Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Sviluppo di un applicazione su reti neurali per la classificazione di capi d'abbigliamento

Relatore: Chiar.mo Prof.
Elena Loli Piccolomini
Correlatore: Chiar.mo Prof.
Gustavo Marfia

Presentata da:
Pierluigi Tartabini

I Sessione
Anno Accademico 2017/2018

"L'importante non è fare quel che si ama
ma amare quel che si fa"
Andrea Bocelli

Introduzione

Il lavoro che ho svolto in questa tesi è la classificazione di capi d' abbigliamento Country tramite una rete neurale, dove per capi si intendono pantaloni ,giacche, guanti, cappelli, gilet, scarpe, accessori, giacconi, tops e gonne.
Il software risultante basa il suo funzionamento su Tensorflow, una libreria di Google che facilita la creazione di reti neurali. Questa tecnologia è stata applicata ad immagini di moda riguardanti lo stile Country e l'obiettivo era quello di riuscire ad indetificare, all'interno di un'immagine, i vari indumenti Country presenti.
Lo sviluppo di questo applicativo mi ha permesso di studiare un argomento odierno come l'Intelligenza Artificiale, andando ad analizzare le relazioni che ci sono tra il campo del Machine Learning e quello della moda. Mondi che di per se sembrano distanti hanno in realtà molta complementarità: grazie allo studio di reti neurali sempre più affini a quella umana oggi è possibile abbinare due o più indumenti presenti nel nostro armadio attraverso una semplice applicazione. Spiegherò il funzionamento di questa e molte altre funzioni nel secondo capitolo, dopo aver introdotto, nel primo, l'argomento generale ed i concetti su cui si basa lo sviluppo di una rete neurale.
Dopo aver illustrato la teoria delle reti neurali e le varie applicazioni nel campo della moda, nel terzo capitolo presenterò il software sviluppato, soffermandomi sui vari steps svolti. Infine, attraverso la statistica descrittiva, illustrerò nella fine del terzo capitolo i

risultati ottenuti motivandoli tramite esempi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Machine Learning

Il lavoro di questa tesi si concentra nell'ambito dell'Intelligenza Artificiale e , più nello specifico , nel campo del Machine Learning. Spesso il Machine Learning viene utilizzato come sinonimo di Intelligenza Artificiale,ma in realtà sono due concetti differenti. L'intento di questo primo capitolo è andare a spiegare l'ambito e le tecniche toccate in questo lavoro , dando una definizione iniziale dei concetti per poi andarli ad esplicare attraverso esempi più concerti e riguardanti la vita reale.
In particolare vengono introdotti e discussi i seguenti argomenti : Machine Learning, Deep Learning e Reti Neurali.

1.1 Analisi del Machine Learning

Oggi,per Intelligenza Artificiale, ci si riferisce a quel campo della Scienza che ha lo scopo di creare sistemi che siano in grado di pensare come un essere umano, ovvero che possano simulare azioni compiute quotidianamente. L'acquisizione di tale pensiero potrà poi portare il sistema in questione a risolvere problemi piu ardui con maggior velocità rispetto a un essere umano. Possiamo vedere infatti questa disciplina come una statistica

applicata dove l'acquisizione sempre maggiore di informazioni da parte della macchina porterà allo sviluppo di una sua conscenza interna , il più possibile simile a quella umana. L'Intelligenza Artificiale è formata da vari ambiti,ed uno di questi è appunto il Machine Learning. Il Machine Learning **(ML)** indica un ambito di ricerca all'interno dell'Intelligenza Artificiale volto a realizzare sistemi in grado di appprendere autonomamente senza la necessità di essere specificatamente programmati per svolgere tale compito.
Tale passaggio introduce una nuova visione di programmazione , nella quale l'obiettivo non è più dare in pasto un input per ricevere un output dalla macchina , bensì far acquisire a quest'utlima sempre più consapevolezza delle istruzioni che ha eseguito in modo tale che giunga al risultato prendendo decisioni autonomamente.

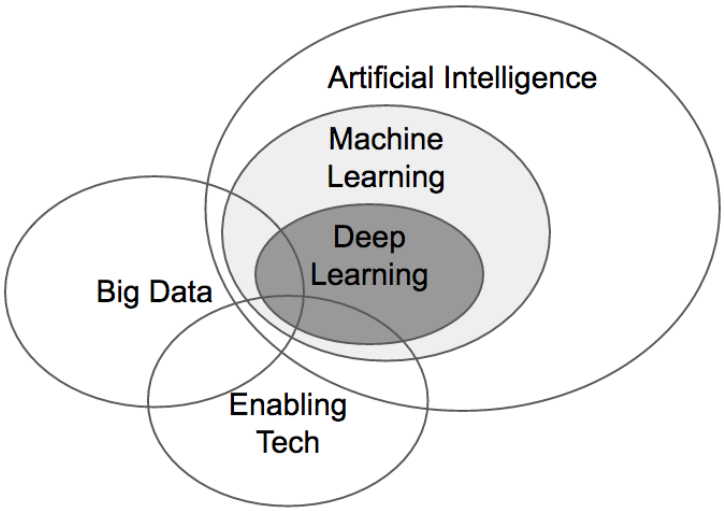

Figura 1.1: Schema generale Intelligenza Artificiale

La definizione di Machine Learning venne data da Herbert Simon nel 1959:
"Il Machine Learning è quella branca dell'informatica che permette ad una macchina di imparare ad eseguire un compito senza essere stata esplicitamente programmata per farlo".

Un'altra definizione più tecnica venne data da Tom Mitchell nel 1997 :
"Si dice che un programma apprende dall'esperienza E con riferimento ad alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P,migliorano con l'esperienza E".
Detto in altre parole è lo stesso procedimento per il quale il bambino imapara a parlare: dall ' esperienza del mondo circostante apprende le informazioni necessarie ; la misurazione della performance viene fatta dai genitori che incitano il bambino a pronunciare parole semplici(come "mamma" o "papà") ;l' acquisizione di informazioni migliorerà sempre più l' esperienza del bambino il quale imparerà a parlare. Ciò che fa infatti è ricavare regole generali **(modelli)** dall' iterazione col mondo esterno. Per questo motivo l'apprendimento è un processo iterativo, che si ripete e migliora la conoscenza all'aumentare delle informazioni che vengono raccolte. Più si fa esperienza e più si impara.
Nel Machine Learning si tenta di ricreare questo ti po di processo. Alla macchina viene fornito un:

- **algoritmo di apprendimento** , ovvero un programma che descriva un modello di apprendimento

- **dati** che possono essere strutturati (ovvero organizzati intabelle chiave-valore) o non-strutturati (testi, audio, immagini ecc..).

L' algoritmo contiene anche lo scopo dell' analisi dei dati , ovvero cosa ci aspettiamo che la macchina apprenda. I tipi di problemi per i quali viene applicato il **ML** sono principalmente :

- **CLASSIFICAZIONE**: quando è necessario decidere a quale categoria appartiene un determinato dato.

- **REGRESSIONE**: prevedere il valore futuro di un dato avendo noto il suo valore attuale. Un esempio è la previsione della quotazione delle valute o delle azioni di una società.

- **RAGGRUPPAMENTO**: quando si vuole raggruppare i dati che presentano caratteristiche simili.

Il Machine Learning funziona in linea di principio sulla base di due distinti approcci, identificati da Arthur Samuel alla fine degli anni '50, che permettono di distinguere l'apprendimento automatico in due sottocategorie del Machine Learning a seconda del fatto che si diano al computer *esempi* completi da utilizzare come indicazione per eseguire il compito richiesto **(apprendimento supervisionato)** oppure che si lasci lavorare il software senza alcun "aiuto" **(apprendimento non supervisionato)**.

1.1.1 Apprendimento supervisionato

Con questa terminologia si intende la metodologia di apprendimento automatico in cui alla macchina vengono passati degli esempi composti da una coppia di dati contenenti il dato originale e il risultato atteso. L'obiettivo di questo approccio è trovare una regola (o detto più precisamente **modello**) che individui una relazione fra dato e risultato cosicchè, al presentarsi in futuro di circostanze simili, la macchina sappia quali regole applicare per ottenere un determinato risultato.
Affinchè la macchina riesca a trarne una regola il dato deve essere *etichettato*, ovvero gli si deve attribuire una caratteristica dalla quale ne verrà dedotta una regola. Questa tipologia di ML è applicata nei probemi di Calssificazione.

1.1.2 Apprendimento non-supervisionato

A differenza del precedente approccio, nell'apprendimento non-supervisionato non abbiamo dati etichettati da cui ricavare una funzione-modello, di conseguenza non abbiamo nessun riscontro per testare la veridicità del risultato. Alla macchina viene chiesto, quindi, di estrarre una regola che raggruppi i casi presentati secondo caratteristiche che ricava dai dati stessi. Per questo è anche definito apprendimento di caratteristiche (feature learning). Gli algoritmi in questo caso cercano una relazione tra i dati per capire se e come essi siano collegati tra di loro. Non contenendo alcuna informazione preimpostata, l'algoritmo è chiamato a creare "nuova conoscenza" (knowledge discovery). L'applicazione dell'apprendimento non-supervisonato avviene per problemi di Raggruppamento, nei quali è appunto la macchina a dover cercare relazioni fra dati per scaturirne nuove conoscenze e viene utilizzato soprattutto nei *Big-Data*.

1.1.3 Aprendimento per rinforzo

Un altro caso di apprendimento non citato precedentemente è quello per rinforzo : in questo caso il sistema deve interagire con un ambiente dinamico (dal quale riceve dati di input) per raggiungere un obiettivo; se tale obiettivo viene raggiunto il sistema in questione riceve una ricompensa , altrimenti subisce una punizione.
Detto in altri termini, è lo stesso procedimento per cui si impara a guidare : commettendo gli errori si aprende cosa non bisogna fare , migliorando così le prestazioni per raggiungere poi l'obiettivo finale. Attarverso questa tipologia di apprendimento AlphaGo, il software sviluppato da Google Deepmind, ha battuto il 18 volte campione internazionale di scacchi Lee Se-Dol.
Ci sono poi altre sottocategorie di Machine Learning che non indicano un metodo di ap-

prendimento, bensì rappresentano dei metodi attraverso i quali si rappresentano i dati. Uno di questi è sicuramente **l'albero decisionale**, il quale mira a migliorare la probabilità di un risultato : tale tecnica è la base di modelli predittivi nei quali è possibile scoprire le conseguenze (output) di determinate decisioni (input).
Altro esempio concreto viene dal **clustering**, ossia dai modelli matematici che consentono di raggruppare dati, informazioni, oggetti,"simili"; si tratta di un applicazione pratica del Machine Learning dietro alla quale esistono modelli di apprendimento differenti che vanno dall'identificazione delle strutture (cosa definisce un cluster e qual è la sua natura) al riconoscimento degli "oggetti" che devono far parte di un gruppo piuttosto che di un altro.
Un' ulteriore sottocategoria è quella dei **modelli probabilistici** che basano il processo di apprendimento del sistema sul calcolo delle probabilità. Il più noto modello probabilistico è sicuramente la Rete di Bayes.

1.1.4 Applicazioni odierne

Il Machine Learning è utilizzato per molteplici scopi in cui è richiesto alla macchina di prendere delle decisioni sui dati e viene applicato in diversi ambiti , dalla medicina al marketing e le sue applicazioni sono già oggi molto numerose, alcune delle quali entrate comunemente nella nostra vita quotidiana. L'esempio più consono è il **motore di ricerca** :attraverso una o più parole chiave, questi motori restituiscono liste di risultati (le cosiddette SERP – Search Engine Results Page) che sono l'effetto di algoritmi di Machine Learning con apprendimento non supervisionato. Altro esempio comune è quello dei **filtri spam delle e-mail** : questo meccanismo si basa sull'intercettazione ed il riconoscimento di posta elettronica fraudolenta. Analogamente modelli di questo tipo vengono utilizzati nella nel settore Finance per la prevenzione delle frodi (come la clonazione della

carta di credito), dei furti di dati e identità. Un altro modello di apprednimento che sta avendo molto successo nel campo automoblistico è quello di rinforzo , sul quale si basano applicazioni come la **guida autonoma** ed il **riconoscimento vocale**.

1.2 Deep Learning

La traduzione letterale del termine consiste in **"apprendimento approfondito"** : il Deep Learning infatti è una disciplina del Machine Learning che ha lo scopo di creare modelli di apprendimento su più livelli .

Prendiamo ad esempio l'esposizione di una nozione : il nostro cervello la apprende e subito dopo ne espone un'altra poichè ha raccolto gli input dalla prima , li ha elaborati e processati per produrne una successiva , aumentando sempre più il livello di astrazione. Tale tecnica infatti trae spunto dai procedimenti di apprendimento del nostro cervello e , in particolare , alle relazioni tra i neuroni nell' elaborare i dati che ricevono dagli stimoli esterni. L'apprendimento così sviluppato ha la forma concettuale di una piramide : **i concetti più alti sono appresi a partire da quelli più bassi**, poichè ogni volta aggiungiamo sempre più dati in input che vanno a modellare il nostro apprendimento e livello di elaborazione.

Il Deep Learning ci permette di ottenere una macchina che riesce a classificare i dati in entrata (input) e quelli in uscita (output), evidenziando quelli importanti ai fini della risoluzione del problema e scartando quelli che non servono.La rivoluzione apportata dal Deep Learning è tutta nella capacità, simile a quella umana, di elaborare i dati, le proprie conoscenze a livelli che non sono affatto lineari. Grazie a questa facoltà, la macchina apprende e perfeziona funzionalità sempre più complesse. Per ottenere un approccio simile a quello umano il Deep Learning si ispira alle **reti neurali artificiali**.

1.3 Reti Neurali Artificiali

1.3.1 Definizione

Le Reti Neurali Artificiali sono modelli di calcolo matematico-informatico basate sul funzionamento delle reti neurali biologiche,ossia modelli costruiti da interconnessioni di informazioni.

Questi modelli matematici sono composti da neuroni artificiali che si ispirano al funzionamento biologico del cervello umano. I neuroni biologici infatti formano le nostre reti neurali celebrali, ovvero quelle che permettono a ciascun individuo di ragionare, fare calcoli, riconoscere volti, agire ecc.. L'idea alla base della Rete Neurale Artificiale (ANN) è quella di replicare artificialmente il cervello umano simulandone il funzionamento.

1.3.2 Modello di riferimento: le reti neurali biologiche

Il sistema nervoso umano è formato da 3 stadi principali:

- **rete neurale**: riceve continue informazioni, le percepisce e fa appropriate decisioni.
- **recettori**: convertono gli stimoli esterni in impulsi elettrici che vengono inviati alla rete neurale.
- **attuatori**: convertono gli impulsi elettrici generati dalla rete neurale in risposte al sistema esterno

La componente principale sono ovviamente i **neuroni** : si stima che nel cervello umano siano presenti 1 miliardo di neuroni; ogni neurone è collegato ad altri 1000 neuroni attraverso un prolungamento chiamato assone nel quale vengono trasmesse le informazioni in uscita del neurone, mentre quelle in entrata vengono trasmesse attraverso i dentriti.Questi dati vengono poi processati dai somi neuronali,ossia i corpi dei neuroni,

che corrispondono all'unità di calcolo del sistema; essi infatti ricevono e processano le informazioni e, nel caso in cui il potenziale di azione in ingresso superi un certo valore, generano a loro volta degli impulsi in grado di propagarsi nella rete. All'interno dei somi troviamo i neurotrasmettitori, che sono composti biologici di diverse categorie(ammine,peptidi, aminoacidi) sintetizzati nei somi e responsabili della modulazione degli impulsi nervosi. Un' altra componente biologica molto importante della rete neurale sono le sinapsi, ovvero i siti funzionali ad alta specializzazione nei quali avviene lo scambio di informazioni fra neuroni. Ci sono due tipologie di sinapsi :

- **eccitatore**: favoriscono la generazione di elettricità nel neurone postsinaptico.
- **inibitore**: inibiscono la generazione di elettricità nel neurone postsinaptico.

Ogni neurone può ricevere simultaneamente segnali da diverse sinapsi: questa ricezione viene poi misurata in termini di potenziale elettrico in modo da stabilire se è stata raggiunta la soglia di attivazione per genereare ,a sua volta, un impulso nervoso. Quest'ultima proprietà viene implementata anche nelle ANN ed è il fulcro dell'apprendimento: la configurazione sinaptica all'interno di ogni rete neurale biologica è dinamica, ciò significa che il numero di sinapsi può incrementare o diminuire a seconda degli stimoli che riceve la rete, ed è lo stesso principio adottato dalle reti neurali artificiali, le quali hanno un atteggiamento adattivo a seconda dell'utilità o meno dell'informazione.

1.3.3 Neurone di McCulloch-Pitts

Nel 1943 gli scenziati McCulloch e Pitts proposero un primo modello rudiemntale di Rete Artificiale, da cui nacque la denominazione di "neurone matematico".
Il neurone è è l'unità di calcolo fondamentale della rete neurale ed è formato da 3 elementi di base nel modello neurale:

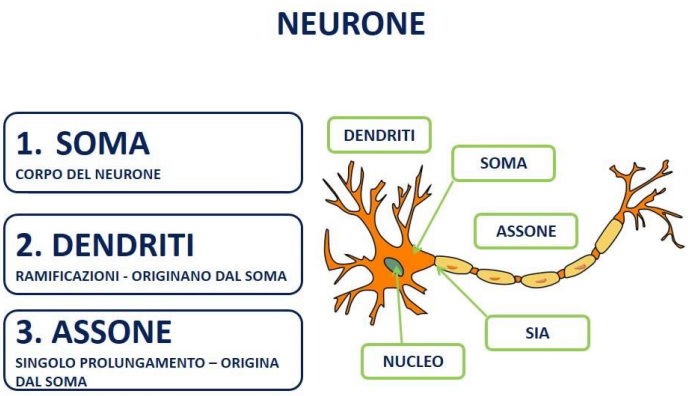

Figura 1.2: neurone biologico

1. **insieme di sinapsi** ,ognuna caratterizzata da un peso; a differenza del modello umano, quello artificiale può avere pesi sia positivi che negativi.

2. **un sommatore** che somma i segnali in input pesati dalle rispettive sinapsi, producendo in output una combinazione lineare degli input.

3. **una funzione di attivazione** per limitare l'ampiezza dell'output di un neurone. Tipicamente per comodità l'ampiezza degli output appartengono all'intervallo [0,1] oppure [-1,1].

Il neurone inoltre include un **valore di soglia** che ha l'effetto, a seconda della sua positività o negatività, di aumentare o diminuire l'input per la funzione di attivazione.

In termini matematici, il modello può essere descritto così:

$u_{\mathrm{k}} = \sum_{j=1}^{m} w_{\mathrm{k}} i * j x_{\mathrm{j}}$

$\mathrm{y}_{\mathrm{k}} = \phi\ (\mathrm{u}_{\mathrm{k}} + \mathrm{b}_{\mathrm{k}})$ dove:

- x_{j} sono i pesi sinaptici del neurone k;

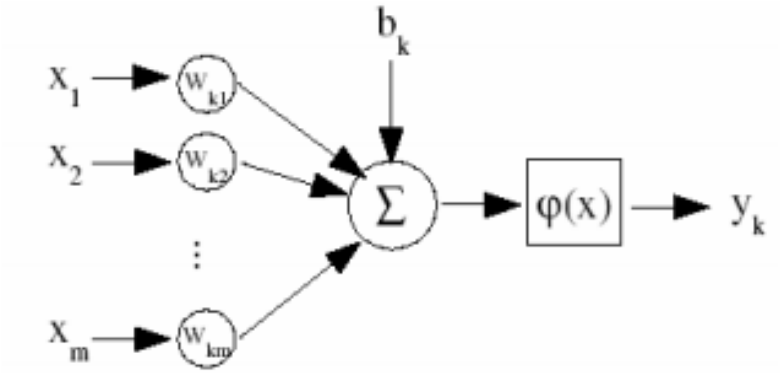

Figura 1.3: neurone matematico

- u_k è la combinazione lineare degli input nel neurone k;
- b_k è il valore soglia del neurone k;
- ϕ (x) è la funzione di attivazione;
- y_k è l'output generato dal neurone k

Le funzioni principali di attivazione della soglia sono 3:

- **Funzione Threshold**

  ϕ (v)=$\begin{cases} 1 sev > 0 \\ 0 sev < 0 \end{cases}$

- **Funzione sigmoide**

  ϕ (v)=$\begin{cases} 1 sev > 1/2 \\ vse - 1/2 < v < 1/2 \\ 0 sev < 1/2 \end{cases}$

- **Funzione linearea tratti**

  ϕ (v)=$\frac{1}{1+e^{-}av}$

Dal '43 alla fine degli anni '50 accadde però ben poco; ci fu un tentativo nel 1949 da parte dello psicologo canadese Donald Olding Hebb di spiegare i modelli complessi del cervello e di estrapolarne le prime ipotesi di apprendimento delle reti neurali .Famoso è il suo postulato chiamato appunto il **Postulato di Hebb sull'apprendimento**: egli afferma che quando l'assone di un neurone A è abbsatanza vicino o da eccitare un neurone B e questo, in modo ripetitivo e persistente, gli invia un potenziale di azione, inizia un processo di crescita in uno o entrambi i neuroni tali da incrementare l'efficienza di A.

1.3.4 Percettrone e MLP

L'importante svolta si ebbe nel 1958 con Frank Rosenblatt il quale presentò il primo schema di rete neurale, il **Percettrone**: tale modello è la forma più semplice di rete neurale usata per classificare pattern linearmente separabili, ovvero piani collocati ai lati opposti di un iperpiano.Il Percettrone consiste in un singolo neurone con soglia e pesi modificabili. L'algoritmo di apprendimento si basa sul *Teorema di convergenza del percettrone* ed il modello di rete è feedforward (gli impulsi si propagano in un unica direzione,come vedremo successivamente). La restrizione del Precettrone era il campo di applicazione :esso infatti si limitava a riconoscere forme,classificarle in gruppi separati e calcolare semplici funzioni (come gli operatori booleani AND,OR e NOT, poichè linearmene separabili).

Il passo successivo a questo modello è stato il **Percettrone Multistrato (MLP)**, in grado di calcolare funzioni non linearmente separabili (come lo XOR). Il MLP consiste in un insieme di ingressi (input layer), uno o più strati nascosti di neuroni (hidden layers) e un insieme di neuroni di uscita (output layer).Le caratteristiche principali di questo tipo di rete sono che:

- ogni neurone include una funzione di attivazione non lineare differenziabile.

- la rete contiene una o più strati nascosti.
- la rete ha un alta connettività.

In questo tipo di rete ogni neurone nascosto o d'uscita di un multilayer perceptron esegue 2 computazioni:

1. **computazione del segnale di funzione**,ovvero il calcolo della funzione di attivazione.
2. **computazione di un vettore gradiente**, ovvero il calcolo per l'aggiornamento dei pesi sinaptici.

Il lavoro di Rosenblatt,criticato da J. Von Neuman nell'opera "The computer and the brain", viene ripreso e perfezionato da Hinton e Williams nel 1986 con **l'algoritmo di retropropagazione dell'errore** : in questo metodo di apprendimento si aggiornano ciclicamente i pesi sinaptici e,al termine di ogni ciclo, si cerca di limare sempre più la differenza che c'è tra il risultato ottenuto e quello auspicato.Le interazioni si arrestano quando la procedura converge, ovvero quando la discrepanza scende al di sotto di un limite prefissato, anche se tale scenario non è sempre garantito.Al termine della fase di apprendimento, alla rete viene richiesto di risolvere un problema reale . Col passare degli anni molti matematici e fisici si sono adoperati nell'ideare l'architettura della rete neurale: possiamo raggruppare gli sviluppi nelle seguenti tipologie di architetture.

1.3.5 Tipologie di architetture della ANN

La tipologia di architettura di rete è strettamente dipendente dall'algoritmo di apprendimento che si vuole utilizzare; in particolare vengono identificati 3 tipologie di reti:

a) **Reti feedforward ad uno strato**: in questa forma di rete ci sono nodi di input (input layer) ed uno strato di neuroni (output layer). Il segnale della rete si propaga in avanti, partendo dal layer di input e terminando in quello di output.

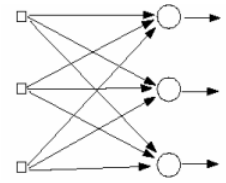

Figura 1.4: Reti feedforward ad uno strato

b) **Reti feedforward a più strati**:questo modello si differenzia dal precedente per la presenza di uno o più strati di neuroni nascosti (hidden layers) tra lo strato di input e quello di output. In quest'architettura aumentano le iterazioni fra neuroni.

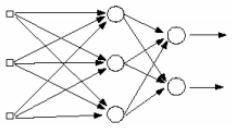

Figura 1.5: Reti feedforward ad più strati

c) **Reti ricorrenti o feedback**: questa tipologia di rete ciclica venne sviluppata nel 1982 dal fisico Hopfield; viene definita ciclica poichè le informazioni fra nodi viaggiano in qualsiasi direzione.La presenza di cicli ha un impatto profondo sulle capacità di apprendimento della rete e sulle sue performance, poichè viene ulteriormente ampliato il campo delle applicazioni del modello. I principali problemi della rete di Hopefield sono che le soluzioni proposte sono difficilmente ammissibili.

Sempre nel 1982 Kohoen sviluppa un tipo di rete neurale chiamata **SOM (Self-Organizing Maps)** sia a feedforward che a feedback: la caratteristica principale

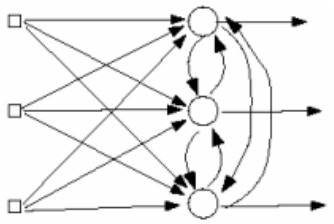

Figura 1.6: Reti ricorrenti o feedback

di tale architettura è modificare la configurazione della mappa dei nodi in base al peso che gli viene assegnato man mano che si accumulano input. In questo modo i nodi con pesi simili si avvicinano, mentre quelli con pesi molto diversi si allontanano.

Nel 1990 viene proposta una variante della MLP da Elman,ovvero una rete ricorrente bidirezionale,con la differenza che viene aggiunto un gruppo di nodi con lo scopo di conservare la configurzione della precedente mappa della rete. Questa modifica comportò miglioramenti nel calcolo delle sequenze temporali.

L'aggiornamento più vicino ai giorni nostri è rappresentato dal lavoro dell' IBM riguardo lo sviluppo di reti neurali basate su materiali a cambiamento di fase; tale modello consisite in un cambiamento dello stato fisico dell' hardaware a seconda dell'impulso elettrico ricevuto: quest'ultimo infatti può provocare una cristallizzazione del materiale o renderlo amorfo, analogamente a ciò che avviene nella rete biologica. Altro lavoro rilevante è sicuramente quello della Intel con il **chip neuromorfici**,ovvero microprocessori che non compiono calcoli con un approccio binario,bensì si scambiano i vari pesi dei segnali. Il prototipo più recente (ancora in fase di sviluppo) si chiama Loihi e contiene al suo interno 130mila neuroni e 130 milioni di sinapsi che gli permettono di apprendere in tempo reale, sulla base dei feedback ricevuti dall'ambiente.

1.3.6 Tipologie di strati

In generale, le Reti Neurali sono formate da 3 tipi di strati principali:

a) **strato degli ingressi (Input)**: ha il compito di ricevere ed elaborare i segnali in ingresso adattandoli ai neuroni che compongono la rete.

b) **strato H (strato nascosto)**: la sua mansione è quella di caricare il processo di elaborazione (solitamente è strutturato in sotto-livelli).

c) **strato di uscita (Output)**: in questo strato vengono raccolti i risultati dello strato H e vengono adattati ale richieste del livello successivo della rete neurale.

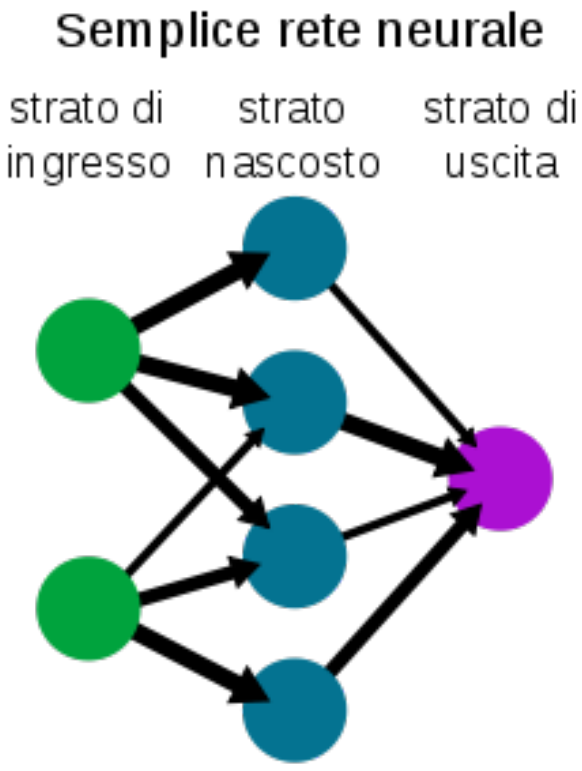

Figura 1.7: neurone biologico

1.3.7 Difetti della Rete Neurale

Sebbene i miglioramenti delle Reti Neurali siano continui, ci sono alcuni limiti di cui è difficile prevedere la tempistica di risoluzione e miglioramento. Il difetto maggiore riscontrato finora è il **black box**, termine inglese che indica la difficoltà di poter analizzare la computazione delle ANN: sono in grado di fornire output corretti, o sufficientemente corretti, ma non permettono di esaminare i singoli stadi di elaborazione che li determinano. Altro scoglio rilevante è l'incertezza di giungere ad un risultato finale , qualora fosse fornito un output, spesso non rappresenta la soluzione perfetta. Infine si può affermare che il periodo di "apprendimento" sia, almeno per ora, molto lungo e difficilmente prevedibile, anche se si spera che con l'utlizzo di nuove tecnologie sarà possibile ovviare questo problema.

Capitolo 2

Moda e Machine Learning

La facilità con cui possiamo trovare una vasta disponibilità di dataset relativi ad immagini ci permette di sfruttare diverse potenzialità di iterazione con la macchina . In questo capitolo verranno illustrati alcune tipologie di sistema che processano immagini inerenti al campo della moda che si basano sul Compter Vision (che verrà approfondito seguentemente): in particolare il focus saranno gli "Style Recommendation Systems", ovvero sistemi basati su modelli di apprendimento in grado di consigliare all'utente outfits ed abbinamenti di vestiti.

2.1 Computer Vision

Come citato precedentemente,il Computer Vision è quella materia che ci permette di acquisire una fotografia bidimensionale e di interpretare il contenuto di quest'ultima per estrapolarne infine delle informazioni. Lo scopo ultimo del Computer Vision è quello di cercare di riprodurre la vista umana: l' uomo infatti guarda un' immagine e sa quali oggetti la compongono ,la loro locazione e come interagiscono. Intuitivamente questa disciplina è strettamente collegata con l'Intelligenza Artificiale ed il Machine Learning.

Possiamo vedere infatti il Computer Vision come un punto d'incontro tra Intelligenza Artificiale, Machine Learning e reti neurali: il Computer Vision prende in ingresso l' immagine la quale, attraverso le reti neurali, viene scansionata e associata a delle informazioni che permetteranno l' apprendimento della macchina.
Il Computer Vision nasce a partire dagli anni '60 per poi svilupparsi più concretamente negli anni '80,grazie all'introduzione di sistemi per applicazioni industriali.

2.1.1 Campi applicativi

La tematica più frequente per la quale viene applicato il Computer Vision è cercare di comprendere se l'immagine contiene o meno un determinato oggetto o attività; questo tipo di problema viene chiamato **Object Recognition**, il quale potrebbe essere definito come un processo di allenamento di un classificatore finalizzato al riconoscimento della categoria a cui l'oggetto appartiene.
La categorizzazione di un oggetto passa per due punti cruciali:

a) **elaborazione dell' immagine**

b) **estrazione features**, ossia informazioni dell' immagine in questione che siano rilevanti per quanto riguarda l'apprendimento.

. L' elaborazione delle immagini(o image processing) è composta da due operazioni principali, ovvero l'Edge-detector e la Segmentazione.
La prima operazione consiste in un eliminazione di quelli che sono i dettagli meno significativi al fine del riconoscimento dell' oggetto, lasciando inalterata la fisionomia geometrica dell' immagine.

Figura 2.1: Esempio di Edge-detection

L' operazione successiva è la Segmentazione, che consiste nel partizionare l' immagine data in regioni disgiunte ed omogenee, dove per omogenee si intendono parti di immagine in cui i pixel sono monocromatici.

Figura 2.2: Esempio di Object-Segmentation

Una volta che l' immagine è stata elaborata si passa all'estrazione delle features che, come già detto, hanno il compito di individuare le caratteristiche peculiari dell' oggetto.Per estrarre informazioni vengono applicate diverse tipoloige di algoritmi tra cui i più rilevanti sono SIFT e SURF; le informazioni sono estraibili anche dalle Deep Neural Networks e, in particolare, dalle Convolutional Neural Network (CNN) nelle quali l'estrapolazione dell' informazione viene attuata attraverso i layers che compongono la rete.

Le features possono essere di due tipologie:

- Features globali: rappresentano la globalità dell' oggetto e non dipendono dalle forme (inefficaci).

- Features locali: sono le caratteristiche dell' immagine rilevabili (efficaci per l' identificazione dell' oggetto).

Altro campo di applicazione del Computer Vision è sicuramente l' **Object Detection**. Questa tecnologia si occupa di rilevare istanze di oggetti semantici in una detereminata classe(come ad es. animali, esseri umani, edifici ecc..) dove ogni istanza ha le sue caratteristiche specifiche che contribuiscono ad identificare la classe (ad es. in ogni cerhio il centro è equidistante da qualsiasi punto della circonferenza). Tuttavia non bisogna confondere la classificazione delle immagini con l' Object Detection: in generale, se si desidera classificare un' immagine in una determinata categoria, si utilizza la classificazione;se invece si vuole identificare la posizione degli oggeti in un' immagine si ususfruisce dell' Object Detection. In realtà le due tecniche sono diverse ma non contrapposte: mol-

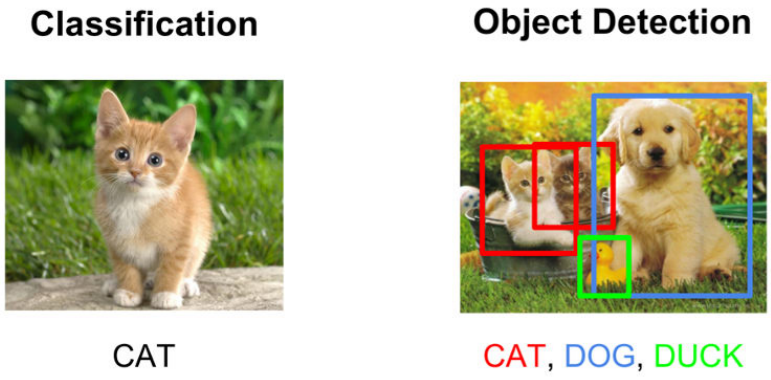

Figura 2.3: Differenze tra Classificazione ed Object Detection

to spesso infatti si utilizza la Classificazione per generare le informazioni dell' intera immagine e successivamente, con l' Object Detection, si riduce sempre più la regione dell 'immagine, fino ad ottenere un riquadro ben specifico al quale viene associata un *etichetta*. Ovviamente più c'è disponibilità di immagini da cui ricavare informazioni/etichette, migliore sarà l' apprendimento del modello: ultimamente questo problema è poco rilevante, poichè oggigiorno siamo in grado di trovare ingenti quantità di immagini relative a moltissime categorie.
Una categoria interessata a questa branca del Machine Learning è sicuramente la moda e l' incontro fra i due ambiti ha dato origine a quelli che vengono chiamati **SystemRecommendation**.

2.2 System Recommendation

I System Recommendation non sono altro che modelli di apprendimento il cui intento è quello di riuscire non solo a classificare le diverse tipologie di vestiario (pantalone,camicia,scarpe,capotto,sciarpa ec...), bensì essere in grado di catalogare i capi di abbigliamento in base allo stile di appartenenza (Rock,Calssico,Hypster,Casual ecc...) ed accoppiarli di conseguenza.
Riportato all' esperienza di vita reale il System Recommendation corrisponderebbe allla commessa che ci assiste nel momento in cui compriamo un indumento; Recommendation infatti ,in inglese, significa "raccomandazione, consiglio" ed è appunto ciò che il sistema in questione deve essere in grado di fare : in base al nostro stile ed ai nostri gusti deve essere abile nel consigliarci un capo piuttosto che un altro e,fattore ancor più rilevante, nel momento in cui scegliamo un determinato abito deve essere capace di suggerirci qualcosa da abbinarci, come farebbe una commessa di un negozio.

I System Recommendation sono alla base di siti web ed e-commerce, il cui obiettivo è quello di minimizzare tempi di overload e spingere il cliente a comprare oggetti complementari a quello scelto. Un esempio calzante è sicuramemte Netflix, il quale consiglia film che siano simili a quelli visti precedentemente dall'utente;stesso discorso per Amazon e Pandora, la quale consiglia musica basandosi sui brani ascoltati precedentemente dallo user. Per quanto questi sistemi lavorino bene per oggetti come libri, film e musica, finora restituiscono soluzioni non soddisfacenti per quanto riguarda articoli come vestiti. In questo caso infatti ciò che è ancora difficile da "processsare" per il modello è il fattore estetico di un capo d' abbigliamento: una gonna potrebbe risultare un abbinamento migliore con quella t-shirt piuttosto che un altra;di conseguenza, la decisione di comprare o meno un indumento dipende non solo dall'articolo stesso, bensì anche da quanto è compatibile o meno con altri capi.
Perciò l'obiettivo del System Recommendation non è solamente quello di saper consigliare l'utente, bensì di poter mostrare più scenari combinativi tra capi d'abbigliamento, basandosi su criteri sia estetici sia di altro ambito.

2.2.1 Problemi ricorrenti

Come accennato precedentemente,il System Recommendation non è in grado di dare attualmente soluzioni abbastanza soddisfacenti; i problemi che causano quest' incompletezza sono stati divisi in due filoni principali, ovvero la *costruzione del modello* e la ricerca e la *classificazione di combinazioni basate sulla qualità*. Le difficoltà riguardo la costruzione del modello derivano soprattutto dalla rappresentazione di abiti come *visual features* : è infatti in base alle caratteristiche dei singoli indumenti che il modello potrà cercare di abbinare quest' ultimi. Molto spesso infatti è difficile estrarre features dall' immagine per problemi inerenti la pulizia del background dell' immagine e la gestione

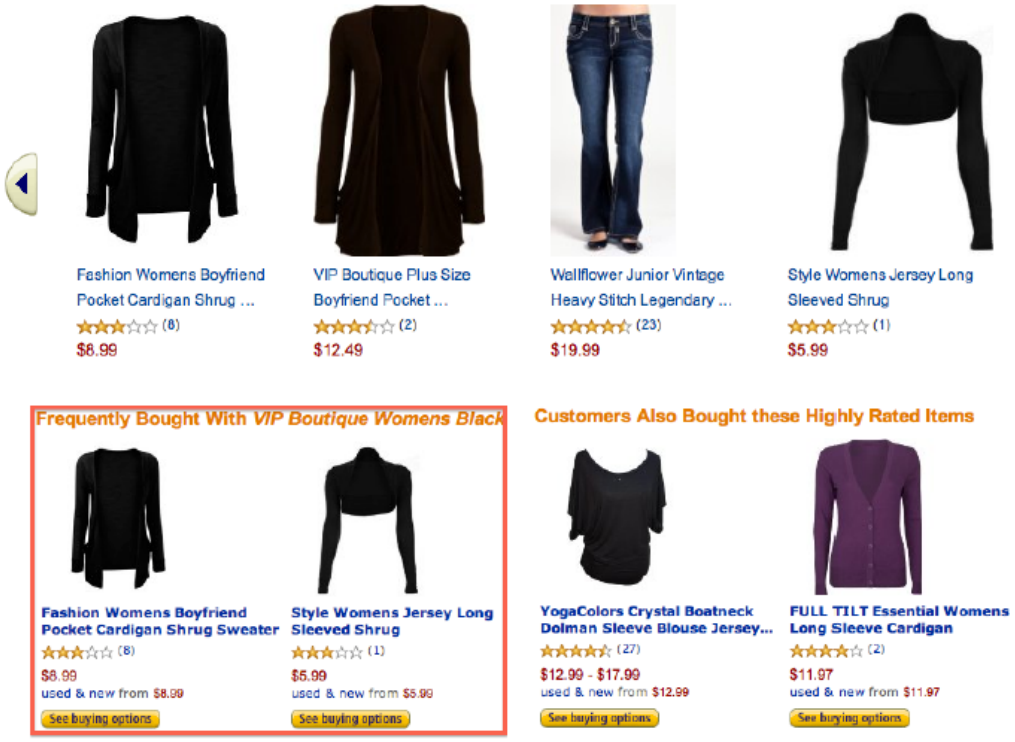

Figura 2.4: Esempio di System Recommendation su Amazon

della memoria di cui necessitiamo per permettere queste operazioni.

Altro problema non di poco conto è la person-detection, ovvero riusicre ad individuare la sagoma della persona nell' immagine.

2.2.2 Algoritmi di System Recommendation

Per risolvere questo problema verranno proposti alcuni algoritmi. Un tipo di approccio potrebbe essere dividere il dataset di n immagini in n_{train} per il training e n_{test} per il testing, dove ogni immagine è formata da più parti (come testa,gambe,braccia ec..). Assumendo che le parti vengano indicate con gli indici p=[1,2,3...,P], lo scopo dell' algoritmo è associare features ad ogni parte del corpo, anche quelle che sono nascoste. Queste ultime vengono indicate con $p_v = \{ 1,2,....,|p_v| \}$. Ipotizziamo che ogni parte j

dell' immagine i ha una parte descrittiva $h_ij \in \Re_K$, la parte descrittiva può riferirsi a colore,struttura o altre caratteristiche visuali. Possiamo concludere quindi che l'immagine i th può essre descritta come segue:

$H_i^T := [h_i1^T, h_i2^T, ..., h_iP^T]$.

Questo algoritmo viene chiamato **Tuned Perceptual Retrieval (PR)**.

Altro algoritmo utilizzato per ovviare il problema della person-detection è il **Complementary Nearest Neighbor Consensus (CNNC)** il cui scopo è lo stesso dell'algoritmo precedente, ovvero cercare di definire delle caratteristiche per tutte le parti che compongono la foto. Matematicamente la forumla base è la seguente:

$N_q = \text{argmin}\ K \sum j \in \text{pdist}(h_qj, h_ij)$.

Con questa equazione si cercano di raggruppare le immagini che hanno somiglianze a quella q data in input. Una volta che sono sate raggruppate le immagini simili l' obiettivo è definire features per le parti nascoste dell' immagine. A questo proposito vengono accumulate rappresentazioni delle parti nascoste come:

$\{ h_ij \mid N_q, j \in p_h \}$.

Un' altra metodologia di risoluzione del problema è il **Gaussian Mixture Model (GMM)**. Un modello di distribuzione gaussiana è una funzione di densità di probabilità parametrica rappresentata come una somma ponderata della densità dei componenti. GMM sono comunemente usati come modello parametrico della distribuzione di probabilità di misurazioni continue o caratteristiche in un sistema biometrico. L' obiettivo è quello di campionare, attraverso queste distribuzioni, le regioni di alta densità delle parti visibili per consigliare abiti da abbinare.

L' equazione della GMM è la seguente :

$p(H \mid \lambda) = \sum_{i=1}^{M} w_i g(H)$

In quest' equazione $g(H \mid u_i, \sum_i) = N(H \mid u_i, \sum_i)$, $\lambda = (u, \sum, w)$ sono i paramteri

della distibuzione Gaussiana multivariata.

Ipotizzando che il vettore dell' immagine query sia :

H ^{T} $_{q}$ =[h $_{q}$1 ^{T},h $_{q}$2 ^{T},.....,h $_{q}$N ^{T}]

con una parte mancante m, l' obiettivo è calcolare il valore più probabile h $_{q}$m della parte mancante.

Un metodo alternativo per il recupero di immagini utilizzando la struttura a cluster dei dati consiste nell' utilizzo di una struttura dati formata da parole nella quale si vanno a ricercare le combinazioni più ricorrenti (ad es. l' accoppiamento ricorrente di due colori). Questo tipo di struttura prende il nome di **topic model** ;un algoritmo che genera questa struttura è il **LDA** model di Markov Chain.

In quest' equazione le parole sono rappresentate da ω, i topics da θ ed α e β sono date da:

$p(\omega \mid \alpha , \beta) = \int p(\alpha|\theta) \prod_{n}^{i=1} \sum_{z_i} p(z_i|\theta)p(\omega|z_i, \beta)d\theta$

Ultima tecnica proposta è la **Texture Agnostic Retrival (TAR)** , basata sulle nozioni sui consumatori.

Per quanto riguarda invece la *classificazione delle combinazioni*, un problema sicuramente non trascurabile è la mancanza di nozioni basilari di cui noi usufruiamo quando ci vestiamo: i System Recommendation odierni non tengono conto di variabili come la stagione nella quale ci troviamo, il luogo in cui dobbiamo andare o magari l' occasione per la quale stiamo scegliendo il nostro outfit. Possiamo quindi descrivere formalmente il problema della classificazione come segue:

Dato un capo d'abbigliamento, trovare k combinazioni di alto livello che contengano l' articolo, dove la qualità è determinata dal modello di classificazione .

Tradotto in termini matematici protremmo descrivere il problema come segue:

Data un immagine i contenente uno o più articoli (occhiali,scarpe,gonne ecc...), la de-

scrizione H_i di un immagine è dato da $H_i{}^T$:[$H_i1{}^T$, $H_i2{}^T$,...., $H_iP{}^T$] $\in \Re{}^P K$, h_ij 1K= 1, $\forall$ j $\in$[1,2,....,P].

Dove P è il numero delle parti che costituiscono l'immagine, K è la grandezza del codice per rappresentare le parti e $h_ij \in \Re{}^K$ corrisponde la rappresentazione delle parti j dell'immagine i.

L' obiettivo è allenare un modello predittivo M(H,q), dove H=[H_1,H_2,.....,H_n].

Dato in input un vettore $H_q{}^T = [,h_q1{}^T,h_q2{}^T,.....,h_qP{}^T] \in \Re{}^P K$, con la mancazna di una colonna nella entry corrispondente alla parte m, il problema del modello di apprendimento consiste nel predire il valore della colonna mancante.

2.2.3 Retrieval System

Nella sezione precedente sono stati presentati algoritmi che avevano lo scopo di apprendere e trovare la parte mancante di un immagine data in input; se quindi avessimo un *descrittore* di una gonna, quale sarebbe la giusta camicia da abbinarci? La query, nel momento della risoluzione del problema,verrà mappata in modo tale da servire successivamente per recuperare immagini basandosi sul loro contenuto (CBIR).

Un esperimento è stato fatto con dataset "The Fashion-136K" utilizzato per il **Training**, creato per scansionare fotografie fashion da internet: nella stessa immagine abbiamo due grandi sottogruppi di vestiti, ovvero indumenti per la parte superiore del corpo *(top)* ed indumenti per la parte inferiore del corpo *(bottom)*, mentre le coordinate (x,y) che rappresentano la locazione spaziale del capo sono state annotate dalle *fashionistas* [1]. Analizzando il dataset sono emerse molte relazioni fra top e bottom come l'uso ricorrente di colori grigiastri o neri per top colorati; questi pattern verranno utilizzati successivamente per l'apprendimento del modello.

[1]Una persona che crea o promuove l' alta moda , ovvero un fashion designer o un fashion editor

La fase successiva al Training è stata quella del **Retrieval**, nella quale sono state prese immagini di vestiti bottom dal dataset "Fashion-1QK" utilizzate come queries per recuperare top dal dataset "The Fashion-136K".Il Retrieval è stato fatto secondo due criteri:

- **Ricerca di abbinamento di colori** (attraverso i pattern citati precedentemente)
- **Ricerca dei solid colors**, ovvero della tinta unita.

E' utile notare che attraverso questa procedura si allena il Retrieval System con immagini (del Fashion-1QK) che sono completamente differenti dall' altro dataset.
Il risultato dell' esperimento è una lista classificata di immagini disponibili data in input una query presa dal dataset Fashion-1QK.
Chi ha giudicato il risultato ottenuto per dare validità all' esperimento sono state le fashionistas. Poichè ogni query ha un elenco di 5 output possibili, sono stati mantenuti i primi risultati da ogni algoritmo utilizzato in modo tale da verificare la qualità del System Retrieval.Ad ogni fashionista è stata presentata un immagine query e una griglia di immagini conteneti i risultati per ogni algoritmo , chiedendo loro di valutare ogni soluzione di accoppiamento proposta con -1 (bad match) ,0 (neutral match), 1 (good match) o 2 (excellent match). Lo scopo di questo lavoro infatti è stato quello di studiare le performance degli algoritmi in base ai Retrieval differenti.
Nella ricerca della tinta unita è emerso che l' algoritmo CNNC ha ottenuto il miglior punteggio (0.48 su una scala da 0 a 1), seguito dal TAR (0.42);dopo il TAR l' algoritmo che ha totalizzato più punti è il GMM con 0.398, a seguire il PR con 0.383 ed infine il MCL. Per quanto riguarda la ricerca dell' abbinamento dei colori l' algoritmo più soddisfacente è stato il TAR (votazione di 0.40), seguito da GMM (0.31), CNNC (0.30) ed infine MCL e PR. Dovendo stipulare un sondaggio finale, è stato osservato che il CNNC

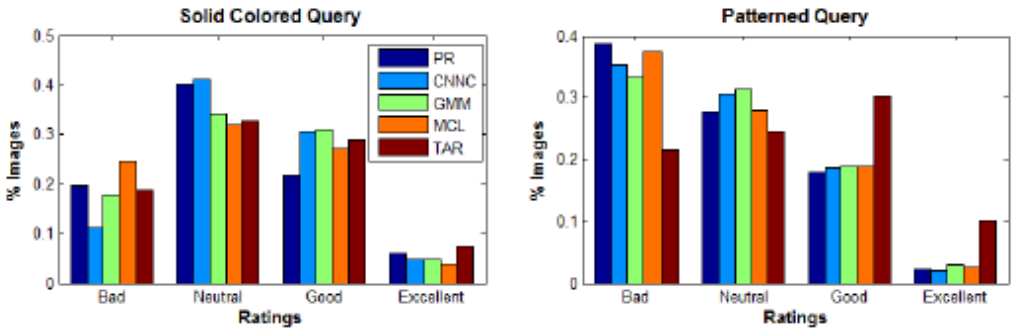

Figura 2.5: Risultati algoritmi System Retrieval

ha le performance migliori per quanto riguarda la tinta unita mentre il TAR è più consono per l' abbinamento. L'esperimento ha anche videnziato che le fashonistas preferiscono soluzioni di tinta unita poichè, nel caso in cui si considerino solid queries, la maggior parte dei voti si concentra sulla fascia neutral;quando invece si prendono in considerazione le pattern queries le fashionistas tendono a preferire abbinamenti cromatici deboli. Altra considerazione che is può dedurre da questo esperimento è la maggior importanza dell' abbinamento di colori rispetto all' omogneeità struttura dell' indumento (rombi,pois, ecc...).

2.3 CHIC

Un altro lavoro simile a quello visto precedentemente è quello di Manasi Vartak e Samuel Madden che, nel 2013, hanno presentato CHIC (Combination-Based Recommendation System).

CHIC è un Recommendation System costruito su una larga scala di dataset crowd-sourced e web-scraped: ciò permette al sistema di essere allargato a diverse categorie di dataset, a differenza degli altri che hanno una scalarità più limitata.

Il focus di CHIC è quello di produrre abbinamenti di alta qualità attrarverso 2 step

Figura 2.6: Nella figura sono state prese le migliori 3 opzioni consigliate dai vari algoritmi per la query "gonna".

crcuiali del processo:

- usare dataset per allenare il modello predittivo che calcolerà poi la qalità delle combinazioni.

- l' utilizzo dell' algoritmo C-Search per per computare e classificare efficientemente le combinazioni.

Per costruire il modello sono state estratte le caratteristiche in due livelli: nel primo vengono processate le immagini di capi d'abbigliamento individuali usando tecniche(il

background detection, *comutazione dell'istogramma dei colori* ecc...) per estrarre le features di base come il set di colori, la struttura dell' immagine e la forma (in seguito vengono estratte anche caratteristiche come il materiale dell' oggetto,prezzo ecc...).
Successivamente vengono processate combinazioni di indumenti per estrarre *hybrid features*; dopo questa fase si procede alla fase finale della costruzione , avendo collezionato un set di una cinquantina di hybrid features.
Le combinazioni trovate sinora sicuramente non rappresentano un alto livello di qualità, per questo viene eseguito un approccio pre-computazionale dove vengono valutate tutte le possibili combinazioni fra gli articoli (a discapito di un costo computazionale moto elevato). Tuttavia questo tipo di approccio è infattibile per alcune piattaforme online (solo per la categoria Tops Amazon possiede 60mila articoli); di conseguenza si cerca di ottimizzare il tutto salvando, insieme al top, anche il collegamento al capo più vicino all' interno del dataset;per questo, quando navighiamo, ci appaiono anche prodotti simili a quallo che abbiamo selezionato.

2.3.1 Ricerca e Classificazione

Dato un articolo in entrata come query,CHIC intraprende una ricerca interattiva per accumulare combinazioni. Supponendo che la query sia composta da un singolo capo i_Q non presente nel database,CHIC utilizza l'articolo più vicino a quello della ricerca per trovare l' immagine i_{DQ} nel database, ovvero l' immagine più approssimativa a quella ricevuta in input.
A seguire CHIC localizza coppie di articoli con alta qualità di macthing e le immagazzina in una lista nella quale la classificazione avviene attraverso un modello regressivo, fino a quando non si ottiene la combinazione migliore.
Infine la combinazione risultante viene computata, attraverso il modello di classificazione,

con la hybrid features presenti nel database per quell' abbinamento.

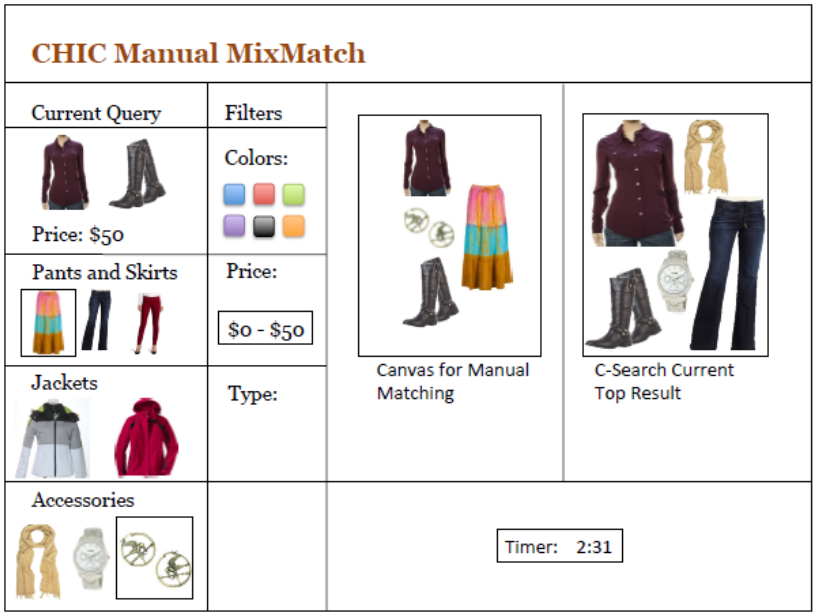

Figura 2.7: Interfaccia di funzionamento di CHIC

2.4 Heterogeneous Information Network

Un altro modello di System Recommendation è stato proposto da Hanbit Lee e Sang-go Lee della School of Computer Science and Engineering of Seoul.

Il loro datase è composto da 7458 insiemi contenente ognuna 2500 articoli, per un totale di 18499 capi d' abbigliamento presi da uno shopping mall online[2]. Gli insiemi consistono in vestiti,scarpe, accessori come mostrato nella tabella sottostante.

Sono state definite 4 macro-categorie dalle descrizioni degli indumenti: Catgeoria, come ad es. Cardigan,t-shirt,gonna,jeans ecc....; Pattern,ovvero lo stile ;Materiale e Colore. A questo dataset viene applicata una rete neurale composta da 6 nodi , in particolare **category**,**item**, **pattern**, **ensemble**, **color** e **material**.I collegamenti non etichettai

[2]Complesso di negozi, supermercati e sim. costruiti in un unici edificio

Attribute	Value Set
Category	Jacket, Suit, Coat, Shirts, T-Shirts, Sweater, Cardigan, Vest, Jeans, Slacks, Cargo, Baggy
Pattern	Striped, Checkered, Twisted, Printed, Dotted, Floral, Camoflage, Paisley, Herringbone
Material	Cotton, Leather, Denim, Wool, Linen, Suede, Corduroy, Fur, Spandex
Color	3000 color clusters

Figura 2.8: Organizzazione insiemi

rappresentano le associazioni fra nodi.

Il modello si basa sul concetto del *meta-path*, ovvero percorsi per esplorare reti di informazioni e studiarne lo schema. I meta-path possono fornire indicazioni per la ricerca e il mining della rete e aiutare ad analizzare e comprendere il significato semantico degli oggetti e delle relazioni nella rete.

Vengono utilizzati due tipi di meta-path:

item $\rightarrow$ X $\rightarrow$ item

item $\rightarrow$ X $\rightarrow$ item $\rightarrow$ ensemble $\rightarrow$ item $\rightarrow$ Y $\rightarrow$ item

dove X,Y $\in$ { CategoryPattern,Color,Ensemble,Material} utilizzando quindi 4+16=20 meta-paths. Questi meta-path ci permettono di associare capi d'abbigliamento che condividono lo stesso attributo X. Per es. se sostituiamo la X del secondo meta-path con Category un indumento "Jeans" potrebbe combaciare con un articolo "T-shirt" . L' apprendimento dei coefficienti di ogni meta-path è stato fatto con 8000 insiemi poichè il resto viene utilizzato per le valutazioni. In seguito per ogni insieme è stato scelto un capo come *target*, mentre gli altri sono usati come query.

Il coefficiente di ogni meta-path è allenato usando la regressione logistica : se il coeff. assume un valore negativo significa che gli articoli di quela categoria trovano raramente

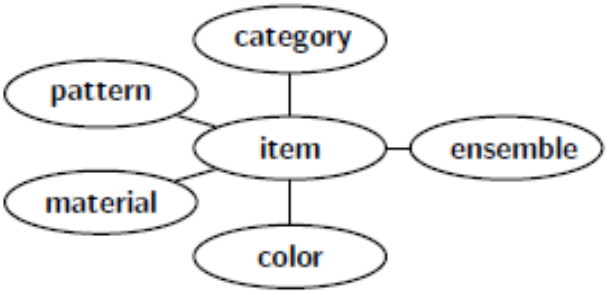

Figura 2.9: Rete neurale utilizzata

una combinazione con altri capi; nel caso invece di un coeff. positivo l' indumento trova frequentemente un altro capo con cui abbinarsi.

No.	Meta-path	Coefficient
(a)	$i \rightarrow c \rightarrow i$	-6.897
(b)	$i \rightarrow p \rightarrow i$	1.090
(c)	$i \rightarrow l \rightarrow i$	-3.041
(d)	$i \rightarrow c \rightarrow i \rightarrow e \rightarrow i \rightarrow c \rightarrow i$	2.088
(e)	$i \rightarrow p \rightarrow i \rightarrow e \rightarrow i \rightarrow p \rightarrow i$	-0.607
(f)	$i \rightarrow p \rightarrow i \rightarrow e \rightarrow i \rightarrow l \rightarrow i$	0.565
(g)	$i \rightarrow l \rightarrow i \rightarrow e \rightarrow i \rightarrow p \rightarrow i$	0.652
(h)	$i \rightarrow l \rightarrow i \rightarrow e \rightarrow i \rightarrow l \rightarrow i$	3.826

Figura 2.10: c denota la categoria, p i pattern, i gli item, e gli ensemble e l sta per colore

Capitolo 3

Applicazione Tensorflow

In questo capitolo andrò ad illustrare il software realizzato il cui obiettivo è quello di localizzare e riconoscere, all'interno di un file formato jpeg o jpg, indumenti appartenenti allo stile di moda Country.
Il cuore del sofware risultante è la rete neurale, con la quale sono state svolte le fasi di training e testing. Un alto step fondamentale precedente a questi due passaggi è la preparazione del database, che verrà approfondita insieme alle altre fasi.
La realizzzione ed implementazione della rete neurale è stata attuata attraverso la libreria software open source Tensorflow: insieme ad altri moduli farò un preambolo dei vari strumenti utilizzati.

3.1 Strumenti utilizzati

Come accennato precedentemente, Tensorflow è una libreria open-source progettata da Google Brain, branca di Google nel campo dell' Intelligenza Artificiale e del Machine Learning; tale libreria consiste in una serie di API in linguagio Python progettate con l' obiettivo di semplificare ed unificare la creazione di modelli di apprendimento automa-

tico.

Attraverso questo software infatti si riesce a creare una rete neurale in modo semplice ed efficace: la particolarità di Tensorflow infatti consiste nel flusso delle sue elaborazioni, poichè ogni operazione matematica eseguita corrisponde ad un nodo, mentre i bordi sono i dati da processare (solitamente array multidimensionali o costanti numeriche).Array ed operatori sono colegati attravreso dei *tensori*, che indicano la direzione del flusso di esecuzione; l' unione di questi elementi da vita ad un grafo.

Di seguito viene proposto un esempio di moltiplicazione tra array per vedere il funzionamento

```
import  tensorflow as tf
#ARRAY
x1 = tf.constant([1,2,3,4])
x2 = tf.constant([5,6,7,8])
#MOLTIPLICAZIONE
risultato = tf.multiply(x1,x2)

print(risultato)

with tf.Session() as sessione:
    writer = tf.summary.FileWriter('logs', sessione.graph)
    print(sessione.run(risultato))
```

Il risultato è un oggetto **Tensor** di cui abbiamo aperto una *sessione* per eseguire il modello: all'interno di questa sessione stiamo scrivendo,con il comando *writer*, una cartella di nome logs che conterrà il file di log della sessione e, in particolare, il grafo risultante. Per poter visualizzare il grafo basta eseguire i seguenti comandi da shell:

```
\$ tensorboard --logdir="."
```

Nella shell ci apparirà un indirizzo che, copiato nel browser, ci permette di accedere a **Tensorboard**, una suite di strumenti di visualizzazione specificatamente creata per rapprrsentare gli eventi delle sessioni di Tensorflow.

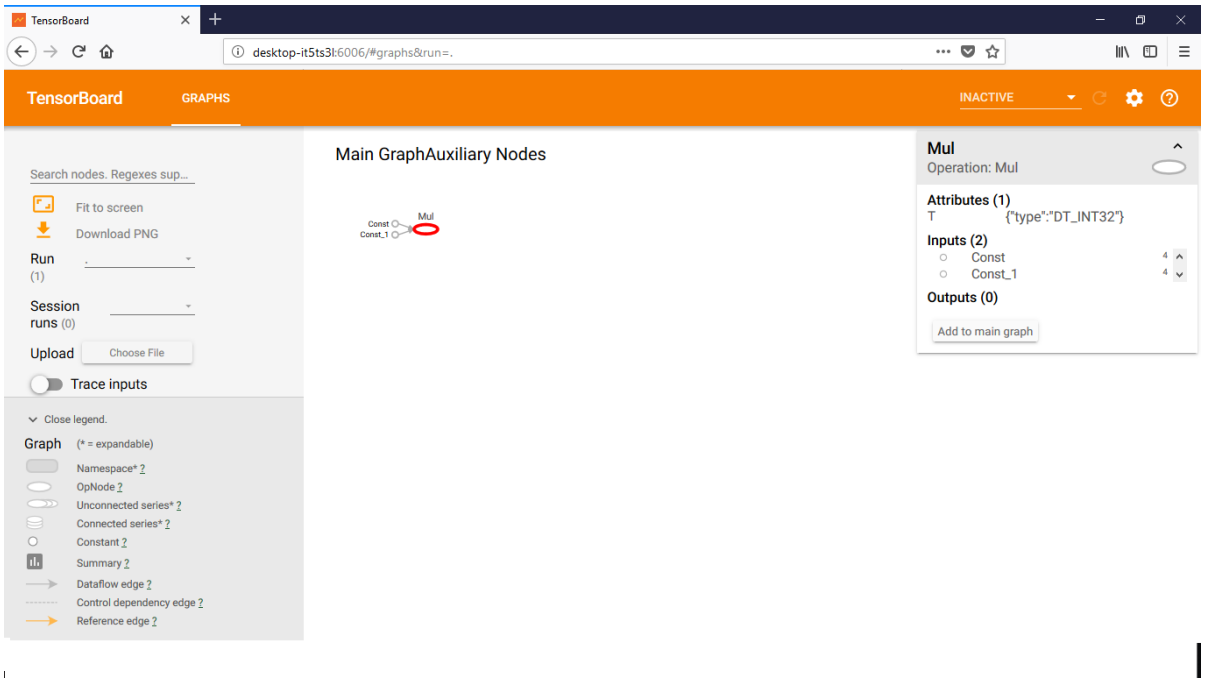

Figura 3.1: Tensorboard

Il grafo centrale corrisponde alla rete neurale e, in alto a destra, una piccola legenda. Sicuramente, per la semplice moltiplicazione di due array, il codice da scrivere è molto, ma per costruire una rete neurale sono bastate poche righe di codice; oltre a ciò i vantaggi sono notevoli:

- possibilità di sfruttare le GPU NVIDIA per il calcolo (e quindi migliori tempi di elaborazione)
- suite di visualizzazione integrata
- possibilità di sviluppo di librerie che utilizzino Tensorflow come backend.

Altre due librerie utili nel campo del Machine Learning sono **Numpy** e **Pandas**.
Numpy è una libreria per il calcolo scientifico e permette di trattare i numeri in maniera più semplice rispetto ai normali linguaggi di programmazione, poichè l' elemento di base di Numpy è l' array.
Pandas invece è una libreria che permette di manipolare dati di forme diverse per favorirne l' analisi; questa libreria è molto utilizzata nel mondo del Data Munging [1].

3.2 Installazione

Come primo approccio ho installato la versione Tensorflow-CPU nel mio PC; nella fase di training ho constatato l' elevato costo tempistico e computazionale del processo (in due settimane la macchina aveva svolto metà degli steps per completare l' apprendimento). Essendo i tempi di elaborazione insostenibili la Professoressa Piccolomini ed il Professor Marfia mi hanno dato la possibilità di ususfruire della macchina presente nel laboratorio di Informatica di via Ranzani, grazie alla quale ho potuto completare il training in 6 giorni.

3.3 Dataset

Parte preliminare e sicuramente fondamentale è la strutturazione del dataset. Come approfondito nel secondo capitolo, le applicazioni software nel campo dell' Object Recognition si fondano su dataset molto ampi formati da due fattori:

- immagine da localizzare
- label da associare alla localizzazione

[1] processo di trasformazione dei dati dal loro formato grezzo (scaricato dalla sorgente) in uno ottimizzato per essere processato da un algoritmo

Il primo Dataset mi è stato fornito dal Prof. Marfia Gustavo, insegnante al Dipartimento di Moda di Rimini, il quale è stato integrato successivamente con un altro set di immagini.

Le immagini provengono dal Dipartimento di Moda e sono relativi a sfilate o book fotografici riguardanti lo stile Country; lo scopo della rete neurale è apprendere 10 articoli Country [2]:

a) **PANTS** (PANTALONI)

b) **JACKETS** (CAPPOTTO)

c) **SKIRTS** (GONNA)

d) **SHOES** (SCARPE)

e) **BAGS** (BORSA)

f) **ACCESSORIES** (ACCESSORI)

g) **GLOVES** (GUANTI)

h) **VESTS** (GILET)

i) **TOPS** (TOP)

j) **HATS** (CAPPELLO)

Il Daset su cui ho implementato l' applicativo è così strutturato:

- una cartella **Images** contenente le immagini;

[2]stile di provenienza americana che trae origine dai vestiti da cowboy e di matrice indiana

- una cartella **Labels** contenente files txt;

  Ogni file txt contiene una o più righe formate da 4 numeri; ogni riga corrisponde alle coordinate del riquadro che localizza l' indumento nell' immagine; in particolare vengono specificati *x, y, width ,heigth*, ovvero x,y, altezza e larghezza del riquadro.

Le directory Images e Labels sono suddivise al loro interno da 10 sottocartelle, una per ogni articolo Country. Spesso l' immagine usata per identificare un pantalone piuttosto

COUNTRY PANTS: 4001
COUNTRY JACKTES: 4002
COUNTRY SKIRTS: 4003
COUNTRY SHOES: 4004
COUNTRY BAGS: 4005
COUNTRY ACCESSORIES: 4006
COUNTRY HATS: 4007
COUNTRY GLOVES: 4008
COUNTRY VESTS: 4009
COUNTRY TOPS: 4010

Figura 3.2: Schema suddivisione directories

che una giacca è la stessa, ciò che cambia sono le coordinate di localizzazione. Il mio lavoro in questa fase di sviluppo è stato scorrere le due directory e le varie sottocartelle per salvare ,in una struttura dati, il nome dell' immagine e le relative informazioni rigurdanti le coordinate *x,y,width* ed *heigth*.

Per fare ciò ho utilizzato PyCharm, un ambiente di sviluppo per Python.

Per salvare le informazioni necessarie al training ho utilizzato i dizionari,ovvero liste di chiave-valore il cui indice è un *etichetta*; la funzionalità del dizionario consiste nella correlazione immediata fra chiave e rispettivo valore; così facendo non si ha più bisogno dell' inidice numerico di un determinato valore per accedervi.

Di seguito riporto la struttura dei dizionari creati:

```
#DIZIONARIO PER INFO IMMAGINE DA LABELS
informazioniLabels={
    "nomeImmagine":"",
    "xmin" : 0,
    "ymin" : 0,
    "xmax" : 0,
    "width" : 0,
    "ymax" : 0,
    "height" :0
}
#DIZIONARIO PER INFO IMMAGINE DA IMAGES
informazioniIm={
                "name" : "",
                "width" : 0,
                "height" : 0,
                "class" : "",
                "xmin" : 0,
                "xmax" : 0,
                "ymin" : 0,
                "ymax" : 0
}
#DIZIONARIO APPARTENEZA CATEGORIA
categoria={
    'nomeImmagine' : "", 'nomeClasse' : "" }
```

Ho creato due strutture dati per salvare le informazioni estrapolate sia dalla directory Images che da quella Labels. Successivamente, dovendo aggiungere la categoria di appartenenza della foto (nomeClasse, che si riferisce a pantalone,gonna ecc...), ho creato il dizionario categoria.Alle 4 coordinate precedenti sono state aggiunte *xmin* ed *ymin*, che non sono altro che il risultato della sottrazione tra i valori "heigth- y" e "width- x". Successivamente ho implementato due *cicli for* per scorrere ambedue le directory Ima-

ges e Labels e salvare i valori necessari. Lo scopo dei cicli infatti era trovare il file txt corrispondente all 'immagine e salvare coordinate e classe di appartenenza nella stessa posizione del dizionario. Ogni volta che viene settato un dizionario viene aggiunto all'interno di un array, struttura dati vista precedentemente nella quale ogni posizione corrisponde ad un' immagine con i relativi valori.

Infine l' utimo passaggio svolto in Python è stato scrivere files CSV dagli array risultatnti dei cicli :

```
# CREO IL FILE CSV TRAIN

keys = arrayDefinitivoTrainingCSV[0].keys()
#for directory in ['train','test']:

with open(r'images\train_labels.csv', 'w') as output_file:
    fieldname = ['filename', 'width', 'height', 'class', 'xmin', 'ymin',
    'xmax', 'ymax']
    #for directory in ['train','test']:
    writer = csv.DictWriter(output_file,fieldnames=fieldname)
    writer.writeheader()
    #writer.writerows(arrayDefinitivoTrainingCSV)
    for i in range(len(arrayDefinitivoTrainingCSV)):
            n=arrayDefinitivoTrainingCSV[i]['name']
            cl=arrayDefinitivoTrainingCSV[i]['class']
            xm=arrayDefinitivoTrainingCSV[i]['xmin']
            xM=arrayDefinitivoTrainingCSV[i]['xmax']
            ym=arrayDefinitivoTrainingCSV[i]['ymin']
            yM=arrayDefinitivoTrainingCSV[i]['ymax']
            w=arrayDefinitivoTrainingCSV[i]['width']
            h=arrayDefinitivoTrainingCSV[i]['height']
```

```
            writer.writerow({'filename': n, 'width': w, 'height': h, 'class':
    cl, 'xmin': xm, 'ymin': ym, 'xmax': xM, 'ymax': yM})
            print(i)

# CREO IL FILE CSV TESTING

keys = arrayDefinitivoTestingCSV[0].keys()

with open(r'images\test_labels.csv', 'w') as output_file:
    fieldname = ['filename', 'width', 'height', 'class', 'xmin', 'ymin',
    'xmax', 'ymax']
    #for directory in ['train','test']:
    writer = csv.DictWriter(output_file,fieldnames=fieldname)
    writer.writeheader()
    #writer.writerows(arrayDefinitivoTrainingCSV)
    for i in range(len(arrayDefinitivoTestingCSV)):
            n=arrayDefinitivoTestingCSV[i]['name']
            cl=arrayDefinitivoTestingCSV[i]['class']
            xm=arrayDefinitivoTestingCSV[i]['xmin']
            xM=arrayDefinitivoTestingCSV[i]['xmax']
            ym=arrayDefinitivoTestingCSV[i]['ymin']
            yM=arrayDefinitivoTestingCSV[i]['ymax']
            w=arrayDefinitivoTestingCSV[i]['width']
            h=arrayDefinitivoTestingCSV[i]['height']

            writer.writerow({'filename': n, 'width': w, 'height': h, 'class':
    cl, 'xmin': xm, 'ymin': ym, 'xmax': xM, 'ymax': yM})
            print(i)


# CREO IL FILE CSV

keys = arrayDefinitivoCSV[0].keys()
```

```
print(keys)
with open('images\COUNTRY_labels.csv', 'w') as output_file:
    fieldname = ['filename','width' , 'height', 'class' , 'xmin' ,'ymin' ,
    'xmax' ,  'ymax' ]
    #for directory in ['train','test']:
    writer = csv.DictWriter(output_file,fieldnames=fieldname)
    writer.writeheader()
    #writer.writerows(arrayDefinitivoTrainingCSV)
    for i in range(len(arrayDefinitivoCSV)):
            n=arrayDefinitivoCSV[i]['name']
            cl=arrayDefinitivoCSV[i]['class']
            xm=arrayDefinitivoCSV[i]['xmin']
            xM=arrayDefinitivoCSV[i]['xmax']
            ym=arrayDefinitivoCSV[i]['ymin']
            yM=arrayDefinitivoCSV[i]['ymax']
            w=arrayDefinitivoCSV[i]['width']
            h=arrayDefinitivoCSV[i]['height']

            writer.writerow({'filename':n,'width':w,'height':h,
    'class':cl,'xmin':xm,'ymin':ym,'xmax':xM,'ymax':yM})
            print(i)
```

I file CSV sono necessari per la creazione di file TFRecords che serviranno poi alla rete neurale per far partire l'apprendimento (come vedremo in seguito). Sono stai creati i file CSV con le immagini per training e testing; infine ho creato un fle csv delle immagini totali presenti. Per incolonnare precisamente (secondo le richieste di Tensorflow) i valori *class,xmin,xmax,ymin,ymax,width* ed *heigth* ho fatto un altro script Python nel quale ho definito l' ordine della successione dei valori a seguito del nome dell' immagine. Qui di seguito riporto il codice:

```
import numpy as np
import pandas as pd
np.random.seed(1)
#TRAIN
full_train_labels = pd.read_csv(r'images\train_labels.csv')
print(full_train_labels)
full_train_labels.head()
grouped = full_train_labels.groupby('filename')
print(grouped)
grouped.apply(lambda x: len(x)).value_counts()

#TEST
full_test_labels=pd.read_csv(r'images\test_labels.csv')
print(full_test_labels)
full_test_labels.head()
grouped_test=full_test_labels.groupby('filename')
print(grouped_test)
grouped_test.apply(lambda x: len(x)).value_counts()
```

Successivamente ho trasformato i file CSV riguardanti Training e Testing in files **TFRecord**,un formato di file specifico per inizializzare i parametri del file di configurazione del modello.

La conversione dei files da CSV a TFRecord avviene mediante le seguenti righe di comando da shell:

```
\$ python generate_tfrecord.py --csv_input=images\train_labels.csv
    --image_dir=images\train --output_path=train.record
\$ python generate_tfrecord.py --csv_input=images\test_labels.csv
    --image_dir=images\test --output_path=test.record
```

Un altro file necessario per l'apprendimento del modello è il **labelmap**, ovvero una *mappa delle etichette*. Questo file (il cui formato è pbtxt) è una mappatura delle classi che il modello deve apprendere, ed oguna di esse è indicizzata tramite un id.

Di seguito riporto il mio labelmap:

```
item {
  id: 1
  name: 'COUNTRY PANTS'
}

item {
  id: 2
  name: 'COUNTRY JACKETS'
}

item {
  id: 3
  name: 'COUNTRY SKIRTS'
}

item {
  id: 4
  name: 'COUNTRY SHOES'
}

item {
  id: 5
  name: 'COUNTRY BAGS'
}

item {
  id: 6
  name: 'COUNTRY ACCESSORIES'
```

```
}

item {
  id: 7
  name: 'COUNTRY HATS'
}

item {
  id: 8
  name: 'COUNTRY GLOVES'
}

item {
  id: 9
  name: 'COUNTRY VESTS'
}

item {
  id: 10
  name: 'COUNTRY TOPS'
}
```

3.3.1 Labeling

Come accennato nell' introduzione del capitolo è stato aggiunto, nel corso della preparazione del dataset, un secondo set di foto appartenente al mondo Counrty.
Il Dataset proviene, come nel caso del primo, dal Dipartimento di Moda di Rimini e mi è stato fornito dal Professor Marfia.
La decisione di aggiungere dati al training è stata presa poichè nel primo dataset erano

presenti 127 foto totali; considerando che viene usato il 70% del Dataset per il training ed il 30% per il testing, le foto per l' apprendimento risulatavano poche.

Sebbene le foto appartengano alla stessa insiemistica, la tipologia di immagini è diversa poichè il secondo set di foto appartiene principalmente a sfilate di moda, mentre nel primo erano presenti per lo più immagini derivate da book fotografici o shootings.

La differenza fondamentale tra i due Dataset è però il **labeling**, ovvero l' assegnazione dell' etichetta ad una particolare zona dell 'immagine : nel primo set infatti le coordinate di localizzazione degli articoli Country erano già presenti nella directory Labels, a differenza del corrente.

Per effettuare il labeling delle immagini ho utilizzato "labelImg",un software che, data in input un immagine, permette di localizzare ed etichettare una sua area, restituendo come output un file XML contenente le coordinate della localizzazione effettuata. Nella

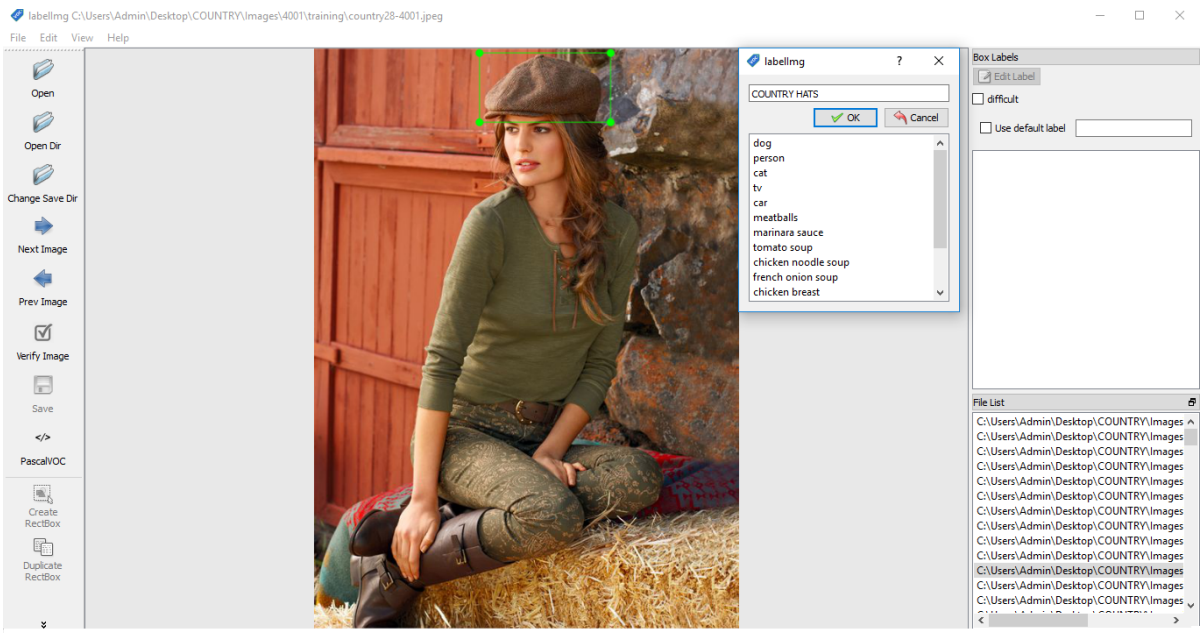

Figura 3.3: Interfaccia labelImg

figura sovrastante possiamo vederne il funzionamento: il riquadro verde indica l' area che vogliamo etichettare attraverso la finestra vicino a destra (nell' esempio ho inquadrato la zona dell capello Country assegnandogli la classe "COUNTRY HATS").
Successivamente ho convertito il file XML risultante in un file CSV con la seguente porzione di codice Python:

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin',
    'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

```
def main():
    for folder in ['train','test']:
        image_path = os.path.join(os.getcwd(), ('images/' + folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/' + folder + '_labels.csv'), index=None)
        print('Successfully converted xml to csv.')


main()
```

Il file CSV ottenuto è stato aggiunto a quello precedente.

3.4 Training

Dopo aver organizzato i dati da processare, ho configurato la fase successiva, ovvero quella del Training.

Tensorflow fornisce una raccolta di modelli pre-addestrati all' Objet Recognition basati su dataset COCO,set di dati Kitti e open Images. Alcuni modelli (come il modello SSD-MobileNet) hanno un'architettura che consente una rilevazione più rapida ma con minore precisione, mentre alcuni modelli (come il modello Faster-RCNN) offrono un rilevamento più lento ma con maggiore precisione. Inizialmente ho provato ad utilizzare il modello Faster-RCNN-Inception-V2 ma non sono riuscito a configurarlo con il sistema operativo Windows 10, di conseguenza ho ripiegato sul modello **ssd_mobilenet_v1_coco**.

Ogni modello ha un proprio file di configurazione nel quale ho settato i paramteri necessari per inizializzare l' apprendimento; in particolre ho modificato la PATH dei files di

Modelli addestrati da COCO

Nome del modello	Velocità (ms)	COCO mAP [^ 1]	Uscite
ssd_mobilenet_v1_coco	30	21	scatole
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	scatole
ssd_mobilenet_v1_quantized_coco ☆	29	18	scatole
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	scatole
ssd_mobilenet_v1_ppn_coco ☆	26	20	scatole
ssd_mobilenet_v1_fpn_coco ☆	56	32	scatole
ssd_resnet_50_fpn_coco ☆	76	35	scatole
ssd_mobilenet_v2_coco	31	22	scatole
ssd_mobilenet_v2_quantized_coco	29	22	scatole
ssdlite_mobilenet_v2_coco	27	22	scatole
ssd_inception_v2_coco	42	24	scatole
faster_rcnn_inception_v2_coco	58	28	scatole

Figura 3.4: Lista di alcuni modelli di Tensorflow

label, training e testing.

Di seguto riporto il file di configurazione del modello:

```
model {
  ssd {
    num_classes: 10
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
```

```
}
matcher {
  argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
  }
}
similarity_calculator {
  iou_similarity {
  }
}
anchor_generator {
  ssd_anchor_generator {
    num_layers: 6
    min_scale: 0.2
    max_scale: 0.95
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    aspect_ratios: 3.0
    aspect_ratios: 0.3333
    reduce_boxes_in_lowest_layer: true
  }
}
image_resizer {
  fixed_shape_resizer {
    height: 300
    width: 300
  }
}
```

```
box_predictor {
  convolutional_box_predictor {
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    use_dropout: false
    dropout_keep_probability: 0.8
    kernel_size: 3
    box_code_size: 4
    apply_sigmoid_to_scores: false
    conv_hyperparams {
      activation: RELU_6,
      regularizer {
        l2_regularizer {
          weight: 0.00004
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
    }
  }
}
feature_extractor {
  type: 'ssd_inception_v2'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
```

```
      l2_regularizer {
        weight: 0.00004
      }
    }
    initializer {
      truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
      }
    }
    batch_norm {
      train: true,
      scale: true,
      center: true,
      decay: 0.9997,
      epsilon: 0.001,
    }
  }
  override_base_feature_extractor_hyperparams: true
}
loss {
  classification_loss {
    weighted_sigmoid {
    }
  }
  localization_loss {
    weighted_smooth_l1 {
    }
  }
  hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
```

```
        max_negatives_per_positive: 3
        min_negatives_per_image: 0
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    normalize_loss_by_num_matches: true
    post_processing {
      batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 100
      }
      score_converter: SIGMOID
    }
  }
}

train_config: {
  batch_size: 24
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
```

```
    }
  }
  fine_tune_checkpoint:
    'C:\\Windows\\System32\\venv\\models\\research\\object_detection\\ssd_mobilenet_v1_co
  from_detection_checkpoint: true
  load_all_detection_checkpoint_vars: true
  # Note: The below line limits the training process to 200K steps, which we
  # empirically found to be sufficient enough to train the pets dataset. This
  # effectively bypasses the learning rate schedule (the learning rate will
  # never decay). Remove the below line to train indefinitely.
  num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    ssd_random_crop {
    }
  }
  max_number_of_boxes: 50
}

train_input_reader: {
  tf_record_input_reader {
    input_path:
    'C:\\Windows\\System32\\venv\\models\\research\\object_detection\\train.record'
  }
  label_map_path:
    'C:\\Windows\System32\\venv\\models\\research\\object_detection\\training\\labelmap.p
}

eval_config: {
  metrics_set: "coco_detection_metrics"
```

```
  num_examples: 305
}

eval_input_reader: {
  tf_record_input_reader {
    input_path:
   'C:\\Windows\\System32\\venv\\models\\research\\object_detection\\test.record'
  }
  label_map_path:
   'C:\\Windows\\System32\\venv\\models\\research\\object_detection\\training\\labelmap.pbt
  shuffle: false
  num_readers: 1
}
```

Le path evidenziate in rosso riguardano i files di label,training e testing. Oltre a questi percorsi ho modificato:

- *num_classes*: 10
- *num_examples*: 305, poichè il Dataset finale è composto da 432 immagini, di cui il 70% vanno usate per il training.

Inoltre si può evincere che è stata scelta la SIGMOIDE come funzione di attivazione. Successivamente ho fatto finalmente partire il training attraverso il seguente comando da shell:

```
\$ python train.py --logtostderr --train_dir=training/
   --pipeline_config_path=training/ssd_inception_v2_pets.config
```

Una volta eseguito il comando inizierà il training. Nella shell verranno riportati gli steps che il modello sta eseguendo : in ogni iterazione il modello processa informazioni ed

abbassa il *loss*, ovvero la soglia di precisione con la quale classifica le immagini (inizialmente parte da un valore intorno ai 70 per poi scendere sino a valori prossimi all' 1).
Il training può essere stoppato in qualsiasi momento per poi essere riattivato; nei paramteri del file di configurazione c'è una voce che indica il numero massimo di steps: solitamente non si raggiunge mai quella soglia, bensì si stoppa l' apprendimento una volta ragiunto il loss adeguato. Utilizzando la virtual machine installata nel computer del laboratorio ho impiegato all' incirca 6 giorni per effettuare il training, poichè ogni step impiegava dai 15 ai 35 secondi per concludersi.

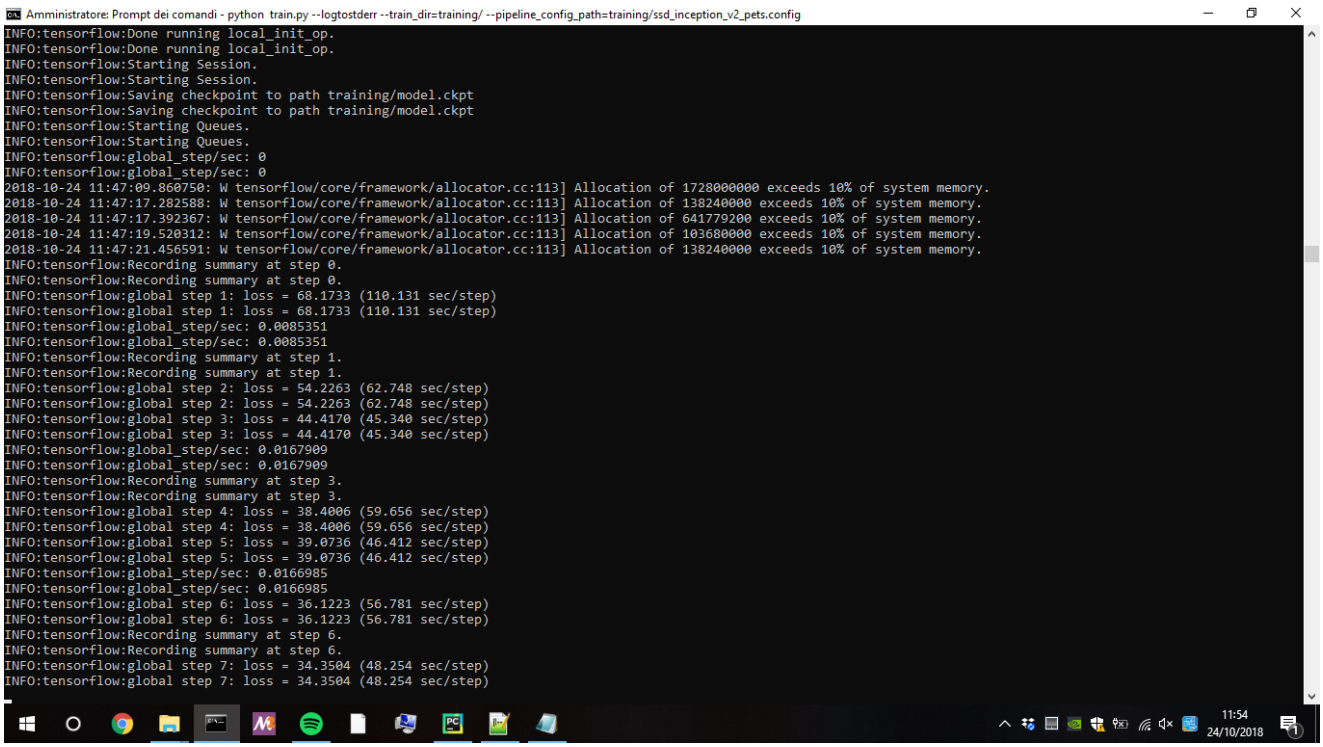

Figura 3.5: Terminale all' avvio del training

3.5 Testing

Una volta finito il training sono passato al testing.

Il testing serve per verificare se il modello ha appreso conoscenza delle classi specificate nella labelmap e se, soprattutto, è in grado di identificare una categoria all' interno dell' immagine.

Come acccennato ad inizio capitolo Tensorflow produce il grafo risultante delle elaborazioni svolte: per poter usufruire del modello addestrato si deve esportare il grafo ottenendo così il **frozen inference graph**, ovvero il file contentente il *classificatore* finale.

Per esportare il modello bisogna eseguire il seguente comando da shell:

```
python export_inference_graph.py --input_type image_tensor
    --pipeline_config_path training/faster_rcnn_inception_v2_pets.config
    --trained_checkpoint_prefix training/model.ckpt-XXXX --output_directory
    inference_graph
```

dove le XXX vanno sostituite con il numero di step svolti per il training, e l'output directory corrisponde alla cartella che si crea contentente l'inference graph. Portata a termine l' esportazione ho testato il mio modello con le immagini che non fossero state processate nel training. Per il testing ho utilizzato *object_detection_tutorial.ipynb*, uno script di Python messo a disposizione da Tensorlfow; il file serve per verificare il corretto funzionamento delle librerie: vengono date in input due immagini (una di due cani e l' altra di un paesaggio) per vedere se è stato installato tutto corettamente. Ho modificato questo script sostituendo il modello di default con il mio classificatore ed ho sostituito le due immagini con le foto Country.

Di seguito riporto lo script:

```
from distutils.version import StrictVersion
```

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or
    later!')
from utils import label_map_util
from utils import visualization_utils as vis_util
# What model to use.
MODEL_NAME = 'countryGraph'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for
    the object detection.
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('training', 'labelmap.pbtxt')
```

```
NUM_CLASSES = 10
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
   max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

# If you want to test the code with your images, just add path to the images
   to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
   'image{}.jpg'.format(i)) for i in range(1,5) ]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
def run_inference_for_single_image(image, graph):
  with graph.as_default():
    with tf.Session() as sess:
      # Get handles to input and output tensors
      ops = tf.get_default_graph().get_operations()
      all_tensor_names = {output.name for op in ops for output in op.outputs}
      tensor_dict = {}
```

```
    for key in [
        'num_detections', 'detection_boxes', 'detection_scores',
        'detection_classes', 'detection_masks'
    ]:
      tensor_name = key + ':0'
      if tensor_name in all_tensor_names:
        tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
            tensor_name)
    if 'detection_masks' in tensor_dict:
      # The following processing is only for single image
      detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
      detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
      # Reframe is required to translate mask from box coordinates to image
coordinates and fit the image size.
      real_num_detection = tf.cast(tensor_dict['num_detections'][0],
tf.int32)
      detection_boxes = tf.slice(detection_boxes, [0, 0],
[real_num_detection, -1])
      detection_masks = tf.slice(detection_masks, [0, 0, 0],
[real_num_detection, -1, -1])
      detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
          detection_masks, detection_boxes, image.shape[0], image.shape[1])
      detection_masks_reframed = tf.cast(
          tf.greater(detection_masks_reframed, 0.5), tf.uint8)
      # Follow the convention by adding back the batch dimension
      tensor_dict['detection_masks'] = tf.expand_dims(
          detection_masks_reframed, 0)
    image_tensor =
tf.get_default_graph().get_tensor_by_name('image_tensor:0')

    # Run inference
    output_dict = sess.run(tensor_dict,
                           feed_dict={image_tensor: np.expand_dims(image,
```

```
    0)})

      # all outputs are float32 numpy arrays, so convert types as appropriate
      output_dict['num_detections'] = int(output_dict['num_detections'][0])
      output_dict['detection_classes'] = output_dict[
          'detection_classes'][0].astype(np.uint8)
      output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
      output_dict['detection_scores'] = output_dict['detection_scores'][0]
      if 'detection_masks' in output_dict:
        output_dict['detection_masks'] = output_dict['detection_masks'][0]
  return output_dict
for image_path in TEST_IMAGE_PATHS:
  image = Image.open(image_path)
  # the array based representation of the image will be used later in order
    to prepare the
  # result image with boxes and labels on it.
  image_np = load_image_into_numpy_array(image)
  # Expand dimensions since the model expects images to have shape: [1, None,
    None, 3]
  image_np_expanded = np.expand_dims(image_np, axis=0)
  # Actual detection.
  output_dict = run_inference_for_single_image(image_np, detection_graph)
  # Visualization of the results of a detection.
  vis_util.visualize_boxes_and_labels_on_image_array(
      image_np,
      output_dict['detection_boxes'],
      output_dict['detection_classes'],
      output_dict['detection_scores'],
      category_index,
      instance_masks=output_dict.get('detection_masks'),
      use_normalized_coordinates=True,
      line_thickness=8)
  plt.figure(figsize=IMAGE_SIZE)
```

```
plt.imshow(image_np)
```

Essendo uno script formato .ipybn va eseguito nel browser come Tensorboard attraverso il comando:

```
jupyter notebook object_detection_tutorial.ipynb
```

con il quale accediamo a Jupyter, una piattaforma open sorce per l' elaborazione interattiva. Inizialmente ho effettuato il testing con le immagini che non avevo utilizzato per il training, ma il risultato è stato alquanto sconfortante: difficilmente infatti avveniva un' adeguata identificazione del capo d' abbigliamento e,quando si verificava, non sempre era corretta; in più la localizzazione all' interno dell' immagine rappresentava aree non conformi col capo d' abbigliamento in questione.
Il testing è stato fatto in concomitanza del secondo capitolo, che mi ha permesso di capire le motivazioni per cui non avveniva l' object detection.
Le cause principali sono:

a) person-detection: come spiegato per le applicazioni presenti nel mercato, anche il mio lavoro incorre in problemi di person-detection. Le immagini con le quali è stato effettuato l' apprendimento infatti sono foto di sfilate o shooting fotografici nelle quali, inevitailmente, è presente l'indossatore. Riuscire a distinguere la sagoma umana e separarla dai vestiti che indossa risulta ancora difficile per i modelli di apprendimento; a ciò va aggiunta un' ulteriore complicazione per quanto riguarda la posizione del corpo umano: molto spesso infatti le modelle sono fotografate in pose non uniformi (seduti,di profilo ec...) e ciò facilita l' errore.

   Prendiamo ad es. una modella che indossa una giacca Country con i caratteristici

filamenti posti nella parte inferiore del braccio , e supponiamo che l' indossatrice abbia una posa con le braccia aperte: in questo caso la particolarità della giacca Country sono sicuramente gli accessori laterali sopra citati, di conseguenza sono un fattore rilevante per l' etichettatura della categoria. Nel momento del labeling dell' immagine la localizzazione avverrà nell' area che circoscrive la zona dei filamenti, andando a caripre anche i pixel sottostanti che riguardano il background o altri elementi della foto, non propedeutici all' apprendimento.

b) diversificazione degli elementi: vedendo esempi su Internet di altri progetti riguardanti la classificazione di immagini attraverso l' Object-Detection, ho constatato che gli ambiti che venivano processati riguardavano categorie "monotematiche".
Prendiamo ad es. un progetto che ho consultato riguardante la classificazione di procioni. L' animale non si presenta in natura in forma diverse da quella che comunemente conosciamo, ovvero con un muso pronunciato, le orecchie a punta ed un pelo grigiastro; le foto possono essere differenti per luce, angolazione, ma le caratteristiche peculiari del procione sono le stesse e le tonalità di colore principali che il modello apprende rimangono invariate da immagine a immagine.
Tale discorso è incompatibile con i capi d' abbigliamneto fashion: oggigiorno infatti ci sono sempre più tendenze e, anche uno stile tradizionale come il Country, presenta sfumature diverse al suo interno.Prendiamo come esempio la gonna Country: nel dataset ho usufruito di articoli diversi fra loro per tipologia (a fiori, a quadri, monocromatiche ecc...) e struttura(lunghe, corte ec...). Nel momento in cui il modello si allena a riconoscere la "COUNTRY SKIRTS" in realtà si sta esercitando nel riconoscere diverse sfaccettature della categoria "gonna" e non una singola architettura dell' indumento; ciò lo discosta dall' apprendimento della classe.

c) pulizia del background: Stesso discorso della person-detection. Come affrontato nel secondo capitolo, anche la pulizia del background di un' immagine è importante affinchè il modello riesca a classificare, poichè se lo sfondo circostante al capo d' abbigliamento ha tonailtà simili a quelle dell' articolo potrebbe trarre in inganno il modello. Di conseguenza si preferisce processare immagini il cui sfondo sia bianco.

Alla luce delle considerazioni effettuate ho testato il mio modello con immagini che avessero le seguenti carateristiche:

- assenza della sagoma umana (per la maggior parte delle foto)
- background bianco

Di seguito riporto i rislutati ottenuti:

- **COUNTRY HATS** da figura 3.6 a figura 3.9
- **COUNTRY SHOES** da figura 3.10 a figura 3.13
- **COUNTRY JACKETS** da figura 3.14 a figura 3.17
- **COUNTRY SKIRTS** da figura 3.18 a figura 3.21
- **COUNTRY BAGS** da figura 3.22 a figura 3.25
- **COUNTRY TOPS** da figura 3.26 a figura 3.29

Non ho inserito le immagni di altre classi perchè la classificazione non avveniva o non era precisa. In particolare le categorie che mi hanno dato maggiori problemi sono state:

- COUNTRY PANTS: il modello non riesce ad identificare i pantaloni Country a causa del vasto assortimento di modelli che ci sono per questa categoria (Jeans,Denim,taglio

Figura 3.6: Cappello Country

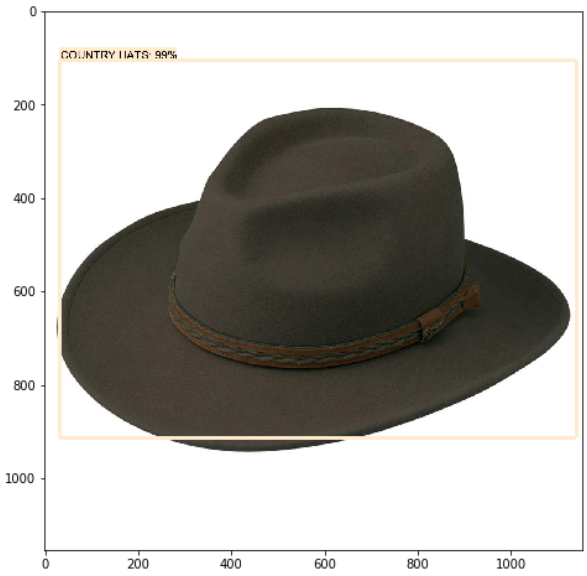

Figura 3.7: Cappello Country

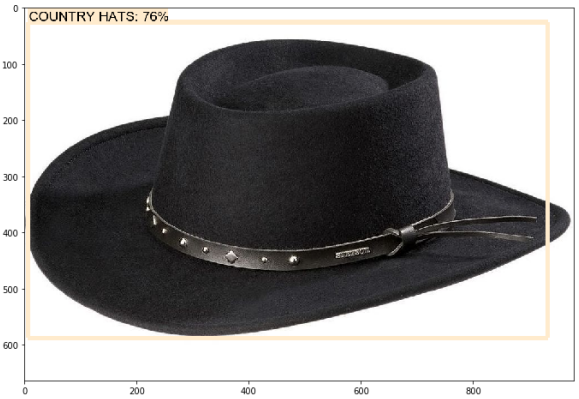

Figura 3.8: Cappello Country

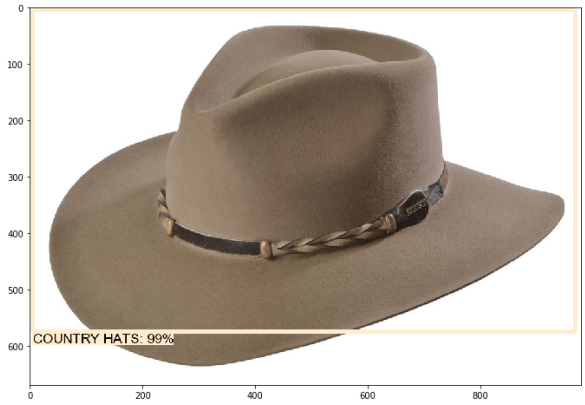

Figura 3.9: Cappello Country

Figura 3.10: Stivale Country

Figura 3.11: Stivale Country

Figura 3.12: Stivale Country

Figura 3.13: Stivale Country

Figura 3.14: Giaccone Country

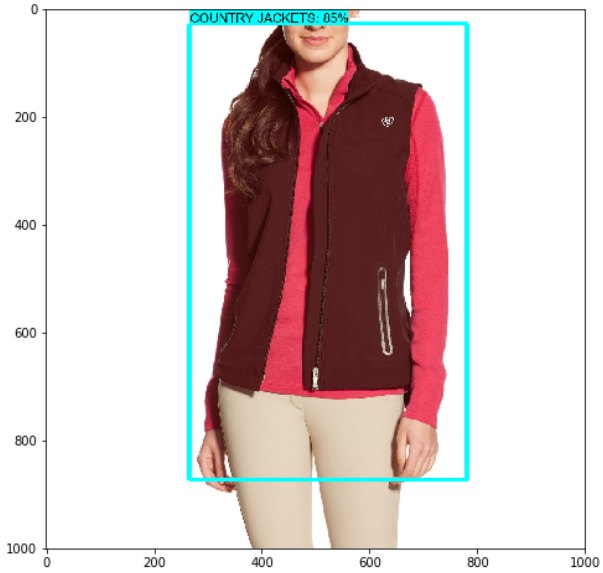

Figura 3.15: Giaccone Country

Figura 3.16: Giaccone Country

Figura 3.17: Giaccone Country

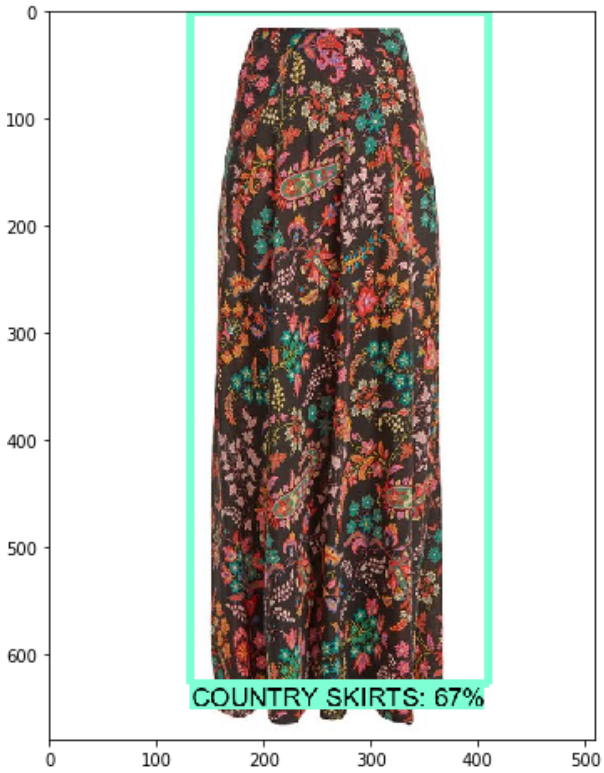

Figura 3.18: Gonna Country

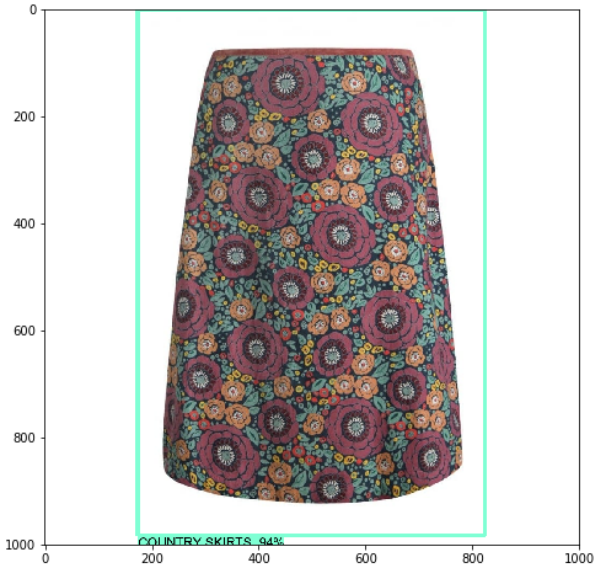

Figura 3.19: Gonna Country

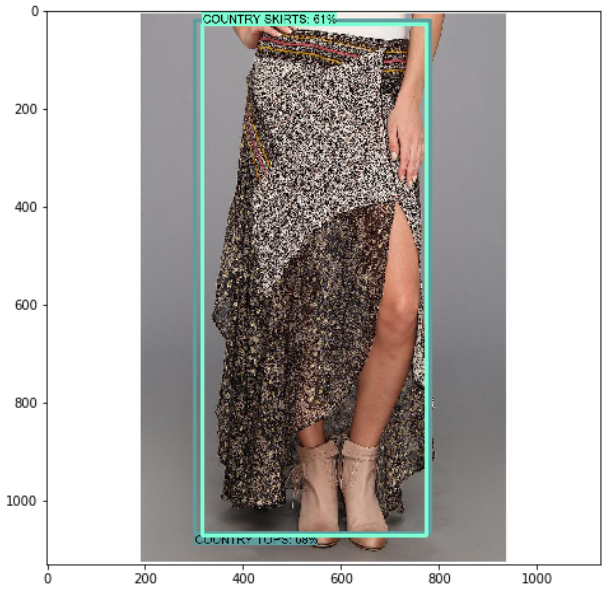

Figura 3.20: Gonna Country

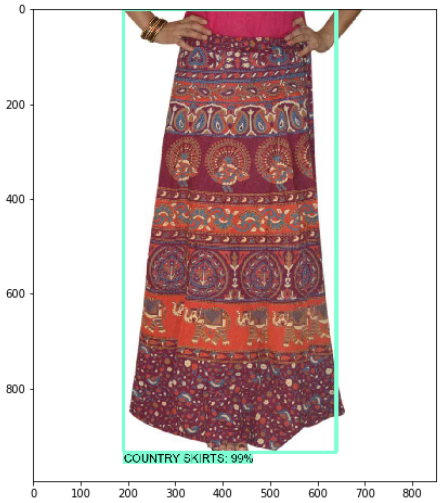

Figura 3.21: Gonna Country

Figura 3.22: Borsa Country

Figura 3.23: Borsa Country

Figura 3.24: Borsa Country

Figura 3.25: Borsa Country

Figura 3.26: Top Country

Figura 3.27: Top Country

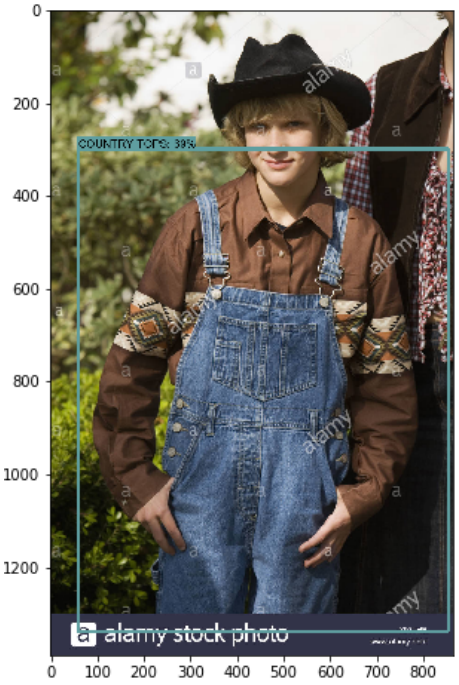

Figura 3.28: Top Country

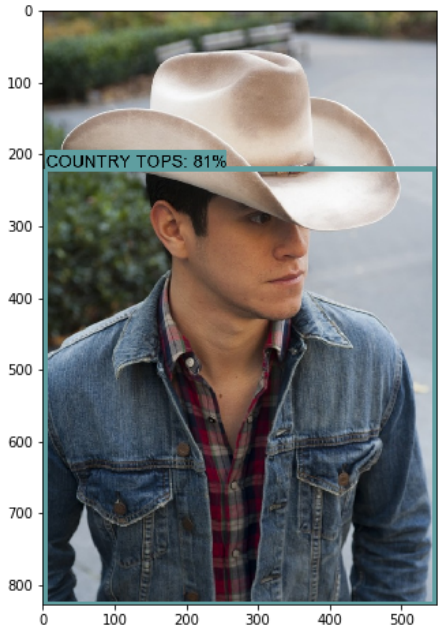

Figura 3.29: Top Country

classico,con tasche ec..), in più molto spesso i pantaloni sono soovrapposti ad altri indumenti come giacconi che non ne permettono un identificazione.

- COUNTRY VESTS: per quanto riguarda i gilet Country il mio modello difficilmente lo individua poichè in quasi tutti i casi, il gilet è sottostante al giaccone ed è abbinato ad esso. Di conseguenza molto spesso il modello confonde questi due capi.

- COUNTRY GLOVES :i guanti sono difficilmente classificabili per il mio modello perchè, presumo, non abbia avuto molte immagini per il training.

3.6 Statistica descrittiva

Infine ho fatto uno studio dei risultati ottenuti dal classificatore attraverso la statistica descrittiva, ovvero quella parte della statistica che si occupa della descrizione di un insieme di dati.
I dati presi in considerazione sono di tipi *quantitativo*, cioè che sono associabili a delle osservazioni (immagine classificata); i dati quantitativi possono essere discreti (hanno valori in un insieme numerabile) o continui (hanno valori in un intervallo di numeri). È importante descrivere i dati poichè ci consente di avere informazioni fondamentali su ciò che essi rappresentano. Oltre che numericamente, l' insieme di dati può essere rappresentato anche graficamente.
Oggetto dello studio statistico sono state le percentuali di classificazione del modello riguardanti la categoria COUNTRY JACKETS, essendo una classe nella quale la classificazione è avvenuta nella maggior parte dei casi e con un ampio range di accuratezza (tra il 60% ed il 98%). Ho preso come campione significativo la classificazione di 30 immagini di giacconi Country riportando in un file CSV la percentuale di accuratezza con la quale

il modello classificasse l' immagine (ho trasformato il valore in percentuale a decimale). Tramite R Studio, un ambiente di sviluppo per il lionguaggio R, ho importato il file ed eseguito le analisi statistiche.

```
> mioFile <- "R/mioFile.csv"
> File <- read.table(mioFile, header=TRUE, sep= ";")
> value<-File[,2]
> #STAMPO I VALORI DEL DATASET
> print(value)
 [1] 0.64 0.81 0.76 0.00 0.93 0.85 0.56 0.87 0.76 0.85 0.73
[12] 0.00 0.98 0.60 0.84 0.90 0.00 0.00 0.84 0.56 0.90 0.00
[23] 0.78 0.57 0.81 0.74 0.67 0.70 0.00
```

Successivamente ho eseguito le misure del centro, in particolare :

- **Media semplice**: rapporto tra la somma dei dati numerici ed il numero dei dati.

```
> #CALCOLO LA MEDIA
> mean(value)
[1] 0.6086207
```

- **Meidana**: il valore centrale di una serie n di dati ordinati

```
> #CALCOLO LA MEDIANA
> median(value)
[1] 0.74
```

- **Varianza**: Deviazione standard quadratica [3]

```
> #CALCOLO LA VARIANZA
> var(value)
```

[3] indice di dispersione statistico

```
[1] 0.111898
```

- valori **minimo** e **massimo**

```
> #CALCOLO IL VALORE MINIMO PRESENTE NEL DATASET
> min(value)
[1] 0
> #CALCOLO IL VALORE MASSIMO PRESENTE NEL DATASET
> max(value)
[1] 0.98
```

Dalla descrizione numerica del datatset emerge che il valore medio di predizione del COUNTRY JACKETS è 0.60 (quindi 60%), i valori minimi e massimi invece sono 0 e 98%.

Successivamente ho eseguito l' analisi grafica del campione, sempre attraverso le funzioni di R. In particolare ho creato un **istogramma** per vedere la fequenza con la quale i valori di predizione compaiono:

```
> #ISTOGRAMMA FREQUENZA VALORI RISULTANTI
> hist(value, main = "Istogramma valori")
```

Come possiamo vedere, i valori che maggiormente emergono sono quelli compresi nella fascia tra l' 80 ed il 100%; sebbene questa rilevazione possa far pensare che il modello classifica quasi sicuramente l' immagine, vediamo come anche la fascia che va tra lo 0 ed il 20% è molto frequente. Questo dato sottolinea quanto sia importante la struttura dell' immagine con cui si fa testing, poichè si passa da una predizione accurata ad una non-classificazione. L' ultimo grafico costruito è il **plot**, utile per visualizzare come si distribuiscono i valori all' interno del range selezionato.

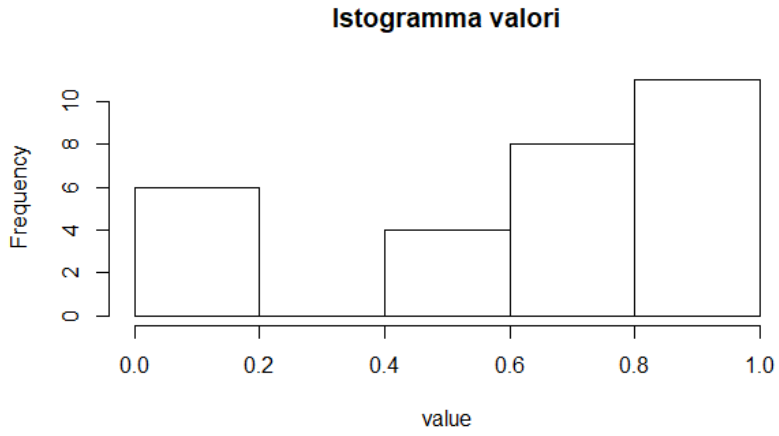

Figura 3.30: Istogramma predizioni

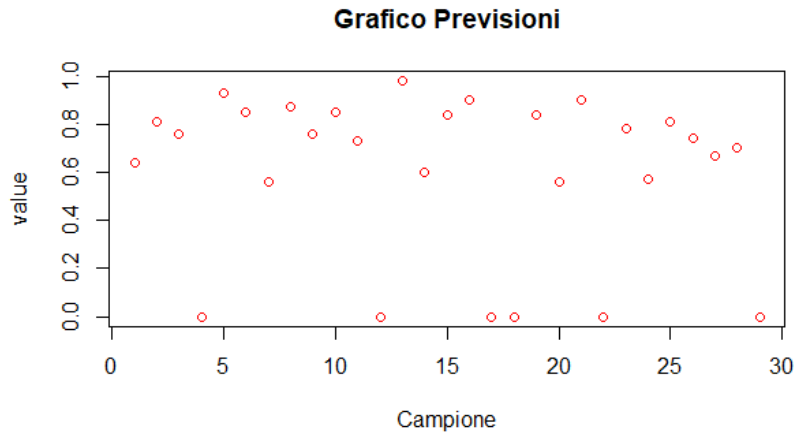

Figura 3.31: Grafico distribuzione campioni

Capitolo 4

Conclusioni

Al termine del lavoro svolto nella tesi si può giungere alla conclusione che il classificatore implementato è più accurato ed abile nel riconoscere alcune categorie di vestiti Country piuttosto che altre. La differenza di livello di predizione, come sostenuto nel capitolo precedente,consiste nella struttura e nella tipologia di indumento che si apprende: per la rete neurale è più facile comprendere categorie che hanno una struttura simile per ogni modello ed una colorazione che appartenga alla stessa scala cromatica omogenea piuttosto che articoli i cui modelli sono differenti per queste caratteristiche. Riassumendo il concetto con un esempio la rete neurale riconosce più facilmente lo stivale Country perchè, nella maggior parte dei casi, quest' ultimo ha sempre lo stesso aspetto (forma allungata, colori scuri e lucidi, tacco posteriore ecc....) a differenza di una borsa la quale può essere di colori (nelle immagini ho inserito appositamente una rossa per esplicitare il concetto) e forme diverse (a tracolla,pochette,con accessori ecc...).

Altro aspetto non meno rilevante è il software e l hardware che si utilizza: per implementare l' applicazione nel mio computer ho dovuto utilizzare ssd_mobilenet_v1_coco, un modello di apprendimento pre-addestrato conforme al sistema operativo Windows;nella git di Tensorflow ci sono modelli la cui velocità e livello di accuratezza è migliore, di

conseguenza dovrebbero restituire risultati più conformi. Inoltre ho scaricato la libreria di Tensorflow che si basa sulla CPU poichè, per processare immagini, è sufficiente la potenza di calcolo dell' unità di elaborazione centrale. Ciò non toglie che si possa utilizzare la libreria di Tensorflow basata sulla GPU che dovrebbe essere più veloce nell' eseguire i calcoli, diminuendo così i tempi del training.

Ultima deduzione dopo il testing riguarda la struttura dell' immagine che si processa nella fase del training: come accennato nel secondo capitolo e verificato nel terzo, la rete neurale ha un miglior apprendimento se riceve in input immagini in cui il soggetto principale sia il capo d' abbigliamento etichettato. Altro aspetto che in alcuni casi complica l' apprendimento è la presenza di un sagoma umana poichè, inevitabilmente, nel momento del Labeling si devono etichettare anche aree dell' immagine nelle quali è presente il rosa della pelle che distorce dalla colorazione dell' articolo.

L' applicazione ha quindi ampi margini di miglioramento, sia per quanto riguarda l' implementazione del software sia per quanto riguarda la struttura del dataset.

Tra gli sviluppi futuri si potrebbe implementare il software in un applicazione per smartphone che sia in grado di suggerire all' utente se comprare o meno un articolo basandosi sul dataset di indumenti già presenti nell' armado di casa, facendo una foto del capo d' abbigliamento che si vuole comprare; ovviamente l' applicazione si estenderebbe anche per altri stili e non solamente per il Country.

Ringraziamenti

E' arrivata la parte più difficile da scrivere della tesi, i ringraziamenti. Riassumere in poche righe l' importanza delle persone che ho incontrato in questo percorso è infattibile, poichè oguno si meriterebbe una pagina a se.
Vorrei innanzitutto ringraziare la Professoressa Piccolomini ed il Professor Marfia, perchè mi hanno dato la possibiltà di approfondire un argomento odierno ed interessante, seguendomi in ogni passo del lavoro.Immenso è il ringraziemnto che voglio fare ai miei genitori: non solo per il sostentamento economico senza il quale questi 4 anni non sarebbero stati possibili, ma soprattutto perchè mi hanno fatto vivere indirettamente esperienze fondamentali che mi hanno formato e che porterò sempre con me; di questo non smetterò mai di ringraziarli. Un altro grazie "DI CUORE" va ai miei compagni di facoltà: se non fosse stato per loro probailmente dopo poco sarei tornato a casa, invece ho trovato amici sinceri con cui passare 4 anni bellissimi, sempre pronti a darmi una mano sia a livello di studio sia a livello di svago. I particolare il mio ringraziamento va ad Anna, Gallo ed Enzino, che probailmente lo santificheranno per la pazienza che gli ho fatto consumare. E poi ci sono loro due, Manny e Luca, compagni di viaggio che sono diventati fratelli insostituibili senza i quali sarebbe stato tutto più difficile. E poi Sanfo,Pietro Marche e tutti gli altri, grazie di cuore.
Ringrazio tutti i miei amici di giù: Fiorè,Amianto,Mattia,Squ,Frankie,Edo e tutti gli

altri che ritrovo con piacere quando scendo; gli amici di scuola Ale e Nico; ringrazio Enrico, i ragazzi di "Roberta" e tutta la banda di Petriolo e tutti gli altri amici con cui ho mantenuto un legame distanza permettendo.

Ringrazio i miei coinquilini,vecchi e nuovi, Giovanni e tutti gli altri scapestrati dello studentato. Un altro grazie va a chi mi ha permesso di spernacchiare col trombone in questi anni: i ragazzotti della Roveri, la Gas Gas Band,l'orchestra dell'Università e tanti altri.

Insomma, GRAZIE A TUTTI.

Bibliografia

[1] https://en.wikipedia.org/wiki/Deep_learning

[2] http://www.simonefavaro.it/2017/04/07/introduzione-machine-learning/

[3] http://www.intelligenzaartificiale.it/deep-learning/

[4] https://www.ai4business.it/intelligenza-artificiale/deep-learning/reti-neurali/

[5] http://www.dsi.unive.it/ srotabul/files/AppuntiRetiNeurali.pdf

[6] https://www.datacamp.com/community/tutorials/object-detection-guide

[7] https://ieeexplore.ieee.org/document/6574671

[8] Large Scale Visual Recommendations From Street Fashion Images,Vignesh Jagadeesh, Robinson Piramuthu, Anurag Bhardwaj, Wei Di, Neel Sundaresan eBay Research Labs, 2065 East Hamilton Avenue, San Jose, CA-95128

[9] CHIC: A Combination-based Recommendation System Manasi Vartak,Samuel Madden

[10] Style Recommendation for Fashion Items using Heterogeneous Information Network Hanbit Lee,Sang-goo Lee

[11] Machine Learning: predire le scelte del consumatore, Marco Silvestri

[12] Sviluppo di una applicazione per il riconoscimento di capi d' abbigliamento in collezioni fotografiche in ambito di moda, Giuseppe Danesi

[13] A tu per tu col Machine Learning,Alessandro Cucchi