

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Fondamenti di Data Science e Machine Learning

SocialMapper

Chiara Menniti, Filippo Avagliano

ANNO ACCADEMICO 2022/2023

INDICE

1	Introduzione	3
1.1	Scopo del progetto	3
1.2	Decisioni Progettuali	5
2	Stato dell'arte	7
2.1	Object Detection	7
2.1.1	Logo Detection	9
2.1.2	Optical Character Recognition	9
3	Reti Utilizzate	11
3.1	YOLOv5	11
3.1.1	Architettura single-stage object detectors	12
3.2	Logohunter	20
3.3	CarRecognition	21
3.4	EasyOCR	22

INDICE

4 Implementazione	25
4.1 Creazione di un unico ambiente	25
4.2 Download delle immagini	26
4.3 YOLO	27
4.4 CarRecognition	28
4.5 Logohunter	28
4.6 EasyOCR	30
4.7 Creazione output	30
4.8 Email	32
5 Conclusioni	38

CAPITOLO 1

INTRODUZIONE

1.1 Scopo del progetto

Instagram è uno dei social media più utilizzati a livello globale; dal 2010 milioni di utenti - il miliardo di utenti attivi è stato raggiunto tra il 2021 e il 2022 - pubblicano foto dei loro momenti più interessanti, dei propri eventi sociali, taggando a loro volta altri utenti, lo utilizzano come canale di promozione delle proprie attività commerciali e hobbi-stiche.

Conoscenza comune è che l'utilizzo di qualsiasi sito web, non limitandosi ai social media, comporti la condivisione di informazioni, dai propri dati anagrafici alla propria posizione - per quest'ultima è sufficiente un'analisi della *request* di connessione iniziale - ma ciò che distingue

1. INTRODUZIONE

Instagram da molti altri social media è il quasi completo affidamento alle immagini, molto più foriere di verità di un post testuale o di un tweet.

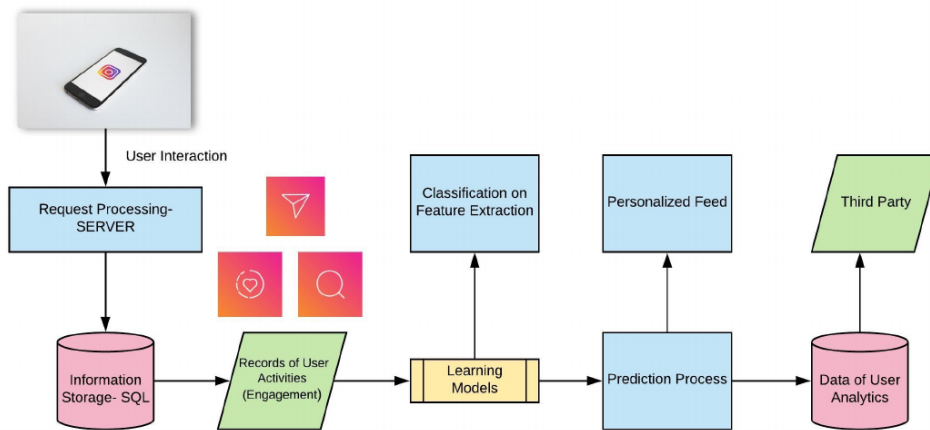


Figura 1.1: Pipeline di Instagram [1]

Pertanto, della *userbase* totale, solo in pochi sembrano essere veramente coscienti del numero di informazioni che ogni giorno condividono spontaneamente mediante la loro attività e di come queste possano essere utilizzate, per scopi leciti o illeciti, dalla piattaforma stessa e da eventuali terze parti. Instagram, infatti, fa utilizzo di sofisticati algoritmi di machine learning per prelevare da ogni immagine i potenziali interessi dell'utente, che possono essere utilizzati (insieme ai *tags* da lui aggiunti) per personalizzare la sua esperienza, mostrandogli i contenuti che si ritengono possano coinvolgerlo emotivamente o per curiosità e spingerlo a spendere quantità sempre maggiori del proprio tempo sulla piattaforma, una serie di azioni che nel marketing si definisce *engagement*.

Le informazioni estratte non restano sotto il controllo di Instagram ma sono successivamente rivendute ad attività commerciali, che ne posso-

1. INTRODUZIONE

no fare uso per pubblicità mirata: *se non stai pagando per un prodotto, allora il prodotto sei tu.*

1.2 Decisioni Progettuali

I sistemi di reti utilizzati sono quattro.

YOLOv5 è un sistema di riconoscimento di oggetti addestrato sul dataset **COCO** - permette di isolare oggetti generici ricorrenti all'interno delle immagini ed estrarre una regione interessata per successive elaborazioni. Con questa rete è possibile identificare interessi o caratteristiche generiche dell'utente.

Logohunter è un sistema basato su YOLOv3, il cui scopo è individuare loghi appartenenti a marche commerciali. Dopo aver selezionato gli elementi che sono identificati come tali da YOLOv3, il sistema esegue un confronto tra questi e delle immagini di alta qualità di loghi dati in input. Questo sistema è uno degli strumenti di analisi più importanti nel nostro contesto, in quanto fornisce le informazioni necessarie per costruire pubblicità mirata.

CarRecognition prende in input immagini di automobili da prospettiva variabile e ne riconosce il modello.

EasyOCR è un sistema di OCR con supporto ad oltre 80 lingue, basato su PyTorch.

1. INTRODUZIONE

Seguono i cinque account selezionati per questo lavoro, con username completo e valido a Dicembre 2022:

- Andrea Galeazzi (@andreagaleazzi)
- Emelia Hartford (@ms.emelia)
- Agriturismo Ferdy (@ferdy_wild)
- Tyler McElhaney (@tynology)
- LeBron James (@lebron)

Ognuno di essi è specializzato in un determinato ambito (es. automobili), in modo da massimizzare le informazioni prelevabili da ogni rete. Con lo stesso scopo, come caso di studio, per ciascun utente è stato selezionato un campione di 20 fotografie.

Il progetto presenta inoltre uno *scheduler* con intervallo di tre ore, che, dopo aver eseguito il login sulla piattaforma, esegue uno scaricamento multiplo di immagini di un profilo dato in input tramite la libreria *instagrapi*. Per ridurre la complessità, si è scelto di scaricare e lavorare su foto singole, ma eventuali sviluppi futuri possono prevedere l'elaborazione di post più complessi quali album e *reel* e allargare la dimensione del campione.

CAPITOLO 2

STATO DELL'ARTE

2.1 Object Detection

Il riconoscimento di oggetti - *object detection* - all'interno di immagini è stato, e continua ad essere, uno degli argomenti più studiati dall'avvento del Deep Learning. In generale, le metodologie più utilizzate possono essere classificate in due tipi: ***one stage*** e ***two stages***. Nell'approccio *one stage* l'isolamento dei singoli oggetti e la loro eventuale classificazione avvengono nello stesso *step*; al contrario, nell'approccio *two stages* i due compiti sono svolti in passi successivi.

Per quanto riguarda le metodologie *one stage*, una delle reti più importanti proposte è sicuramente **YOLO**, che al momento rappresenta

2. STATO DELL'ARTE

lo stato dell'arte dell'*object detection* in tempo reale e di cui si parlerà in dettaglio nelle sezioni successive.

La tecnica **Single Shot MultiBox** è anch'essa di tipo *one stage*; proposta da Liu et al. [2], si basa sull'architettura VGG-16, a cui è stata aggiunta una serie di livelli ausiliari per migliorare l'estrazione di *features*.

Le tecniche **Fast R-CNN** [3] e **Faster R-CNN** [4] sono tecniche *two stages*. La prima fase è chiamata *region proposal*: l'immagine viene data in input ad una CNN in modo da generare una mappa di caratteristiche, da questa viene generata una "proposta" di regioni dell'immagine su cui verrà eseguita la classificazione. La seconda fase ha lo scopo di perfezionare la classificazione stessa. Nella Faster R-CNN la proposta delle regioni viene eseguita da una rete separata.

La tecnica **Feature Pyramid Networks** [5] è stata applicata al riconoscimento di oggetti per la prima volta da Lin et al. e consiste nella costruzione di una piramide formata dalla medesima immagine, prelevata più volte con dimensioni sempre minori. Da ogni livello sono estratte individualmente le varie *features*, che vanno a creare a loro volta una piramide di caratteristiche. Questa architettura migliora il riconoscimento di oggetti a diversa scala.

RetinaNet [6] si basa sulla tecnica precedente ma aggiunge una funzione di loss detta *focal loss* per migliorare la classificazione.

2.1.1 Logo Detection

Il riconoscimento dei loghi è un ambito specializzato del riconoscimento di oggetti ma la sua vasta gamma di applicazioni in ambito commerciale lo ha reso oggetto di studio indipendente. La maggior parte delle tecniche utilizzate - inclusa quella presentata in questo lavoro - si basa sull'utilizzo di una prima rete per l'individuazione di oggetti identificabili come loghi; successivamente si procede con la loro classificazione.

In **Large Scale Open-Set Deep Logo Detection** [7] sono utilizzate le reti VGG16 e RFBNet, quest'ultima addestrata mediante un dataset di loghi con *bounding box*. Non sono utilizzate etichette relative alle classi, ma si cerca un *match* tra l'immagine di input e i loghi nel dataset.

Un approccio simile è proposto da Fehérvári et al. in **Scalable Logo Recognition using Proxies**, [8] che nei loro esperimenti utilizzano ResNet50 e YOLOv3 per il riconoscimento di possibili loghi. L'identificazione dei loghi trovati è stata affidata alla rete SE ResNet.

2.1.2 Optical Character Recognition

Con OCR si indica il riconoscimento ottico di caratteri, in forma di stampa o scritti a mano, all'interno di un'immagine e nella loro conversione in un formato *machine readable*.

Levenshtein OCR [9] è una tecnica innovativa che vede il processo di riconoscimento come una sequenza iterativa - la predizione

2. STATO DELL'ARTE

iniziale è l'input di un transformer che, insieme alle *features* visive, viene utilizzato per il risultato finale.

Tian et al. propongono in **Detecting Text in Natural Image with Connectionist Text Proposal Network** [10] una rete basata su VGG-16 per l'estrazione iniziale di *feature*, a loro volta input per una RNN. Il risultato sono delle *anchor box* (*text proposal*), accompagnate da una possibile classificazione e delle coordinate di raffinamento del risultato.

Una metodologia più tradizionale è infine data da **TextBoxes++ (TextBoxes++: A Single-Shot Oriented Scene Text Detector)** [11], che prevede anch'essa l'utilizzo di VGG-16, seguita da da 6 livelli aggiuntivi di convoluzione.

CAPITOLO 3

RETI UTILIZZATE

3.1 YOLOv5

YOLO è lo stato dell'arte degli algoritmi sul riconoscimento degli oggetti in tempo reale, creata nel 2015 e pre-addestrata sul dataset COCO. È costituita da una singola rete neurale, divisa in componenti, per processare un'intera immagine. L'immagine è divisa in regioni e l'algoritmo predice la probabilità di un oggetto e definisce dei riquadri (*bounding box*) per ogni regione. È ben conosciuta per la sua velocità e accuratezza ed è stata usata per molte applicazioni come: salute, sorveglianza di sicurezza e macchine a guida autonoma.

3. RETI UTILIZZATE

3.1.1 Architettura single-stage object detectors

Le architetture *single-stage* (come YOLO) sono composte da tre componenti:

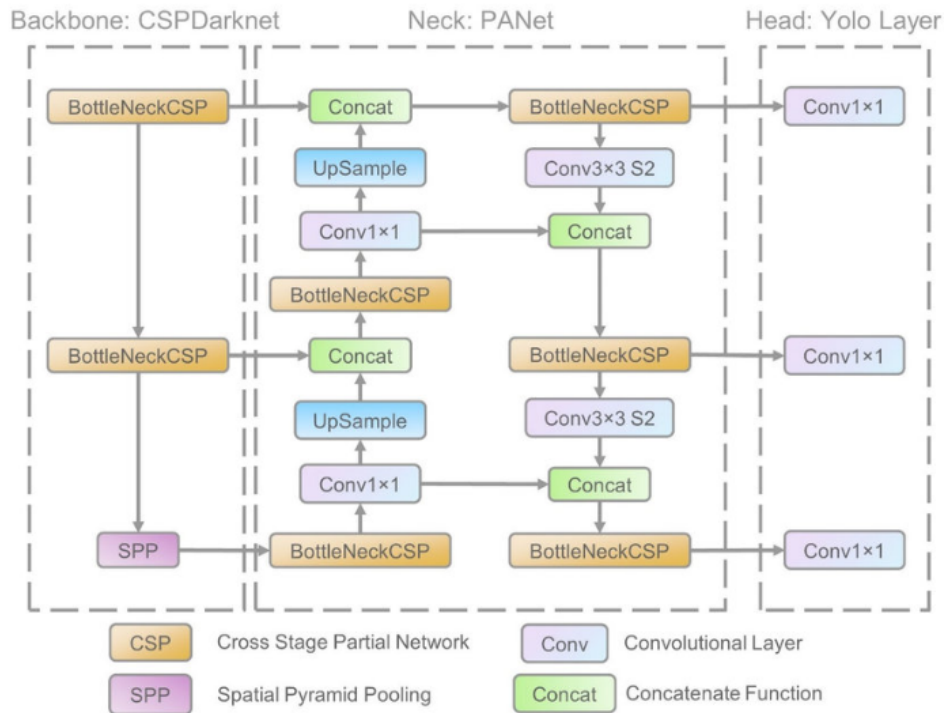


Figura 3.1: Architettura single-stage detector [12]

- **Model Backbone:** CSPDarknet (Cross Stage Partial) è una rete pre-addestrata usata per estrarre una ricca rappresentazione delle caratteristiche per le immagini. Questo aiuta a ridurre la risoluzione spaziale dell'immagine ma permette di aumentare la risoluzione delle caratteristiche (canali).
- **Model Neck:** PANet (Path Aggregation Network) viene utilizzato per estrarre caratteristiche piramidali. Questo aiuta il modello a generalizzare bene su oggetti con dimensioni e scale diverse.

3. RETI UTILIZZATE

- **Model Head:** utilizzato per eseguire le operazioni della fase finale. Applica i riquadri di selezione alle *feature maps* e restituisce l'output finale: classi, punteggio e riquadri che delimitano l'area dell'oggetto riconosciuto.

In ordine, la rete CSPDarknet riceve i dati in input per l'estrazione di *feature*, che poi sono inviate a PANet per la *feature fusion*. Infine, YOLO Layer produce i risultati del rilevamento (classe, punteggio, posizione, dimensione).

YOLOv5 è stato rilasciato con cinque diverse dimensioni. Come

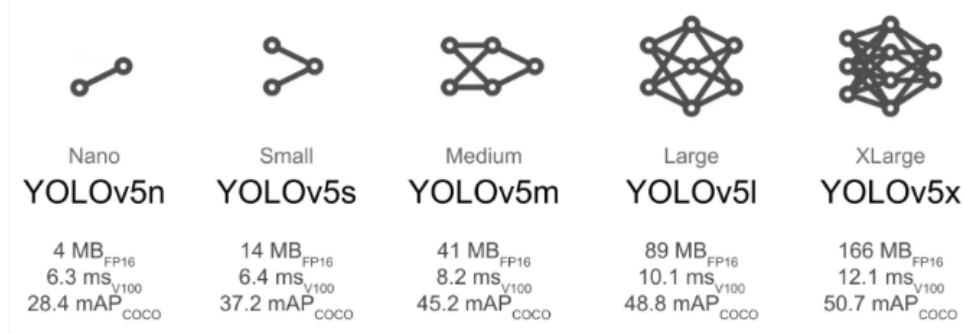


Figura 3.2: Modelli YOLOv5 [13]

si può notare dal grafico in figura 4, a mano a mano che aumentano il numero di livelli e parametri, aumenta anche sensibilmente il tempo impiegato per riconoscere gli oggetti ma allo stesso tempo si ha un modello migliore in termini di precisione nel riconoscimento degli stessi.

Tutti questi modelli, nonostante le loro differenze, seguono sostanzialmente la struttura presentata.

3. RETI UTILIZZATE

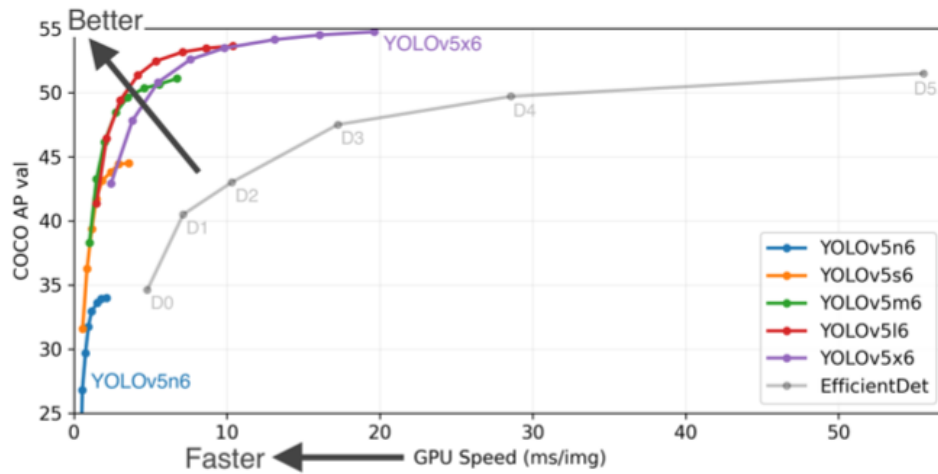


Figura 3.3: Grafi di YOLOv5[13]

CSP-Darknet

Essendo una *deep network*, YOLO utilizza *residual block* e *dense block* per consentire il flusso di informazioni diretto agli strati più profondi e per superare il problema della scomparsa del gradiente (*vanishing gradient*).

Vanish gradient problem è un problema che talvolta si presenta quando si addestrano algoritmi di apprendimento automatico tramite *gradient descent*; si verifica soprattutto nelle reti neurali di apprendimento profondo, ma anche nelle RNN. Le derivate parziali ottenute ad ogni passo servono a calcolare il gradiente quanto più si va in profondità nella rete. Poiché i gradienti controllano quanto la rete apprende durante l'addestramento, se questi sono molto piccoli o nulli, l'addestramento può essere minimo o nullo, con conseguenti scarse prestazioni predittive.

In particolare, il modello CSP è basato su DenseNet. Nella rete **Dense-**

3. RETI UTILIZZATE

Net, al contrario di quanto avviene nelle CNN tradizionali, ogni livello convoluzionale prende in input l'output di tutti i livelli precedenti quindi ogni livello è connesso ad ogni altro. Per L livelli vi sono quindi $\frac{L(L+1)}{2}$ connessioni.

Questa struttura:

- favorisce la propagazione delle caratteristiche,
- incoraggia il riutilizzo delle stesse,
- riduce il numero di parametri della rete.

Tuttavia, uno dei svantaggi dell'uso di blocchi densi e residui è la ridondanza delle informazioni del gradiente; con la strategia Cross Stage Partial si riduce la quantità eccessiva di informazioni, troncando il flusso del gradiente stesso.

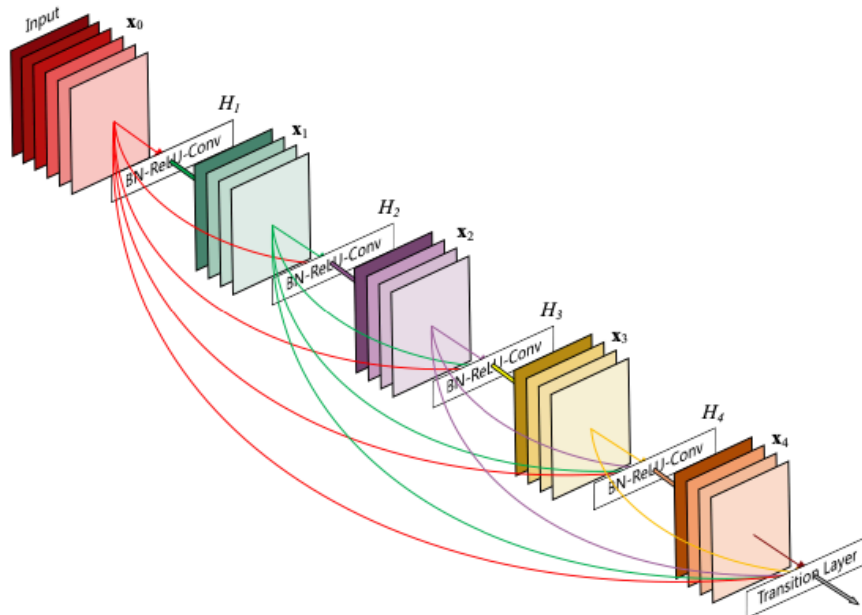


Figura 3.4: Layers nella rete DenseNet [14]

Path Aggregation Network

È una rete con caratteristiche piramidali, è stata utilizzata nella versione precedente di YOLO per migliorare il flusso di informazioni e per aiutare nella corretta localizzazione dei pixel nel compito di collocazione del riquadro. In YOLOv5 questa rete è stata modificata con l'applicazione della strategia CSPNet come mostrato nella figura dell'architettura della rete.

Pyramid Pooling (SPP)

Poiché l'operazione di convoluzione viene applicata con una finestra scorrevole, può accettare input di dimensioni diverse, con conseguente output variabile. Le CNN, invece, hanno i livelli completamente connessi tra loro e possono accettare input di dimensioni fisse. Quindi, le immagini vengono rimodellate in una dimensione specifica prima di essere inserite in una CNN, e si viene a creare un'ulteriore deformazione dell'immagine con una inevitabile riduzione della risoluzione. SPP rappresenta la soluzione a questo problema, perché mantiene le informazioni spaziali in **bins** (sezioni) spaziali locali. Il numero di *bins* e la loro dimensione sono fissi per ogni livello. In ogni *bin* spaziale vengono raggruppate le risposte di ciascun filtro. Nell'immagine seguente viene eseguito un *pooling* a tre livelli:

- Nel primo livello di *pooling*, l'output ha un singolo bin e copre un'immagine completa.
- Nel secondo livello, la mappa delle caratteristiche viene raggruppata in modo da avere 4 *bins*.

3. RETI UTILIZZATE

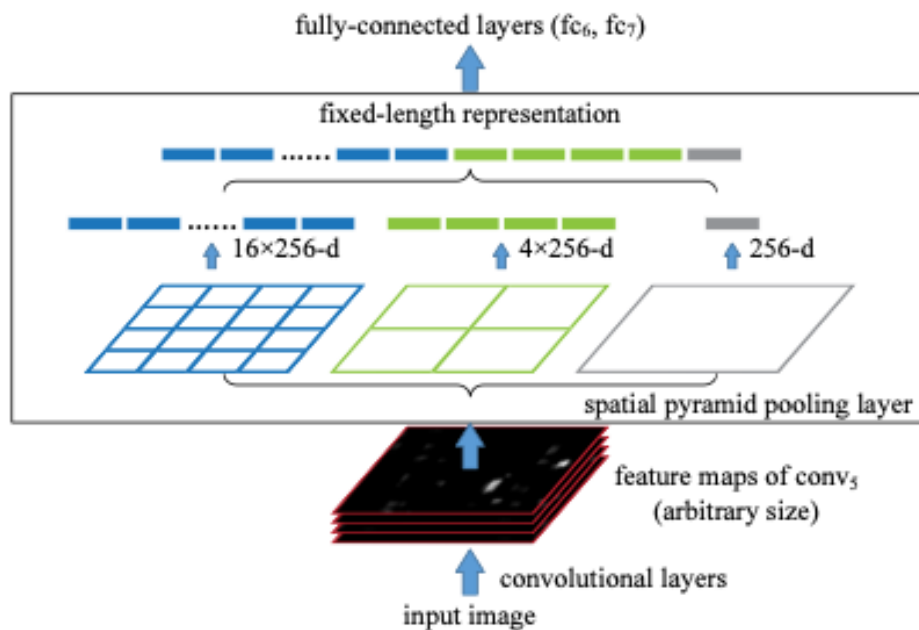


Figura 3.5: Suddivisione in sezioni in SPP [15]

- Nel terzo livello, la mappa delle caratteristiche è ridotta ad una sola regione.

L'output di tutti i livelli è appiattito e concatenato per ottenere un output di dimensione fissa, indipendentemente dalle dimensioni di input.

Il blocco SPP esegue un'aggregazione delle informazioni che riceve dagli ingressi e restituisce un'uscita di lunghezza fissa. In questo modo ha il vantaggio di aumentare significativamente il campo recettivo e di segregare le caratteristiche contestuali più rilevanti senza ridurre la velocità della rete. Questo blocco viene utilizzato anche nella versione 3 di YOLO dalla *backbone* per separare le caratteristiche più importanti. Invece in YOLOv5 è stato utilizzato SPPF, che è solo un'altra variante del blocco SPP, per migliorare la velocità della rete - come

3. RETI UTILIZZATE

indica il nome stesso, in cui la "F" finale è l'aggettivo *fast*.

Funzione di attivazione

Per YOLOv5 gli autori hanno scelto la funzione di attivazione SiLU e Sigmoid.

- SiLU è l'acronimo di **Sigmoid Linear Unit**, in matematica è indicata come funzione di attivazione Swish. È utilizzata negli strati nascosti insieme alle operazioni di convoluzione.

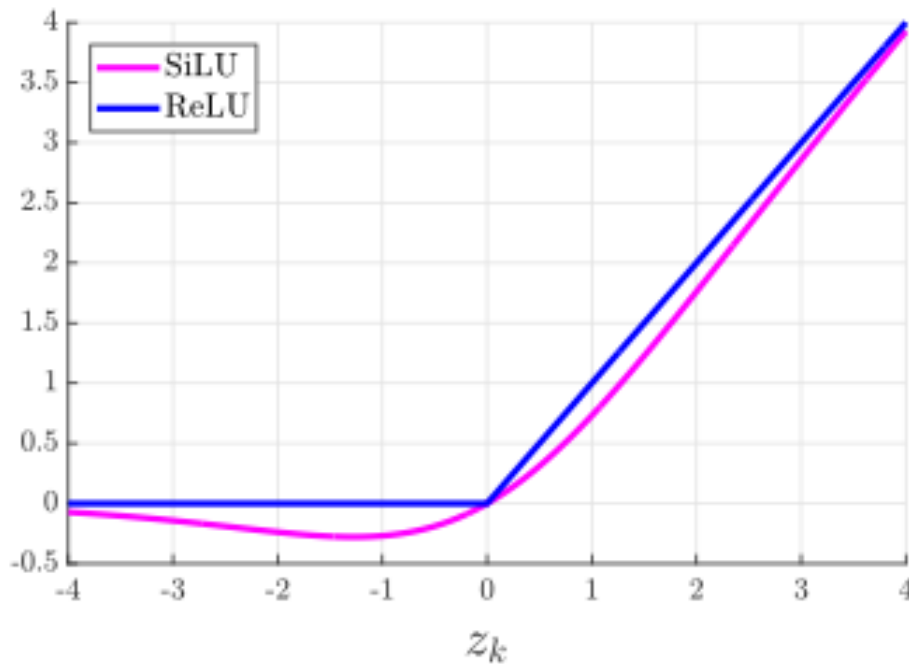


Figura 3.6: SiLU activation function [16]

- Al contrario, la funzione di attivazione Sigmoid è utilizzata nello strato di uscita.

3. RETI UTILIZZATE

Loss Function

YOLOv5 restituisce tre output: la classe degli oggetti rilevati, i loro *bounding boxes* e il punteggio. Così, la rete usa BCE (Binary Cross Entropy) per calcolare la loss sugli oggetti e sulle classi.

YOLOv5 vs YOLOv3

Oltre a YOLOv5 utilizzata è stata utilizzata anche la rete YOLOv3, in particolare in Logohunter.

Le principali differenze tra YOLOv5 e YOLOv3 sono due [17]:

- **Backbone feature extraction:** YOLOv5 usa CSP-Darknet come struttura *backbone*; infatti in YOLOv3, nonostante fosse anche questa basata su Darknet, non era presente la strategia di rete Cross Stage Partial.
- **Neck:** in YOLOv3 viene utilizzata la Feature Pyramid Networks [18], sviluppata nel 2017 dalla Facebook Artificial Intelligence Research. È una topologia in cui una mappa di caratteristiche diminuisce gradualmente nella dimensione, ma aumenta di nuovo e viene concatenata con mappe di caratteristiche precedenti con dimensioni corrispondenti. Le mappe di caratteristiche con dimensione diversa vengono quindi inviate ad una diversa *head*. Come si può vedere dall'immagine seguente, YOLOv3 utilizza 3 *head* di rilevamento.

3. RETI UTILIZZATE

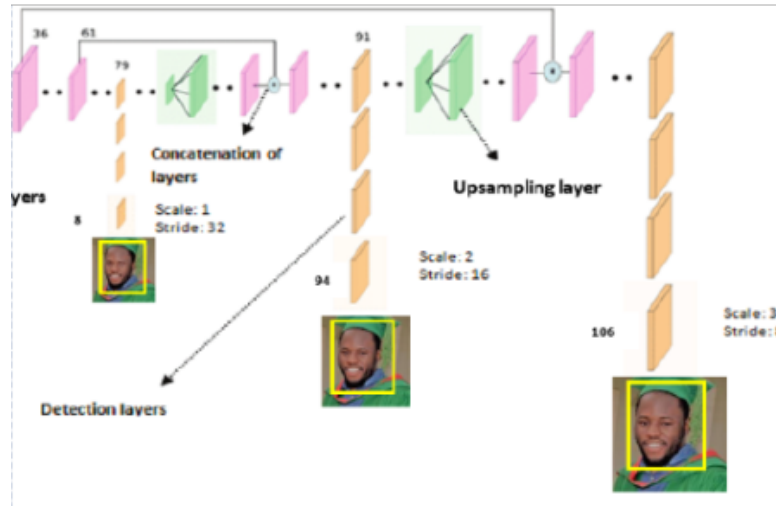


Figura 3.7: Feature Pyramid Network (FPN) [18]

3.2 Logohunter

La rete Logohunter segue il processo indicato da Herrmann et al. [19], basato sulla seguente strategia:

- Rilevamento dei loghi all'interno di un'immagine attraverso la rete YOLOv3 preaddestrata sul dataset *LogosInTheWild*, utilizzando un'implementazione tramite Keras [20]
- Identificazione dei loghi tramite la verifica della somiglianza tra i loghi presenti nel dataset e quelli nelle immagini, con l'utilizzo della funzione di distanza del **coseno** applicata sulle features di entrambi; l'estrazione delle caratteristiche avviene grazie ad una seconda rete preaddestrata.

Il dataset *LogosInTheWild* è stato creato da Herrmann et al. e comprende 871 brands per un totale 32850 box annotate manualmente. I loghi inoltre, sono riprese da diverse angolazioni e hanno dimensioni

3. RETI UTILIZZATE

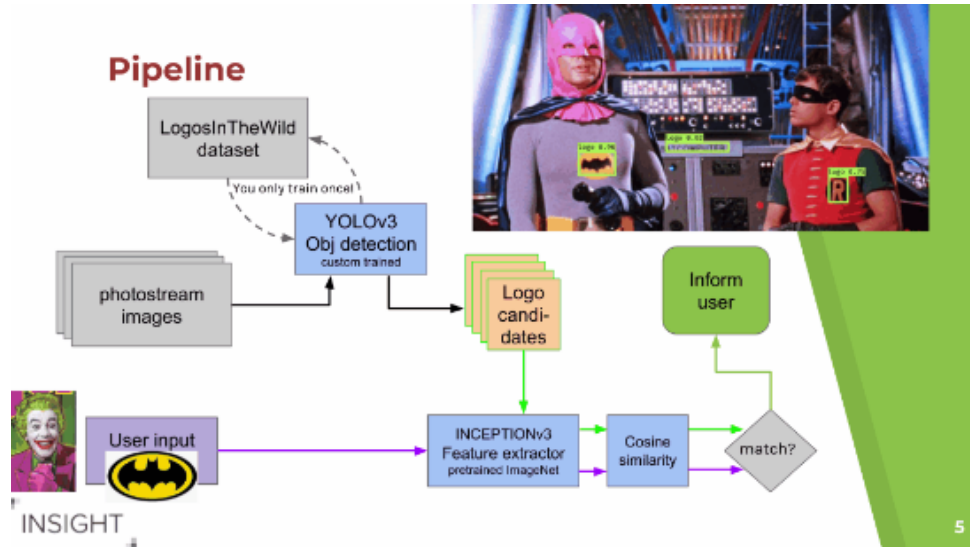


Figura 3.8: Pipeline [21]

variegate.

Per l'estrazione delle caratteristiche di questi ultimi - oltre che delle immagini di input - si è optato per la rete *InceptionV3*; introdotta in **Rethinking the Inception Architecture for Computer Vision** [22] e creata per essere efficiente e scalabile mediante l'utilizzo di convoluzioni asimmetriche, è oggi utilizzata soprattutto per la classificazione.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}$$

Figura 3.9: Formula della similarità del coseno

3.3 CarRecognition

La rete *CarRecognition* è un'implementazione della rete Resnet-152 preaddestrata sul dataset *Cars Dataset* di Stanford, contenente

3. RETI UTILIZZATE

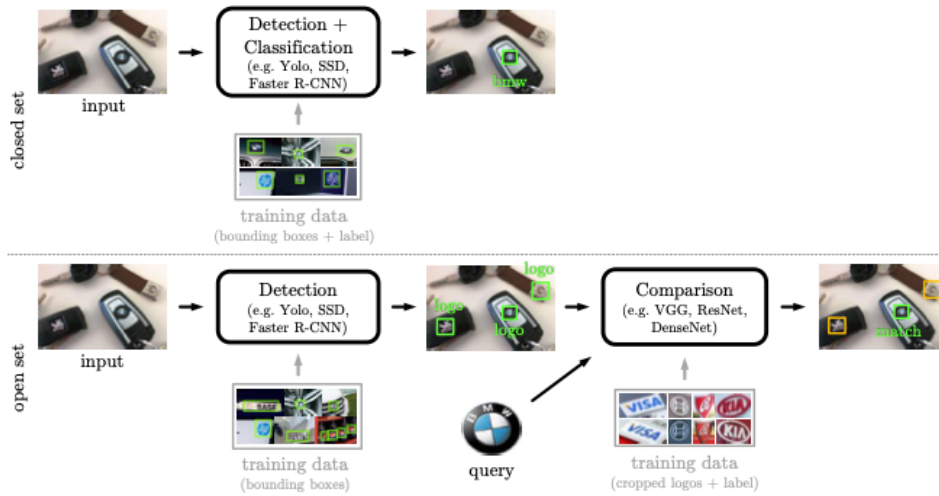


Figura 3.10: Metodologie precedenti vs metodologie correnti di *logo detection* [19]

16.185 immagini per 196 classi in totale.

La ResNet-152 è una CNN a 152 livelli organizzata in blocchi chiamati "*residual blocks*". Nasce come possibile soluzione ai problemi di *exploding gradient* o *vanishing gradient*, attraverso una tecnica chiamata *skip connection*, connessioni tra livelli non adiacenti.

3.4 EasyOCR

EasyOCR [24] è un sistema *open source* implementato tramite PyTorch. La fase di rilevamento del testo (detection) avviene tramite l'algoritmo CRAFT (*Character Region Awareness For Text Detection*) [25]: è una CNN che data in input una immagine restituisce un punteggio sulla regione in cui rileva un carattere singolo (*region score*) e un punteggio per raggruppare i caratteri in un unico testo (*affinity score*). Per il

3. RETI UTILIZZATE

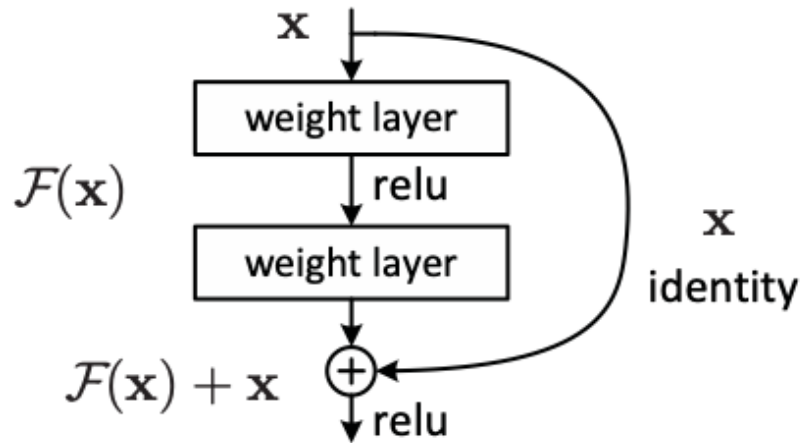


Figura 3.11: Residual block [23]

riconoscimento dei caratteri fa utilizzo di un sistema di apprendimento supervisionato.

Invece per il riconoscimento del testo globale è utilizzata una rete neurale convoluzionale ricorrente (CRNN) [26] così costituita:

- livelli di convoluzione che estraggono le *features* dalle immagini,
- livelli ricorrenti che fanno predizioni per ogni *frame* della sequenza di *feature* (LSTM) ,
- livello di trascrizione che traducono le predizioni in una sequenza di etichette.

Nell'implementazione EasyOCR il modulo di estrazione di caratteristiche è costituito da una ResNet.

3. RETI UTILIZZATE

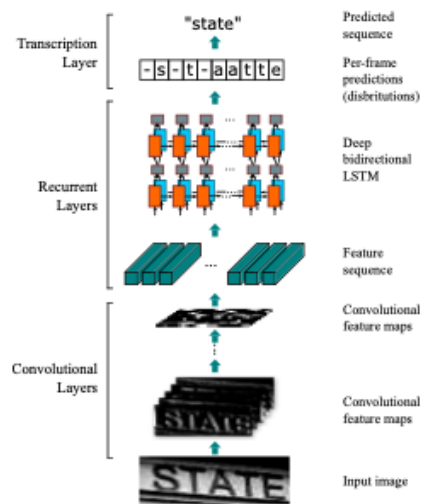


Figura 3.12: Architettura CRNN [26]

CAPITOLO 4

IMPLEMENTAZIONE

4.1 Creazione di un unico ambiente

Il primo passo è stata la creazione di un unico ambiente in cui permettere l'esecuzione di tutti i sistemi in successione; questi presentano dipendenze diverse e, in alcuni casi, incompatibili. Infatti le reti YOLOv5 e EasyOCR sono basate su PyTorch con versioni rispettivamente 1.7.0 e superiore a 1.4.0; la rete Logohunter fa utilizzo di PyTorch (a causa dell'affidamento a YOLOv3) e di TensorFlow (InceptionV3) per l'estrazione di features dai loghi di input; infine la rete CarRecognition utilizza la rete ResNet-152 via TensorFlow.

Questo ha imposto la creazione di un ambiente in cui fossero presenti sia TensorFlow, sia PyTorch, e il prelevamento completo delle reti (con

4. IMPLEMENTAZIONE

l'eccezione di EasyOCR, installato tramite pip) da GitHub in modo da permettere modifiche *ad hoc* al sorgente.

Una delle principali problematiche è stata la dipendenza di Logohunter dalla versione 2.2.4 di Keras, API di alto livello per Tensorflow, di cui supporta esclusivamente la versione 1.x; al contrario, per la rete Car-Reognition è stata espressa una preferenza per TensorFlow 2.x.

L'ambiente corrente utilizza le versioni più recenti di Keras e TensorFlow, per entrambi 2.11.0; la compatibilità con Logohunter è stata garantita rimuovendo il supporto per GPU multiple, che presentava codice specifico per ogni versione di TensorFlow, e modificando l'*import* di alcune librerie.

Successivamente, sono stati modificati i *path* all'interno dei sorgenti in modo da permettere l'esecuzione da un main principale per l'intero progetto; questa avviene tramite la funzionalità *system* della libreria *os*, che invoca la funzione principale di ogni sistema con eventuali parametri.

Ulteriori modifiche ai file di output sono trattate nella sezione dedicata a ciascuna rete.

4.2 Download delle immagini

Per il download delle immagini si utilizza la libreria *instagrapi* [27], scelta in quanto non presenta dipendenze da eventuali cookie all'interno del browser; inoltre le API fornite sono aggiornate all'anno 2022 e aggirano eventuali controlli anti-bot di Instagram. La piattaforma infatti presenta numerosi controlli al fine di evitare operazioni di *scraping* e download di massa: le sue API ufficiali sono sottoposte a

4. IMPLEMENTAZIONE

frequenti cambiamenti nei metodi, mentre il *layout* HTML delle pagine prevede modifiche invisibili all'utente finale, quali il nome delle classi CSS o l'id degli elementi, ma che comprometterebbero il funzionamento di uno *scraper*.

instagrapi, dopo una procedura automatica di login, permette il download di qualsiasi tipo di media pubblicabile sulla piattaforma, ma in questo studio si è deciso di prelevare solo immagini singole.

Dopo aver selezionato un alto numero di media recenti - in generale tra i 80 e i 200 - si isolano le foto e si procede con uno scaricamento limitato, relativo a 20 immagini campione. Nonostante l'alta affidabilità della libreria, a volte è possibile imbattersi nei controlli sui login da parte di Instagram, rendendo necessario l'inserimento di un codice di verifica inviato alla mail dell'account utilizzato.

4.3 YOLO

La prima rete ad essere utilizzata è YOLOv5. Le impostazioni utilizzate sono le seguenti:

- -save-crop: salva le regioni contenenti ogni oggetto in un'immagine separata dello stesso formato, oggetti appartenenti allo stesso tipo sono raggruppati in una cartella che prende il nome dell'etichetta corrispondente;
- -save-txt: salva il risultato relativo ad ogni immagine in un file *.txt*;
- -save-conf: salva il grado di confidenza di ogni oggetto riconosciuto all'interno del file di output.

4. IMPLEMENTAZIONE

Il file sorgente di generazione output per una singola immagine è stato inoltre modificato per ottenere il nome della classe individuata (in origine, questo prevedeva solo l'id della stessa) ed eseguire un semplice conto delle volte che ogni oggetto è stato trovato.

4.4 CarRecognition

La rete CarRecognition è strettamente dipendente da YOLO in quanto esige che ogni fotografia di input contenga una sola automobile. Questa condizione è rispettata facilmente grazie all'operazione di ritaglio eseguita in precedenza da YOLO: se per le immagini di un utente è stata generata una cartella *cars*, gli elementi al suo interno costituiranno gli input di CarRecognition.

L'output della rete consiste in un singolo file *.json* per utente, contenente un vettore di risultati. Ogni elemento del vettore era in origine composto dal modello riconosciuto e dal grado di confidenza del risultato, ma è stato modificato per indicare anche il percorso del ritaglio di origine.

4.5 Logohunter

L'estrazione delle caratteristiche dei loghi per il confronto con i dati di input avviene a tempo di esecuzione; questo ha reso necessaria la scelta di un insieme ridotto di loghi da proporre alla rete per non sovraccaricare le risorse hardware. Abbiamo quindi deciso di non utilizzare il dataset *LogosInTheWild* che, a causa della massiccia presenza di fotografie da diverse angolazioni, avrebbe richiesto una selezione manuale dei loghi in posizione frontale e un'estesa manipolazione delle

4. IMPLEMENTAZIONE

immagini rimanenti.

Abbiamo invece scelto il dataset *Logos-images-dataset* [28], su cui è stata effettuata una selezione di rimozione di tutti i *brand* non presenti in *LogosInTheWild*. In seguito è stata effettuata una pulizia del database eliminando immagini corrotte, rimuovendo le trasparenze tramite la libreria *OpenCV*, e portando tutte le immagini a tre canali. Sono stati generati due insiemi di loghi, *logos* e *logos_complete*: la prima contiene due varianti di logo per ogni marca, la seconda sette varianti; prevedibilmente, l'accuratezza del sistema migliora fornendo un maggior numero di varianti.



(a) Due varianti



(b) Sette varianti

Figura 4.1: Confronto dei risultati

Le opzioni di esecuzione scelte prevedono il salvataggio del risultato in un file di testo *.txt* tramite l'opzione *-outtxt* e l'impostazione di una soglia di confidenza (*-confidence*) pari a 0.5. Tale parametro indica che solo gli elementi indicati come loghi da YOLOv3 con confidenza pari o superiore a 50% vanno sottoposti al confronto.

4.6 EasyOCR

In EasyOCR si procede istanziando una classe Reader contenente le lingue da identificare; per questo lavoro sono state istanziate solo le lingue Italiano ed Inglese, al fine di evitare problemi di compatibilità e massimizzare i risultati sugli utenti scelti.

Per ogni immagine l'output è costituito da un array di oggetti contenenti un elemento di testo trovato e la confidenza della rete per questo.

4.7 Creazione output

L'output finale è costituito da un file `.json` contenente i risultati singoli ottenuti precedentemente. Per le reti YOLO e Logohunter è stato aggiunto un contatore rispettivamente per ogni oggetto e ogni marca trovata in tutte le immagini.

Si include un esempio parziale del caso di studio *andreagaleazzi*.

Listing 4.1: Esempio di risultato finale

```
1 {  
2   "ocr": [  
3     "NuoVI",  
4     "1",  
5     "IFA9z",  
6     "..."  
7   ],  
8   "yolo": {  
9     "car": 7,  
10    "cell phone": 7,
```

4. IMPLEMENTAZIONE

```
11     "person": 16,  
12     "cake": 2,  
13     "dining table": 2,  
14     "..."  
15 },  
16 "car_recognition": [  
17     {  
18         "picture": "[path]",  
19         "label": "Ram C/V Cargo Van Minivan 2012",  
20         "prob": "0.4144"  
21     },  
22     {  
23         "picture": "[path]",  
24         "label": "Fisker Karma Sedan 2012",  
25         "prob": "0.1729"  
26     },  
27     {  
28         "picture": "[path]",  
29         "label": "Audi TT RS Coupe 2012",  
30         "prob": "0.5985"  
31     },  
32 ],  
33 "logohunter": {  
34     "apple": 2,  
35     "mazda": 1,  
36     "microsoft": 1,  
37     "citroen": 1,
```


4. IMPLEMENTAZIONE

```
38     "ferrari": 1,  
39 }  
40 }
```

4.8 Email

Tra le motivazioni principali che portano una piattaforma a prelevare dati aggiuntivi dei propri utenti vi è sicuramente la pubblicità mirata. Questa, come indicato dal nome, consiste nel proporre all'utente finale contenuti adeguati ai suoi interessi e ai *brand* di cui fa uso - massimizzando la probabilità di vendita. Per questo motivo il nostro template pubblicitario fa uso delle informazioni ricavate da Logohunter. Queste stesse informazioni sono state utilizzate anche per un template di *phishing*. Il *phishing* consiste nel fingersi una terza parte affidabile per convincere l'utente a condividere informazioni personali; il template per questo compito consiste in una finta ricevuta di acquisto. Il *layout* della pagina è stato creato dinamicamente tramite la libreria Python *tinyhtml* [29] ed è semplice ed essenziale in modo da adeguarsi a molteplici brand; la credibilità è garantita proprio dalle informazioni prelevate dall'utente vittima.

Ulteriori informazioni quali gli oggetti "acquistati" e i loro prezzi, oppure le immagini dei prodotti all'interno della mail pubblicitaria sono state prelevate dall'e-commerce **ebay**.

Per ogni utente sono stati estratti i tre loghi con maggior presenza - basandosi sul contatore aggiunto. Il risultato è diventato parametro di input per una *query* di ricerca, basata sulle API per sviluppatori privati

4. IMPLEMENTAZIONE



Hi lebron

Here are the details of your payments:

Product	Category	Price
Kobe Bryan Gold Purple Swingman Jersey #24 Los Angeles Lakers Mens Yellow	Basketball-NBA	71.99€
Los Angeles Lakers #24 Kobe Bryant Jersey Stitched Throwback Purple	Basketball-NBA	44.99€
Los Angeles Lakers #24 Kobe Bryant Gold Hardwood Classics Sewing Jersey new	Basketball-NBA	34.99€
		151.97€

Don't recognize this payment?

If you did not authorize this payment, please visit [Los-angeles-lakers Payment Cancellation](#)

Figura 4.2: Template di *phishing*

fornite da ebay.

Poiché le moderne REST API sono basate sul protocollo di autenticazione **OAuth 2.0** con limite giornaliero delle richieste e modifica del *token* ogni 24 ore, è stato necessario utilizzare le API tradizionali con protocollo **Auth'n'Auth**. Il protocollo tradizionale garantisce *token* con validità di 18 mesi e per sessioni multiple; ciò tuttavia ha comportato un output della richiesta in formato XML.

Listing 4.2: Esempio di richiesta

```
1 https://svcs.ebay.com/services/search/FindingService/v1?  
   OPERATION-NAME=findItemsByKeywords&SECURITY-APPNAME=  
   GennaroE-SM2023-PRD-9805bf466-0f3650b4&keywords=apple
```

La richiesta in 4.2 è un'invocazione al metodo `findItemsByKeyword` che utilizza come parametro il nome del *brand*; la sua risposta - nel Li-

4. IMPLEMENTAZIONE

sting 4.3 - fornisce un elenco di oggetti con nome contenente la parola chiave; ad ognuno sono associati il prezzo corrente ed una fotografia di riferimento.

Listing 4.3: Esempio di risposta

```
1 <?xml version="1.0" ?>
2 <findItemsByKeywordsResponse xmlns="http://www.ebay.com/
  marketplace/search/v1/services">
3   <ack>Success</ack>
4   <version>1.13.0</version>
5   <timestamp>2022-12-08T13:18:02.441Z</timestamp>
6   <searchResult count="61">
7     <item>
8       <itemId>154843103473</itemId>
9       <title>Apple iPad 6th Gen 9.7" 32GB 128GB Silver Gray
        WiFi or Cellular Unlocked Good</title>
10      <globalId>EBAY-US</globalId>
11      <primaryCategory>
12        <categoryId>171485</categoryId>
13        <categoryName>Tablets & eBook Readers</categoryName>
14      </primaryCategory>
15      <galleryURL>https://i.ebayimg.com/thumbs/images/g/
        aaEAA0SwX4ZjgiBP/s-l140.jpg</galleryURL>
16      <viewItemURL>https://www.ebay.com/itm/Apple-iPad-6th-Gen
        -9-7-32GB-128GB-Silver-Gray-WiFi-Cellular-Unlocked-Good
        -/154843103473?var=0</viewItemURL>
17      <autoPay>true</autoPay>
18      <postalCode>600**</postalCode>
19      <location>Glenview, IL, USA</location>
```

4. IMPLEMENTAZIONE

```
20    <country>US</country>
21    <shippingInfo>
22        <shippingServiceCost currencyId="USD">0.0</
          shippingServiceCost>
23        <shippingType>Free</shippingType>
24        <shipToLocations>Worldwide</shipToLocations>
25        <expeditedShipping>true</expeditedShipping>
26        <oneDayShippingAvailable>false</oneDayShippingAvailable>
27        <handlingTime>0</handlingTime>
28    </shippingInfo>
29    <sellingStatus>
30        <currentPrice currencyId="USD">149.99</currentPrice>
31        <convertedCurrentPrice currencyId="USD">149.99</
          convertedCurrentPrice>
32        <sellingState>Active</sellingState>
33        <timeLeft>P3DT12H42M36S</timeLeft>
34    </sellingStatus>
35    <listingInfo>
36        <bestOfferEnabled>false</bestOfferEnabled>
37        <buyItNowAvailable>false</buyItNowAvailable>
38        <startTime>2022-02-12T02:00:38.000Z</startTime>
39        <endTime>2022-12-12T02:00:38.000Z</endTime>
40        <listingType>FixedPrice</listingType>
41        <gift>false</gift>
42        <watchCount>1483</watchCount>
43    </listingInfo>
44    <returnsAccepted>true</returnsAccepted>
45    <condition>
46        <conditionId>2030</conditionId>
```

4. IMPLEMENTAZIONE

```
47     <conditionDisplayName>Good - Refurbished</  
        conditionDisplayName>  
48 </condition>  
49 <isMultiVariationListing>true</isMultiVariationListing>  
50 <topRatedListing>true</topRatedListing>  
51 </item>  
52 [...]   
53 </searchResult>  
54 <paginationOutput>  
55   <pageNumber>1</pageNumber>  
56   <entriesPerPage>100</entriesPerPage>  
57   <totalPages>199757</totalPages>  
58   <totalEntries>19975601</totalEntries>  
59 </paginationOutput>  
60 <itemSearchURL>https://www.ebay.com/sch/i.html?_nkw=apple&  
    _ddo=1&_ipg=100&_pgn=1</itemSearchURL>  
61 </findItemsByKeywordsResponse>
```


4. IMPLEMENTAZIONE

ONLINE SHOPPING

SAVE 40%

Everything At One Place

Making Lives Effortless






ShopKart Brings A Week Long Sale.

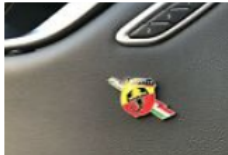


Buy Products Upto 60% Off

Season's Must-Haves




Los-angeles-lakers



Abarth



Hp



Visit Us & Avail Amazing Discount..!!

If you are no longer interested, you can [unsubscribe instantly](#)

EMAIL VIA MAILGET

Figura 4.3: Template di pubblicità mirata

CAPITOLO 5

CONCLUSIONI

Con il lavoro svolto si vuole sensibilizzare l'utente nel prestare attenzione alle immagini postate con leggerezza; queste rappresentano una possibile fonte di informazioni più veritiere e accurate di quanto si immagini. La pubblicità ed il *phishing* sono soltanto due delle possibili applicazioni, tra le più semplici in quanto non necessitano di ulteriori fonti; ma con una ricerca più approfondita, ad esempio attraverso la rete di contatti, i post testuali o un controllo incrociato con altri *social network*, è possibile ricavare informazioni di natura più sensibile anche senza l'inserimento diretto dell'utente.

Eventuali sviluppi futuri prevedono l'aggiunta di ulteriori reti neurali - riconoscimento dei volti, di animali, di paesaggi... - ed il miglioramento di quelle presenti, che hanno risentito di forti limitazioni hardware.

BIBLIOGRAFIA

- [1] S. L. H. Nandyala, “Privacy impact assessment: Instagram,” 01 2018.
- [2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot MultiBox detector,” in *Computer Vision – ECCV 2016*, pp. 21–37, Springer International Publishing, 2016.
- [3] R. Girshick, “Fast r-cnn,” 2015.
- [4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.
- [5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,”

BIBLIOGRAFIA

- 2016.
- [6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” 2017.
 - [7] M. Bastan, H.-Y. Wu, T. Cao, B. Kota, and M. Tek, “Large scale open-set deep logo detection,” 2019.
 - [8] I. Fehervari and S. Appalaraju, “Scalable logo recognition using proxies,” 2018.
 - [9] C. Da, P. Wang, and C. Yao, “Levenshtein ocr,” 2022.
 - [10] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, “Detecting text in natural image with connectionist text proposal network,” 2016.
 - [11] M. Liao, B. Shi, and X. Bai, “Textboxes++: A single-shot oriented scene text detector,” *IEEE, Transactions on Image Processing*, vol. 27, pp. 3676–3690, aug 2018.
 - [12] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, “A forest fire detection system based on ensemble learning,” *Forests*, vol. 12, p. 217, 02 2021.
 - [13] Ultralytics, “Yolov5.” <https://github.com/ultralytics/yolov5>.
 - [14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
 - [15] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *Com-*

BIBLIOGRAFIA

- puter Vision – ECCV 2014*, pp. 346–361, Springer International Publishing, 2014.
- [16] S. Elfving, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning,” 2017.
- [17] U. Nepal and H. Eslamiat, “Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs,” *Sensors*, vol. 22, 01 2022.
- [18] K. Oguine, O. Oguine, and H. Bisallah, “Yolo v3: Visual and real-time object detection model for smart surveillance systems(3s),” 09 2022.
- [19] A. Tüzkö, C. Herrmann, D. Manger, and J. Beyerer, “Open set logo detection and retrieval,” 2017.
- [20] “GitHub - qqwweee/keras-yolo3: A Keras implementation of YOLOv3 (Tensorflow backend) — github.com.” <https://github.com/qqwweee/keras-yolo3>. [Accessed 13-Dec-2022].
- [21] A. Monteux, “Logohunter.” <https://github.com/ilmonteux/logohunter>.
- [22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” 2015.
- [23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” June 2016.
- [24] JaiedAl, “Easyocr.” <https://github.com/JaiedAI/EasyOCR>.

BIBLIOGRAFIA

- [25] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," June 2019.
- [26] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," 2015.
- [27] adw0rd, "instagrapi." <https://github.com/adw0rd/instagrapi>.
- [28] R. PAWAR, "logos images dataset." <https://www.kaggle.com/datasets/inputblackboxoutput/logoimagesdataset>.
- [29] niklasf, "tinyhtml." <https://github.com/niklasf/python-tinyhtml>.