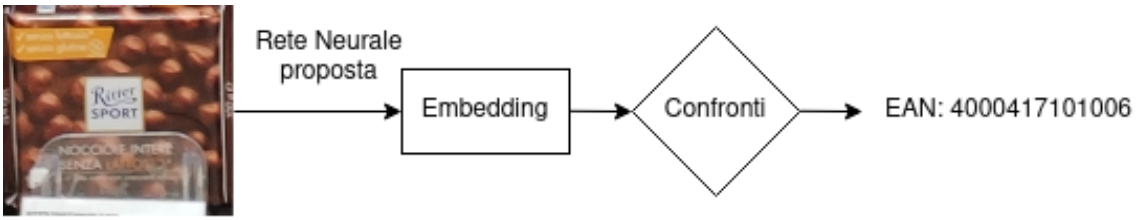


Figura 1.3: La rete neurale studiata e presentata in questa tesi vuole individuare l'EAN di ogni singolo prodotto presente in uno scaffale di un punto vendita

Capitolo 2

Stato dell'arte

In questo capitolo è presentanto lo stato dell'arte ([Wei et al.] [8]) relativo al riconoscimento di prodotti in ambito *retail* tramite tecniche di *Computer Vision* e *Deep Learning* dopo una descrizione del problema del riconoscimento di immagini simili più generica, non confinato al solo ambito *retail*.

2.1 Il problema del riconoscimento di immagini simili

Il problema del riconoscimento di immagini simili ha da sempre destato interesse da parte della comunità scientifica. In particolare, riuscire a distinguere tra classi simili è un compito davvero complesso, anche per la creazione di *dataset* correttamente annotati in quanto è richiesto un personale specializzato allo scopo: ad esempio, annottare un *dataset* di fiori a livello di singola specie è fattibile solo per esperti del settore. Per questo motivo, si sono cercati metodi per utilizzare le reti neurali anche con pochi dati dato l'incredibile costo economico che può avere produrli.

Un problema chiave che, quindi, attanaglia il *Deep Learning*, e il *Machine Learning* in generale, è la mancanza di *dataset* annotati e di alta qualità, necessari per l'addestramento delle soluzioni. Poiché solitamente le reti neurali necessitano di un elevato numero di dati per poter essere allenate correttamente, i ricercatori hanno cercato strade alternative e una soluzione si è ottenuta scavalcando il problema e introducendo l'addestramento *one-shot* [7], dove per effettuare l'allenamento sono necessarie solo una, o al più alcune, immagini di una certa classe. Inoltre, non si risolve un problema di classificazione[1] ma si fanno valutazioni e confronti sulla distanza di informazioni semantiche delle immagini (rappresentata da un vettore di *embedding*),

[1]La classificazione prende un *input* e restituisce direttamente la corrispettiva classe, sulla base della conoscenza della rete

in modo da individuare, in fase di inferenza, immagini simili e, conseguentemente, le *label*.

2.1.1 La nascita delle reti siamesi

Nel 1993 nacque l'idea di utilizzare delle reti neurali siamesi per verificare l'autenticità di firme mediante la verifica della similarità di immagini, come mostrato nel lavoro di ricerca di [Bromley et al.] [24] e la cui architettura è mostrata in Figura 2.1, dove possiamo vedere due reti neurali identiche che prendono due *input* diversi per restituire una misura di distanza.

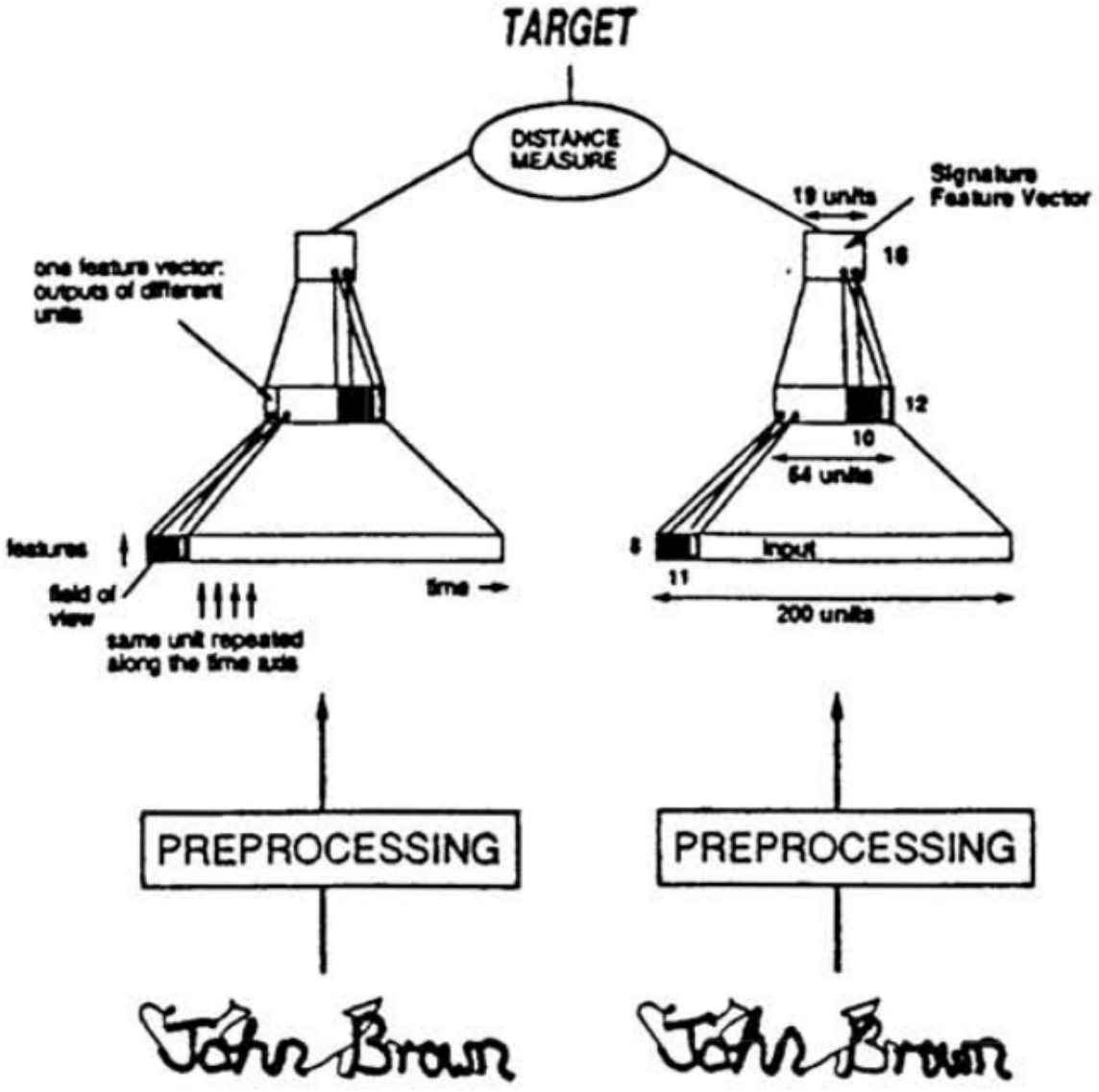


Figura 2.1: L'architettura proposta nel 1993 era formata da due semplici reti neurali identiche (da qui il nome siamesi) che processavano due input diversi. [24]

Una rete neurale siamese è composta da due reti che accettano *input* distinti e queste sono collegate mediante delle funzioni che si occupano di calcolare delle metriche tra le *feature* estratte dalle reti. Inoltre, le due reti sono perfettamente uguali, inclusi i parametri, e godono anche della proprietà di simmetria cioè dare un *input A* alla prima rete e un *input B* alla seconda è uguale a dare un *input B* alla prima e un *input A* alla seconda.

Quando una rete siamese viene allenata *one-shot*, essa prende in *input* due immagini diverse e restituisce un valore che mostra la loro somiglianza mediante una qualche metrica che può essere la distanza euclidea o la *cosine similarity*. Se le due immagini contengono lo stesso oggetto, la rete neurale, nel caso in cui si usi la distanza euclidea, restituirà un valore prossimo a zero e, se si usa la *cosine similarity*, restituirà un valori prossimo a 1.

Quando la dimensionalità dei vettori (e conseguentemente dello spazio) è troppo grande, i punti nello spazio sono tra loro molto distanti e la distanza euclidea tra ogni punto sarà sempre un valore molto grande, verificandosi quindi il problema della maledizione della dimensione (*curse of dimensionality*). L'utilizzo della *cosine similarity*, invece, è meno sensibile a questa problematica in quanto invece di calcolare la distanza tra i punti valuta l'angolo tra i vettori che congiungono l'origine a questi punti; un angolo basso tra i vettori implica che essi stanno "puntando" nella direzione, presupponendo quindi una certa somiglianza.

Nel Capitolo 3.2 si vedrà che l'ultimo strato della rete neurale implementata subirà la normalizzazione L2, rendendo quindi i vettori di *embeddings* di norma unitaria; questa accortezza renderà le due metriche equivalenti a meno di un fattore di scala in quanto i punti sono tutti posizionati in una sfera di raggio unitario.

Il lavoro di [Bromley et al.] non faceva ancora uso delle reti neurali convoluzionali (attualmente molto usate per estrarre *feature* da immagini) poiché esse sono solo da pochi anni usate intensamente grazie all'incremento della potenza di calcolo (soprattutto GPU).

2.1.2 *One-shot learning*

Nel 2015, il lavoro di [Koch et al.] [7] consiste nell'utilizzo dello stesso concetto di rete siamese presentato più di 20 anni prima, applicato al riconoscimento di lettere appartenenti al *dataset* Omniglot (Figura 2.2), raggiungendo un'accuratezza del 92%, utilizzando una rete neurale siamese convoluzionale e allenando la rete *one-shot*. Ha poi usato la stessa rete per valutare la bontà della generalizzazione utilizzando il dataset MNIST (Figura 2.3), ottenendo un'accuratezza del 70.3%.

Figura 2.2: Il *dataset* Omniglot contiene esempi di lettere provenienti da cinquanta alfabeti, dal latino fino alla lingua artistica Klingon

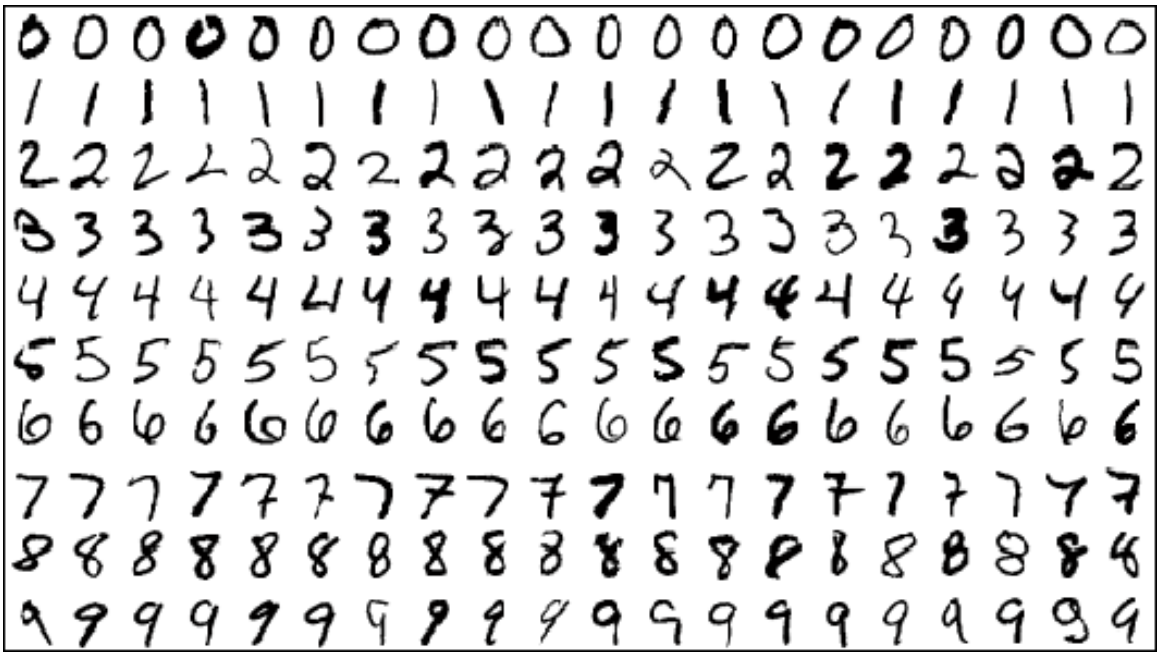


Figura 2.3: Il *dataset* MNIST contiene immagini di cifre scritte a mano, 60′000 nel training set e 10′000 nel test set

Quest'ultimo risultato ha mostrato un'interessante capacità di generalizzare l'estrazione delle *feature*: la rete è stata in grado di ottenere un'accuratezza interessante su un *dataset* mai visto prima, rendendo questa proprietà molto interessante ai nostri scopi per le problematiche che verranno trattate nel Capitolo 3.1.2.

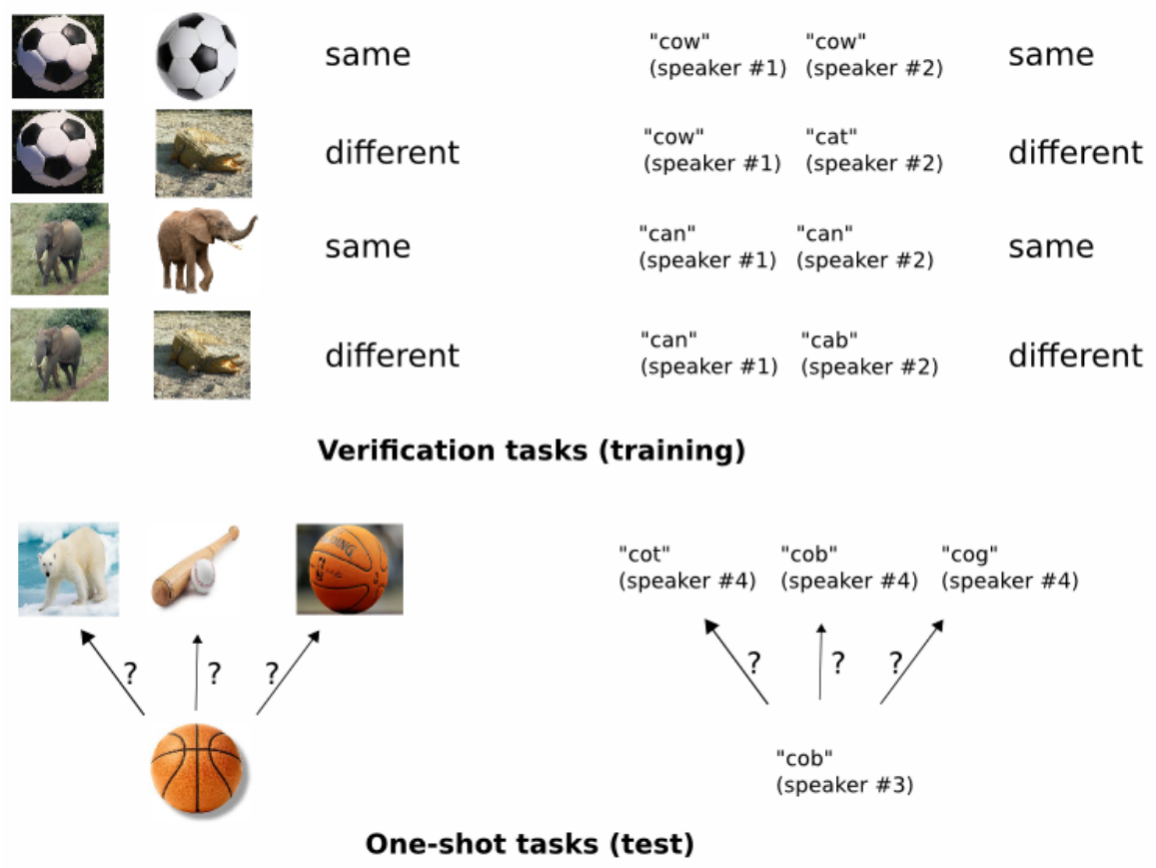


Figura 2.4: La strategia proposta in [7]: 1) Addestrare un modello per discriminare tra un insieme di coppie uguali/differenti (same/different) 2) Generalizzare per valutare nuove categorie sulla base delle *feature*.

2.2 Riconoscimento di immagini simili in ambito *retail - Computer Vision*

Negli ultimi decenni, la *Computer Vision* ha subìto un notevole sviluppo in ambito accademico e commerciale e, riguardo il riconoscimento di immagini simili, uno degli algoritmi più importanti è SIFT [2] (qualche anno dopo, è arrivato SURF [3] che ne vuole essere un miglioramento prestazionale). SIFT permette di estrarre feature significative dalle immagini ed ha numerose proprietà come l'invarianza alla rotazione, alla traslazione e alla scalatura (queste robustezze si pagano con un elevato costo

computazionale). Il calcolo delle feature *hand-crafted*[2] [12] avviene individuando le regioni caratteristiche dell'immagine tramite i *keypoint* (punti di interesse con un alto contenuto informativo), che sono poi caratterizzati calcolando un descrittore (che è un vettore di misure che può essere usato per allenare un classificatore, come la Support Vector Machine (SVM)) capace di distinguere un *keypoint* da un altro. Tuttavia, essendo le *feature* estratte da SIFT *hand-crafted*, non sono in grado di modellare adeguatamente la complessità del problema, risultando sensibile a variazioni di luminosità, alla tipologia di sensore di acquisizione, ecc.. Inoltre, da test interni eseguiti presso Grottini Lab, si è visto come SIFT risulta particolarmente sensibile ai riflessi e, addirittura, lo stesso oggetto scattato con due *smartphone* differenti non sempre viene riconosciuto, rendendo assolutamente inadatto questo approcci per il compito che si vuole risolvere.

Per questo motivo, è cresciuto l'interesse verso il mondo dell'Intelligenza Artificiale, in particolare il *Deep Learning*, per realizzare reti neurali atte ad estrarre le *feature* in maniera automatica. Attualmente, l'approccio maggiormente utilizzato per l'estrazione di *feature* da immagini è realizzato tramite reti neurali convoluzionali (*convolutional neural network*, CNN).

Prima di illustrare le ricerche più importanti che hanno cercato di risolvere il problema del riconoscimento dei prodotti in ambito *retail*, mostriamo quali sono i *dataset* attualmente più usati: *GroZi-120* [16], *GroZi-3.2k* [17], *Grocery store* [19] e *Freiburg grocery* [18]. Il problema di questi *dataset* è che presentano *packaging* non presenti nel territorio italiano (rendendo impossibile realizzare dei *test* in qualche punto vendita locale) e molte di queste immagini sono ottenute in ambienti controllati e non in scenari reali come, invece, è necessario per la risoluzione del compito presentato in questa tesi. Inoltre, non sono annotate a livello di EAN ma a livello di categoria (ad esempio, *coffee*, *chocolate*, *spices*, *sugar* come visibile in Figura 2.8) e, per ultimo, risultano contenere un numero di immagini limitato: il più grande di questi *dataset*, *GroZi-3.2k*, contiene *8,350* immagini nel *training set* e *3,235* nel *test set*.

Dunque, per i motivi appena illustrati, non saranno usati questi *dataset* ma si procederà alla realizzazione di un nuovo *dataset*, il cui processo è descritto nel Capitolo 3.1.

[2]Le feature *hand-crafted* sono feature scelte a mano come ad esempio l'individuazione di spigoli, angoli, istogrammi di colore, ecc.

Figura 2.5: Esempio del *dataset GroZi-120*. A sinistra è visibile un insieme di immagini di prodotti ottenute *in vitro* (ottenute in ambiente controllato), a destra le immagini degli stessi prodotti *in situ* (ottenute in contesti reali). `http://grozi.calit2.net/files/grozi_slam.pdf`

Figura 2.6: Esempio del *dataset Grocery store*. A sinistra è visibile un insieme di immagini definite dagli autori *iconic* (ottenute in ambiente controllato), a destra le immagini *natural* degli stessi prodotti (ottenute in contesti reali). `https://github.com/marcusklasson/GroceryStoreDataset`

Figura 2.7: Esempio di alcune immagini del *dataset GroZi-3.2k*, `https://drive.google.com/file/d/1QdNBMijT8PozTwLEK8fFuG9WiGtuHFeu`

Figura 2.8: Esempio di quattro classi del *dataset Freiburg Grocery*, `https://github.com/PhilJd/freiburg_groceries_dataset`

2.3 Riconoscimento di immagini simili in ambito *retail - Deep Learning*

Nel corso degli anni, ci sono state delle ricerche per cercare di riconoscere i prodotti a partire da immagini nell'ambito *retail* con reti neurali convoluzionali. Ad esempio, in [20], è stata usata *Inception V3* per classificare otto diverse classi di prodotti in scaffali reali di punti vendita. L'accuratezza in negozi reali è dell'87,5% e, considerando le poche classi, è un risultato sicuramente migliorabile.

Un lavoro più recente e che si riguarda più da vicino il nostro problema è quello di [Tonioni et al.] [21] che illustra un approccio in grado di riconoscere un elevato numero di classi. Nell'articolo, gli autori discutono una *pipeline* composta da un *Detector* che ha il ruolo di ottenere, a partire dalla foto dello scaffale, i singoli prodotti e un *Embedder* che è la rete estrattrice di *feature* (rete neurale convoluzionale *VGG16* allenata con *Triplet Loss*), da cui si ottiene un *embedding* dell'immagine ottenuto calcolando le *feature* MAC (attivazioni massime di convoluzioni). Il dataset che hanno utilizzato è il *GroZi-3.2k*, creato dagli autori del lavoro di ricerca [23] che contiene *3'288* diverse classi di prodotti alimentari ed hanno ottenuto risultati di *recall* pari a *57,07% PR* e una *mean average multi-label classification accuracy - mAMCA* del *31.57%*.

2.4 FaceNet

In virtù del lavoro di [Tonioni et al.] e per le motivazioni espresse nel Capitolo 3.2, l'approccio utilizzato in questa tesi è simile al lavoro di [Schroff et al.][4], intitolato *"FaceNet: A unified embedding for face recognition and clustering"*. In questo studio, gli autori hanno realizzato una tecnica per riconoscere volti di persone utilizzando una rete neurale convoluzionale che "mappa" ciascun volto in un vettore di dimensione 128. Tale vettore è un *embedding*: esso contiene parte della semantica dell'*input* e, in uno spazio degli *embedding*, *input* semanticamente simili sono vicini. Per questo motivo, una volta ottenuto questo vettore, è facile individuare il volto più simile: basta cal-

colare il vettore più vicino nello spazio degli *embedding*. Per l'addestramento della rete neurale, hanno usato triplette di facce (*anchor*, faccia simile all'*anchor*, faccia diversa dall'*anchor*) generate proponendo diversi metodi di creazione delle triplette. Gli autori propongono principalmente due tecniche per la creazione delle triplette: *online* e *offline*. La tecnica *offline* implica il calcolo delle triplette prima di iniziare l'allenamento mentre la tecnica *online* impone di calcolarle durante l'allenamento, all'interno del *batch* di immagini. In entrambi i casi, la *loss* risulta essere la *Triplet Loss*, il cui significato geometrico è mostrato nella Figura 2.9 e rappresentata analiticamente dalla seguente formula:

$$TripletLoss(A, P, N) = max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0) \qquad (2.1)$$

dove A è l'*anchor*, P è il positivo della stessa classe di A, N è il negativo di una classe differente rispetto ad A, α è un margine tra la coppia di positivi e negativi e *f(x)* è l'*embedding* di x.

L'obiettivo della *Triplet Loss* è, quindi, quello di realizzare uno spazio degli *embedding* dove *embedding* di immagini simili sono vicini e, viceversa, immagini diverse devono avere *embedding* distanti.

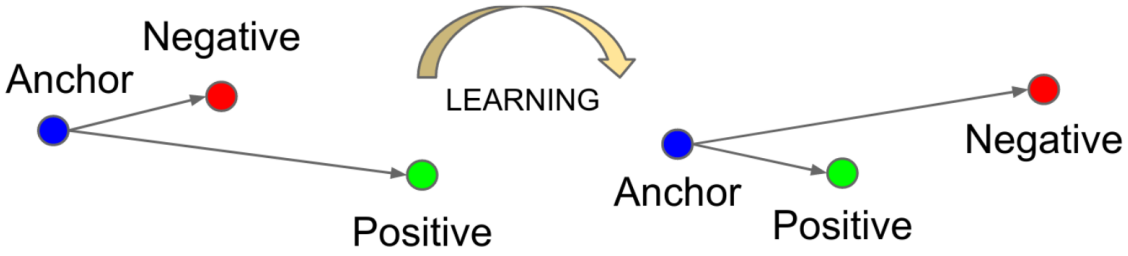


Figura 2.9: *Triplet Loss* [4]: essa minimizza la distanza tra un *anchor* e un positivo, entrambi con la stessa *label*, e massimizza la distanza tra l'*anchor* e un negativo di una *label* diversa

Infine, nella Figura 2.10, è mostrato uno schema dell'architettura proposta in *FaceNet*.

Intuitivamente, quindi, invece di classificare gli input, una rete che utilizza questa *loss* cerca di individuare le differenze tra gli *input* imparando le loro similarità. Da qui si intuisce un enorme vantaggio di questa tipologia di reti neurali: permettono di aggiungere un numero indefinito di nuove classi senza riallenarsi e questo è uno dei motivi principali che hanno spinto l'adozione di questo approccio. Ad esempio, nel contesto in cui è inserita la rete *FaceNet*, essa potrebbe essere usata in un sistema di riconoscimento volti in cui si possono variare i volti riconosciuti cambiando gli *embedding* nel *dataset* di confronto.

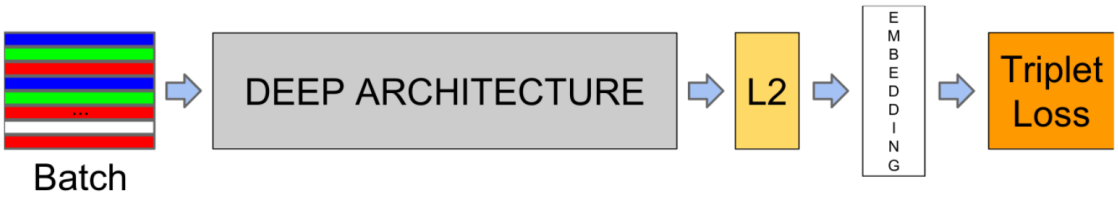


Figura 2.10: Architettura del modello [4]: schematicamente, la rete consiste in un batch di immagini dato in ingresso ad una rete neurale convoluzionale seguita da una normalizzazione L2 che restituisce l'*embedding*. La loss è la *Triplet Loss*.

2.4.1 Nota sulle possibili configurazioni di triplette

Sulla base della definizione della *loss* (Formula 2.1) è possibile realizzare tre diverse configurazioni di triplette:

- *easy triplets*: sono triplette dove la *loss* collassa a 0 perché:

  $\|f(A) - f(P)\|^2 + \alpha < \|f(A) - f(N)\|^2$

- *hard triplets*: sono triplette dove il negativo è più vicino all'*anchor* che al positivo.

  $\|f(A) - f(N)\|^2 < \|f(A) - f(P)\|^2$

- *semi-hard triplet*: sono triplette dove sia ha questa condizione:

  $\|f(A) - f(P)\|^2 < \|f(A) - f(N)\|^2 < + \|f(A) - f(P)\|^2 + \alpha$

Ognuna di queste definizioni dipende da dove si trova il negativo, relativamente all'*anchor* e al positivo.

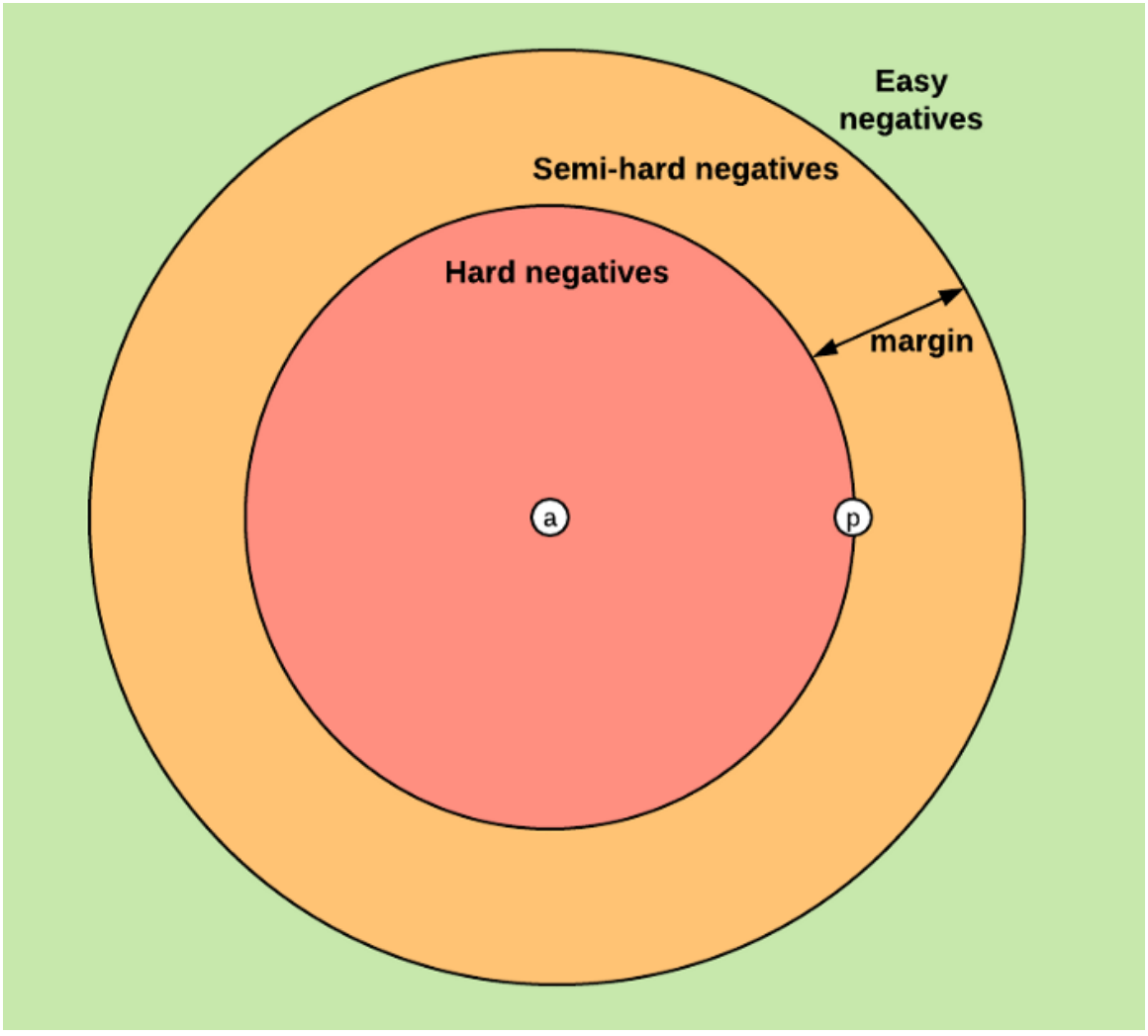


Figura 2.11: Posizionamento dei negativi nello spazio degli *embedding*, secondo la configurazione illustrata nella Sezione 2.4.1. Fonte: `https://omoindrot.github.io/triplet-loss`

Capitolo 3

Materiali e metodi

In questo capitolo è descritto come il *dataset* è stato raccolto, le difficoltà incontrate durante il processo e come è stata progettata la rete neurale.

3.1 Descrizione della raccolta del dataset e del risultato ottenuto

Come anticipato nei precedenti capitoli, il problema che questa tesi intende risolvere non è particolarmente studiato nell'ambito del *Deep Learning*; per questo motivo, non esistono *dataset* preesistenti e liberamente accessibili[1] adatti al lavoro presentato, contenenti quindi un numero elevato di immagini ottenute da scatti (in condizioni reali), realizzati con *smartphone*, di prodotti presenti negli scaffali di vari punti vendita ed etichettati con il giusto EAN.

Si è dovuto, quindi, realizzare un nuovo *dataset* composto di foto ottenute in condizioni reali di prodotti presenti in scaffali, etichettati con i corrispettivi EAN al fine di ottenere immagini come quelle mostrate nella Figura 3.1.

Per questo scopo, insieme all'azienda che mi ha ospitato durante il tirocinio, si è proceduto a scattare le foto di tutti gli scaffali presenti in un punto vendita della catena Acqua & Sapone, di un punto vendita Oasi (Gruppo Gabrielli) e di un punto vendita Si con Te (CE.DI.MARCHE) oltre al reparto cioccolate di un Iper. Insieme a queste foto sono stati raccolti gli EAN dei prodotti presenti nei corrispettivi scaffali tramite uno scanner di codici a barre. In particolare, sono state realizzate le seguenti foto di bay: 176 al punto vendita Acqua&Sapone, 333 al punto vendita Oasi, 160 al punto vendita Si con Te e 4 al punto vendita Iper.

[1] Alcune aziende che operano in questo settore dichiarano di avere *dataset* di questo tipo, ma non li forniscono liberamente

(a) EAN: 8002910057701

(b) EAN: 7640110704721

(c) EAN: 8002190002644

(d) EAN: 4005900694843

Figura 3.1: Esempio del nostro dataset

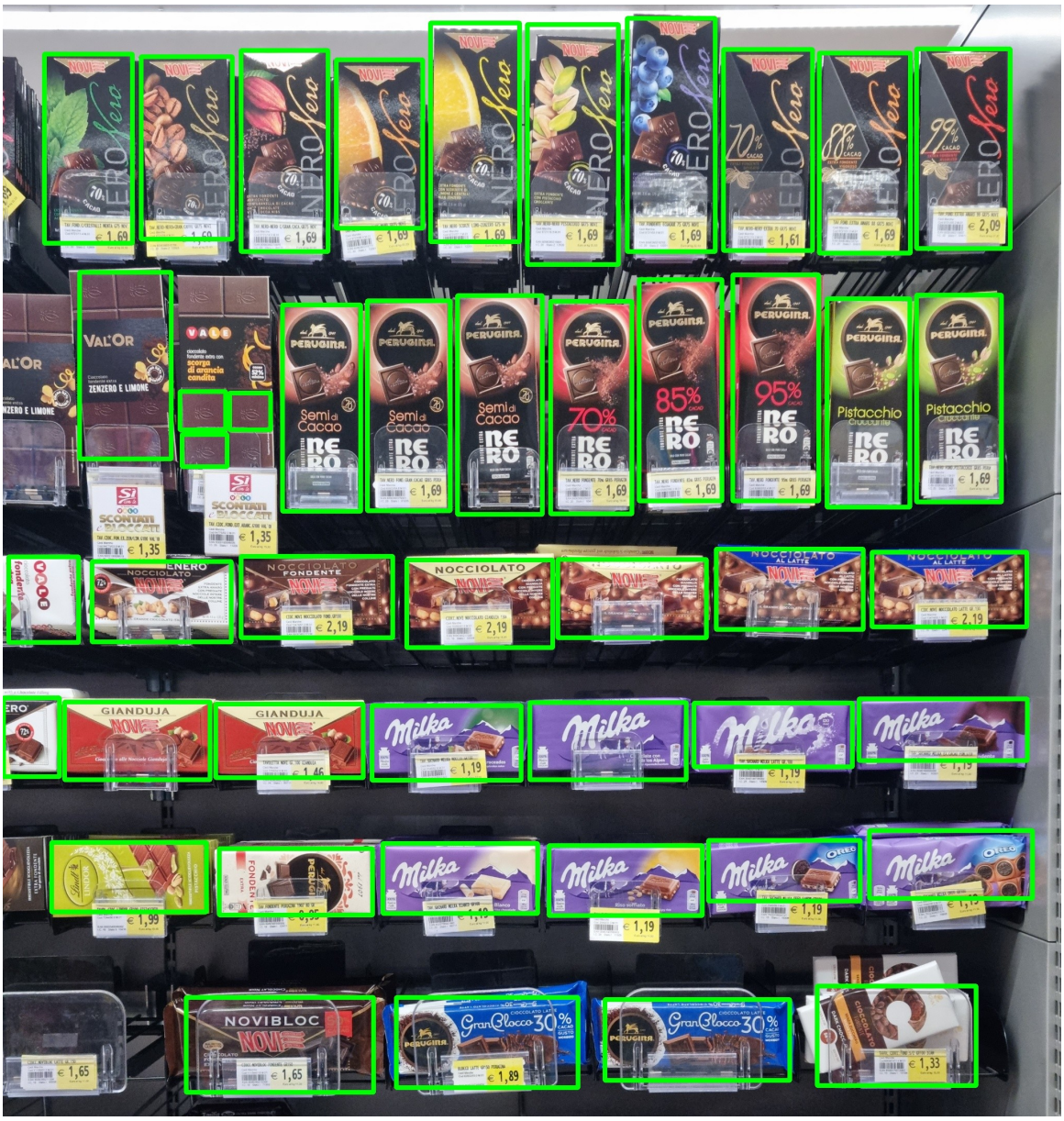


Figura 3.2: Immagine di esempio sull'individuazione dei *bounding box* tramite la rete di [Goldman et al.]

Successivamente, è stata utilizzata la rete neurale di [Goldman et al.][1][2] per effettuare la *detection* dei prodotti presenti nelle foto (Figura 3.2), ottenendo numerose immagini di singoli prodotti contenuti nella foto di uno scaffale. Si è usata tale rete poiché risolve un compito particolarmente complesso ovvero rilevare, in maniera accurata, i singoli prodotti in scaffali densamente popolati e, allo stato dell'arte attuale, risulta essere una delle più efficienti ed efficaci. Inoltre, tale rete è stata allenata con il *dataset* **SKU110K**, composto di *11,762* immagini di scaffali densi di prodotti di migliaia di punti vendita di tutto il mondo, dagli Stati Uniti all'Asia orientale, per un totale di $1.74 * 10^6$ *bounding box*. [3]. Tale rete è in grado anche di classificare le immagini individuate ma gli autori hanno associato, ad ogni immagine, la classe *"object"*.

Purtroppo, i *bounding box* realizzati da tale rete non sempre coincidono con le nostre necessità (individua alcuni falsi positivi come cartellini dei prezzi, Figura 3.4, oppure suddivide un unico prodotto in più prodotti) e, per questo motivo, è stato necessario creare un annotatore, realizzato in *Python* e dotato di interfaccia grafica realizzata con *tkinter*[4], che ci permettesse di creare manualmente dei *bounding box* personalizzati per poter realizzare il *dataset* con le migliori immagini possibili e che, infine, ci permettesse di associare, ad ogni immagine, il giusto EAN.

Inoltre, ad ogni immagine, è stata associata una categoria derivante da un albero delle categorie messo a punto da GS1 nel 1999 come risultato del lavoro di un gruppo di aziende che hanno voluto rispondere all'esigenza di avere una classificazione merceologica comune per i prodotti[5]. In particolare, tale categorizzazione è composta di cinque livelli, dove il quinto livello ha un grado di granularità massima. Ad esempio, il primo livello può essere DROGHERIA ALIMENTARE, BEVANDE, il secondo livello PRODOTTI FORNO E CEREALI, ACQUA, il terzo livello WAFER, ACQUA GASSATA, il quarto SEMPLICI, 0-50 CL, il quinto PORZIONATO, PLASTICA.

Per i nostri scopi, ci siamo fermati ad annotare fino al secondo livello di questo albero in quanto ritenuto un livello di granularità sufficiente. In fase di creazione del *dataset* e di etichettamento delle immagini, le categorie del secondo livello sono state convertite in numeri, rendendo la struttura finale del *dataset* quella presente in Figura 3.3, dove la cartella radice (in questo esempio iper) contiene le immagini del punto vendita Iper, la sottocartella (nell'esempio è presente la 0102 ma ovviamente possono essere di più) rappresenta la *label* della categoria (0102, CIOCCOLATA) e, all'interno di quest'ultima sottocartella sono presenti altre sottocartelle con il nome degli EAN

[2] `https://github.com/eg4000/SKU110K_CVPR19`

[3] `https://retailvisionworkshop.github.io/detection_challenge_2020/`

[4] `https://docs.python.org/library/tkinter.html`

[5] `https://gs1it.org/migliorare-processi/relazione-industria-distribuzione-best-practice-ecr/albero-categorie-classificazione-condivisa-prodotti/`

con all'interno le corrispettive immagini. La categorizzazione è nella forma *xxyy* dove *xx* rappresenta il primo livello dell'albero delle categorie e *yy* il secondo livello.

Questa categorizzazione è utile in fase di inferenza in quanto è possibile effettuare i confronti solo su un sotto-insieme del *dataset*, aumentando quindi le *performance* temporali a causa di un minor numero di confronti da realizzare e riducendo gli errori come, ad esempio, quello visibile in Figura 4.3.

Il risultato finale è stato l'ottenimento di un *dataset* composto di 35′802 immagini di 14′426 EAN distinti, divisi in 57 categorie. In media, di ogni EAN sono stati raccolti, quindi, circa $2,5$ immagini.

In Figura 3.1 possiamo vedere quattro prodotti, di varie categorie, con i rispettivi EAN.

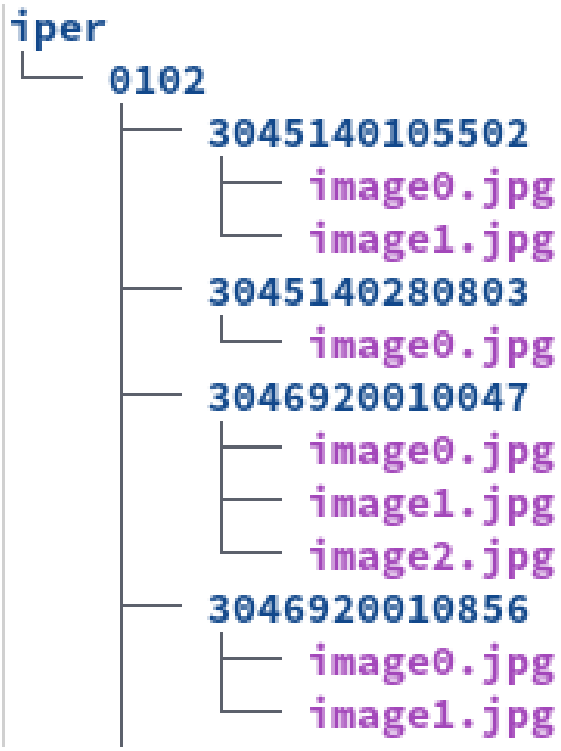


Figura 3.3: *Screenshot* sulla struttura del *dataset*.

3.1.1 EAN *0000000000000*

Abbiamo, inoltre, aggiunto alcune immagini di falsi positivi riconosciuti erroneamente dalla rete di [Goldman et al.] come articoli dotati di EAN *0000000000000*, come visibile in Figura 3.5. In questo modo, una parte di falsi positivi che la nostra rete processerà avrà EAN *0000000000000* e riusciremo ad individuarli facilmente e ad escluderli in modo da migliorare i risultati e mitigare leggermente le criticità della rete di [Goldman et al.].

Figura 3.4: Esempio di due cartellini di prezzo individuati dalla rete di [Goldman et al.]. Queste *detection* non sono state inserite nel *dataset*.

Questa particolare classe, comunque, non è stata utilizzata durante l'allenamento della rete ma solo in fase di inferenza.

Figura 3.5: Esempio di due immagini rumorose riconosciute come prodotti dalla rete di [Goldman et al.], inserite nel *dataset* come EAN *0000000000000*, ma non usate nell'addestramento.

3.1.2 Problemi riscontrati nella realizzazione del dataset

Durante la realizzazione del dataset, sono stati riscontrati numerosi problemi causati sia dalla rete di [Goldman et al.] e sia, in parte minore, dall'applicazione Store Audit di Grottini Lab usata per scattare le foto e per permettere un primo processamento delle foto degli scaffali da parte della rete di [Goldman et al.].

Un grosso problema riguarda la qualità delle ottiche degli *smartphone*; infatti, la qualità delle immagini dipende fortemente dalla qualità delle lenti e pertanto è consigliato non scendere sotto i 12 *megapixel* altrimenti essa risulta essere particolarmente rumorosa come è possibile vedere nella Figura 3.8.
Inoltre, ci sono stati una serie di problemi durante la raccolta del *dataset*, riguardante il *packaging* dei prodotti, così riassumibili:

- Anche a parità di EAN, ci possono essere *packaging* differenti a causa di fisiologici cambiamenti dovuti all'esigenza di aggiornare le grafiche o i marchi in esso presenti, anche per adattare le confezioni a nuove promozioni o collaborazioni aziendali esterne. In questi casi, abbiamo mantenuto il *packaging* nuovo scartando le immagini del *packaging* vecchio e immagini di *packaging* che non contengono parti dedicate a promozioni.

- I prodotti possono essere raggruppati in involucri trasparenti e la rete di [Goldman et al.] può effettuare la *detection* di ogni singolo prodotto presente nella confezione (non è detto che individua solo la singola grande confezione). In particolare, il problema è stato individuato con le casse di bibite.

- Sono numerosi i *packaging* non rigidi (come pasta e biscotti); le foto dei prodotti differiscono in base a come è distribuito il prodotto al loro interno.

- Alcuni *packaging* sono altamente riflettenti; i riflessi riducono fortemente la visibilità del prodotto come visibile in Figura 3.6, rendendo impensabile anche adottare tecniche di OCR su queste tipologie di prodotti come ipotetico sviluppo futuro.

- Problemi di posizionamento non frontale e/o allineamento non corretto. Nel caso dei prodotti posizionati non frontalmente, li abbiamo annotati ma sono stati attualmente esclusi nel *dataset*.

- A parità di EAN, può corrispondere uno stesso prodotto con *packaging* differente come visibile in Figura 3.9.

I problemi riscontrati nella creazione del *dataset* si avranno anche in fase di inferenza. Per ottenere il migliore risultato possibili si dovrà cercare di realizzare la foto dello scaffale nella maniera migliore possibile, curando la posizione dei prodotti e, se possibile, l'illuminaizione, andando a ridurre i riflessi in modo da facilitare il lavoro della rete di [Goldman et al.].

Figura 3.6: Confezione altamente riflettente. I riflessi rendono quasi illeggibile le scritte presenti nel *packaging*. Tale foto è stata, inoltre, scattata in condizioni ideali

(a) Lindt 70% cacao (b) Lindt 85% cacao (c) Lindt 99% cacao

Figura 3.7: Esempio di una prima problematica: prodotti con EAN diversi ma visivamente molto simili

(a) Immagine di scarsa qualità

(b) Immagine di buona qualità

Figura 3.8: Differenza di qualità delle immagini del nostro *dataset*: a sinistra è presente l'immagine di un prodotto realizzato con un'ottica scarsa. A destra, invece, è presente lo stesso prodotto ottenuto con un'ottica migliore

(a) Packaging con mascotte m&m's rossa

(b) Packaging con mascotte m&m's gialla

Figura 3.9: In queste due immagini sono presenti due prodotti con stesso EAN ma con packaging leggermente differente.

(a) Packaging standard (b) Packaging speciale Europei di Calcio Euro2020 (c) Packaging con promozione Amazon Prime

Figura 3.10: Queste tre immagini riguardano il packaging di uno stesso prodotto (stesso EAN), cambiato durante le acquisizioni del *dataset* nel periodo Giugno-Settembre 2021.

3.2 Rete Neurale

Poiché i prodotti nei vari punti vendita variano continuamente sia in numero che in tipologia, non si è potuto utilizzare una rete di classificazione con un numero fisso di classi altrimenti, per ogni nuovo articolo o variazione di *packaging* di uno preesistente, si sarebbe dovuto allenare nuovamente la rete di classificazione. Per questi motivi, si è dovuto utilizzare una rete che permettesse di essere indipendente dal numero delle classi.

Il nostro approccio, quindi, vuole associare il giusto EAN alla foto del prodotto tenendo conto della similarità tra le immagini, rendendosi indipendente dal numero delle classi e dalle modifiche ai *packaging*. Comunque, il riconoscimento di un nuovo prodotto richiede che all'interno del *dataset* usato per il confronto degli *embedding* sia presente la nuova immagine con il nuovo *packaging*, con il corrispettivo EAN. Infatti, la rete presentata genererà, per ogni immagine, un vettore di *embedding* che sarà poi confrontato, tramite calcolo della *cosine similarity*, con gli *embedding* delle immagini usate come *database* di confronto e dotate di *label* (che è l'EAN).

Come già anticipato nel precedente capitolo, l'approccio utilizzato per la risoluzione del nostro problema è fortemente ispirato al lavoro di [Schroff et al.][4], spostando il dominio applicativo, dal riconoscimento di volti all'individuazione degli EAN dei prodotti.

Siccome il nostro dataset risulta essere non eccessivamente grande, abbiamo usato una rete neurale convoluzionale con un numero di parametri inferiori rispetto a quello di FaceNet: da 140 milioni di parametri della rete usata in FaceNet a poco più di 18 milioni della nostra rete. Tale supposizione è stata, inoltre, supportata dai risultati sperimentali.

In particolare, si è deciso di utilizzare la rete neurale MobileNetV2 [10] (senza strato finale di classificazione), seguita da uno strato di densi con dimensionalità di 256. L'output della rete sarà, quindi, un *embedding* di dimensione 256 su cui è stato inoltre applicata la normalizzazione L2[6] come in *FaceNet*.

Una volta allenata la rete, per calcolare l'immagine più simile a quella presa in *input* si calcola la *cosine similarity* tra l'*embedding* dell'immagine di *input* e l'*embedding* di ciascuna immagini utilizzata per la verifica della similarità.

La *cosine similarity* tra due generici vettori numerici A e B è rappresentata dalla seguente formula ed è un numero compreso tra -1 e $+1$ dove valori prossimi a $+1$

[6]Tale normalizzazione ha il compito di rendere la norma di ogni *embedding* pari a 1, rendendo lo spazio degli *embedding* di raggio 1.

indicano una più alta somiglianza (+1 corrisponde a vettori uguali):

$$cosine\ similarity = \frac{A \cdot B}{\|A\| \, \|B\|} \tag{3.1}$$

L'EAN dell'immagine con la più alta *cosine similarity* sarà (idealmente) l'EAN dell'immagine in input alla rete.

La scelta di calcolare le triplette online e non offline è derivato anche dalle conclusioni di [Hermans et al.][9]. Essi, infatti, mostrano come l'utilizzo di triplette calcolate online possa aumentare di molto l'accuratezza del modello e ridurre i tempi di allenamento. Oltre a ciò, l'allenamento con tutte le possibili triplette non sarebbe comunque stata una strada percorribile in quanto la complessità diventerebbe $\mathcal{O}(\frac{N^3}{C})$ dove N è il numero di campioni e C è il numero di classi, rendendo l'allenamento quasi impossibile per *dataset* grandi.

Gli autori dell'articolo [9] introducono una variante della *Triplet Loss* chiamata *Batch Hard* (nel seguito nominata *Triplet Hard Loss*) dove, per ogni immagine all'interno del *batch*, si selezionano le immagini positive e le immagini negative più difficili all'interno del *batch* (utilizzano, cioè, *hard triplets*, Sezione 2.4.1).

Inoltre, definire in maniera statica le triplette potrebbe portare a creare triplette con un negativo troppo "semplice": infatti, prendendo la Formula 2.1, si può notare come il valore sarebbe sempre 0 se l'immagine negativa fosse troppo distante, nello spazio degli *embedding*, dall'*anchor*. Infatti, il valore $\|f(A) - f(N)\|^2$ risulterebbe essere molto più grande di $\|f(A) - f(P)\|^2$. L'ideale è, quindi, individuare quelle triplette dove l'*anchor* ha come "vicino più vicino" il negativo invece che il positivo, cioè $\|f(A) - f(N)\|^2 < \|f(A) - f(P)\|^2$.

Nota sul *soft margin*

Il ruolo della *hinge function* $(max(margin + \bullet,\ 0))$[7] è di evitare di correggere le terzine "già corrette", come discusso nel Capitolo 3.2. Dati gli incrementi di *performance* ottenuti nel lavoro di [Hermans et al.], abbiamo sostituito la *hinge function* con un'approssimazione usando la funzione *softplus*: $ln(1 + exp(\bullet))$. La funzione *softplus* ha un comportamento simile a quello della *hinge function*, ma decade esponenzialmente invece di avere un *cut-off* rigido, quindi quando parliamo di *soft margin* intendiamo questa particolare implementazione.

[7] La *Triplet loss* vista nella Formula 2.1 è una particolare *hinge function*. Il termine $\bullet$ è, nel nostro caso, $\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2$

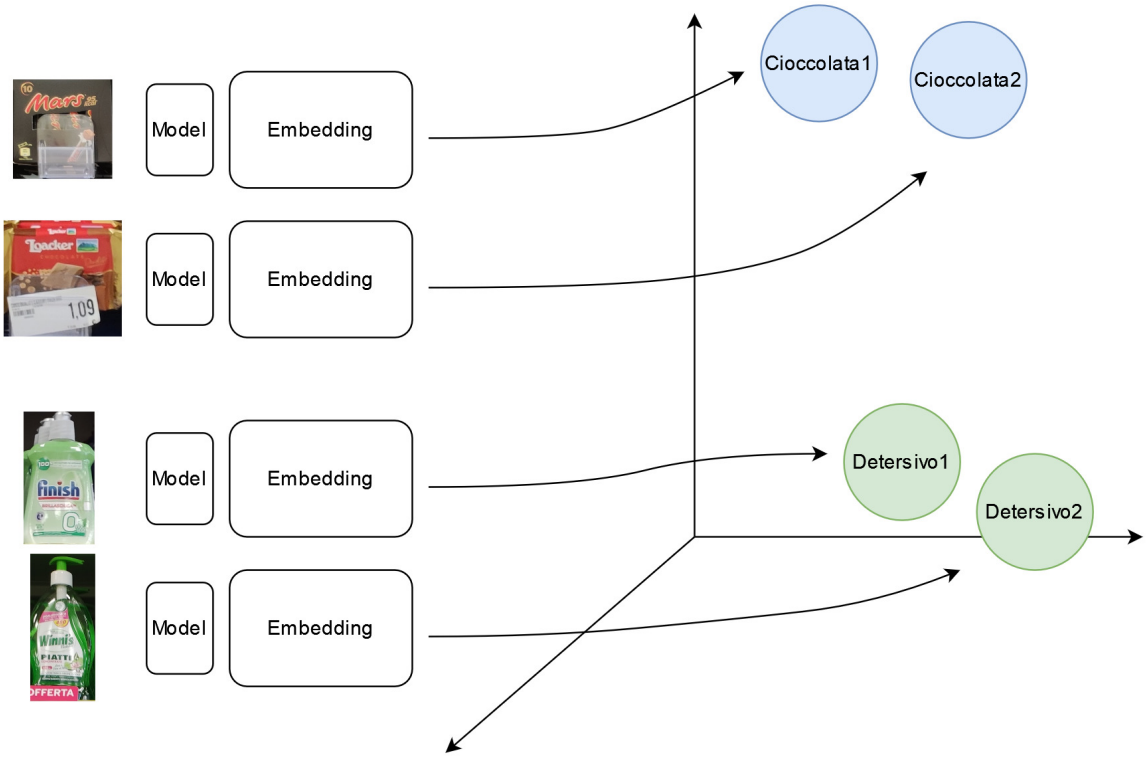


Figura 3.11: L'approccio utilizzato restituisce in *output* un *embedding* che proiettano gli elementi in uno spazio metrico dove elementi simili sono vicini e elementi dissimili sono lontani

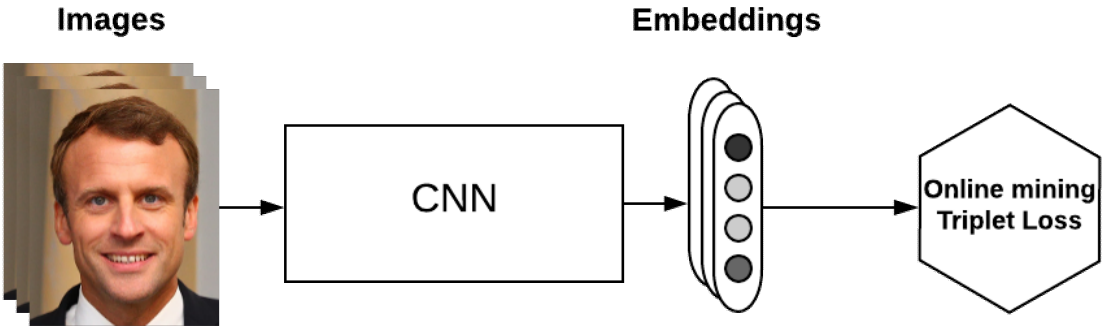


Figura 3.12: *Triplet Loss* con il calcolo delle triplette *online* all'interno di un *batch*. Fonte: `https://omoindrot.github.io/triplet-loss`

Capitolo 4

Risultati

L'ottenimento dei risultati presentanti in questo capitolo deriva dall'utilizzo della *pipeline* composta dallo scatto dell'immagine tramite l'app Store Audit di Grottini Lab, il successivo processamento in *cloud* tramite la rete di [Goldman et al.] per ottenere i singoli prodotti e, infine, il processamento di queste immagini tramite la rete neurale presentata in questa tesi, che andrà quindi, semplificando, a restituire in *output* l'EAN di ogni immagine. Bisogna precisare che i risultati ottenuti dipendono non solo dalle prestazioni della rete qui presentata, ma anche dalle prestazioni della rete di [Goldman et al.] in quanto, alcune volte, interpreta un singolo prodotto come tanti piccoli sottoprodotti (Figura 4.6) oppure va ad individuare come prodotti alcuni oggetti estranei presenti negli scaffali.

4.1 Protocollo sperimentale

La rete neurale illustrata nel Capitolo 3.2 è stata implementata tramite il framework *Tensorflow* con il linguaggio di programmazione *Python*. Durante l'allenamento, sono stati utilizzati i seguenti parametri: la rete adibita all'estrazione di *feature* è la *MobileNetV2* (decisa dopo diversi confronti), la dimensione dell'*embedding* è pari a *256*, la dimensione di ogni immagine è *224x224* a 3 canali, la *loss* è la *Triplet Hard Loss* con margin di 1.0 (*soft margin*), l'ottimizzatore della *loss* è Adam [14] con Learning Rate di 0.001 e l'allenamento è avvenuto in 80 epoche e, infine, la *batch size* è pari a *128*. Sono stati, inoltre, caricati i pesi di *ImageNet* [13], applicando quindi la tecnica del *transfer learning* per velocizzare e migliorare l'apprendimento ed è stato anche utilizzato l'*early stopping* per evitare l'*overfitting* andando a valutare, per ogni epoca, l'andamento della *validation loss* e salvando i pesi che avevano la *validation loss* minore.

Il *dataset* è stato diviso in *training* e *validation set* con percentuali dell'*80%* e del *20%*. Inoltre, prima di effettuare questa operazione di divisione, sono stati rimossi dal *dataset* totale 196 EAN (357 immagini di 17 categorie) presenti nel *dataset* del punto vendita Acqua&Sapone e presenti negli altri punti vendita: questi EAN isolati li abbiamo utilizzati come *test set* e sono, quindi, immagini che la rete neurale non ha mai visto, simulando quindi un uso realistico.

Infine, tutto il nuovo *dataset* è stato utilizzato per realizzare i confronti cioè, in fase di inferenza (con una foto nuova), ogni immagine prodotta dalla rete di [Goldman et al.] è data in *input* alla nostra rete che restituisce in *output* un vettore di *embedding*. Per ognuno di questi vettori viene calcolata la *cosine similarity* tra tutti gli *embedding* del dataset adibito al confronto al fine di individuare il vettore (che è associato ad un EAN) con la più alta *cosine similarity* e, conseguentemente, l'immagine più simile. In questo modo si può ottenere l'EAN.

Nel seguito, faremo riferimento ad una accuratezza calcolata nel seguente modo:

$$Accuratezza(set) = \frac{\#\ immagini\ del\ set\ associate\ al\ corretto\ EAN}{\#\ set} \tag{4.1}$$

4.1.1 Confronto tra *backbone*

Prima di decidere di utilizzare la rete MobileNetV2 come *backbone* (la rete che estrae le *feature*), abbiamo eseguito dei *test* di confronto valutando le prestazioni in termini di accuratezza. Le reti testate sono state la MobileNetV2, MobileNetV3 Large e Small [26], VGG16 [25] e Xception [28]. La VGG16 non è stata inserita nella tabella poiché la *loss* tendeva a rimanere costante, probabilmente a causa del numero elevato di parametri che ha rispetto alla numerosità del dataset. Nella Tabella 4.1 sono riportate le percentuali di accuratezza *TOP1*, *TOP5* e *TOP10* di ogni *backbone* testata. Per avere un confronto coerente, sono stati utilizzati gli stessi dati e parametri durante le diverse sessioni di allenamento. Il *dataset* di test contiene 357 immagini (196 EAN) di 17 categorie di un punto vendita Acqua&Sapone ed è stato usato tutto il *dataset* raccolto per il confronto (escluse, ovviamente, le 357 immagini).

Tabella di confronto dell'Accuratezza	*TOP1* (%)	*TOP5* (%)	*TOP10* (%)
MobileNetV2	43.4	69.2	75.6
MobileNetV3Small	41.1	60.5	70.5
MobileNetV3Large	40.6	55.7	64.4
Xception	18.7	36.13	42.5

Tabella 4.1: Confronto tra le reti usate come *backbone*

Come riportato in questa Tabella, la rete MobileNetV2 ottiene i migliori risultati di accuratezza.

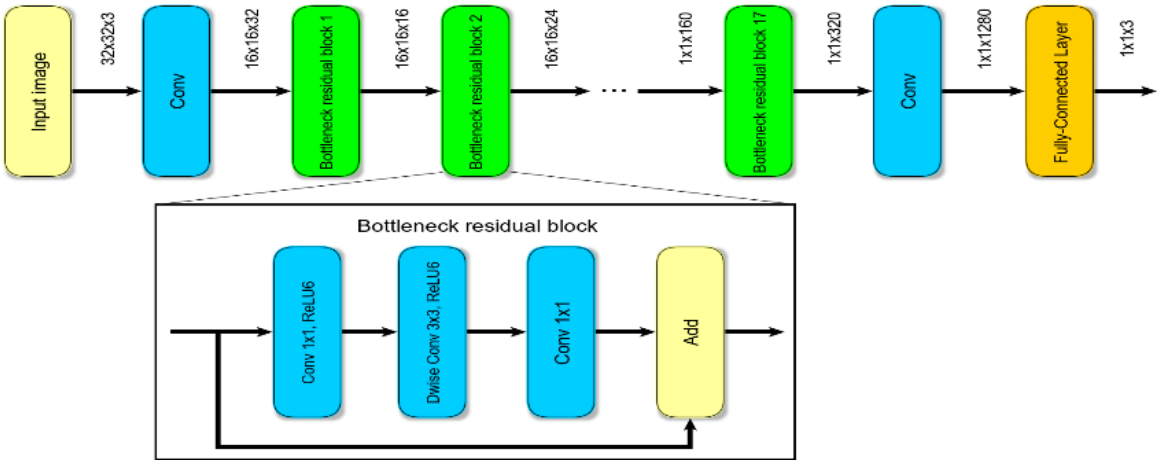


Figura 4.1: Architettura della rete MobileNetV2. Nella nostra architettura, l'ultimo strato *Fully Connected* non è stato utilizzato. Fonte: [27]

4.2 Risultati ottenuti in un contesto reale

Facendo l'analisi solo su due categorie esemplificative (delle 17 categorie del *dataset* di test) usando la rete con MobileNetV2 come *backbone*, si ottengono questi valori di accuratezza:

- Categoria "Cura persona/Igiene personale": *TOP1*: 50.0%, *TOP5*: 68.42%, *TOP10*: 76.31%
- Categoria "Cura persona/Igiene orale": *TOP1*: 35.41%, *TOP5*: 68.75%, *TOP10*: 79.16%

Già da questi risultati possiamo intuire che il corretto EAN è, almeno due volte su tre, contenuto nella *TOP5*. Guardando la Figura 4.2 possiamo vedere come esistano dei prodotti estremamente simili tra di loro: in questo esempio, ci sono piccolissimi cambiamenti nelle scritte presenti nelle etichette e nel colore dei flaconi. Tuttavia, lampade di colore diverso che illuminano lo scaffale o ombre posizionate in maniera differente mettono in difficoltà il riconoscimento del giusto prodotto da parte della nostra rete.

Inoltre, la categoria "Cura persona/Igiene orale" presenta molti articoli appesi sui ganci e non appoggiati direttamente ai ripiani, complicando ulteriormente il riconoscimento a causa delle variazioni di angolatura del prodotto.

Casi particolari

Nelle Figure 4.2 e 4.3 possiamo notare due diversi problemi incontrati. In particolare, nella prima figura notiamo la presenza del problema del riconoscimento di prodotti estremamente simili tra di loro (*Fine-Grained Classification*). Notiamo come alcune immagini risultano essere particolarmente rumorose a causa dell'utilizzo di *smartphone* con lenti di scarsa qualità, rendendo il lavoro di riconoscimento difficile anche ad un umano. In particolare, la rete tende ad associare, ad un *input* rumoroso, un EAN di un'immagine a sua volta rumorosa, facendoci pensare che il rumore sia stata interpretata come una *feature*.

La seconda Figura mostra un simpatico errore: la nostra rete confonde i ceci con le nocciole (essendo *pattern* visivamente simili) e riteniamo che questo problema possa essere risolto con un aumento del *dataset* di allenamento.

Figura 4.2: *TOP10* di un particolare detersivo. La nostra rete non riesce sempre a districarsi tra prodotti molto simili tra loro, come in questo caso.

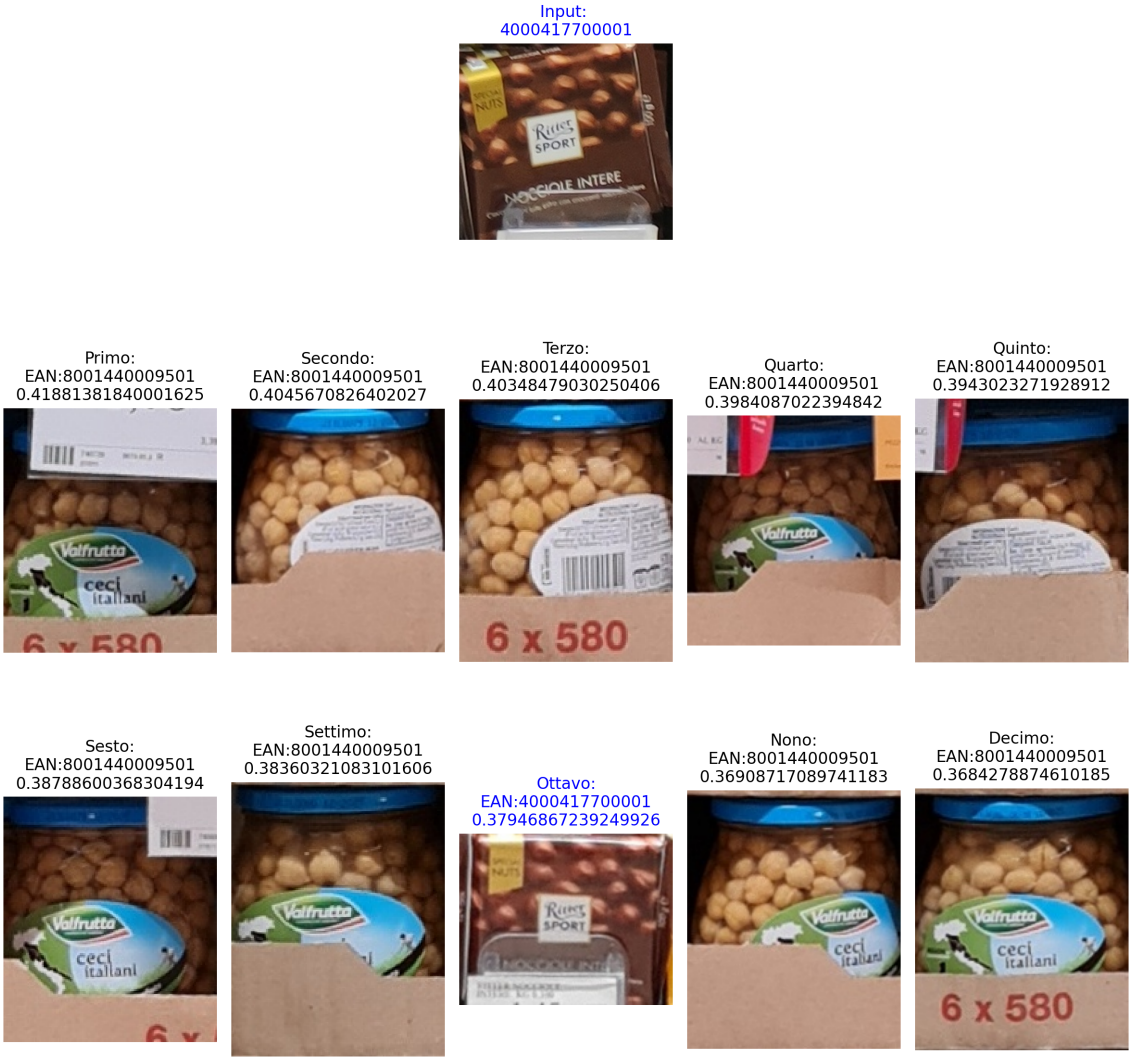


Figura 4.3: *TOP10* di una tavoletta di cioccolata. Notiamo come la nostra rete confonda nove volte su dieci le nocciole con i ceci.

4.3 Risultati ideali ottenuti in laboratorio

I risultati ottenuti nel *bay* di prova (visibile in Figura 4.7) sono stati ottenuti confrontando le immagini ottenute dalla rete di [Goldman et al.] di questo scaffale con 163 EAN della categoria cioccolate ottenute presso un punto vendita Iper. In particolare, abbiamo ottenuto valori di accuratezza del $95,6\%$: bisogna comunque considerare che questo è un risultato ottenuto in un ambiente controllato e, inoltre, le due tavolette di cioccolato del secondo ripiano sono state divise, ciascuna, in due parti. Quest'ultimo punto porta alla problematica visibile nella Figura 4.6 che si ripercuote in una doppia individuazione del prodotto da parte della rete (valuta le due immagini tagliate come due prodotti interi). A volte, un singolo prodotto può essere diviso anche in più parti e, soprattutto in questi casi, la nostra rete può associare le immagini tagliate a prodotti completamente diversi per cercare di ovviare a questo problema e come anticipato nei capitoli precedenti, abbiamo aggiunto un EAN *0000000000000* nel dataset usato per i confronti con immagini di falsi positivi della rete di [Goldman et al.], i cui risultati sono visibili nella Figura 4.5.

4.3.1 Risultati in laboratorio con utilizzo di immagini *render*

Durante i *test* di laboratorio, sono state anche provate altre combinazioni nella posizione dei prodotti: le tavolette di cioccolata orizzontali non individuate correttamente (individuate come due o più prodotti) sono state posizionate in posizione verticale ed è stato rimosso il prodotto con le promozioni di *Amazon Prime* (terzo prodotto del secondo scaffale di Figura 4.7) in quanto il *packaging* cambiava sensibilmente da quelli presenti nel nostro *dataset* rendendo il riconoscimento corretto quasi impossibile.

Nel *dataset* delle immagini di confronto sono state inserite delle immagini ideali provenienti da *rendering*, come mostrato in Figura 4.4, che non sono mai state viste dalla nostra rete durante l'allenamento. In particolare, il *render* del prodotto in Figura 4.4a rappresenta un prodotto nuovo, non presente nel nostro *dataset*.

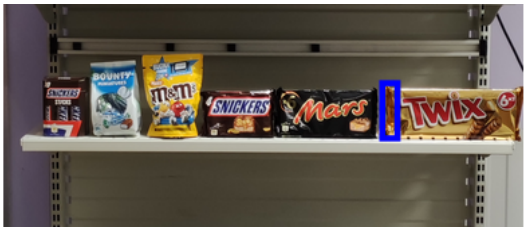
Nello scaffale riorganizzato è stato aggiunto questo nuovo prodotto, non presente nel nostro *dataset*, ed è stato comunque riconosciuto correttamente, confermandoci le capacità di generalizzazione della nostra rete.

Questo nuovo tentativo ha consentito di raggiungere un'accuratezza del 100% ma bisogna essere consapevoli che questo dato è assolutamente ideale, non raggiungibile in un contesto reale.

(a) m&m's PEANUT da 500g

(b) m&m's CHOCOLATE in posizione verticale

Figura 4.4: Esempio di due immagini ottenute da processi di *rendering*



16	5000159404235
17	0000000000000
18	5000159461849

Figura 4.5: Il bounding box della figura di sinistra contiene un falso positivo della rete di [Goldman et al.] ma, in fase di inferenza, la nostra rete è stata in grado di associarla ad un EAN fitizio *0000000000000*

4.3.2 Commento sulle prestazioni temporali

Le prestazioni temporali dell'inferenza sono interessanti: in 118 secondi[1] si hanno i risultati di più di 350 immagini, senza utilizzo di *GPU*. Questi due minuti scarsi tengono conto del tempo necessario a generare gli *embedding* delle immagini di input e a confrontarli con il *dataset* appositamente realizzato, che contiene più di 35′000 immagini.

Questi risultati sono stati ottenuti tramite l'implementazione della *cosine similarity* con **Numba**[2], un compilatore *just in time* per *Python* che ha permesso un aumento di velocità di diversi ordini di grandezza rispetto al codice nativo (interpretato) di *Python*.

Individuazione soglia *cosine similarity*

Nei risultati appena mostrati, non è stata applicata nessuna sogliatura minima nel valore di *cosine similarity* in quanto, allo stato attuale, riuscire a distinguere risul-

[1] Dell Inspiron 5593, 16GB RAM, Intel i5-1035G1, Fedora 34
[2] `https://numba.pydata.org/`

tati rumorosi da risultati corretti ma con un valore basso di *cosine similarity* porta all'esclusione di un numero non indifferente di corrette individuazioni.

(a) (b)

(c)

Figura 4.6: Le Figure (a) e (b) sono quelle individuate dalla rete di Goldman et al.. Il risultato desiderabile è, invece, in (c).

Figura 4.7: Bay di prova, composto da quattro ripiani, di prodotti del marchio MARS presso Grottini Lab

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi è stata proposta una possibile *pipeline* per risolvere il compito del riconoscimento degli EAN dei prodotti negli scaffali. Come visto nella sezione dei risultati, essi sono particolarmente incoraggianti e futuri miglioramenti si potranno avere riallenando la rete di [Goldman et al.] (anche utilizzando il nuovo *dataset* poiché, durante il processo di annotazione manuale, sono state anche salvate le coordinate dei *bounding box*) in modo da cercare di ridurre quei fenomeni di falsi positivi quali i cartelli dei prezzi e le parti metalliche degli scaffali oppure le divisioni dei prodotti in sottoprodotti (Figura 4.6).

Per migliorare le *performance* della rete neurale proposta, si può inoltre pensare di utilizzare le informazioni della *TOP5* o *TOP10* per cercare di individuare il corretto prodotto quando essi risultano essere molto simili, come visibile in Figura 4.2. Infatti, nella maggior parte dei casi il prodotto corretto risulta essere nella *TOP10*.

Un ulteriore aumento del *dataset* raccolto, con conseguente riallenamento della rete e aumento degli *embedding* nel *dataset* di confronto, può portare ad ottenere risultati migliori così come l'utilizzo di diverse tecniche di Intelligenza Artificiale che saranno sviluppate in futuro dalla comunità scientifica. Miglioramenti dello *stitching* nell'app Store Audit di Grottini Lab porteranno a sicuri benefici, soprattutto con l'utilizzo di *smartphone* con scarse ottiche e punti vendita con corsie particolarmente strette, che non consentono di inquadrare tutto lo scaffale con un'unica foto.

Inoltre, nei prossimi anni ci sarà sicuramente un aumento della qualità delle ottiche degli *smartphone* che porterà a realizzare foto sempre migliori: questo sarà fondamentale per applicare tecniche di OCR al fine di aumentare l'accuratezza dell'intera *pipeline*. Investimenti in questo campo del mondo *retail* sono da non ignorare anche in virtù del fatto che, quando il planogramma è completamente corretto, si può avere un incremento delle vendite del 7.8% e un miglioramento del profitto dell'8.1% in sole due settimane [22].

In Figura 5.1 è presente una prima possibile versione di una interfaccia web per la visualizzazione del risultato della *pipeline* al fine di vedere lo stato corrente dello scaffale, potendo verificare i prodotti posizionati tramite gli EAN di ciascuno di loro.

Inoltre, può risultare interessante procedere alla creazione o acquisto da aziende specializzate di un'anagrafica degli EAN contenente, per ognuno di questi, le informazioni del prodotto quali il nome, il marchio, il peso della confezione ecc.. Ciò renderebbe possibile l'ottenimento di nuove e interessanti informazioni quali, ad esempio, statistiche sul maggior marchio presente in uno scaffale (*share of shelf*).

Questa tipologia di sistemi può essere destinata anche ai grandi *brand* del settore *retail*, oltre che ai punti vendita, per poter fare delle statistiche sulla percentuale di loro prodotti presenti in un dato scaffale, per verificare la correttezza del planogramma e per valutare i prezzi di vendita esposti dai punti vendita per realizzare statistiche interne. Mentre i primi due punti appena espressi sono stati implementati (per le statistiche basta procurarsi l'anagrafica degli EAN), per il terzo punto è necessario integrare un meccanismo di lettura dei prezzi dai cartellini, realizzabile con un'ulteriore rete neurale allenata allo scopo.

Inoltre, attualmente la rete non individua "buchi" negli scaffali derivanti da rotture di *stock*; pertanto, potrebbe essere interessante implementare un algoritmo che valuta la distanza tra ogni *bounding box* contiguo e, se superiore ad una certa soglia, inferire un prodotto mancante.

Come ulteriore sviluppo futuro per l'incremento del *dataset*, si potrebbe pensare di usare reti generative (come le **GAN**) per aumentare il numero di campioni e migliorare tutta la *pipeline*.

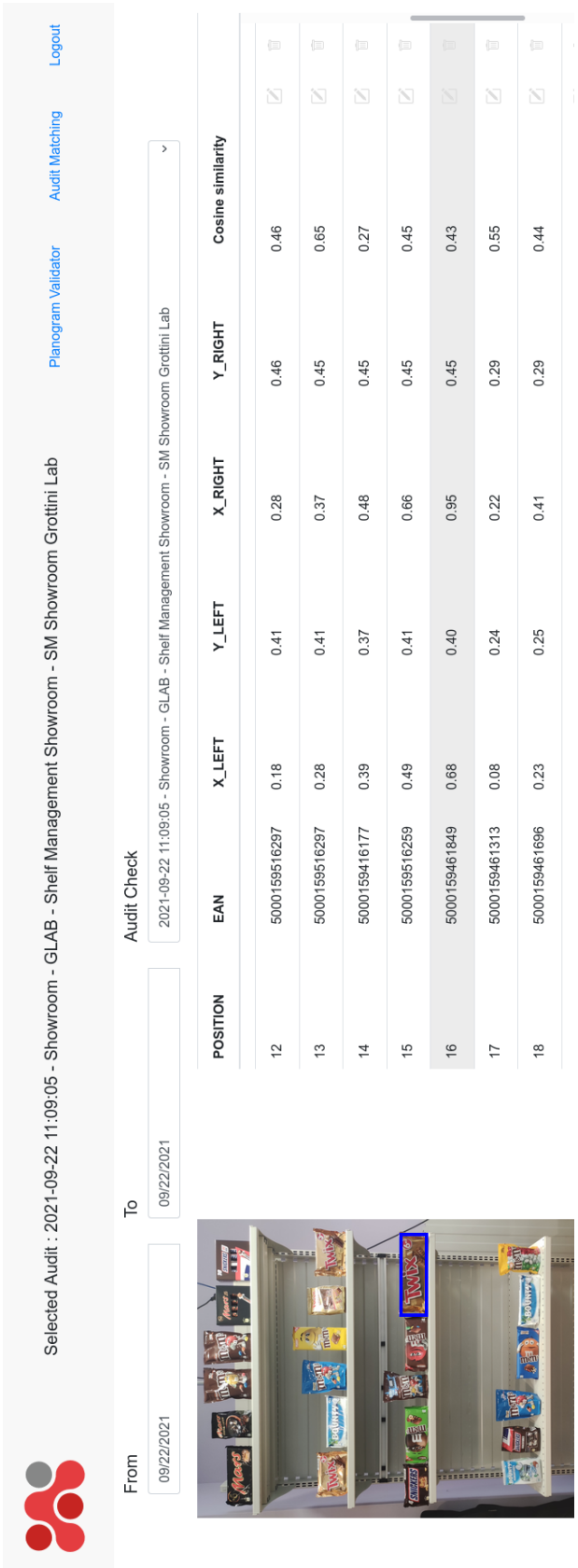


Figura 5.1: Prima versione di una pagina che visualizza i risultati della *pipeline*, realizzata da Grottini Lab

Glossario

Bay Singolo modulo di uno scaffale. 9, 27, 47, 50

EAN L'EAN, *European Article Number*, è un codice europeo che rappresenta, in maniera univoca, un certo prodotto. Può essere di 8 o 13 cifre come ad esempio 5000159416177. Tale codice è rappresentato graficamente dal codice a barre del prodotto, che lo rende quindi facilmente leggibile da un lettore ottico dedicato. 9, 13, 20, 27, 30, 31, 33, 37, 38, 41–43, 47, 48, 51, 52

Image Stitching Lo stitching delle immagini è quel processo in cui sono combinate immagini dello stesso soggetto ma scattate da punti di vista leggermente differenti. 51

OCR Con il termine OCR, acronimo di *Optical Character Recognition*, si intende l'insieme di quelle tecniche atte a riconoscere caratteri dell'alfabeto da un'immagine e a renderli comprensibili ad un calcolatore. 33, 51

Planogramma Il planogramma rappresenta la disposizione dei prodotti in uno scaffale, posizionati all'interno di un punto vendita, al fine di aumentarne le *performance* e, conseguentemente, gli acquisti da parte dei clienti. 11, 51, 52

Ripiano È il palchetto di uno scaffale. 9, 47, 50

Scaffale Elemento costituito da una serie di ripiani orizzontali sovrapposti l'uno all'altro sui quali si dispongono oggetti vari. È composto da più bay. 7, 11–14

Bibliografia

[1] Eran Goldman et al. «Precise Detection in Densely Packed Scenes». In: *Proc. Conf. Comput. Vision Pattern Recognition (CVPR)*. 2019.

[2] D.G. Lowe. «Object recognition from local scale-invariant features». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[3] Herbert Bay, Tinne Tuytelaars e Luc Van Gool. «SURF: Speeded Up Robust Features». In: *Computer Vision – ECCV 2006*. A cura di Aleš Leonardis, Horst Bischof e Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[4] Florian Schroff, Dmitry Kalenichenko e James Philbin. «FaceNet: A unified embedding for face recognition and clustering». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, giu 2015. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

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

[6] Iaroslav Melekhov, Juho Kannala e Esa Rahtu. «Siamese network features for image matching». In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, dic 2016. DOI: 10.1109/icpr.2016.7899663. URL: http://dx.doi.org/10.1109/ICPR.2016.7899663.

[7] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov et al. «Siamese neural networks for one-shot image recognition». In: *ICML deep learning workshop*. Vol. 2. Lille. 2015.

[8] Yuchen Wei et al. «Deep Learning for Retail Product Recognition: Challenges and Techniques». In: *Computational Intelligence and Neuroscience* 2020 (nov. 2020). A cura di Massimo Panella, pp. 1–23. DOI: 10.1155/2020/8875910. URL: http://dx.doi.org/10.1155/2020/8875910.

[9] Alexander Hermans, Lucas Beyer e Bastian Leibe. *In Defense of the Triplet Loss for Person Re-Identification*. 2017. eprint: arXiv:1703.07737.

[10] Mark Sandler et al. «MobileNetV2: Inverted Residuals and Linear Bottlenecks». In: (2018). eprint: arXiv:1801.04381.

[11] Rajesh Ray. *Supply chain management for retailing*. New Delhi: Tata McGraw-Hill Education, 2010. ISBN: 9780070145047.

[12] Loris Nanni, Stefano Ghidoni e Sheryl Brahnam. «Handcrafted vs. non-handcrafted features for computer vision classification». In: *Pattern Recognition* 71 (nov. 2017), pp. 158–172. DOI: 10.1016/j.patcog.2017.05.025. URL: https://doi.org/10.1016/j.patcog.2017.05.025.

[13] Jia Deng et al. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[14] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[15] Bikash Santra e Dipti Prasad Mukherjee. «A comprehensive survey on computer vision based approaches for automatic identification of products in retail store». In: *Image and Vision Computing* 86 (2019), pp. 45–63. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2019.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0262885619300277.

[16] Michele Merler, Carolina Galleguillos e Serge Belongie. «Recognizing Groceries in situ Using in vitro Training Data». In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383486.

[17] Marian George e Christian Floerkemeier. «Recognizing Products: A Per-exemplar Multi-label Image Classification Approach». In: *Computer Vision – ECCV 2014*. A cura di David Fleet et al. Cham: Springer International Publishing, 2014, pp. 440–455. ISBN: 978-3-319-10605-2.

[18] Philipp Jund et al. «The Freiburg Groceries Dataset». In: *CoRR* abs/1611.05799 (2016). arXiv: 1611.05799. URL: http://arxiv.org/abs/1611.05799.

[19] Marcus Klasson, Cheng Zhang e Hedvig Kjellström. «A Hierarchical Grocery Store Image Dataset with Visual and Semantic Labels». In: *CoRR* abs/1901.00711 (2019). arXiv: 1901.00711. URL: http://arxiv.org/abs/1901.00711.

[20] Idawati Bustan Timothy Chong e Mervyn Wee. «Deep Learning Approach to Planogram Compliance in Retail Stores». In: *Stanford, CA, USA* (2016).

[21] Alessio Tonioni, Eugenio Serra e Luigi di Stefano. «A deep learning pipeline for product recognition on store shelves». In: *CoRR* abs/1810.01733 (2018). arXiv: 1810.01733. URL: http://arxiv.org/abs/1810.01733.

[22] M. Shapiro. «Executing the best planogram». In: *Norwalk, CT, USA* (2009).

[23] Marian George e Christian Floerkemeier. «Recognizing products: A per-exemplar multi-label image classification approach». In: *European Conference on Computer Vision*. Springer. 2014, pp. 440–455.

[24] Jane Bromley et al. «Signature verification using a "siamese" time delay neural network». In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993), pp. 669–688.

[25] Karen Simonyan e Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[26] Andrew Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV].

[27] Ulzhalgas Seidaliyeva et al. «Real-Time and Accurate Drone Detection in a Video with a Static Background». In: *Sensors* 20.14 (2020). ISSN: 1424-8220. DOI: 10.3390/s20143856. URL: https://www.mdpi.com/1424-8220/20/14/3856.

[28] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV].

Ringraziamenti

In quest'ultimo capitolo della tesi intendo ringraziare tutte le persone che hanno contribuito alla realizzazione di della tesi.

Un sentito grazie va al mio relatore, *Prof.* Emanuele Frontoni, che mi ha messo in contatto con l'azienda Grottini Lab e con la quale ho collaborato fin dal primo istante senza problemi e per il suo incredibile lavoro nella divulgazione delle potenzialità e delle infinite applicazioni dell'Intelligenza Artificiale.

Ringrazio anche il mio correlatore Rocco Pietrini, *R&D Engineer* presso Grottini Lab, che mi ha guidato e pazientemente aiutato nella progettazione e realizzazione del codice e nell'applicazione delle idee illustrate in questa tesi.

Non posso non ringraziare anche tutto lo staff di Grottini Lab per le *soft skills* che mi hanno permesso di acquisire e il VRAI - Vision, Robotics and Artificial Intelligence del dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche che ha contribuito alla realizzazione del *dataset*, così come tutti i punti vendita *partner* di Grottini Lab che hanno autorizzato le nostre acquisizioni.

Grazie a tutti i miei colleghi e compagni di corso con cui ho stretto amicizia e preparato numerosi esami e che mi hanno sostenuto in questi anni.