POLITECNICO
DI TORINO

– POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica (Computer Engineering)

Tesi di Laurea Magistrale

Estrazione di relazioni semantiche frequenti tra oggetti in immagini segmentate

Relatori
prof. Elena Baralis
dott. Andrea Pasini

Candidato
Antonino Ventura

Aprile 2020

Indice

Elenco delle tabelle

Elenco delle figure

Parte I

Background

Capitolo 1

Introduzione

L'avanzamento tecnologico degli ultimi decenni ha permesso lo svilupparsi di sistemi intelligenti che hanno totalmente cambiato le abitudini della nostra vita quotidiana. Con l'avanzare degli studi nel campo dell'intelligenza artificiale e del machine learning sono nati sistemi in grado di apprendere il modo in cui prendere decisioni autonome. Il *deep learning* è, ad esempio, una tecnica che permette di estrarre informazioni via via più astratte da un insieme di dati. Per fare ciò vengono addestrati modelli, chiamati reti neurali, il cui compito è quello di produrre in output un tipo di informazione coerente a ciò che gli è stato dato in input. Nel caso dell'analisi di immagini, l'estrazione delle informazioni avviene tramite l'uso di reti neurali convoluzionali composte da più livelli organizzati in modo gerarchico. L'informazione in uscita alla rete sarà il risultato dell'apprendimento realizzato ad ogni livello, ognuno dei quali aggiunge ad essa un livello di dettaglio sempre più astratto.

Tra le diverse applicazioni del deep learning possiamo citare le operazioni di *image understanding* che comprendono la classificazione, il riconoscimento e la segmentazione degli oggetti all'interno delle immagini. In questo caso le reti neurali permettono lo sviluppo di sistemi artificiali che, ricevendo in ingresso grosse quantità di immagini, siano in grado di estrarre da esse tutte le informazioni necessarie

ai fini dell'apprendimento.

Lo scopo di questa tesi è quello di analizzare e progettare nuove tecniche di estrazione di informazioni dai dati, in modo da descrivere le relazioni che legano gli oggetti contenuti all'interno delle immagini. La principale particolarità del procedimento introdotto in questa tesi riguarda l'utilizzo di valori discreti per descrivere la posizione relativa tra gli oggetti in modo da rilevare in modo preciso le posizioni tra gli oggetti e permettere una migliore caratterizzazione semantica. Ciò permette inoltre di considerare la forma effettiva di ogni singolo oggetto senza ricorrere alla semplificazione del *boundingbox* (riquadro rettangolare che contiene l'oggetto), tipicamente usato in letteratura. I pattern estratti dai dati etichettati possono quindi venire utilizzati per supportare e validare, tramite informazioni contestuali, i risultati dei metodi di *Deep Learning* per la segmentazione semantica delle immagini. Di seguiro verranno descritti i contenuti trattati nei vari capitoli.

Nel secondo capitolo verranno descritte le tecniche di Machine e Deep learning in modo da evidenziare le loro caratteristiche principali. Verrà analizzato il funzionamento della reti neurali multi-strato e successivamente le tecniche che hanno portato alla nascita delle reti neurali convoluzionali.

Nel terzo capitolo verrà discusso come le reti neurali convoluzionali sono applicate nella Computer Vision. Nello specifico verranno descritte le principali architetture e caratteristiche per ogni tecnica di *Image Understanding*: la classificazione delle immagini, la detection e la segmentazione ponendo una maggiore attenzione sulla segmentazione panottica. Verranno poi elencate le varie metriche per la valutazione dei risultati ottenuti e in seguito la descrizione dei dataset più importanti.

Nel quarto capitolo verranno descritte le caratteristiche e le varie applicazioni di una Knowledge base. Verrà poi mostrato come l'*Anomaly Detection* sia in grado di rilevare eventuali anomalie dalla base di conoscenza attraverso le informazioni contestuali.

Nel quinto capitolo verrà presentata nel dettaglio l'architettura del processo

proposto in questo lavoro, spiegando le varie tecniche e i vari algoritmi utilizzati.

Infine, nel sesto capitolo, verranno mostrati i risultati degli esperimenti effettuati a valle della validazione del modello proposto.

Capitolo 2

Machine learning e deep learning

Negli ultimi decenni, nel campo dell'Intelligenza Artificiale, stanno prendendo il sopravvento tecnologie sempre più evolute e capaci di risolvere problemi, svolgere compiti e attività tipici della mente e dell'abilità umana. L'acquisizione di tale abilità potrà, dunque, permettere alla macchina di risolvere problemi sempre più complicati con una velocità maggiore rispetto a quella dell'uomo. Il *Machine Learning* e il *Deep Learning* sono tecniche di apprendimento automatico appositamente sviluppate per questi obiettivi. Queste due tecniche, introducono una nuova visione di programmazione che consiste nell'apprendere il modo con cui risolvere un determinato problema direttamente dai dati, anziché mediante algoritmi sviluppati ad hoc.

2.1 Machine learning

Il *Machine Learning* è il campo di ricerca che studia algoritmi e sistemi in grado di apprendere dai dati, senza programmazione esplicita, sintetizzando nuova conoscenza. Un sistema con queste caratteristiche, analizzando dati in ingresso, è spesso in grado di mostrare in output dati con un livello di accuratezza prossimo a quello desiderato. È facile, quindi, intuire come lo studio e lo sviluppo dell'apprendimento

automatico, sia diventato il fulcro della ricerca sull'Intelligenza Artificiale trovando soluzione a compiti molto lunghi ed elaborati.

2.2 Deep learning

Il *Deep Learning* rappresenta una branca del *Machine Learning* ed un'evoluzione della stessa, nata con l'obiettivo di imitare le attività e l'apprendimento del cervello umano in maniera più dettagliata attraverso l'utilizzo di un apprendimento su più livelli di astrazione: è caratterizzato dal fatto che i dati, provenienti da un livello precedente, vengono utilizzati come input di quello successivo, estraendo informazioni più complesse con l'aumentare della profondità. Si può pensare ad ogni livello di apprendimento come una delle aree che compongono la corteccia cerebrale: così come il cervello impara per tentativi attivando nuovi neuroni, anche nelle architetture preposte al *Deep Learning* gli stadi di astrazione si modificano in base alle informazioni in ingresso. Tutte le tecniche di *Deep Learning* si basano sull'ideazione di nuove architetture di reti neurali artificiali. Il termine *Deep* deriva dal fatto che questi modelli, per eseguire compiti sempre più complicati presentano svariati *layer* concatenati l'uno con l'altro.

Uno dei problemi del *Deep Learning*, riguarda la complessità del modello. Infatti man mano che la profondità della rete aumenta si ha più possibilità di causare *overfitting*.

2.2.1 Overfitting

Dato un classificatore addestrato su un determinato insieme di addestramento (Training Set) è necessario valutarlo su un altro insieme (Validation Set o Certification Set). Da questo confronto è possibile estrarre degli indici che permettono di valutare il classificatore e permettono di confrontare diversi classificatori tra loro. È assolutamente indispensabile che gli indici di prestazione vengano calcolati su

un insieme di campioni non usati durante la fase di addestramento (il *validation set*) in modo da rilevare problemi come l'*overfitting* dei dati, spiegato di seguito. Qualsiasi modello di *Machine Learning*, apprende la conoscenza dalle immagini che sono contenute all'interno del training set. Esiste una misura, *loss function*, che permette di capire quanto l'algoritmo è riuscito ad imparare dai dati che gli sono stati forniti. Questa misura può avere due diversi tipi di errori, quelli irriducibili, i quali non possono essere ridotti indipendente dall'algoritmo che è stato applicato e gli errori riducibili che possono essere:

- un errore restituito dalla differenza tra il valore che è stato predetto dal modello e quello che si cerca di predire; quando il valore di questo errore è molto elevato allora il modello sta riscontrando il problema dell'*underfitting* (Figura 2.1(b));
- un errore che misura la variabilità che può avere la predizione su certi dati; un valore molto elevato indica che il modello ha difficoltà nell'individuare le predizioni corrette.

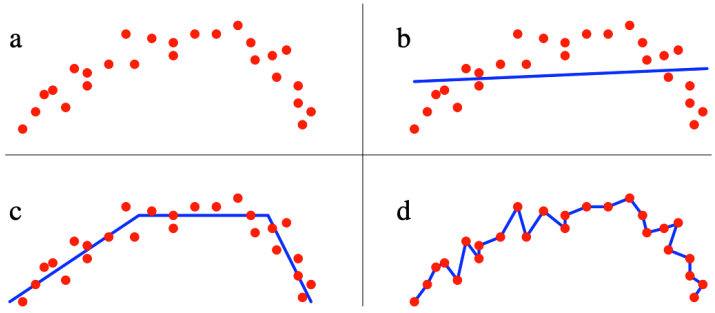

Figura 2.1: Ricostruzione di una curva da (a) un insieme di punti: (b) modello con *underfitting*, (c) modello corretto, (d) modello con *overfitting*. [20]

Quando un modello soffre di *overfitting* (Figura 2.1(d)), significa che è un modello che ha appreso in maniera eccessiva gli esempi contenuti all'interno del

training set, non riuscendo a garantire delle ottime prestazioni nel valutare ulteriori informazioni. Imparare eccessivamente i dati del training set, significa apprendere tutto ciò che fa parte del *dataset*. Per evitare questo problema bisognerebbe valutare le performance finali del modello di classificazione su un test set. Ciò nonostante, risulta più efficiente l'utilizzo di un ulteriore *dataset*, il *validation set* per valutare le performance del modello alla fine di ogni allenamento sul training set, questo permette di supervisionare l'andamento dell'apprendimento e generalizzare le nozioni che sono state apprese. Tuttavia non è semplice capire i motivi per cui un modello tende ad andare in *overfitting*. Per questo motivo l'utilizzo di ulteriori tecniche permettono di migliorare la capacità dell'algoritmo di classificazione nel generalizzare i dati appresi.

Nella prossima sezione vengono descritte le caratteristiche principali di una rete neurale.

2.3 Reti neurali

Una rete neurale artificiale è un modello matematico ispirato ad una rete neurale biologica, con l'obiettivo di riprodurre le funzioni tipiche del cervello umano. Una rete di questo tipo è realizzata, così come per la biologica, da unità semplici chiamate neuroni artificiali, caratterizzati dal fatto di avere molti ingressi ed una sola uscita calcolata dalla somma pesata degli input.

Come già accennato, le reti, applicate al *Deep Learning*, sono strutturate su più livelli, dove il grado di profondità ne determina il grado di complessità del calcolo. Nel corso degli anni, sono state realizzate diverse architetture con compiti ben precisi dando vita a diverse reti neurali artificiali tra cui il MLP (Multi layer perceptron) e la CNN (Convolutional Neural Network).

2.3.1 Perceptron

Il primo modello di rete neurale, introdotto da Rosenblatt tra la fine degli anni '50 e gli inizi degli anni '60, fu rappresentato dal percettrone, ideato per risolvere dei problemi di classificazione in due classi. Questo modello (Figura 2.2)

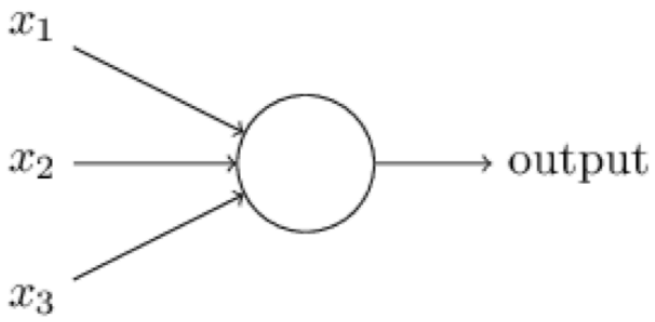

Figura 2.2: Esempio di percettrone di Rosenblatt. [11]

è, quindi, un algoritmo di classificazione, che riceve in ingresso un numero n di elementi, $x_1, x_2, \ldots, x_n$, per poi restituire un singolo output binario (0 o 1, a seconda dell'appartenenza dei valori osservati alla classe in esame):

$$output = \begin{cases} 0, & se \sum_i x_i \cdot \omega_i \leq soglia \\ 1, & se \sum_i x_i \cdot \omega_i > soglia \end{cases}$$

Possiamo scrivere la suddetta formula in termini vettoriali come $\sum_i x_i \cdot \omega_i = x \cdot \omega$ e spostare a sinistra il valore *soglia*, introducendo, così, un nuovo termine chiamato *bias* ed indicato con b, definito come $b \equiv -soglia$:

$$output = \begin{cases} 0, & se\ x \cdot \omega + b \leq 0 \\ 1, & se\ x \cdot \omega + b > 0 \end{cases}$$

Rosenblatt, riuscì a dimostrare che l'algoritmo da lui ideato, converge se le due classi in esame sono linearmente separabili, ossia, se esiste un iper-piano lineare che riesca a separare le due classi in modo corretto.

Un perceptron è quindi la versione più semplice di rete neurale, composta da due strati. Il primo strato, chiamato *input layer*, ha una dimensione pari al numero di elementi in ingresso alla rete, chiamati neuroni di input. Ogni nodo, che appartiene al primo strato, è collegato al nodo di output, costituendo, in questo modo, la risposta che la rete neurale elabora per ciascun neurone di input.

Learnig

L'algoritmo di learnig del percettrone [11] è il seguente:

- **Inizializzazione**: all'istante di tempo $t = 0$, vengono inizializzati i pesi $\omega_i(0)$ e bias $b_i(0)$ in modo casuale $\forall i = 1, \ldots, n$;
- **Presentazione di un esempio di training**: viene introdotto un vettore di input $X = x_1, \ldots, x_n$ con il relativo valore di output Y;
- **Calcolo del valore di attivazione**: viene calcolato l'output partendo da X, utilizzando la formula descritta in precedenza;
- **Aggiornamento**: viene aggiornato il vettore dei pesi secondo la seguente formula:

  $$\omega_i(t+1) = \omega_i(t) + \eta \cdot (y - a(t)) \cdot x_i(t)$$

  dove $a(t)$ indica l'output calcolato e η è un valore compreso tra 0 e 1 Quindi viene calcolata la differenza tra l'output ottenuto al punto 3 e quello corretto, per poi moltiplicarla per il vettore delle feature in input al tempo t. Il risultato viene, poi, sommato al vettore dei pesi corrente $\omega_i(t)$, ottenendo, così, un nuovo vettore $\omega_i(t+1)$.

Questo tipo di algoritmo viene ripetuto fino a quando tutti gli elementi saranno classificati in maniera corretta oppure per un numero di iterazioni che viene dichiarato inizialmente. La rete riesce a classificare in maniera esatta ogni elemento delle due classi, solamente se queste sono linearmente separabili, altrimenti, nel

caso un cui una retta non basti per separare le due classi, il percettrone non riesce a classificare due classi, che non siano linearmente separabili, in maniera corretta.

2.3.2 Multi layer perceptron

Per risolvere le limitazione del percettrone venne ideata una rete neurale contenente uno o più strati nascosti tra lo strato di input e quello di output. Un modello di questo tipo è chiamato Multi Layer Perceptron (MLP), in quanto l'informazione viaggia dai nodi di ingresso fino ai nodi di uscita senza scambio di informazioni tra i nodi che appartengono allo stesso strato. La Figura 2.3 mostra un esempio di percettrone multistrato contenente quattro neuroni di input e due neuroni di output con un solo strato intermedio.

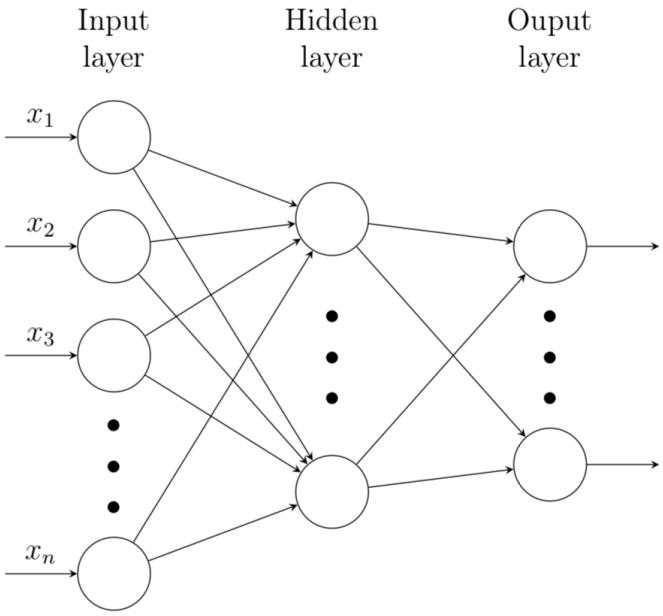

Figura 2.3: Esempio di percettrone multistrato con un solo strato intermedio. [11]

Ciascun nodo di input è collegato ad ogni nodo che fa parte dello strato intermedio, in quanto contiene nodi che non appartengono né al primo, né all'ultimo strato. Ogni nodo è connesso a tutti i nodi dello strato precedente, ma due neuroni

dello stesso strato non possono essere collegati tra di loro. Infine, per rendere il modello capace di risolvere problemi non lineari, ogni nodo calcola il proprio output mediante una funzione non lineare detta funzione di attivazione.

Con il trascorrere degli anni, l'utilizzo di reti neurali con molteplici *hidden layer* è diventato sempre più frequente, costituendo così la base del *Deep Learnig*. Ovviamente, all'aumentare del numero dei *layer* intermedi, la complessità del modello e il numero di dati richiesti in input per arrivare al risultato finale, aumenta notevolmente.

I MLP generano risultati di classificazione migliori rispetto a quelli con un unico livello solamente se le classi sono separate da superfici non lineari. Infatti, se una rete multistrato processasse solamente funzioni di trasferimento lineari, allora questa sarebbe equivalente ad una rete mono-livello [25].

L'esempio più semplice di rete neurale, non contiene connessioni retroattive, ovvero l'informazione viaggia in un'unica direzione, dall'input all'output, senza scambio di dati tra nodi dello stesso livello oppure tra nodi appartenenti ad un livello diverso. In questo caso la rete è chiamata *feed-forward*. Queste reti non hanno memoria, poiché il valore ottenuto in uscita, considera solamente il valore corrente dei dati in ingresso e quello dei pesi calcolati. Invece, un modello più complesso di rete neurale, contiene dei collegamenti in retroazione all'interno dello stesso livello oppure tra due livelli diversi. In questo caso, la rete è detta *feedback* e il valore ottenuto in uscita considera il valore corrente dei pesi, il valore dei dati in ingresso e dal valore delle precedenti uscite (reti neurali con memoria).

Nel prossimo paragrafo, verrà discusso il funzionamento dei MLP e quali sono le loro caratteristiche principali.

2.3.3 MLP: funzionamento e caratteristiche

I dati, che vengono inviati in ingresso ad un MPL, sono dei vettori numerici formati da componenti che costituiscono le informazioni principali del problema esaminato. Nella fase di apprendimento, i vettori di esempio, estratti da tutti i casi possibili del problema reale, sono forniti alla rete in maniera sequenziale. Per ogni vettore esempio, quindi, i pesi relativi alla connessione sono riadattati seguendo una determinata procedura, ovvero la regola di addestramento. Lo scopo, dunque, è quello di indirizzare i pesi delle connessioni a dei valori tali da rendere minimo l'errore che può essere commesso dalla rete, raggiungendo un'opportuna accuratezza [25].
Per l'apprendimento di una rete neurale artificiale, sono utilizzati diversi metodi:

- **Supervised Learning**: dove tutti i dati che sono presenti all'interno dei dati allenamento (training set) sono etichettati. Quindi [26] per ogni osservazione $x_i, i = 1, \ldots, n$, all'interno del training set verrà associata una risposta y_i;
- **Unsupervised Learning**: in questo metodo, per ogni osservazione x_i contenuta all'interno del training set non risulta associata nessuna risposta y_i;
- **Semi-Supervised Learning**: questo criterio, permette di utilizzare l'idea dei due metodi precedenti, ovvero, per una parte del training set vengono utilizzati dati etichettati, mentre per la parte rimanente è formata da dati non etichettati;
- **Reinforcement Learning**: questo metodo, ha un approccio completamente diverso rispetto ai criteri appena descritti. Con questo metodo di apprendimento, viene assegnato un premio o una penalità, rispettivamente se il modello, in fase di training, avrà preso delle decisione corrette o sbagliate.

Il percettrone multistrato, permette di ottenere risultati molto accurati, in quanto vengono effettuate numerose trasformazioni ai dati iniziali, negli strati intermedi.

In questi strati, viene applicata una funzione di attivazione (come per il percettrone singolo) che trasforma i dati in ingresso in valori che possono essere utilizzati dagli strati successivi, o direttamente dallo strato finale.

Nel caso di un Multi Layer Perceptron, indicheremo pesi, bias e funzione di attivazione come segue:

- ω^l_{jk} rappresenta il peso per il collegamento tra il $k-esimo$ neurone relativo al $(l-1)-esimo$ strato e il $j-esimo$ neurone relativo al $l-esimo$ strato:
- b^l_j corrisponde all'$j-esimo$ bias del $l-esimo$ strato;
- a^l_j corrisponde all'$j-esimo$ funzione di attivazione del $l-esimo$ strato, dove la somma è su tutti i k nodi dell'$l-esimo$ strato.

$$a^l_j = \sigma\left(\sum_k \omega^l_{jk} a^{l-1}_k + b^l_j\right)$$

Indichiamo $\sum_k \omega^l_{jk} a^{l-1}_k + b^l_j$ come z^l, che rappresenta la somma su tutti di tutti i nodi sull'$l-esimo$ strato. L'equazione, quindi può essere riscritta come:

$$a^l_j = \sigma(z^l)$$

dove σ è la funzione di attivazione utilizzata per i neuroni

Con il trascorrere degli anni sono state utilizzate molteplici funzioni di attivazione: binarie, lineari e non lineari. Vediamo le più comuni [11], nel prossimo paragrafo.

2.3.4 Funzioni di attivazione

Il risultato restituito dalla rete neurale è intensamente condizionato dalla funzione di attivazione che viene utilizzata perché occupa un ruolo molto importante sia nel processo che nella velocità di convergenza della rete.

A seguire verranno discusse le principali funzioni di attivazione.

Step Function

La Step Function, σ_{step} (Figura 2.4), nota come "funzione di Heaviside", viene definita come:

$$\sigma(z)_{step} = \begin{cases} 0, & se\ z \leq 0 \\ 1, & se\ z > 0 \end{cases}$$

Questa funzione può assumere solamente due tipi di valori: un valore uguale a 0, per

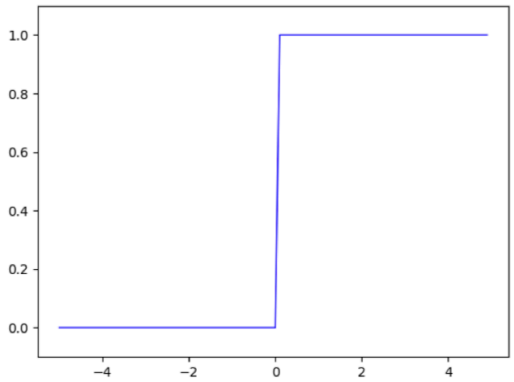

Figura 2.4: Diagramma di una Step Function. [11]

grandezze della variabile indipendente minori o uguali a 0, mentre viceversa, assume valore 1. Questo tipo di funzione, però, può avere diversi problemi. Il primo riguarda la sensibilità del percettrone a minime variazioni dei pesi e del bias, in quanto, basta una minima variazione per ribaltare totalmente il risultato che potrebbe passare da 0 a 1 o viceversa, perdendo così molte informazioni quantitative relative all'array di input. Un altro problema, legato all'utilizzo di questa funzione, è il fatto che non è differenziabile; la funzione gradino (Step Function), essendo una funziona discreta e non continua, potrebbe causare diversi problemi, nei successivi della rete.

Per ovviare ai problemi introdotti dalla funzione gradino, spesso viene considerata la funzione sigmoide utilizzata come funzione di attivazione dello strato di output per i modelli di classificazione.

Funzione Sigmoide

La funzione sigmoide, $\sigma_{sigmoid}$ (Figura 2.5), viene definita come:

$$\sigma(z)_{sigmoid} = \frac{1}{1 - e^{-z}}$$

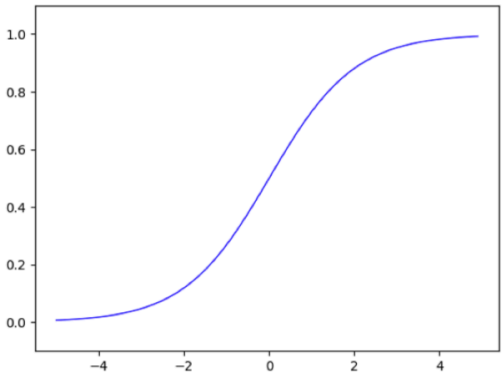

Figura 2.5: Diagramma di una funzione sigmoide. [11]

La funzione sigmoide può assumere qualsiasi valore compreso nell'intervallo [0,1] $\in \Re$, in modo tale da risultare continua e differenziabile. Questo tipo di funzione converge verso il valore 1 all'aumentare di z. L'utilizzo di tale funzione, dunque, permette di superare il problema, della suscettibilità dell'output a minime variazioni nei pesi e nei bias, restituendo un numero reale compreso tra 0 e 1.

Funzione tangente iperbolica

Un altro tipo di funzione che può essere utilizzata per la funzione di attivazione è la funzione di tangente iperbolica (Figura 2.6), definita come:

$$\sigma(z)_{tanh} = \frac{e^{2z} - 1}{e^{2z} + 1}$$

la quale restituisce valori compresi nell'intervallo $[-1,1] \in \Re$,

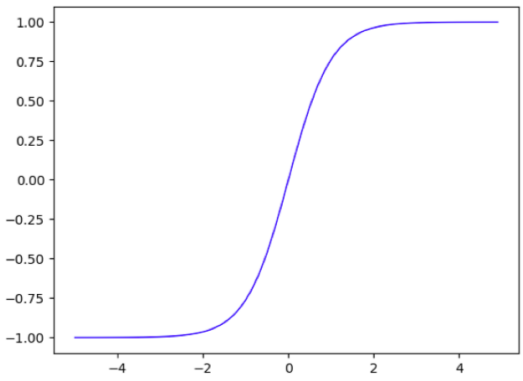

Figura 2.6: Diagramma di una funzione tangente iperbolica. [11]

Funzione Rectified Linear Unit (ReLU)

La funzione ReLU (Figura 2.7), definita come:

$$\sigma(z)_{ReLU} = max\{0, z\}$$

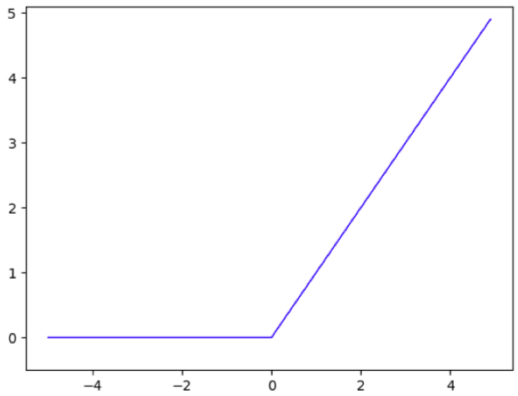

Figura 2.7: Diagramma della funzione Reflicted Linear Unit. [11]

Quindi la funzione restituisce solamente valori positivi per z positive, mentre per i valori negativi la funzione assumerà un valore pari a 0. La ReLU è la funzione di attivazione utilizzata maggiormente negli ultimi anni nell'impiego delle rete neurali artificiali, in quanto le derivate prima di ϕ_{tanh} e $\phi_{sigmond}$ tendono velocemente

a 0. Il motivo è dato dal fatto che l'allenamento della rete neurale si basa sulla discesa del gradiente, e quindi la moltiplicazione per un valore molto vicino allo 0 porta gli strati più profondi della rete ad un apprendimento molto più lento. Questo problema è noto con il nome "Vanishing gradient". Inoltre, un altro vantaggio di questa funzione riguarda la facilità di calcolo, ciò significa, sopratutto per reti neurali molto profonde, con un notevole risparmio di calcolo computazionale.

Funzione Softmax

Nel caso in cui si abbiano solamente due classi, per problemi riguardanti la classificazione, la funzione sigmoide sarebbe opportunamente utilizzata. Posto che il numero di classi sia maggiore di due allora viene utilizzata la funzione di attivazione softmax [11]:

$$\sigma(z) = \frac{e^{zj}}{\sum_k e^{zk}}$$

Riscrivendo l'equazione utilizzando la notazione presentata nel paragrafo precedente avremo:

$$a_j^L = \frac{e^{zj}}{\sum_k e^{zk}}$$

Le funzioni di attivazione a_j^L relative allo strato di uscita di una rete neurale, presentano un interpretazione probabilistica, ovvero ogni a_j^L assume un valore che appartiene all'intervallo [0,1]:

$$\sum_j a_j^L = \frac{\sum_j e^{zj}}{\sum_k e^{zk}}$$

Questo fa si che la somma dei valori dei neuroni di output della rete sia uno. Quindi a_j^L viene interpretata come la probabilità della rete neurale che l'elemento considerato possa appartenere alla $j - esima$ classe.

2.3.5 MLP: utilizzo e limiti

Fino ad ora abbiamo visto come funziona un MLP e quali sono le sue caratteristiche. Un esempio, abbastanza complesso, di come il percettrone multistrato viene utilizzato, è quello riguardante la classificazione di immagini. Consideriamo, per esempio, il *dataset* MNIST [28] che contiene immagini in scala di grigi con dimensione 28 × 28 pixel che raffigurano un numero intero da 0 a 9 scritti a mano (Figura 2.8). Il compito del MLP è, quindi, quello di determinare il numero cor-

Figura 2.8: Esempi del *dataset* MNIST. [29]

rispondente da 0 a 9, classificando in modo corretto l'immagine. L'immagine sarà letta dal classificatore come un array di pixel, dove ognuno di questi potrà assumere un valore da 0 a 255 a seconda dell'intensità del colore che vi è associato. Quindi l'array di input sarà formato da 784 neuroni, ovvero le *feature* dell'immagine, 15 neuroni appartenenti ad uno strato nascosto e 10 neuroni nello strato di output (Figura 2.9).

Questo esempio ci fa capire che l'utilizzo di una rete neurale, completamente connessa per l'estrazione delle *feature*, presenta un enorme problema. Per rappresentare, ad esempio, un'immagine a colori (RGB) con dimensione 8 × 8 pixel, saranno necessarie tre matrici, uno per ogni canale RGB, della stessa dimensione,

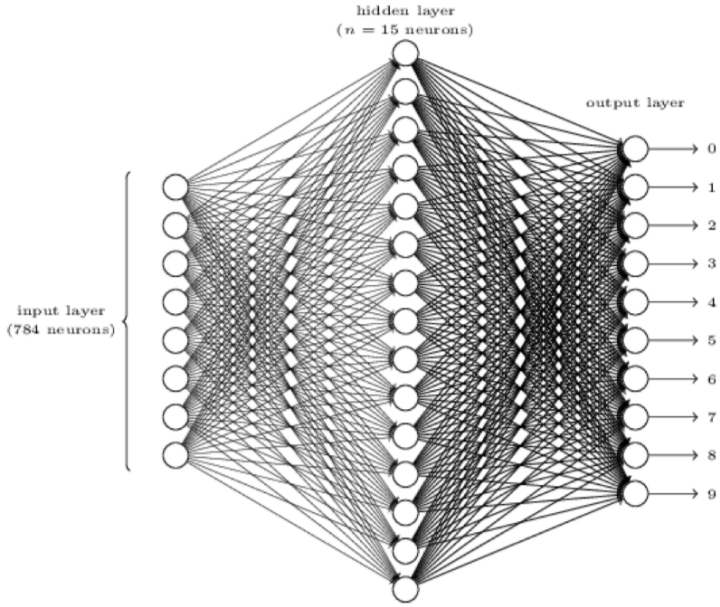

Figura 2.9: Il percettrone multistrato utilizzato nel *dataset* MNIST [29]

che può essere rappresentata come una matrice di dimensione $8 \times 8 \times 3$, conosciuta come *tensore*, con 192 strati nel primo strato. Un'immagine RGB di 256×256 avrà 196.608 *feature* solo nello strato di input. Quindi il numero di neuroni cresce notevolmente con l'aumentare delle dimensioni di un'immagine. Ma il più grande problema consiste nel fatto che questo tipo di rete neurale non è invariante rispetto a traslazioni e distorsioni dell'immagine [11]. Una soluzione a tale problema [28] consiste un nuovo metodo per l'estrazione delle *feature* delle immagini. Quest'ultima consiste nel dividere l'immagine in diverse sezioni e per ognuna di queste estrarre delle *feature* moltiplicando i filtri per matrici di pesi chiamate *filtri*.

Questa tecnica permise la nascita di una nuova tipologia di rete: le reti neurali convoluzionali che approfondiremo nel prossimo capitolo.

2.4 Convolutional neural networks

Nel paragrafo precedente, abbiamo visto, come le reti neurali artificiali vengono utilizzate per la risoluzione di problemi particolarmente complessi, come la classificazione delle immagini, ma anche come il percettrone multistrato risulta inadatto per questo tipo di lavoro. Per questo motivo per il riconoscimento delle immagini vengono utilizzate le reti neurali convoluzionali (Convolutional neural networks, CNN) che utilizzano un'operazione matematica chiamata convoluzione; questo tipo di operazione va a sostituire il prodotto tra matrici, che veniva utilizzato nei MLP. Inoltre le reti convoluzionali, a differenza del percettrone multistrato, non sono *fully connected*, ovvero i neuroni non sono tutti connessi tra loro, ma solamente con un loro sottospazio. Vediamo nello specifico in cosa consiste l'operazione di convoluzione.

2.4.1 Operazione di convoluzione

Questo tipo di operazione sta alla base delle CNN e si occupa principalmente di estrarre le *feature* delle immagini. Date due funzioni f e g allora viene definita la convoluzione tra le due come:

$$(f * g)(t) = \int f(\tau)g(t-\tau)d\tau$$

dove $(f * g)$ indica la convoluzione tra le due funzioni.

Nel caso di due funzioni discrete avremo:

$$(f * g)(t) = \sum_{k=-\infty}^{+\infty} f(k)g(t-k)$$

Il risultato di queste due convoluzioni può essere letto come un mescolamento tra le due, facendo scorrere la seconda funzione (filtro) sulla prima [11]. Applicando la convoluzione sulle immagini, essendo queste formate da matrici, il filtro che gli sarà associato sarà anche una matrice, chiamata *kernel* (o matrice di convoluzione).

Definendo l'immagine (I) e il kernel (K) avremo:

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

dove i, j, m, n sono coordinate dello spazio bidimensionale su cui giacciono I e K.

2.4.2 Architettura di una rete neurale convoluzionale

Per la costruzione di una CNN, vengono considerati tre tipi di strati diversi:

- *convolutional*;
- *pooling*;
- *fully-connected*.

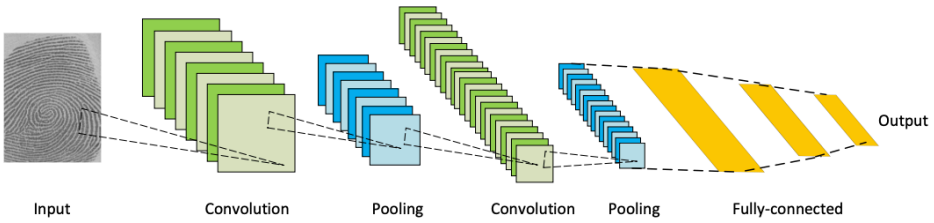

Figura 2.10: Esempio di rete neurale convoluzionale basata sul riconoscimento di documenti testuali. [21]

Ciascuna rete neurale, per garantire l'invarianza circa traslazioni e distorsioni, basa il proprio funzionamento su tre idee diverse: il campo recettivo locale, i pesi e i bias condivisi, e il *pooling* che permette di ridurre le dimensioni della *feature map*, ovvero la mappa generata partendo dallo strato di input.

Nelle CNN, ogni immagine viene rappresentata come una matrice di neuroni con dimensioni $m \times n$, dove il nodo (i, j) sarà associato all'intensità del pixel (i, j) dell'immagine di input. Come accennato, tutti neuroni non saranno connessi con tutti i nodi del primo strato nascosto, ma solamente con una parte di questi, quindi

la sezione di input associata al primo neurone dello strato *hidden* sarà considerato il suo *Local Receptive Field* (Figura 2.11). Affinché si possa determinare il valore dei *local receptive field* relativo agli altri nodi dello strato di input, si deve spostare il primo valore ottenuto di uno step k verso destra, chiamato *shift* (Figura 2.12).

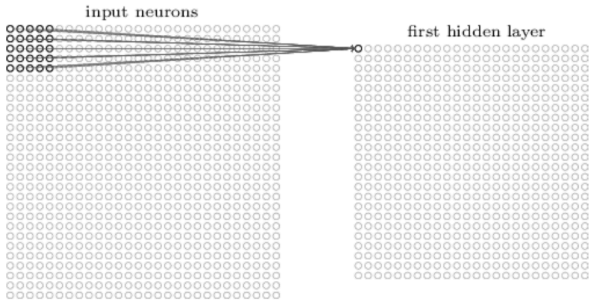

Figura 2.11: Connessione tra il primo nodo dello strato nascosto ed il *Local Receptive Field* [29]

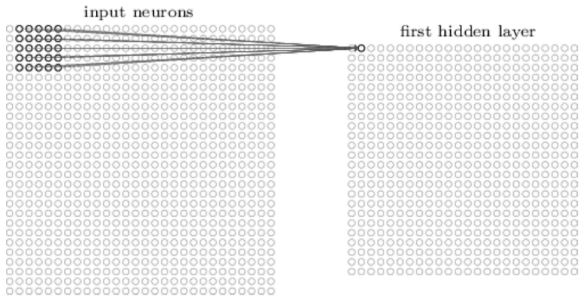

Figura 2.12: Connessione tra il secondo nodo dello strato nascosto ed il relativo *Local Receptive Field*. [29]

Consideriamo l'esempio delle figure 2.11 e 2.12 ogni *Local Receptive Field* è formato da 25 neuroni, mentre il primo strato nascosto, avrà una dimensione di 24 $\times$ 24. Il nodo (j, k) del primo strato nascosto, avrà il seguente output:

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} \omega_{l,m} a_{j+l,k+m}\right)$$

dove σ è la funzione di attivazione, e $a_{x,y}$ l'attivazione di posto (x, y). Come possiamo notare dalla formula appena descritta, ciascun nodo dello strato nascosto, utilizza gli stessi *bias* e gli stessi pesi per tutti i nodi che vi fanno parte; quindi questo permette di apprendere la stessa caratteristica anche se si trovano in zone diverse dell'immagine. Se una caratteristica viene rilevata trovandosi in una posizione particolare dell'immagine, questa verrà rilevata anche se presente in altre posizioni dell'immagine, evidenziando così l'invarianza alle traslazioni [11].

Strato Convoluzionale

Lo strato convoluzionale di una CNN, è uno strato che contiene all'interno un'operazione di convoluzione, dove per determinare il valore del nodo di output viene utilizzato sempre lo stesso filtro, che successivamente sarà condiviso per calcolare il valore che sarà assegnato a ciascun nodo dello strato nascosto. In questo modo, saranno utilizzati un numero di pesi che sarà uguale alla dimensione del filtro ed il relativo *bias*, basandosi sul concetto della condivisione di cui abbiamo appena discusso. Ovviamente, all'interno di uno strato di convoluzione si possono trovare molteplici filtri, ciascuno di essi imparerà una caratteristica diversa dell'immagine in esame. Per determinare la *feature map* bisogna considerare 3 parametri [11]:

1. **Profondità**: corrisponde al numero di filtri utilizzati;
2. **Stride**: equivale al passo con cui si fa scorrere il filtro, variando il *Local Receptive Field*;
3. **Zero-Padding**: permette di controllare le dimensioni dell'output, ovvero permette di aggiungere all'input un bordo di soli zeri, evitando così di perdere caratteristiche nel passaggio tra uno strato e l'altro della rete neurale.

La dimensione dell'output di uno strato convoluzionale è ottenuta da:

$$O = \frac{I - F + 2P}{S} + 1$$

Indichiamo con:

- I: dimensione dell'input;
- F: dimensione dei filtri utilizzati;
- P: quantità di *zero-padding*;
- S: *stride*.

Essendo che la convoluzione restituisce una funzione lineare, allora, ricavata la *feature maps*, si applica la funzione di attivazione $ReLU$, e quindi assegnare 0 a tutti i nodi che avranno valori negativi. Una rete neurale di questo tipo, può contenere più stati convoluzionali, dove i primi strati, partendo dallo strato di input, estraggono feature più a basso livello, mentre quelli più vicini allo strato di output determinano le feature più significative dell'immagine che si sta esaminando.

Strato di Pooling

Come abbiamo già accennato, l'obbiettivo dello strato di *pooling* è quello di ridimensionare l'immagine. Consiste nel dividere la matrice di pixel in tante sotto-matrici per poi sostituirli con una lo statistica approssimativa, così da ottenere una matrice molto più piccola rispetto a quella di partenza. In questo modo, cambiano la lunghezza e la larghezza della matrice, ma la profondità rimane invariata.

Esistono due tecniche di *Pooling* principali:

1. **Max Pooling**: consiste nel recuperare da ogni sotto-matrice solamente il pixel con il valore maggiore rispetto agli altri (Figura 2.13).
2. **Average Pooling**: con questo metodo, la matrice ridotta, viene calcolata facendo la media tra tutti i valori della sotto-matrice (Figura 2.14).

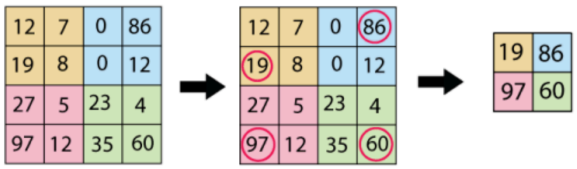

Figura 2.13: Esempio di Max Pooling in una matrice 4 x 4 [11]

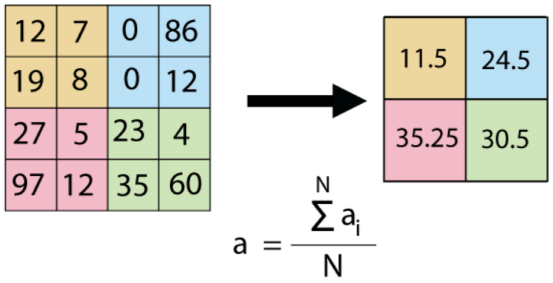

Figura 2.14: Esempio di Average Pooling in una matrice 4 x 4 [11]

Strato Fully-Connected

Effettuate tutte le operazioni riguardanti tutti gli strati convoluzionali e quelli di *pooling* verrà restituita una matrice $W \times H \times D$. Successivamente la matrice calcolata sarà rappresentata all'interno di un vettore $W \cdot H \cdot D$. L'ultimo strato della rete realizzato con un MLP e avrà un numero di neuroni di output uguale alle classi che sono state definite per la classificazione. Il vettore finale, calcolato tramite la funzione di attivazione *softmax*, corrisponderà alla probabilità che viene assegnata dalla rete neurale che l'oggetto in esame appartenga alla classe $i-esima$;

Capitolo 3

Image understanding

Con la *Computer Vision* è possibile analizzare un'immagine bidimensionale interpretandone il contenuto ed estrapolarne una miriade di informazioni (Image Recognition). Nell'ultimo decennio, questa disciplina è stata legata con l'Intelligenza Artificiale ed il *Machine Learning* visti fino ad ora. Nello specifico, le reti neurali convoluzionali, viste nel capitolo precedente, sono state applicate al rilevamento, al riconoscimento di oggetti, alla classificazione, alla regressione, alla segmentazione di oggetti, ecc. [30]. L'*Image understanding* è possibile grazie allo sviluppo di un sistema artificiale che sia in grado di elaborare immagini in modo da comprendere ed estrarre da esse informazioni utili ai fini dell'apprendimento. Le CNN non rendono necessario lo sviluppo di algoritmi ad hoc per l'estrazione delle feature, poiché provvedono esse stesse all'elaborazione delle immagini. L'immagine in esame viene successivamente suddivisa in più gruppi di pixel, ad ognuno di essi viene quindi assegnata un opportuno insieme di *feature* per permetterne la classificazione. Sempre più spesso questo compito viene svolto da reti neurali complesse che hanno bisogno di una grande quantità di dati per effettuare le operazione di apprendimento.

Nei paragrafi successivi verranno evidenziate le principali architetture e caratteristiche che riguardano le tecniche di *Image Understanding*.

3.1 Classificazione di immagini

Con la classificazione si intende il task di assegnare una singola etichetta ad un dato (un'immagine nel nostro caso) in ingresso al modello. Le CNN, vengono ampiamente utilizzate per la classificazione delle immagini soprattutto nel campo della medicina. Infatti, una delle principali applicazioni è la diagnosi del cancro mediante immagini istopatologiche. Recentemente, le CNN sono state usate per la diagnosi del cancro al seno cosicché i risultati venissero confrontati con i risultati di una rete addestrata con un *dataset* di descrittori fatti a mano. Un'altra tecnica basata sulle CNN riguarda la diagnosi del carcinoma mammario. Questo lavoro è stato diviso in due fasi, nella prima fase vengono identificati esempi che non presentavano mitosi, mentre nella seconda fase venne eseguito una *Data Augmentation* (che permette di aumentare il numero di dati presente nel training set) in modo da ovviare il problema riguardante la simmetria della classe. Allo stesso modo è stato sviluppato un altro algoritmo, sempre basato sulla CNN, in grado di classificare i segnali stradali [30]

Esistono diverse architetture di classificazione basate sulle reti convoluzionali, che hanno rivoluzionato il campo del riconoscimento delle immagini, dalla più semplice AlexNet [31], a quelle più complesse come VGG[32] e ResNet [36].

3.1.1 AlexNet

L'architettura LeNet [33] può essere considerata come la prima architettura di Deep Learning basata sulle CNN. Inizialmente le CNN erano limitate al riconoscimento dei numeri scritti a mano (viste come esempio nel capitolo precedente).Queste prime tecniche non funzionava molto bene con altri tipi di immagine. In successore di LeNet, chiamato AlexNet [31], è considerato la prima architettura di *deep* CNN in grado di mostrare risultati rivoluzionari sulle attività di classificazione

e di riconoscimento delle immagini. Questa architettura, ha migliorato la capacità di apprendimento delle CNN, rendendole più profonde, ovvero aggiungendo più strati nascosti, e applicando diverse strategie di ottimizzazione dei parametri [30].

All'inizio degli anni 2000, l'evoluzione dei sistemi hardware, hanno limitato la capacità di apprendimento delle *deep* CNN. Per superare tali limiti, AlexNet è stata addestrata in parallelo su due GPU NVIDIA GTX 580. La profondità è stata estesa da 5 (LeNet) a 8 livelli permettendo di migliorare la generalizzazione a diversi livelli di risoluzione dell'immagine. Tuttavia, l'elevato numero di layer ha introdotto il cosiddetto problema dell'*overfitting* (vedi 2.2.1). Per questo motivo, è stato applicato il sotto-campionamento sovrapposto e la normalizzazione della risposta locale. Un'altra risoluzione a questo problema è stata introdotta sulla base del lavoro di Hinton [34], il quale ha permesso di realizzare un algoritmo (chiamato dropout) il cui compito è quello di saltare alcune unità della rete in maniera casuale durante il processo di training, in modo da imporre l'apprendimento di feature più robuste. Inoltre è stata applicata la ReLU come funzione di attivazione non saturante, per migliorare la frequenza con cui la rete converge, ovviando al problema dell'attenuazione dei gradienti (Vanishing gradient). Altre modifiche introdotte da AlexNet riguardando l'uso di filtri di grandi dimensioni (11x11 e 5x5) ai livelli iniziali della rete. Grazie a tutto ciò l'efficacia dell'apprendimento con l'uso di AlexNet è stata comprovata, portando così ad una significativa evoluzione nell'ambito delle CNN, dando il via ad una nuova serie di ricerche per nuovi tipi di architetture [30].

Un esempio di architettura AlexNet è mostrato nella Figura 3.1.

3.1.2 VGG

Gli ottimi risultati che sono stati prodotti dall'utilizzo delle reti neurali convoluzionali nel campo del riconoscimento delle immagini, ha permesso studiare e progettare architetture, basate sulle CNN, sempre più moderne. VGG, è un architettura CNN, che si basa su un algoritmo semplice ed efficiente, realizzato con 19

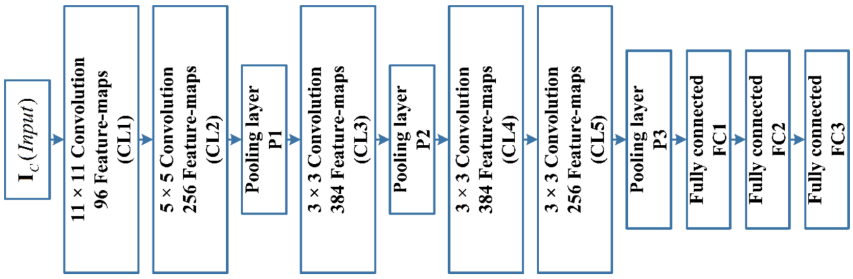

Figura 3.1: Diagramma di un'architettura AlexNet che contiene 4 strati convoluzionali e 3 strati *fully-connected* [30]

livelli di profondità, rispetto agli 8 di AlexNet e Znet [35]. Il principio di ZNet suggerisce che i filtri di piccole dimensioni possono migliorare le prestazioni delle CNN, infatti, sulla base di questi risultati, VGG ha sostituito i filtri 11x11 e 5x5 con una colonna di filtri 3x3, dimostrando, così, in maniera sperimentale che il l'utilizzo simultaneo di filtri di piccole dimensioni simula l'uso di filtri di dimensioni maggiori (5x5 e 7x7). L'utilizzo di questi filtri così piccoli, hanno permesso di ridurre la complessità computazione e apprendere una combinazione lineare delle *feature maps*, riducendo il numero di parametri. Per ottenere ciò sono state introdotte anche convoluzioni con dimensioni 1x1. Successivamente, per ottenere maggiori prestazioni della rete neurale, è stato utilizzato un algoritmo di *max pooling* dopo lo strato di convoluzione, con l'utilizzo del *padding* per mantenere la risoluzione spaziale, evitando di incombere al problema della perdita di feature. Il limite più grosso di questa architettura si basa sull'utilizzo di 138 milioni di parametri che lo rendono molto più pesante dal punto di vista computazione e difficile da implementare su sistemi con bassa potenza di calcolo [30]

3.1.3 ResNet

La ResNet, formata da 152 strati, viene considerata come un estensione delle *deep* CNN perché venne introdotto il concetto dell'apprendimento residuo, rivoluzionando l'intera architettura delle reti neurali convoluzionali. Questa tecnologia permette di elaborare un nuovo metodo utile per la costruzione di sistemi di *Deep Learnig* basati sulle reti neurali. Nonostante questa architettura abbia una profondità maggiore rispetto alle precedenti, sviluppa una complessità computazionale minore. Infatti è stato dimostrato [37], che una ResNet con 50/101/152 strati presenta meno errori sulla classificazione delle immagini rispetto ai 34 strati di una rete normale, ottenendo un miglioramento delle prestazioni di circa il 28% su di un famoso *dataset* utilizzato per il riconoscimento delle immagini, ovvero COCO [14].

3.2 Detection

Si definisce Detection il task di trovare regioni rettangolari di un'immagine, nelle quali sono rappresentati oggetti di interesse. Queste regioni, dette *Bounding Box* vengono poi classificate per descrivere l'oggetto in esse contenuto (vedi Figura 3.2). A seguire sono elencate diverse architetture basate sulle reti neurali convoluzionali che ci aiutano a capire come il Deep Learnig viene applicato ai modelli che servono per il riconoscimento degli oggetti.

Nel 2014, venne proposta una nuova rete neurale, chiamata *Region-based Convolutional Neural Network* (R-CNN) [38] in grado di risolvere problemi di *Object Detection* che si basano sull'identificazione di diversi oggetti che appartengono ad una stessa immagine. Successivamente, questa rete è stata migliorata anno dopo anno, lasciando spazio alle reti Fast R-CNN e Faster R-CNN che venivano utilizzate nella segmentazione di singole istanze.

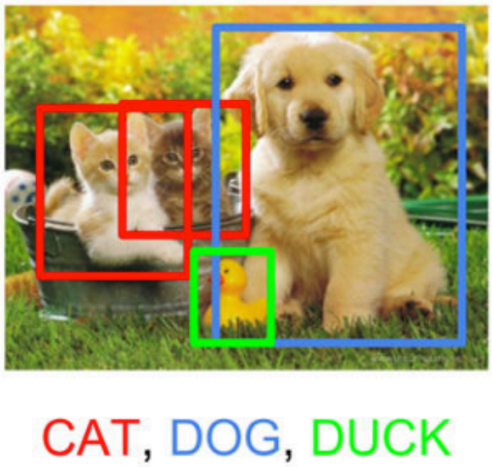

Figura 3.2: Esempio di detection e classificazione di un immagine rappresentante alcuni animali. [58]

3.2.1 R-CNN

Le rete R-CNN fu la prima architettura di CNN per *object detection* basata sull'uso di un *region proposal* che consiste nell'estrarre le aree delle immagine da analizzare in modo da classificarle e rilevare tutti gli oggetti presenti su di essi [39]. L'architettura R-CNN si basa, dunque, sul seguente funzionamento (Figura 3.3)[11]:

- *Region Proposal*: vengono estratte circa 2000 regioni di riferimento (RoI) utilizzando la propria euristica, tramite l'algoritmo Selective Search [41];

- *Resize della RoI*: consiste nel modificare la dimensione dell'immagine in modo da rispettare le dimensioni attese dalla CNN. La rete neurale non si occuperà della classificazione delle feature bensì della loro estrazione;

- *SVM (Support Vector Machine)*: permette di classificare le RoI in base alle feature estratte nel passo precedente;

- *Regressione della bouding box*: si occupa di ridimensionare la bounding box dell'oggetto in esame.

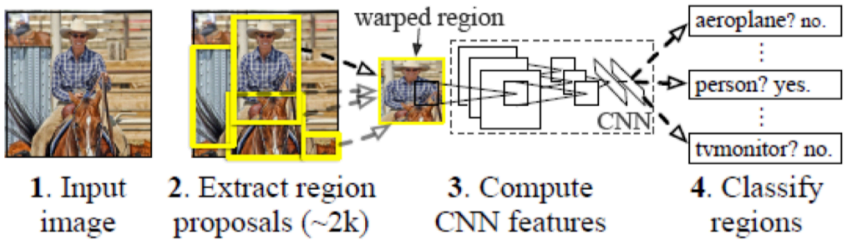

Figura 3.3: Schema di funzionamento del R-CNN. [11]

Il problema principale di questa architettura riguarda l'elevato calcolo computazionale, infatti, nella fase di *training* per ogni RoI bisogna:

1. allenare la rete neurale;
2. allenare il classificatore per ogni classe, in quanto R-CNN non utilizza un solo classificatore;
3. allenare l'algoritmo di regressione singolarmente.

Lo svantaggio principale, invece, nella fase di *testing* riguarda il partizionamento dell'immagine in circa 2000 sezioni anche se questa contiene un solo oggetto di interesse. Come già accennato, per ovviare ai problemi rilevanti da questa architettura molto costosa, fu implementa una nuova tecnica in modo da accelerare i tempi di lavoro.

3.2.2 Fast R-CNN

L'anno successivo allo sviluppo della R-CNN venne introdotta una nuova architettura, in grado di risolvere i problemi di lentezza che ne derivavano. La Fast R-CNN [40] propone una nuova sequenza di operazioni rispetto al suo predecessore,

ovvero le operazioni relative all'estrazione delle regioni di interesse e quelle della classificazione. Infatti l'estrazione delle RoI avviene, adesso, sull'ultima *feature map* estratta dalla rete. Facendo in questo modo, il tempo di elaborazione, nella fase di *training* si riduce drasticamente, in quanto è possibile ottenere un *feature extractor* perfettamente allenato in modo da estrarre le feature migliori per la fase di classificazione. Inoltre, non viene più utilizzato un classificatore SVM per ogni classe, ma questi sono stati sostituiti con un singolo classificatore *softmax*. Nonostante questa miglioria, il Selective Search, che si occupa come abbiamo detto di estrarre le regioni di interesse, risulta essere ancora molto lento perché estrae un numero elevato di regioni. Per questo motivo è stato pensata un'architettura in grado di integrare l'operazione di estrazione delle feature all'interno dell'architettura stessa.

3.2.3 Faster R-CNN

Faster R-CNNfaster (Figura 3.4) permette di migliorare, ancora una volta, le prestazioni del precedente algoritmo in modo da modificare il modo in cui vengono selezionate le regioni di interesse. In questa architettura, la parte relativa alla

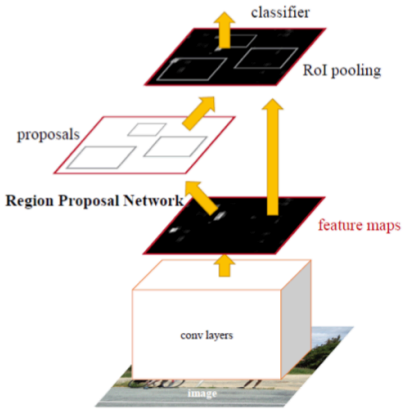

Figura 3.4: Architettura Faster R-CNN [11]

Region Proposal viene inserita all'interno della rete neurale e partecipare così nella

fase di *training*. Cosi facendo il Selective Search, che si occupava della fase dell'estrazione delle RoI viene sostituito dall'aggiunta di ulteriori strati alla rete neurale che permettono di ricavare le feature mediante una rete convoluzionale a tutti gli effetti, la Region Proposal Network (RPN), spiegata più avanti. In questo modo la classificazione viene effettuata solamente sulle regioni che la RPN fornisce agli strati successivi aumentando le prestazione nella fase si *testing*. Faster R-CNN, a parità di input, permette di aumentare la velocità di elaborazione di Fast R-CNN di circa 250 volte.

Region Proposal Network

La rete neurale in esame, consiste nel ricavare da un'immagine in ingresso l'insieme delle aree in cui vengono rilevati gli oggetti. A ciascuna di queste viene poi associata una probabilità che l'oggetto si trovi nell'area in questione. Successivamente all'ultimo strato della prima rete neurale, contenuta all'interno dell'architettura Faster R-CNN, viene fatta scorrere una finestra di dimensione $n \times n$, lungo la *feature maps*. Quest'operazione con lo scopo di rilevare i riquadri delle regioni di interesse, chiamate "ancore". Al variare di diversi parametri, ne verranno individuate al massimo k [11]. Quindi per una *feature maps* di dimensione $W \times H$ saranno estratte $W \cdot K \cdot k$ ancore. Una volta individuate le ancore queste verranno nuovamente elaborate dalla rete neurale. Gli strati seguenti saranno analoghi a quelli proposti dalla Faster R-CNN includendo così uno strato di pooling, un classificatore di tipo softmax ed un regressore relativo alle bounding box. All'architettura Faster R-CNN, negli anni successivi, sono state applicate ulteriori migliorie permettendo di definire il Mask R-CNN [45] utilizzato nei problemi di *istance segmentation* di cui discuteremo nel paragrafo successivo.

3.2.4 Mask R-CNN

La Mask R-CNN [45], viene considerato come una estensione del Faster R-CNN, ma a differenza di quest'ultimo, viene aggiunto un nuovo componente, insieme alla classificazione delle classi e alla determinazione delle bounding-box. Questo permette di predire i contorni di un oggetto in una regione di interesse. In particolare, per ogni RoI, una Mask R-CNN, permette di generare una maschera di dimensione 28×28 che indica quali pixel della regione di interesse appartengono all'oggetto. Successivamente la maschera verrà allargata fino ad ottenere le stesse dimensioni della *bounding box* di riferimento. Nella Figura 3.6, viene mostrata la struttura dell'architettura in questione. Per riassumere, una Mask R-CNN, per ogni istanza che viene individuata permetterà di restituire in uscita la classe a cui appartiene, la *bounding box* calcolata con la regressione ed una maschera binaria che sarà sovrapposta all'area selezionata [46].

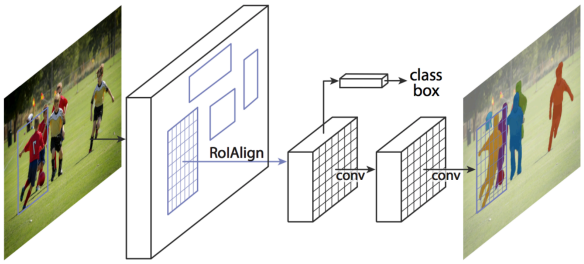

Figura 3.5: Architettura della Mask R-CNN. [11]

Con lo sviluppo di questa architettura, è stato introdotto il concetto del RoI Align, che va a sostituire l'algoritmo di RoI che abbiamo visto per le precedenti architetture, per l'estrazione delle feature. Con questo nuovo approccio, la regione estratta dalla *feature maps* non avrà più un valore approssimativo, come veniva fatto in precedenza producendo una perdita di informazioni, ma sarà esattamente uguale all'area estrapolata dalle feature.

Vediamo, nel paragrafo successivo, un nuovo processo architetturale per la object detection che si differenzia totalmente rispetto a quelle che abbimo visto fino ad ora.

3.2.5 YOLO

YOLO [43] (You-Only-Look-Once) (Figura 3.6) è un'architettura molto semplice rispetto a quelle che abbiamo appena visto ed utilizza una sola CNN per svolgere al compito dell'*object detection*. In particolare non utilizza una region proposal network per funzionare. Il funzionamento di YOLO, si basa principalmente

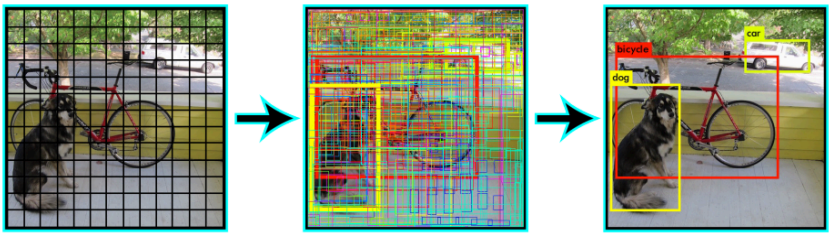

Figura 3.6: Applicazione di un'architettura YOLO. [11]

sul ridimensionare l'immagine di input ad una grandezza fissa per poi dividerla in una griglia di dimensione $S \times S$ che viene analizzata e classificata applicando una CNN. Il risultato, così ottenuto, consiste in un vettore che contiene le coordinate della regressione ed il vettore di appartenenza alle classi che la rete neurale assegna alla *bounding box* per ciascuna classe. Quindi, possiamo notare, che l'elaborazione dell'immagine avviene in un'unica operazione comportando così una maggiore velocità in termini computazionali delle architetture R-CNN. La rete viene, invece, applicata $S \times S$ volte, su ognuna delle celle definite, che in ogni caso rappresenta un numero molto più basso rispetto alle regioni che venivano analizzare dalla *region proposal*.

3.3 Segmentazione

La segmentazione delle immagini, è una parte fondamentale per i sistemi di *Computer Vision* che consiste nel partizionare un'immagine in diverse regioni rappresentanti i diversi oggetti. Questo processo si basa sull'estrazione, dal dominio dell'immagine, di una o più regioni, ovvero insiemi di pixel, connessi tra loro. La segmentazione, ricopre un ruolo di particolare importanza in diverse applicazioni tra cui, per esempio, immagini mediche, videosorveglianza e realtà aumentata.

Sono stati sviluppati numerosi algoritmi di segmentazione, da quelli che considerano un valore di soglia, alcuni che si basano sugli istogrammi, fino ad algoritmi più avanzati. Si parla di *semantic segmentation* quando il modello è anche in grado di stabilire la classe per ognuna delle regioni individuate. Negli ultimi anni, le reti di *Deep Learnig*, hanno prodotto una nuova generazione di modelli di segmentazione semantica delle immagini con un notevole miglioramento delle prestazioni, raggiungendo valori di accuratezza molto elevati.

Nella Figura 3.7 si può osservare il risultato di una segmentazione di alcune immagini utilizzando un modello di *Deep Learnig*.

Il risultato di una tecnica di segmentazione deve rispettare alcune proprietà:

- la segmentazione deve essere completa, ovvero tutti i pixel dell'immagine devono appartenere ad almeno una regione della partizione;
- l'insieme di pixel di una regione dell'imagine deve essere connesso;
- tutte le regioni di un'immagine devono essere separate tra di loro;
- deve soddisfare il criterio di omogeneità di feature(s) derivanti da componenti spettrali definiti in base all'intensità, dal colore o la *texture*.

Distinguiamo tre tipi di segmentazione:

- *Semantic Segmentation* (Figura 3.8(b));

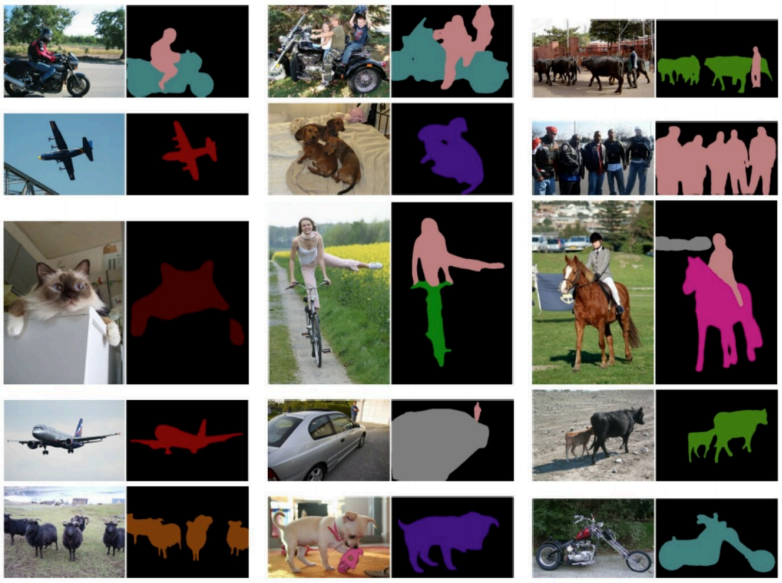

Figura 3.7: Segmentazione semantica di alcune immagini utilizzando un modello di *Deep Learning*. [21]

- *Istance Segmentation* (Figura 3.8(c));

- *Panoptic Segmentation* (Figura 3.8(d));

Come anticipato, effettuare una *segmentazione semantica* significa dividere un'immagine in diversi insiemi di pixel che devono essere opportunamente etichettati e classificati in una specifica classe (come ad es. animali, esseri umani, edifici ecc..). Quando è necessario effettuare la segmentazione semantica di ogni singola istanza (*semantic instance segmentation* o, più brevemente indicata come *instance segmentation*), l'obiettivo diventa quello di identificare (*Object Detection*), classificare al fine di determinarne l'esatta posizione e differenziare ogni oggetto dagli altri (anche se appartenenti alla medesima classe). L'*Instance Segmentation* è impegnativa perché richiede il rilevamento corretto di tutti gli oggetti in un'immagine e la loro segmentazione. Combina quindi elementi provenienti dai compiti classici di visione artificiale come l'*Object Detection*, in cui l'obiettivo è classificare singoli oggetti e

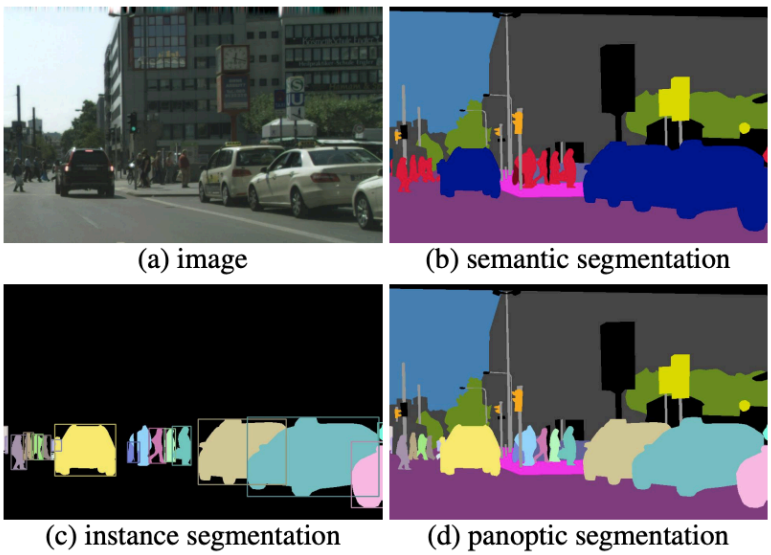

Figura 3.8: Per una data immagine (a), si mostra il ground truth per: (b) semantic segmentation, (c) instance segmentation, (d) panoptic segmentation. [13]

localizzarli utilizzando un *bounding box*, e *semantic segmentation*, in cui l'obiettivo è classificare ciascun pixel in un insieme fisso di categorie senza differenziare le istanze dell'oggetto [10]. Tuttavia, nonostante la classificazione delle immagini e l'*Object Detection* possano sembrare molto simili tra loro, in realtà sono due tecniche diverse. Mentre la classificazione si occupa, come già spiegato, di classificare un'immagine in una determinata categoria, per identificare la posizione degli oggetti in un'immagine, si utilizza la *Object Detection*. Le due tecniche non sono contrapposte, 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 una etichetta. La segmentazione panottica (*panoptic segmentation*), si occupa, invece, di includere tutto ciò che è visibile in uno scenario verso una visione globale e unificata della segmentazione. Ad ogni pixel di un'immagine viene assegnato un'etichetta semantica ed un ID d'istanza. Così, i pixel che avranno la stessa etichetta e lo stesso ID apparterranno allo stesso oggetto.

3.3.1 Panoptic segmentation

Come abbiamo visto, l'obiettivo della segmentazione semantica è quello di classificare ogni pixel in modo da attribuirgli le informazioni delle classi, mentre la segmentazione d'istanza si occupa di considerare ogni istanza di un oggetto in maniera separata. La segmentazione panottica, invece combina la segmentazione semantica e di istanza in modo tale che a tutti i pixel sia assegnata un'etichetta di classe e tutte le istanze degli oggetti siano segmentate in modo univoco. L'obiettivo nella segmentazione panottica è, quindi, quello di eseguire una segmentazione unificata, ma per fare ciò, analizziamo prima alcuni concetti di base. Le regioni di un'immagine vengono suddivise in due categorie: "*thing*" e "*stuff*". Una regione "*thing*" rappresenta un oggetto numerabile come per esempio: persone, auto, ecc.. Viene quindi associata al concetto di istanza di un oggetto. Le regioni "*stuff*", invece, sono aree amorfe dell'immagine con delle *texture* molto simili tra loro. Esempi di *stuff* possono essere: strada, cielo, vegetazione ecc.. Queste regioni non sono associate al concetto di istanza. Lo studio delle regioni "*thing*" rientra nella categoria dell'*object detection* e della segmentazione d'istanza, mentre lo studio di quelle di categoria "*stuff*" fanno parte della segmentazione semantica. Ad ogni pixel di un'immagine viene assegnata un'etichetta semantica ed un ID d'istanza, così, i pixel che avranno la stessa etichetta e lo stesso ID apparterranno allo stesso oggetto. A differenza della segmentazione dell'istanza, ogni pixel nella segmentazione panottica appartiene ad una sola istanza così da non comportare la possibilità di avere oggetti sovrapposte.

Come già accennato, sono state studiate ed elaborate diverse architetture per la segmentazione delle immagini basate sulle reti neurali convoluzionali, come per il Mask R-CNN, relativo alla segmentazione d'istanza, spiegato nel precedente paragrafo. A seguire verranno introdotte alcune architetture tra le più famose per la segmentazione semantica, la U-Net e la PSP-Net.

3.3.2 U-Net

La CNN che si basa sull'architettura U-Net [46][47] consiste in una serie di codifiche e contrazioni che servono per estrarre il contesto dell'immagine, seguite da una sequenza di decodifica ed espansione simmetrica. Questi ultimi processi servono per ricavare la classificazione di ogni singolo pixel. Ogni *step* che riguarda la contrazione riceve in ingresso un immagine è caratterizzato da 4 blocchi concatenati, ognuno dei quali presenta due strati convoluzionali, uno strato di *batch normalization* ed una funzione di attivazione ReLU, ed in fine un ultimo blocco di *max-pooling*. Per ogni blocco, il numero di filtri che vengono applicati, sarà il doppio di quello precedente, ovvero si parte da 64 *feature maps* estratte dal primo blocco, fino ad arrivare a 512 feature del quarto blocco. Durante questa fase notiamo che le informazioni riguardanti le feature vengono incrementate, mentre quelle che riguardano le informazioni spaziali diminuiscono tramite l'operazione di campionamento. La seconda parte dell'architettura, ovvero quella che riguarda l'espansione, è formata anch'essa di 4 blocchi e ognuno di questi permette di concatenare l'immagine sovra-campionata con quella che corrisponde al percorso di contrazione e successivamente applicare le due unità convoluzionali, identiche a quelle della fase precedente. Alla fine, le feature che usciranno dall'ultimo livello saranno di dimensione uguale a quella dell'immagine in ingresso ma con un numero più elevato di canali. Per concludere, lo strato di classificazione softmax si occupa di assegnare ad ogni pixel dell'immagine l'etichetta in base alla categoria di appartenenza. Si può osservare l'architettura appena descritta nell'immagine 3.9.

3.3.3 PSP-Net

L'architettura Pyramid Scene Parsing Network (PSPNet)[48] utilizza delle informazioni contestuali globali per una migliore segmentazione. In questo modello,

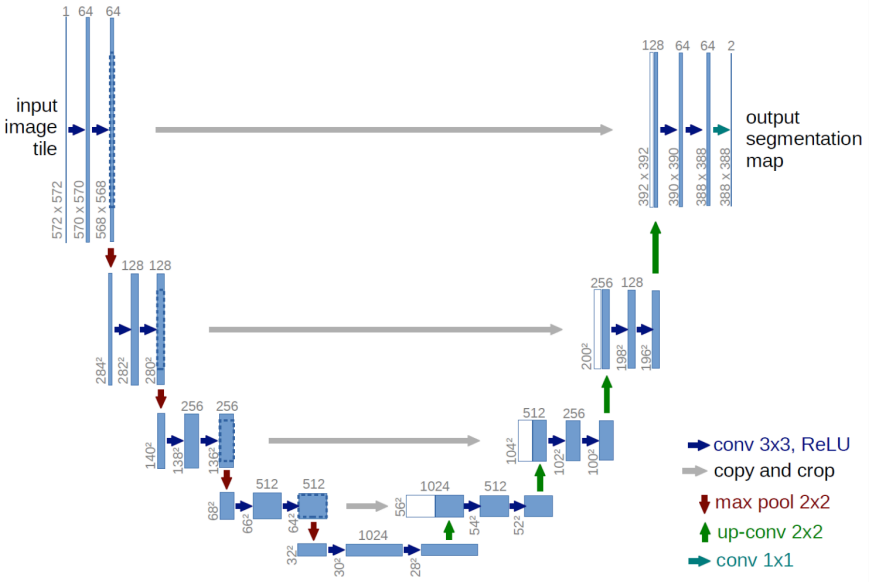

Figura 3.9: Esempio di un'architettura U-Net basata su un'immagine in ingresso di dimensione 572×572. [46]

gli autori, hanno usato un modulo di pooling piramidale al di sopra dell'ultima *feature map* (estratta usando una rete MLP *fully connected*). L'operazione di *pooling* sulle feature è applicata utilizzando 4 scale diverse corrispondenti a 4 livelli piramidali differenti: $1 \times 1, 2 \times 2, 3 \times 3$ *e* 6×6. Per ridurre le dimensioni delle *feature maps* viene applicata un'operazione di convoluzione usando uno strato convoluzionale di 1×1 in modo da sotto-campionarlo. Le *feature* così ottenute vengono infine concatenate alle *feature maps* iniziali in modo da ricavare le informazioni contestuali globali. Successivamente, questi risultati vengono sottoposti ad una nuova elaborazione da parte di un nuovo strato convoluzionale in modo da generare le predizioni riguardanti i pixel. La rete in esame permette dunque di esaminare tutte le feature in per ogni sotto-regione in diverse posizioni in modo da comprendere al meglio l'immagine e avere ottimi risultati che riguardano la segmentazione semantica. Si può osservare l'architettura appena descritta nell'immagine 3.10.

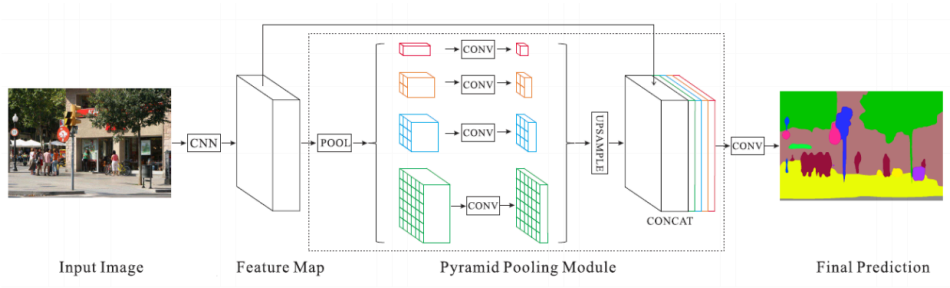

Figura 3.10: Esempio di un'architettura PSP-Net per la segmentazione semantica. [46]

3.4 Metriche di valutazione: Classification

Per valutare le prestazioni di un determinato modello di classificazione, esistono diverse e svariate metriche per analizzare i risultati. Queste funzioni richiedono in input le etichette reali *ground truth* e quelle derivanti dall'inferenza del modello. Restituiscono un valore numerico indice della qualità dei risultati.

Nelle prossime sezioni esamineremo alcune delle metriche di classificazione più diffuse: *accuracy*, *precision*, *recall* e *F1 Score*. Prima di vederle nel dettaglio, può essere utile definire i seguenti concetti:

- True Positive (TP): il set costituito dai campioni etichettati e classificati allo stesso modo;

- False Positive (FP): è un set costituito dai campioni classificati dal sistema come appartenenti ad una classe, ma realmente appartenenti ad un altra;

- False Negative (FN): costituita dai campioni classificati dal sistema come non appartenenti ad una determinata classe che, in realtà, ne fanno parte.

Matrice di confusione

La matrice di confusione è un utile ed efficace strumento di valutazione. Per un problema di classificazione con C classi, la matrice di confusione M è una matrice CxC dove l'elemento Mij corrisponde al numero di campioni appartenenti alla classe come i ma classificati dalla rete come j. Gli elementi della diagonale corrispondono invece ai campioni correttamente classificati. La matrice di confusione più semplice è quella di un problema di classificazione binario. Supponendo di avere due etichette, -1 e 1, la matrice di confusione corrisponde alla Figura 3.11, dove M_{11} corrisponde al vero-positivo. L'elemento M_{12} corrisponde ad un falso-negativo (FN). Similmente, M_{21} è un falso-positivo ed infine M_{22} un vero-negativo (TN).

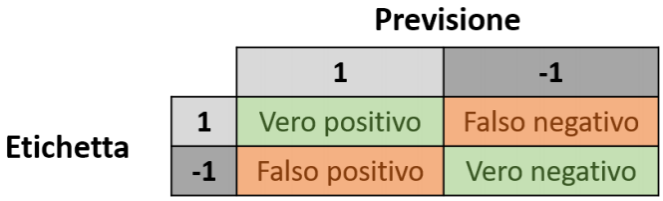

Figura 3.11: Matrice di confusione, caso binario [16]

Per un problema di classificazione multi-classe, la matrice diventa quella delle Figura 3.12, dove nella diagonale figurano i campioni classificati correttamente (veri-positivi), mentre nel resto della matrice sono presenti FP e FN.

Quindi, possiamo concludere che la matrice di confusione permette di capire le prestazioni del modello utilizzato, nel caso in cui le classe coinvolte non sono eccessive. In casi di classificazione in cui il numero di classi sono è elevato, diventa molto difficile estrapolare informazioni utili per una prima valutazione. Per questo motivo vengono utilizzate altre metriche riassuntive come quelle spiegate nei prossimi paragrafi.

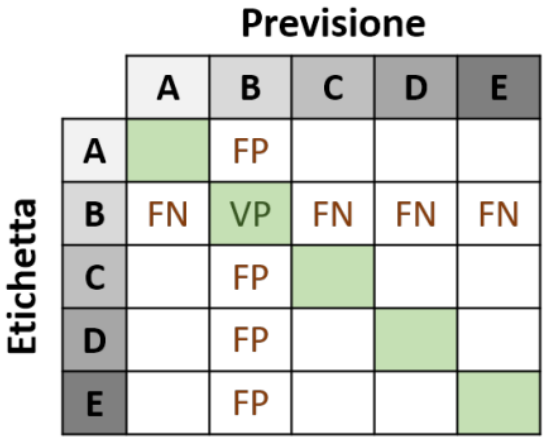

Figura 3.12: Matrice di confusione con 5 classi protect[16]

3.4.1 Accuracy

L'accuratezza (o *accuracy*), è la più semplice funzione per misurare la qualità di un classificatore e corrisponde alla frazione di campioni classificati correttamente.

Esprimiamo l'accuratezza come:

$$FP_i = \frac{\sum_i M_{ii}}{\sum_i \sum_j M_{ji}}$$

In un problema di segmentazione semantica:

- per ogni classe, l'accuratezza, è il rapporto tra il numero di pixel correttamente classificati e il numero totale della singola classe, secondo quanto specificato dal *ground truth*;
- per ogni immagine, il valore di accuratezza è l'accuratezza media di tutte le classi per la singola immagine;
- per l'intero *dataset*, il valore di accuratezza è la accuratezza media di tutte le classi in tutte le immagini.

L'accuratezza globale, invece, è la frazione di pixel correttamente classificati, indipendentemente dalla classe. Quindi possiamo concludere che questa metrica esprime una valutazione generale della bontà del modello, è solo indicativa e non può essere considerata affidabile sopratutto in presenza di *dataset* sbilanciati.

3.4.2 Precision, Recall e F1 Score

La precisione (*precision*) e il richiamo (*recall*) sono due indici che permettono una lettura semplificata della matrice di confusione permettendo di misurare la bontà dei risultati.

Definiamo la *precision* come il rapporto tra il numero dei campioni che sono stati classificati correttamente e il totale degli elementi classificati, compresi i FP:

$$Precision = \frac{M_{ii}}{\sum_j M_{ji}}$$

Viene definito *recall* il rapporto tra il numero totale degli elementi classificati correttamente e il numero totale degli elementi che il sistema dovrebbe recuperare se questo funzionasse alla perfezione:

$$Recall = \frac{M_{ii}}{\sum_j M_{ij}}$$

Queste due metriche sono molto utili per capire l'affidabilità di un sistema, ma spesso può essere comodo considerare un solo valore, per questo motivo, si calcola il valore *F1 Score*, che calcola la media armonica dei due valori calcolati precedentemente:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

F1 può assumere valori compresi tra 0 ed 1, dove 1 indica un risultato perfetto, mentre 0 indica un risultato inattendibile. Nel caso di problemi multi-classe, questo valore, equivale alla media dei valori di F1 di ogni classe. In relazione alla segmentazione semantica, questo parametro è indice della vicinanza tra contorno tracciato e contorno segmentato. Nello specifico:

- per ogni classe, l'F1 medio, è la media degli F1 della classe su tutte le immagini;
- per ogni immagine, il valore di F1 è dato dalla media degli F1 di tutte le classe per la singola immagine;
- per l'intero dataset, l'F1, è la media degli F1 di tutte le classi in tutte le immagini.

3.5 Metriche di valutazione: Detection

Per la valutazione della detection viene utilizzata maggiormente una metrica chiamata *intersection over union* (IoU). Il coefficiente IoU è un parametro che si occupa di valutare il risultato prodotto dalla fase di detection.

- per ciascuna classe (C), l'IoU viene calcolato come il rapporto tra i pixel che vengono classificati correttamente (TP) diviso la somma del numero complessivo di pixel che sono stati etichettati con la classe C o che nella *ground truth* appartengono a quella stessa classe (C). Il numeratore corrisponde quindi all'intersezione tra i pixel di classe C secondo le predizioni e secondo la *ground truth*. Il denominatore, invece, è l'unione dei pixel di classe C per la *ground truth* e per le predizioni.

$$IoU = \frac{TP}{(TP + FP + FN)}$$

- per ciascuna immagine, il valore IoU medio (mIoU) corrisponde alla media dei valori di tutte le classi per l'immagine considerata;

- per tutto il dataset, mIoU corrisponde alla media dei valori di tutte le classi per tutte le immagini.

Nel caso in cui il dataset fosse non bilanciato, si può calcolare il valore IoU medio in base al numero di pixel totali per ogni immagine. A differenza dell'accuratezza, questo valore metrico, descrive al meglio la qualità della segmentazione risultante.

3.6 Metriche di valutazione: Segmentation

Le tecniche di segmentazione hanno diverse metriche per valutare le prestazioni. Nella segmentazione semantica, vengono spesso utilizzate la metrica IoU, che abbiamo appena visto per la detection e la *pixel accuracy*. Per la segmentazione panottica è possibile utilizzare una combinazione di queste due metriche, ma questo provoca dei problemi di simmetria tra le classi in base alla presenza o meno di annotazioni. Per questo motivo è stata utilizzata una nuova metrica chiamata *Panoptic Quality* che tratta in maniera identico i diversi tipi di categorie. Nei prossimi paragrafi verranno espresse le caratteristiche principali di *pixel accuracy* e *Panoptic Quality*.

3.6.1 Pixel accuracy

La pixel accuracy [49] [50] permette di valutare la segmentazione semantica riportando la percentuale dei pixel dell'immagine che sono stati classificati correttamente. Questa metrica, ci permette di calcolare l'accuratezza dei pixel per ogni classe considerando una maschera binaria, dove un vero-positivo rappresenta un pixel appartenente alla classe che è stato associato correttamente (secondo la

maschera), mentre un vero-negativo rappresenta un pixel che viene correttamente identificato come non appartenente alla classe in esame.

$$pixel\ accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

I valori ottenuti possono anche fornire risultati fuorvianti quando la classe ha pochi pixel all'interno delle immagini, contenenti per esempio altre classi più frequenti. La classe minoritaria può non essere classificata correttamente, in quanto il modello classifica tutti i pixel come una delle classi maggioritarie. Ciò implicherebbe comunque valori elevati della *pixel accuracy*. Possiamo quindi concludere che l'elevata precisione dei pixel non implica sempre un'ottima capacità di segmentazione. Questo problema si chiama squilibrio di classe: quando le nostre classi sono estremamente sbilanciate, significa che una classe o alcune classi dominano l'immagine, mentre altre classi costituiscono solo una piccola porzione dell'immagine.

3.6.2 Panoptic quality

Per la segmentazione panottica [13], è stata introdotta una nuova metrica, in quanto le tecniche di valutazione che abbiamo visto fino ad ora sono specializzate per la segmentazione semantica o di istanza e non possono essere utilizzati per valutare l'efficienza della loro combinazione. Per avere una valutazione ottimale della Segmentazione Panottica serve verificare diversi requisiti:

- **Completezza**: la metrica dovrebbe trattare gli oggetti (di entrambe le categorie tra "*thing*" e "*stuff*") in maniera uniforme;

- **Interpretabilità**: la metrica deve essere facile da interpretare facilitandone la comprensione;

- **Semplicità**: inoltre, deve essere semplice da definire ed applicare. Questa proprietà migliora la trasparenza e consente una facile re-implementazione. In relazione a questo, la metrica dovrebbe essere facile da calcolare per consentirne una rapida valutazione.

La Panoptic Quality (PQ) soddisfa queste proprietà misurando la qualità di una previsione panottica relativa alla ground truth seguendo due principi base:

- **Segment Matching**: nessun singolo pixel può appartenere a due segmenti contemporaneamente, ovvero non devono esserci delle previsioni sovrapposte. Inoltre, un segmento calcolato, può essere abbinato alla *ground truth* solo se il suo IoU con la ground truth abbia un valore maggiore di 0,5. Questo requisito fa in modo che la valutazione risulti semplice ed efficiente in quanto le corrispondenze sono uniche e secondariamente risulta abbastanza interpretabile e facilmente comprensibile. Per IoU più piccoli, sarebbero necessarie altre tecniche per l'abbinamento con il *ground truth*, ma è stato dimostrato che i casi con IoU minori o uguali 0,5 si possono verificare raramente nella realtà;

- **PQ Computation**: la PQ viene calcolata per ogni classe in maniera indipendente per poi calcolare la media su tutte le classi. Questo rende quindi la PQ insensibile allo squilibrio di classe. Per ogni classe, infatti, la corrispondenza univoca tra valori predetti e *ground truth* può avere tre valori diversi, ovvero veri-positivi, falsi-positivi e falsi-negativi. Definiamo la PQ come:

$$PQ = \frac{\sum_{(pg)\in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

  dove $\frac{1}{|TP|}\sum_{(pg)\in TP} IoU(p,g)$ rappresenta il valore IoU medio della corrispondenza tra i dati predetti e il ground truth, mentre $\frac{1}{2}|FP| + \frac{1}{2}|FN|$ viene sommato al denominatore per penalizzare le regioni predette senza corrispondenza.

Da notare come tutti i segmenti ricevano tutti la stessa importanza, indipendentemente dalla loro area. Inoltre dividendo e moltiplicando PQ per la dimensione dei TP possiamo notare che la Panoptic Quality può essere vista come la moltiplicazione tra due metriche chiamate *segmentation quality* (SQ) e la *recognition quality* (RQ):

$$PQ = \frac{\sum_{(pg)\in TP} IoU(p,g)}{|TP|} \times \frac{|TP|}{\sum_{(pg)\in TP} IoU(p,g)}$$

dove la RQ non è altro che il valore di F1 che indica quanti oggetti sono trovati correttamente dal modello, mentre la SQ consiste nel mIoU dei segmenti corrispondenti che indica quanto accurata è la segmentazione delle forme dei singoli oggetti. Possiamo, dunque, notare che i due valori non sono indipendenti tra di loro, in quanto SQ viene misurata solo sui segmenti corrispondenti. Possiamo concludere dicendo che la PQ misura le prestazioni di tutte le classi in maniera uniforme utilizzando una formula semplice e facilmente interpretabile, considerando sia la capacità di riconoscere gli oggetti (RQ), sia la qualità della segmentazione (SQ).

3.7 Dataset

Come descritto in precedenza, la qualità del Dataset utilizzato è fondamentale per il *Machine Learning*. Un dataset è definito come un enorme collezione di dati, nel nostro caso immagini etichettate. Più grande sarà il dataset, maggiore sarà la qualità del modello risultante. Solitamente un dataset viene suddiviso in tre parti distinte:

- **Training Set**: questa porzione di dati viene utilizzata per l'addestramento del modello che si vuole realizzare;

- **Validation Set**: porzione di dati utilizzata per validare il modello e scegliere la migliore configurazione dei suoi iper-parametri. La qualità dei risultati viene valutata mediante una delle metriche rappresentate nella sezione 3.4. Valutare le prestazioni di un dataset, ovvero del *training set*, permette di evitare il fenomeno dell'*overfitting* (Sezione 2.2.1);

- **Test Set**: questo insieme, invece, viene utilizzato solo per testare la soluzione finale, scelta tra quelle che hanno la miglior qualità sul *validation set*.

La dimensione dell'intero dataset (D) sarà data dall'unione tra le tre parti, *training set* ($D_{training}$), *validation set* ($D_{validation}$) e *test set* (D_{test}):

$$D = D_{training} \cup D_{validation} \cup D_{test}$$

Una proprietà molto importante del dataset è che l'intersezione tra le tre parti deve essere pari a 0:

$$D_{training} \cap D_{validation} = D_{validation} \cap D_{test} = D_{test} \cap D_{training} = 0$$

La maggior parte delle volte, è impossibile creare dei dataset di enormi dimensioni, per questo motivo esistono numerosi dataset pubblici messi a disposizione da diversi fornitori. Tra i maggiori, nel campo dell'*image understanding* annoveriamo: COCO e ADE20K.

3.7.1 COCO

Il dataset di Microsoft Common Objects in COntext (MS COCO) [14] è utilizzato per l'*object detection*, la *segmentation* e il *captioning*; la versione del 2017 (utilizzato in questo lavoro di tesi) contiene immagini, bounding box e *labels*. Questo dataset, comprende immagini riguardante la vita di tutti i giorni che contengono oggetti comuni nei loro contesti naturali. vediamo alcune particolarità:

- alcune immagini del training set e del validation set non hanno alcuna annotazione;
- COCO 2014 e COCO 2017 usano le stesse immagini, ma con una partizione diversa per le tre parti, training, validation e test set;
- il test set è composto solamente da immagini senza alcuna annotazione;
- il dataset definisce 91 classi (tipo di oggetti), ma solamente 80 di loro sono utilizzati;
- le annotazioni panottiche riguardano 200 classi, ma solamente 133 classi sono utilizzate.

Nella Tabella 3.1 viene mostrato il numero di immagini contenute in ognuno dei tre insiemi del dataset COCO.

Tabella 3.1: Partizione del dataset COCO

Set	*Immagini*
Test	40,670
Training	118,287
Validation	5,000

3.7.2 ADE20K

Tutte le immagini appartenenti a questo dataset [15], sono state annotate con degli oggetti. A loro volta, una buona parte degli oggetti, è stata annotata con le parti che li compongono. Per ogni oggetto, sono presenti ulteriori informazioni, come il fatto che gli oggetti siano occlusi o troncati. Le immagini del *validation set* sono completamente annotate, mentre le immagini del training set non sono annotate in maniera esaustiva. Le informazioni che riguardano questo set di dati sono in continuo aggiornamento.

Nella Tabella 3.2 è stato indicato il numero di immagini contenuti all'interno del *training set*, del *validation set* e del *test set* del dataset ADE20K.

Tabella 3.2: Partizione del dataset ADE20K

Set	*Immagini*
Test	3,000
Training	20,210
Validation	2,000

Parte II

Analisi semantica delle immagini

Capitolo 4

Estrazione di conoscenza dai dati

Al giorno d'oggi, la continua crescita di dati multimediali nella nostra società dell'informazione, mette in risalto la necessità di fornire gli strumenti che servono per analizzarli. Infatti la grande quantità di dati, presenti nelle diverse basi di dati, se opportunamente elaborate ed analizzate, possono fornire un valore aggiunto nella scelta di applicare determinate decisioni. Il *data mining*, per citare un esempio, si occupa di selezionare, esplorare e modificare dati di grandi dimensioni, con lo scopo di estrapolarne le informazioni necessarie per ottenere un'interpretazione dei risultati utile ai fini del lavoro per il quale si basa l'analisi. L'obiettivo del *data mining* è quello di ricercare pattern potenzialmente significativi in modo da estrarre da essi la "conoscenza" che può essere tradotta in un modello per la generalizzazione dei dati a disposizione.

I dati ottenuti vengono successivamente archiviati in una struttura dati, chiamata *Knowledge base* in modo da poter essere consultata ed esaminata.

4.1 Knowledge base

Una *Knowledge base* (KB) [51] non è semplicemente un spazio utile solamente all'archiviazione dei dati, ma può essere vista come uno strumento per l'intelligenza

artificiale in grado di fornire supporto alle decisioni. La conoscenza può essere rappresentata in diversi modi, come per esempio *frame* o *script* in modo da descrivere una spiegazione di un concetto, un tipo di ragionamento per fornire un supporto per qualche decisione da prendere. Un *frame* è definito come un "contenitore" di informazioni riguardanti oggetti di varia natura, mentre uno *script* è una struttura che descrive una sequenza stereotipata di eventi.

Gli strumenti ingegneristici che si basano su una *Knowledge base* permettono di aiutare i progettisti in modo da fornirgli suggerimenti e soluzioni aggiuntive contribuendo così alle decisioni che vengono prese in fase di design. Esistono due tipi di *Knowledge base*: quelle che possono essere interpretate dall'uomo, senza aver bisogno di un ulteriore supporto computazionale e quelle che per essere utilizzate hanno bisogno di ulteriori processi computerizzati. Le *Knowledge base* interpretabili dall'uomo consentono alle persone di accedere e utilizzare le risorse in esse contenute, come per esempio libretti di istruzioni, manuali e possibili risposte a domande frequenti. Questo tipo di informazioni possono essere consultate dagli utenti in maniera interattiva, per esempio tramite un sito web. Così facendo l'utente deve solamente fornire delle semplici informazioni in modo guidato affinché possa ottenere le risposte desiderate. L'altro tipo di *Knowledge base*, può essere compresa solamente dal sistema di elaborazione in maniera automatica. Ovviamente, l'utilizzo di questo tipo di conoscenza si basa su un approccio diverso, in quanto l'interpretazione dei dati avviene mediante lo sviluppo di un software ad hoc che sia in grado di interrogare i dati contenuti al suo interno.

Negli ultimi anni, le *knowledge base*, sono stata largamente utilizzate come risorsa per fornire conoscenza in modo da facilitare la ricerca e aiutare il processo decisionale in vari settori. I progressi riguardanti l'estrazione delle informazioni, hanno portato alla costruzione di basi di conoscenza semantiche e strutturate grazie ad informazioni contenute nel Web (ad esempio, DBPedia [52], YAGO [53])

permettendo così di avere una certa mole di dati su una vasta gamma di argomenti riguardanti, ad esempio, persone, organizzazioni, città, paesi, animali ecc. Oltre alle *knowledge base* riguardanti informazioni generali, esistono basi di conoscenza specifiche in determinati settori come GeneOntology [54] e WordNet [55] che forniscono informazioni utili solo ad un gruppo ristretto di persone, come gli scienziati o i ricercatori. In questo modo le informazioni sono limitate al dominio di interesse dell'utilizzatore. Il *Knowledge Graph* di Google, per esempio, è una grande *knowledge base* che contiene oltre 70 miliardi fatti su persone, luoghi e organizzazioni (con dati aggiornati al 2016) [51]. Facebook sta costruendo *knowledge base*, chiamata "Entity Graph" contenente dai dei propri utenti, con l'obiettivo di potenziare la ricerca sui social network[51]. LinkedIn contiene informazioni riguardanti utenti, società, competenze, e certificati [51].

Per concludere, le *Knowledge base* possono essere classificate in due categorie [51]:

- **curated KB**: sono create e costruite manualmente da un gruppo di persone. Questa tipologia di KB ha un'elevata qualità dei dati, ma con il passare del tempo, questi vengono superati molto velocemente, in quanto non vengono aggiornati di continuo [53];

- **open KB**: raccolgono automaticamente informazioni contenute nel web; la conoscenza che rappresentano è sempre aggiornata ma non hanno una qualità molto alta, come le *curated KB*, i quali essendo costruite a mano hanno un'accuratezza maggiore.

4.1.1 Rappresentazione della KB

Le *Knowledge base*, sono caratterizzate principalmente da due componenti. La prima riguarda la rappresentazione dei dati e descrive come sono rappresentati i fatti, mentre la seconda si basa sulla rappresentazione della conoscenza in modo

da definire le classi, le categorie e le tassonomie per comprendere al meglio la conoscenza contenuta nella KB. La rappresentazione dei dati, avviene tramite una modellazione come entità e relazione, che seguendo rigorosamente uno schema ben definito.

Il Resource Description Framework (RDF) è ampiamente usato come linguaggio di modellazione dei dati per curated KB. In RDF, ogni dato è costituito da una tupla a tre elementi (soggetto, predicato, oggetto) in cui il soggetto e l'oggetto sono collegati da un predicato.

$$\langle soggetto, predicato, oggetto \rangle$$

Il modello di un RDF si compone di tre parti [57]:

1. qualsiasi entità descritta nella KB viene chiamata *Resource*. Ad esempio una pagina web, o un elemento XML possono essere considerati delle risorse. Una risorsa è sempre identificata tramite un "anchor id" o da un *uri* (Uniform Resource Identifier);

2. un *Property* è una caratteristica, una specifica, un attributo o una relazione che viene utilizzata per descrivere un determinato oggetto. Queste possono essere identificate da un nome e possono assumere valori in un determinato dominio;

3. uno *Statement* è una tupla formata da un soggetto (risorsa) da un predicato (proprietà) e da un oggetto (valore). L'oggetto può essere una sequenza di caratteri, un XML, o un'altra risorsa.

Una Knowledge base può essere dunque rappresentata da una serie di tuple collegate tra loro per formare un grafo (cioè un grafo RDF). Un esempio di Knowledge Base RDF è rappresentato in figura 4.1.

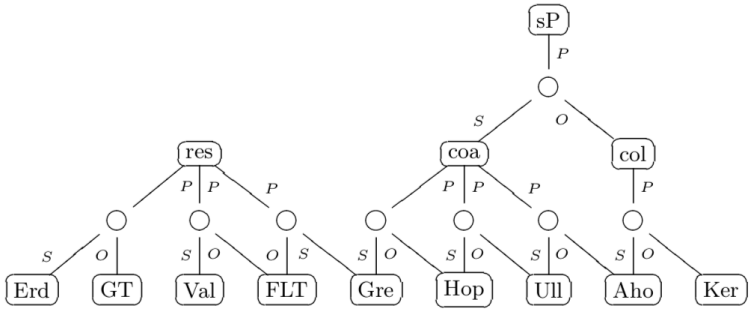

Figura 4.1: Il grafico mostra un esempio di Knowledge Base, dove i cerchi rappresentano i nodi, mentre S, P, O indicano rispettivamente il soggetto, il predicato e l'oggetto [56]

4.2 Applicazioni dell'estrazione di conoscenza dai dati

Nei capitoli precedenti, sono state discusse le modalità in cui il *Deep Learning* viene usato nel processo di *image understanding*. Nello specifico sono state analizzate diverse strutture di rete neurali convoluzionale utilizzate per elaborare e classificare le immagini. I modelli di *Deep Learning* non sempre restituiscono informazioni accettabili, in quanto commettono spesso degli errori banali che non sono coerenti con l'interpretazione che l'essere umano conferisce alla realtà. Inoltre, le CNN sono considerate delle "Black Box", ovvero delle "scatole nere", che ricevendo in ingresso degli iper-parametri e delle funzioni progettate ad hoc, riescono a generare un risultato senza poter comprendere come si è arrivati a calcolarlo. Per questo motivo le CNN non sono considerate interpretabili [67]. Questi problemi, possono essere risolti mediante l'utilizzo di conoscenza preveniente dalle KB. Ad esempio, è possibile *anomaly detection* cercando di capire se la CNN genera risultati "anomali" e quindi non corretti. Le anomalie possono essere trovate conoscendo le distribuzioni dei dati (come si comportano nella norma). In questa tesi, ad esempio, analizziamo come sono frequentemente posizionati gli oggetti nelle immagini. Se la

CNN trova oggetti che non rispettano questi pattern, probabilmente ha commesso un errore. Questo tipo di analisi è detta "di contesto", dove si analizzano e si classificano gli oggetti considerando il significato di tutta l'immagine. Per analizzare il contesto possono essere utili le KB che descrivono situazioni comuni che vengono rappresentate nelle immagini.

4.2.1 Anomaly detection

L'*Anomaly Detection* permette di individuare elementi, avvenimenti o osservazioni che si discostano dalla normalità, chiamate *anomalie*. Uno dei compiti del processo di *anomaly detection* è quello di stabilire quali dati rappresentano un comportamento normale e quali, invece, sono considerate delle anomalie. L'esecuzione di questo processo può comportare delle difficoltà:

- talvolta, non è facile stabile quando un comportamento può essere definito normale e quando invece un'anomalia. Per questo motivo i processi di *anomaly detection* potrebbero restituire dei falsi-positivi e dei falsi-negativi;
- i comportamenti dipendono dal tipo di contesto analizzato. Infatti, un comportamento normale in una determinata area, può essere considerato un'anomalia in un altro dominio.

Nella Figura 4.2 è rappresentato un semplice esempio del processo di *anomaly detection* applicato ad un dataset bidimensionale. Le aree che contengono elementi con comportamenti normali sono indicate con N_1 e N_2, dove si sono allocati la maggior parte degli elementi. I punti o_1 e o_2 e la regione O_3 rappresentano le anomalie che dovrebbero essere rilevate, in quanto si discostano dal comportamento comune dei dati.

Tipi di Anomalie

Le anomalie possono essere classificate in tre categorie [59]:

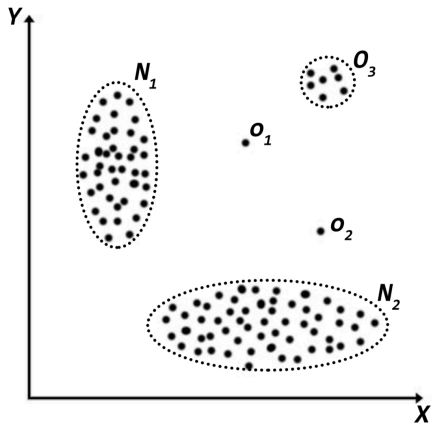

Figura 4.2: Esempio di anomalia puntuale [59]

- **Anomalie puntuali**: si hanno quando un'unica istanza di dati si discosta dal resto del dataset assumendo dei comportamenti diversi rispetto agli altri. Questo dimostra che l'anomalia si trova al di fuori dei confini della regione normale. I singoli punti o_1 e o_2 della Figura 4.2 vengono considerati come anomalie puntuali;

- **Anomalie contestuali**: si verificano quando le informazioni mostrano un comportamento anomalo e in determinati contesti. In tal caso bisogna specificare il contesto nella definizione del problema. La Figura 4.3 mostra il grafico delle temperatura di una città tipica italiana, dove si registrano alcune temperature molto basse. Queste vengono considerate normali durante il mese di Dicembre, in quanto mese invernale, mentre vengono considerate come anomalie durante un mese estivo come Giugno.

- **Anomalie collettive**: si hanno quando un insieme di istanze di dati si comportano in maniera diversa rispetto al resto delle istanze del dataset. Un esempio è rappresentato da una sotto-sequenza anomala in un elettrocardiogramma (vedi Figura 4.4)

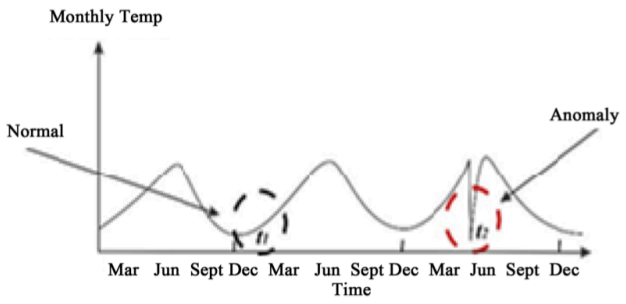

Figura 4.3: Esempio di anomalia contestuale [59]

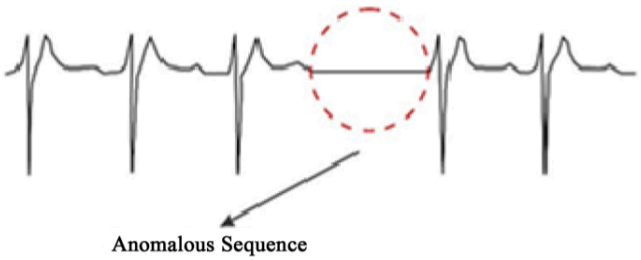

Figura 4.4: Esempio di collettiva [59]

Tecniche di Anomaly Detection

Esistono diverse tecniche di Anomaly detection [60] che possono essere classificate secondo due criteri. Il primo riguarda le tecniche che si basano sulla natura del dataset utilizzato per la progettazione del modello. Il secondo riguarda il metodo con cui le anomalie vengono separate. In base al primo criterio distinguiamo le seguenti tecniche:

- **Apprendimento supervisionato**: a ciascuna istanza del dataset viene assegnata un'etichetta, "anomalo" o "normale" da parte di esperti, in modo tale da trattare la ricerca delle anomalie come se fosse un problema di classificazione. Affinché la classificazione restituisca ottimi risultati, bisogna innanzitutto bilanciare il dataset in modo da avere approssimativamente la stessa quantità

di dati etichettati in entrambe le classi. Un metodo classico usato per il bilanciamento è quello dell'*oversamplig* che permette di aumentare la distribuzione dei dati anomali nel *training set* utilizzato per allenare il classificatore;

- **Apprendimento non-supervisionato**: tecnica utilizzata quando i dati non sono etichettati, considerando i dati "normali" clasterizzabili. Questi modelli nascono dall'esigenza di ridurre i costi ed il tempo necessario per l'etichettatura visto che questo intervento viene fatto solamente da esperti in materie, ciò rende più probabile la generazione di falsi-positivi e falsi-negativi

- **Apprendimento semi-supervisionato**: tecnica utilizzata quando solamente una parte dei dati contenuti nel dataset presentano un'etichetta. Il funzionamento consiste nello sfruttare la parte dei dati etichettati per cercare, nel sottoinsieme senza etichetta, le istante di dati che più si somigliano.

In base al metodo con cui le anomalie vengono separate, distinguiamo le seguenti tecniche:

- **Metodi statistici**: si basano su l'utilizzo di processi stocastici e considerano i dati che si trovano nelle zone ad alta probabilità come normali, mentre quelli che si trovano in zone con basse probabilità vengono considerate anomalie;

- **Metodi basati sulla prossimità**: l'efficienza di questi metodi dipendono dal tipo di metrica utilizzata, infatti l'idea è quella di ipotizzare che le anomalie si trovino in una posizione lontana rispetto a quella dei suoi vicini. Questo metodo si basa sulla distanza oppure sulla la densità. Nel primo caso un'istanza viene considerata anomala quando la distanza rispetto ai suoi vicini supera una certa soglia. Nel secondo caso, invece, il modello mette a confronto la densità del dato e quella dei loro vicini; se la densità calcolata è inferiore a quella degli altri allora viene considerata un'anomalia.

- **Metodi basati sul *Clustering***: ipotizzano che le anomalie facciamo parte di cluster piccoli e sparsi, mentre i dati normali appartengano a cluster di grandi dimensioni e molto densi. Un'anomalia, viene considerata tale se il dato:
  - non appartiene a nessun cluster;
  - è molto distante rispetto al cluster più vicino;
  - appartiene ad un cluster molto piccolo; in questo caso l'intero cluster viene considerato anomalo.

4.2.2 Analisi Contestuale

I tradizionali approcci utilizzati per la categorizzazione degli oggetti, si basano su caratteristiche estetiche come fonte principale per il riconoscimento delle classi di oggetti a cui appartengono. Le caratteristiche dell'oggetto, come il colore, il contorno dei bordi, la *texture* e la forma, non sempre danno dei risultati esatti. Infatti in presenza di rumore, di movimento, o di una scarsa luminosità, la categoria, a cui appartiene l'oggetto, può essere interpretata grazie alla presenza degli agli altri elementi che compongono scena. Le informazioni che riguardano le configurazioni tipiche degli oggetti in una scena, sono state studiate per anni in psicologia e nella visione artificiale, al fine di comprenderne gli effetti nella ricerca visiva, nella localizzazione e nel riconoscimento. Biederman et al. [63] hanno proposto cinque diverse relazioni tra un oggetto e l'ambiente circostante: interposizione, supporto, probabilità, posizione e dimensione. Queste classi permettono di stabilire l'organizzazione degli oggetti coerentemente con il mondo reale. Le classi corrispondenti all'interposizione e al supporto possono essere codificate facendo riferimento allo spazio fisico, mentre la probabilità, la posizione e la dimensione vengono definite relazioni semantiche, in quanto richiedono il significato semantico dell'oggetto stesso. Le relazioni semantiche includono le informazioni sulle interazioni tra i vari

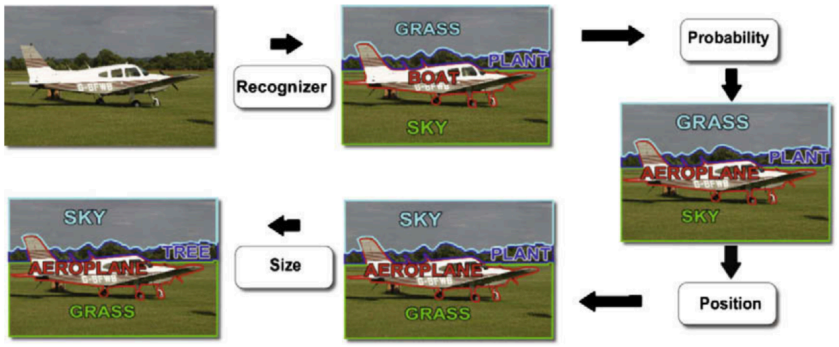

Figura 4.5: Applicazione dell'analisi contestuale in un processo di segmentazione [62]

oggetti della scena e sono spesso utilizzate come caratteristiche contestuali, permettendo di ridurre i tempi di elaborazione. Un esempio è mostrato nella Figura 4.5. Dove un'immagine contenente un aereo, degli alberi, il cielo e l'erba, viene prima elaborata attraverso un modello basato sulla segmentazione, in modo da estrarre le possibili etichette degli oggetti. Senza applicare le informazioni contestuali, possiamo notare che sono presenti diversi errori, assegnando in maniera errata le classi degli oggetti. Il contesto semantico (probabilità) sotto forma di co-occorrenza di oggetti, consente di correggere l'etichetta dell'aereo, ma lascia incorrette le etichette del cielo, dell'erba e della pianta. Il contesto spaziale (posizione) definisce che è più probabile che il cielo appaia sopra l'erba e viceversa. Infine, il contesto di scala (dimensione) corregge il segmento etichettato come "pianta" etichettandola come "albero", in quanto le piante sono solitamente più piccole degli alberi e del resto degli oggetti nella scena [62].

In questo lavoro di tesi è stato applicato un approccio in grado di analizzare la posizione semantica delle classi di oggetti (come vedremo nel prossimo capitolo), che permette di avere un'interpretazione più astratta degli oggetti.

4.2.3 Miglioramento delle performance nell'*image recognition*

Negli ultimi anni, sono stati sviluppati diversi lavori che si basano sulla posizione semantica degli oggetti considerando un insieme di valori discreti. Il lavoro svolto da Galleguillos [68], per esempio, sviluppa un nuovo approccio per la categorizzazione degli oggetti incorporando due tipi di contesto: uno basato sulla semantica ed un altro contesto che si basa sulle informazioni spaziali. Il lavoro svolto, denominato CoLA (Co-occurence, Location and Appareance) utilizza dei campi aleatori condizionali (CRF) per etichettare gli oggetti in maniera più coerente in base alle informazioni contestuali disponibili. L'approccio utilizzato sfrutta l'uso di più segmentazioni prima dell'elaborazione vera e propria in modo da fornire un raggruppamento spaziale di pixel all'interno delle regioni degli oggetti analizzati, in modo da ottenere delle prestazioni migliori rispetto all'utilizzo di algoritmi più semplici come Bag of Features (BoF). La posizione dei due oggetti, dunque, viene definita attraverso la quantizzazione di semplici coppie di feature in modo da permettere l'apprendimento di un sottoinsieme di relazioni spaziali direttamente dai dati. Per l'esecuzione di tale algoritmo sono stati utilizzati due dataset diversi, quali PASCAL 2007 e MSRC, che mostrano come l'utilizzo della combinazione dei due contesti migliorino l'accuratezza dei risultati ottenuti rispetto all'utilizzo della sola ricorrenza.

4.2.4 Obiettivi

L'obiettivo principale di questa tesi è quello di progettare ed esaminare nuove tecniche di estrazione della conoscenza per descrivere il comportamento di oggetti di varia natura nelle immagini. I pattern che verranno estratti dai dati etichettati, verranno poi utilizzati per supportare e validare i risultati dei metodi *Deep-Learning* per la segmentazione semantica degli oggetti.

Il processo presentato è suddiviso in due parti fondamentali. Nella prima parte, vengono calcolate le posizioni relative ai singoli oggetti di ogni immagine grazie alle informazioni che sono fornite all'interno del *dataset* utilizzato. Per ogni immagine, sono stati selezionati, in maniera casuale, due oggetti in modo da etichettare manualmente la relazione relativa alla loro posizione nell'immagine con lo scopo di addestrare l'algoritmo. A partire da questi dati etichettati è stato addestrato un classificatore in grado di calcolare automaticamente le posizioni relativa tra gli oggetti. Successivamente, dopo aver calcolato le posizioni di tutti gli oggetti tramite il classificatore, viene effettuato un processo di estrazione della conoscenza tramite l'elaborazione delle immagini nel *training set*. In questo modo il sistema sviluppato è stato in grado di analizzare le immagini da addestramento e creare automaticamente una base di conoscenza che descrive il comportamento, attraverso delle statistiche, delle diverse classi di oggetti per ciascuna relazione.

Nel prossimo capito verrà spiegato nel dettaglio il funzionamento dell'algoritmo appena descritto.

Capitolo 5

Design dell'architettura proposta

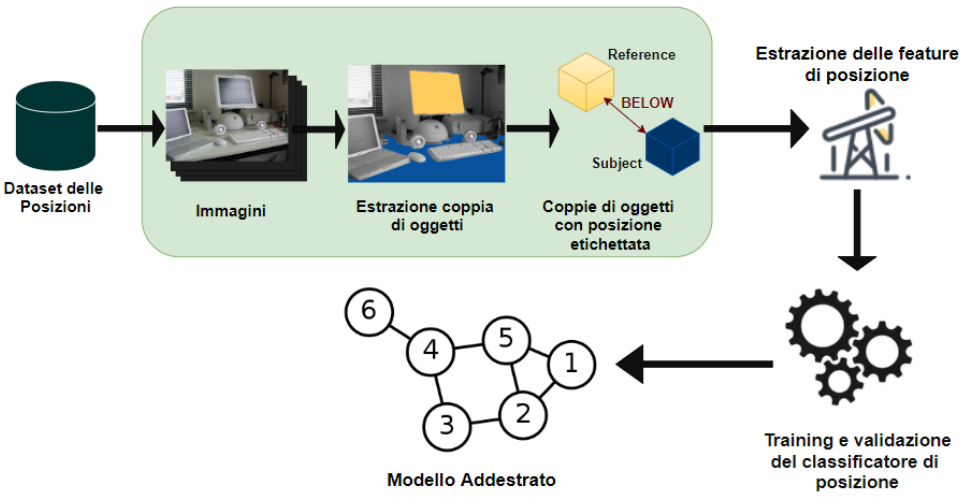

Figura 5.1: Diagramma che rappresenta l'addestramento del classificatore della posizione relativa di due oggetti

In questo capitolo verrà presentata l'architettura del processo proposto in questo lavoro di tesi. A valle di un processo di addestramento del classificatore di posizione, tramite le feature estratte dalle immagini contenute all'interno del *training*

set, verrà costruita automaticamente una base di conoscenza in grado di descrivere, attraverso informazioni statistiche, le relazioni che legano due oggetti qualsiasi contenuti all'interno delle immagini analizzate.

L'architettura presentata può essere suddivisa in due parti fondamentali:

1. Addestramento del classificatore della posizione relativa di due oggetti;

2. Generazione della *Knowledge base*.

La prima parte dell'architettura, come mostrato nella Figura 5.1, descrive il processo di addestramento del classificatore per la predizione della posizione relativa di due oggetti. In ingresso al processo, verranno fornite le immagini segmentate estratte da un sottoinsieme del *training set* contenuto all'interno del dataset MS COCO, insieme alle coppie di oggetti la cui posizione è già stata assegnata attraverso una procedura manuale, in modo da formare il *ground truth*.

Definito così il dominio delle posizioni, verranno poi estratte le *feature* delle immagini, attraverso tecniche descritte in questo capitolo, in modo da poter effettuare l'addestramento e la validazione del classificatore di posizione per ottenere il modello finale.

La seconda parte, , mostrata nella Figura 5.2, riguarda la creazione delle statistiche e di conseguenza la generazione della KB. Questo processo, a differenza della prima parte, prende in input solo le immagine segmentate, corrispondenti al *ground truth*. Esegue poi gli stessi passaggi per l'estrazione delle coppie di oggetti e delle loro *feature*, le quali vengono successivamente utilizzate dal modello di classificazione precedentemente addestrato per estrarre le posizioni relative agli oggetti. Una volte ottenute le posizioni, queste vengono analizzate in modo da ottenere delle statistiche (sotto forma di istogrammi) utili per generare la *Knowledge Base*.

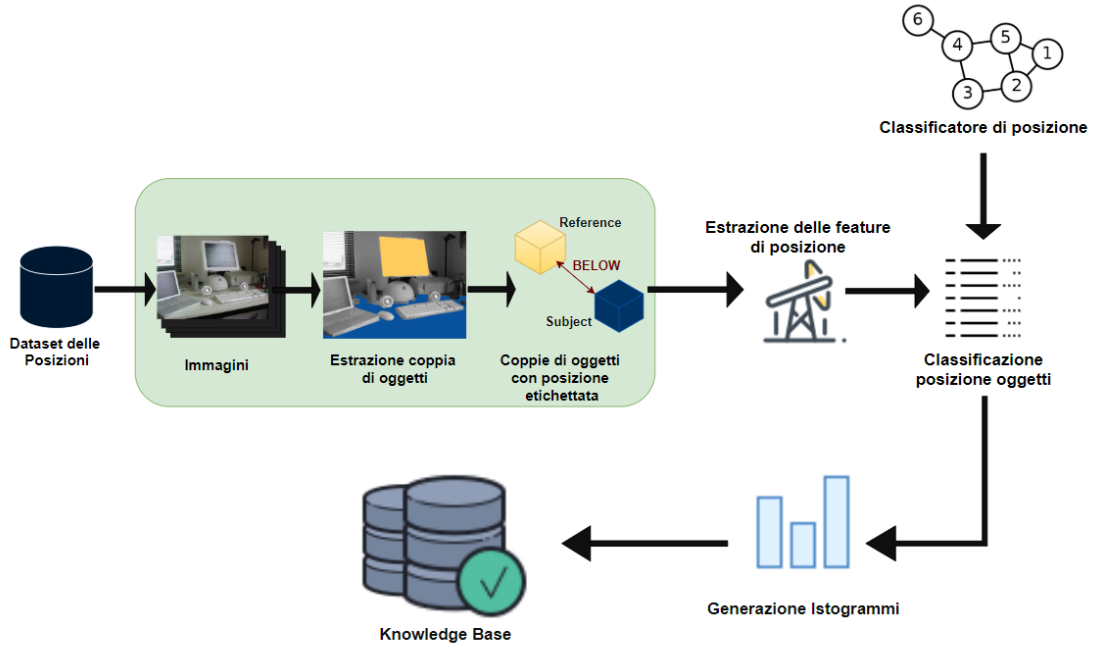

Figura 5.2: Diagramma che raffigura il processo di generazione della Knowledge Base

5.1 Addestramento del classificatore della posizione relativa di due oggetti

Il calcolo delle posizioni relative a due oggetti, indicati in questo lavoro di tesi con i termini *Subject* e *Reference*, consiste nello stabilire in maniera automatica la relazione tra due oggetti qualsiasi, contenuti all'interno di una stessa immagine. Per ottenere questo risultato è stato addestrato un classificatore che, prendendo in ingresso le feature estratte dalle coppie di oggetti nel *training set* e la *ground truth* costruita manualmente, sia in grado di restituire la relazione che lega i due oggetti.

A seguire, sono descritte le categorie relative alle posizioni che sono state utilizzate in questo lavoro.

5.1.1 Categorie di posizione

Per rilevare le posizioni tra gli oggetti, sono stati utilizzati insiemi di valori discreti, come ad esempio: sopra, sotto, intorno, dentro, che permettono di avere una migliore rilevanza semantica, piuttosto che considerare valori continui per le coordinate degli oggetti. Inoltre, l'utilizzo di attributi discreti, ha permesso di esaminare l'effettiva posizione considerando la forma e non solamente il *bounding-box*.

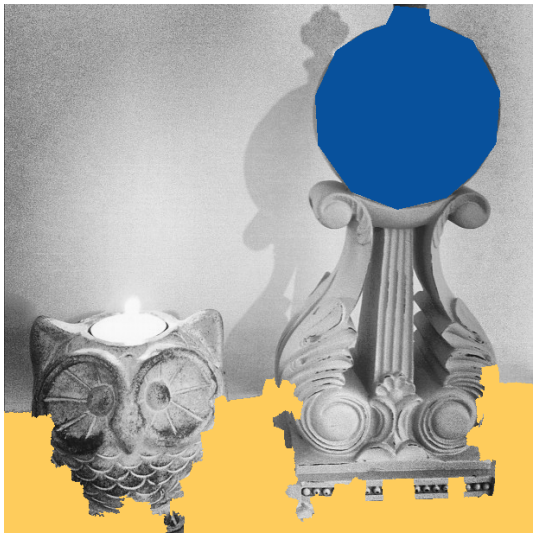

Figura 5.3: Esempio di immagine con la posizione tra i due oggetti etichettata come "above"

Figura 5.4: Esempio di immagine con la posizione tra i due oggetti etichettata come "inside"

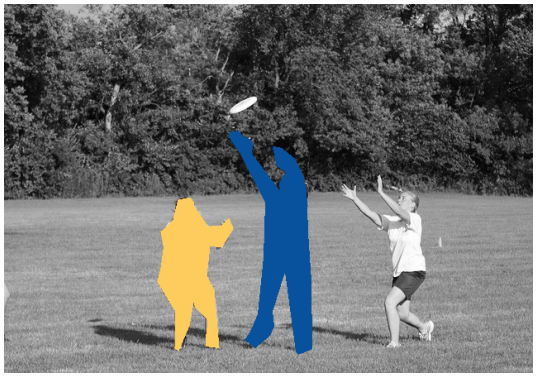

Figura 5.5: Esempio di immagine con la posizione tra i due oggetti etichettata come "side"

Figura 5.6: Esempio di immagine con la posizione tra i due oggetti etichettata come "side-up"

La relazione tra i due oggetti dell'immagine, quindi, viene descritta mediante una delle seguenti classi:

- **Above/Below**: vengono utilizzate quando i due oggetti in esame si trovano nella stessa posizione orizzontale, ma con una posizione verticale differente, separate da uno spazio intermedio non rilevante, a seconda che il *subject* si trova sopra o sotto l'oggetto a cui fa riferimento; nella Figura 5.3 viene mostrata un'immagine dove la relazione tra i due oggetti è stata etichettata come "above";

- **On/Hanging**; quando i due oggetti, sono adiacenti e condividono la stessa posizione orizzontale si utilizza *on* nel caso in cui il *subject* si trova al di sopra rispetto al *reference*, invece, quando il *reference* si trova sopra il *subject* viene applicato l'attributo *hanging*;

- **Inside/Around**: nei casi in cui l'oggetto principale è contenuto all'interno del *reference* viene applicata l'etichetta *inside*, mentre quando il ruolo dei due oggetti è invertito si parla di *around*; la Figura 5.4 mostra un esempio di oggetti etichettati come "inside", in cui un aeroplano (*subject*) è "contenuto" all'interno del cielo (*reference*).

- **Side** quando i due oggetti, come mostrato nella Figura 5.5, sono affiancati, ovvero condividono la stessa posizione verticale, ma hanno una posizione orizzontale differente si definiscono *side*;

- **Side-up/Side-down**: nel caso in cui i due oggetti non sono allineati né orizzontalmente né verticalmente, ad esempio quando *reference* e *subject* si trovino in due angoli opposti dell'immagine, l'etichetta sarà assegnata in base alla posizione dell'oggetto di riferimento a seconda che questo si trova con una posizione verticale più alta o più bassa dell'oggetto a cui si riferisce: *side-up* e

side-down; La figura 5.6 mostra un esempio di immagine in cui i due oggetti sono stati etichettati come "side-up".

5.1.2 Estrazione delle feature di posizione

Come è stato già accennato, per il calcolo delle posizioni sono state utilizzate le feature estratte dalle immagini. Per ogni colonna di pixel dell'immagine, è stato applicato un algoritmo in grado di individuare la categoria relativa alla posizione dei pixel di ogni singolo oggetto rispetto alla posizione dei pixel di un altro oggetto. In questo modo, considerando il numero di colonne con una determinata categoria è stato possibile dedurre la posizione finale dei due oggetti.

Di seguito, viene mostrato l'algoritmo utilizzato per l'estrazione delle feature per ogni coppia di oggetti presenti nell'immagine.

Algoritmo 1: Pseudocodice per il calcolo relativo alle posizioni degli oggetti

input : Lista di tuple contenente gli id degli oggetti per ogni colonna
output: Feature relative alle diverse categorie per ogni coppia di oggetti

```
1  lastObject ← None;
2  lastOccurrence ← None;
3  for all column ∈ image do
4  |   objectList ← getObjects(column);
5  |   numObjects ← lenght(objectList);
6  |   for i ← 0 to numObjects − 1 do
7  |   |   sub ← column[i];
8  |   |   for j ← i + 1 to numObjects − 1 do
9  |   |   |   ref ← column[j];
10 |   |   |   for pos ← 0 to length(column) do
11 |   |   |   |   look for: sr, rs, s+r, r+s, srs, rsr;
12 |   |   |   end
13 |   |   end
14 |   end
15 end
```

L'Algoritmo 1 riceve in ingresso una lista di tuple (*image*) opportunamente raggruppate che contengono gli identificatori degli oggetti presenti in ogni colonna di pixel, in modo da elaborare ed estrarre le diverse *feature* dei due oggetti in esame. Per ogni colonna dell'immagine (riga 3), vengono considerate tutte le possibili coppie univoche formate da tutti gli oggetti presenti in essa (dove *sub* rappresenta il *subject*, riga 7, mentre *ref* identifica il *reference*, riga 9), per poi esaminare una coppia alla volta. Il quarto *loop* dell'algoritmo (riga 10) analizza tutti i pixel della colonna, dove durante questa fase, viene ricercata una tra le seguenti sequenze: *sr*, *rs*, *s+r*, *r+s*, *srs*, *rsr*. Gli indici *s* ed *r* rappresentano rispettivamente il *subject* e il *reference*. La sequenza *sr* indica che i due oggetti sono adiacenti e il *subject* si trova sopra al *reference* identificando così la posizione "on", al contrario la sequenza *rs* viene associata ala posizione "hanging" in quanto il *subject* si trova al di sotto del *reference*. Con la sequenza *s+r*, si identifica la posizione "above" perchè il subject si trova sopra al *reference* ma non sono in contatto tra loro, invece la sequenza *r+s* rappresenta la posizione "below". La serie *srs* indica che la posizione è "around" perchè il *reference* è contenuto all'interno del *subject*, viceversa *rsr* si riferisce a "inside".

Il numero di colonne associate ad ognuna delle sequenze descritte sopra viene utilizzato come *feature* per classificare la posizione di *subject* e *reference*.

In aggiunta all'algoritmo appena descritto, è stato sviluppato un altro algoritmo che permette l'estrazione di feature relative alla distanza, sia verticale che orizzontale, tra i due oggetti, in modo da essere utilizzate dal classificatore per fare una predizione su un eventuale posizione laterale.

Per ogni oggetto dell'immagine è stato calcolato un *bounding box*, tramite una maschera, contenente tutti i pixel dell'oggetto in esame, in modo da poter individuare le coordinate relative alle estremità di ogni lato del rettangolo e raggrupparli all'interno di un array:

$$[boxTop, boxLeft, boxBottom, boxRight]$$

dove rappresentano, in ordine da sinistra verso destra, il punto più alto, il punto più a sinistra, il punto in basso e il punto più a destra dell'oggetto contenuto all'interno del riquadro.

Successivamente è stata calcolata la distanza orizzontale e verticale tra i due oggetti seguendo le seguenti definizione:

$$\Delta_{Y1} = reference_{bottom} - subject_{top}$$

$$\Delta_{Y2} = reference_{top} - subject_{bottom}$$

$$\Delta_{X1} = reference_{right} - subject_{left}$$

$$\Delta_{X2} = reference_{left} - subject_{right}$$

dove *reference* e *subject* indicano due oggetti qualsiasi dell'immagine, mentre *top*, *bottom*, *left*, *right* indicano rispettivamente i punti più in alto, più in basso, più a sinistra e più a destra dell'oggetto al quale si riferiscono.

Ogni feature estratta dall'algoritmo è stata divisa per il valore minimo tra la larghezza del bounding-box del *subject* (*subjectWidth*) e quella del *reference* (*referenceWidth*) in modo da ottenere una normalizzazione bilanciata, così da rendere indipendente la scelta dell'oggetto *subject*:

$$norm = min(subjectWidth, referenceWidth)$$

Per quanto riguarda la normalizzazione delle feature utili al calcolo delle posizioni *Side* sono stati calcolati i seguenti valori:

$$normY = max(abs(\Delta_{Y1}), abs(\Delta_{Y2}))$$

$$normX = max(abs(\Delta_{X1}), abs(\Delta_{X2}))$$

dove $normY$ corrisponde al valore massimo, in valore assoluto, tra Δ_{Y1} e Δ_{Y2}, mentre $normX$ indica il valore massimo, in valore assoluto, tra Δ_{X1} e Δ_{X2}. In questo modo è stato possibile ottenere il risultato finale dividendo le feature ottenute precedentemente come segue:

$$\Delta_{Y1} = reference_{bottom}/normY$$

$$\Delta_{Y2} = reference_{top}/normY$$

$$\Delta_{X1} = reference_{right}/normX$$

$$\Delta_{X2} = reference_{left}/normX$$

Così facendo, i valori delle feature sono stati normalizzati rispetto alla dimensione degli oggetti, in modo tale che il sistema di classificazione possa comprendere al meglio quanto i *bounding box* sono distanti tra loro, relativamente alle dimensioni dei due oggetti.

Ottenute le varie feature sarà possibile utilizzare il classificatore per predire la posizione degli oggetti su tutto il training set.

5.1.3 Training e validazione del classificatore di posizione

Come abbiamo visto, la classificazione permette di assegnare un'etichetta ad ogni dato, in modo da poterlo classificare correttamente nella categoria corrispondente utilizzando le feature per addestrare il classificatore.

Nel nostro caso sono state utilizzate le caratteristiche estratte da ogni coppia di oggetti in ogni immagine, come spiegato nel capitolo 5.1.2. Il *ground truth* utilizzato per addestrare il modello, è stato costruito etichettando manualmente le immagini in relazione alla posizione in cui i due oggetti selezionati come *subject* e *reference* si trovano effettivamente, secondo le regole che sono state discusse in precedenza. In questo lavoro di tesi sono stati analizzati diversi algoritmi di classificazione in modo da capire, tramite le metriche di valutazione viste nel paragrafo 3.4, quale

classificatore restituisce dei risultati migliori per calcolare in maniera automatica le posizione di due oggetti contenente all'interno delle immagini.

Di seguito viene spiegato il funzionamento dei diversi algoritmi di classificazione analizzati.

Classificazione Lineare e Multipla

Il processo di classificazione si basa sul considerare i vettori di feature come punti nello spazio a più dimensioni e di individuare piani o superfici che suddividono tale spazio separando i punti di classe diversa utilizzando un delimitatore (retta, piano o iper-piano) restituito dal processo di training. In base al numero di classi da predire, si possono identificare due tipi di classificazione: Esistono due tipi di classificatori:

- *Classificazione Binaria*;
- *Classificazione Multipla.*

Nel primo caso, i dati da classificare possono appartenere solamente a due categorie, delimitati da un iper-piano o da una superficie curva. Spesso, nel caso di dati presi dal mondo reale, una classificazione esatta è impossibile, in quanto è impossibile trovare una superficie che separi esattamente i due tipi di punti. Per questo motivo esistono numerose strategie in modo da ridurre al minimo il numero di errori. Per fare ciò è necessario stabilire un'adeguata "funzione di errore" da minimizzare che consenta di raggiungere risultati soddisfacenti con una complessità accettabile. Per una classificazione più efficace, è necessario adottare metodi diversi da quelli lineari, per esempio considerare funzioni di grado superiore come una superficie sferica in modo da separare meglio i punti rispetto ad un semplice iper-piano. Nelle applicazioni a problemi reali, non sempre si ha la necessità di classificare solamente 2 diversi tipi di dato. Spesso, come accade nelle immagini, si considerano diversi tipi di categorie. Per questo tipo di analisi viene considerata la *Classificazione Multipla*

dove i punti da classificare possono essere smistati su tre o più insiemi. Esistono diversi algoritmi di classificazione, vediamo quelli utilizzati in questo lavoro:

- Nearest Neighbors;
- SVM;
- Decision Tree;
- Random Forest.

Nearest Neighbors

Questo algoritmo di è uno dei più antichi e semplici per la classificazione. Tuttavia se opportunamente utilizzato produce risultati molto competitivi restituendo risultati migliori rispetto ad altri algoritmi. Il funzionamento si basa nell'etichettare ogni *sample* in base a come sono stati classificati la maggioranza dei suoi k-vicini del dataset di addestramento. Le prestazioni del *Nearest Neighbors* dipendono, quindi, dalla distanza che viene utilizzata per identificare i campioni vicini rispetto al *sample* di riferimento. La metrica della distanza per la classificazione non è uguale per tutti, infatti deve essere adattata al particolare problema da risolvere.

Support Vector Machine

Una Support Vector Machine (SVM), è un algoritmo di classificazione che, dato un insieme di campioni rappresentati come punti di uno spazio multidimensionale, utilizza uno o più iper-piani per la separazione delle classi. Questo tipo di classificatore è molto utile nel caso in cui si hanno molte feature. Le SVM possono usare il *Kernel trick*, ovvero mappare implicitamente l'input in uno spazio di dimensioni maggiori e in questo modo realizzare un classificatore non lineare con l'uso di superfici di separazione più complesse. Tra le funzioni usate per il task di classificazione non lineare consideriamo il Radial Basis Function (RBF) che è di tipo esponenziale.

Decision Tree

Gli alberi di decisione sono algoritmi usati sia nel *Data mining* che nel *Machine learning*. Le foglie di un albero di decisione rappresentano le classi di output e ogni nodo rappresenta uno split dei dati in base ad una delle feature. I rami rappresentano, invece, gli insiemi di valori delle feature. L'algoritmo di *learning* degli alberi di decisione consiste nella suddivisione del dataset in input in base al valore di una delle feature. Tale partizionamento è ricorsivo, ovvero può fare uso di tutte le possibili feature, oppure fermarsi quando ulteriori splitting non porterebbero cambiamenti o miglioramenti nella struttura dell'albero e quindi nelle capacità di classificazione. Questo tipo di classificatore funziona bene in presenza di grandi quantità di dati ed è robusto rispetto alla presenza di feature poco significative per la classificazione. Tuttavia la classificazione basata su alberi di decisione non è sempre accurata, perché questi algoritmi presentano il problema dell'*overfitting* in presenza di un numero eccessivo di feature.

Random Forest

Si tratta di algoritmi di decisione che fanno uso di più alberi di decisione in modo da avere maggiore robustezza dell'output e risultati più accurati rispetto a quelli ottenibili utilizzando un solo albero. L'output è dato da un *majority vote classifier* che assegna l'esempio alla classe a cui è stata assegnata la maggioranza degli alberi facenti parte della *random forest*. L'uso di alberi di decisione multipli permette di attenuare la tendenza all'*overfitting* da parte di ogni singolo albero.

Nel paragrafo successivo viene mostrato come i risultati ottenuti dalla classificazione, vengono utilizzati per costruire la *Knowledge base*.

5.2 Estrazione di relazioni frequenti tra oggetti

In questo paragrafo verrà descritto come il modello di classificazione addestrato, insieme alle statistiche estrapolate dalla predizione delle posizioni, vengano utilizzate per costruire la KB.

La conoscenza, estratta dal processo di classificazione, viene memorizzata nella *Knowledge base* in modo da essere utilizzata per per analizzare le informazioni statistiche che riguardano le relazioni dell'oggetto, per ciascuna coppia di classi. Come abbiamo visto, ogni regola è formata da una rappresentazione Resource Description Framework (RDF) (vedi capitolo 4.1.1), ovvero viene costruita una tripletta contenente i due oggetti (*subject* e *reference*) e la relazione che li lega, calcolata dall'algoritmo di classificazione. Inoltre, per ogni tripletta, vengono calcolati degli istogrammi utili per analizzare la frequenza delle diverse configurazioni per ogni coppia di oggetti relative all'intero dataset.

5.2.1 Generazione della Knowledge base

Come appena accennato per la costruzione della KB sono stati calcolati degli istogrammi associati ad ogni coppia di classi degli oggetti.

L'istogramma è una distribuzione di frequenza che permette di capire quante volte, in termini percentuali, due oggetti con delle classi specifiche (per esempio: "tavolo", "pavimento") si trovano nella posizione definita dal classificatore, analizzando tutte le immagini del training set.

Prima di descrivere le tecniche utilizzate per la generazione della KB vediamo alcune definizioni:

- **Tripletta**: dati due oggetti s (*subject*) e r (*reference*); data una posizione pos che descrive la relazione che intercorre tra due oggetti della stessa immagine, allora una tripletta

$$\mathrm{T} = \langle s, pos, r \rangle$$

definisce la relazione di posizione *pos* che intercorre tra l'oggetto s e l'oggetto r.

- **Istogramma**: data una tripletta $\mathrm{T} = \langle s, pos, r \rangle$, un istogramma è definito da

$$h(\mathrm{T}) = [l(p_0), \ldots, l(p_{N-1})]$$

dove ogni valore $l(p_i)$ rappresenta la probabilità che i due oggetti s e r soddisfino una determinata posizione p_i (tra quelle descritte dl capitolo 5.1.1), mentre N rappresenta il numero di classi di posizioni.

Come si può vedere nell'Algoritmo 2, per la realizzazione degli istogrammi, sono stati utilizzati dei dizionari, uno per ogni *tripletta*. Per ogni immagine del *training set*, sono state estratte tutte le possibili coppie di oggetti contenute in essa. Successivamente, per ogni coppia estratta ed ordinata in ordine alfabetico tramite la funzione *Sort()* (riga 4), è stato applicato il modello di classificazione addestrato precedentemente in modo da estrarne la posizione (riga 5), ovvero la relazione che intercorre tra *subject* e *reference*. L'informazione così ottenuta viene memorizzata all'interno di un dizionario, istanziato alla riga 1. Associato ad ogni chiave del dizionario (classe di *subject* e *reference*, posizione) vi è un contatore che verrà incrementato di uno tutte le volte che tra le immagini analizzate si presenterà la configurazione specificata.

Al termine di tale procedura, l'istogramma è stato normalizzato in termini di percentuale dividendo i suoi valori per il *supporto (sup) (riga 10)*. Il valore del supporto di un istogramma rappresenta tutte le volte in cui la coppia di oggetti appare nella totalità delle immagini. In questo modo è possibile interpretare gli elementi degli istogrammi come una distribuzione discreta di probabilità, la quale descrive la probabilità che due oggetti delle classi specificate assumono una certa posizione. L'istogramma, inoltre, ci permette di capire quanto una distribuzione può essere

Algoritmo 2: Algoritmo per il calcolo dell'istogramma

```
1  Histograms = {};
2  for each Img do
3  |  for each Pair do
4  |  |  Sub, Ref = Sort(Pair);
5  |  |  Pos = position(Sub, Ref);
6  |  |  Histograms[Sub, ref][Pos] =+ 1;
7  |  end
8  end
9  for each h in Histograms do
10 |  h = h/support(h);
11 end
```

significativa o meno. Nel caso in cui tutte le posizioni abbiano la stessa probabilità, o valori molto vicini tra di loro, allora la coppia in esame non ci darà alcuna informazioni utile in quanto nel dataset gli oggetti di quella classe non assumono nessuna posizione frequente. (vedi Listato 5.1). Invece, un buon istogramma, presenta una distribuzione non omogenea. Per esempio, consideriamo un istogramma relativo alla coppia (fiume, cielo):

$$[l(below) = 0.92, l(inside) = 0.0, l(side - down) = 0.04, l(above) = 0.0, \dots]$$

Da questa notazione, è possibile dedurre che nel 92% dei casi, il fiume si trova al di sotto del cielo, mentre tutte le altre posizioni possono essere trascurate.

Per individuare gli istogrammi con queste caratteristiche si è scelto di utilizzare l'entropia come spiegato nel paragrafo successivo.

Listato 5.1: Estratto di KB rappresentante una coppia di oggetti con un istogramma poco significativo

```
"(`mirror-stuff', `stop sign')": {
    "below": 0.25,
    "above": 0.25,
    "side-down": 0.25,
    "around": 0.25,
    "sup": 4
}
```

Entropia

Il concetto di entropia, venne introdotto all'inizio del XIX secolo nel campo della termodinamica classica in connessione con il secondo principio. Nel contesto

termodinamico, l'entropia rappresenta una grandezza dello stato di un sistema fisico in grado di esprimere la capacità del sistema stesso di ospitare trasformazioni spontanee con conseguente perdita di capacità a compiere lavoro quando si verificano tali trasformazioni. In altri termini, il valore dell'entropia cresce, man mano che il sistema subisce dei cambiamenti spontanei e quindi perde parte della sua capacità a subire tali cambiamenti ed eseguire lavoro. L'entropia è quindi un espressione del "grado di disordine" di un sistema. Un aumento del disordine corrisponde ad un aumento dell'entropia, al contrario, una diminuzione del disordine è associata ad una diminuzione dell'entropia. Il concetto di entropia potrebbe essere esteso ad aree di applicazione lontane dalla fisica, in primo luogo la teoria dell'informazione.

Negli anni '40, Shannon riuscì a trovare l'equazione con cui calcolare il livello di imprevedibilità di una fonte di informazione, notando che la formula che era riuscito a trovare, era identica a quella con cui Boltzmann aveva calcolato l'entropia di un sistema termodinamico. Il problema, per Shannon, era quello di misurare quanta informazione contenesse un determinato messaggio e di conseguenza, quanto costa inviarlo, dato un sistema di trasmissione e le difficoltà che il canale di trasmissione stesso (solitamente con del rumore) può trovare. La sua idea era quella di eguagliare il grado di ignoranza al disordine: il "messaggio" è la quantità di informazione che fa passare il destinatario da uno stato di incertezza ad uno stato di minor incertezza. L'entropia, dunque, misura la quantità di incertezza o di informazione presente in un segnale aleatorio, interpretato come il limite inferiore della compressione dei dati [19]. L'informazione relativa all'entropia, dunque, può essere espressa come:

$$I = -log_2 P$$

dove P rappresenta la probabilità che si verifichi un determinato evento, ottenendo un valore misurato in bit. La relazione di Boltzmann:

$$S = log_2 P$$

permette, anch'essa, di ottenere un valore espresso in bit, unità di misura dell'informazione. Possiamo concludere che vale la relazione:

$$I = -S$$

dove un aumento di entropia corrisponde ad una perdita di informazione su un determinato sistema e viceversa.

Definita la quantità di informazioni di un determinato evento, l'entropia è calcolata come l'informazione media per tutti i possibili eventi di una distribuzione:

$$E = -\sum Plog_2P$$

Il valore dell'entropia risulterà essere quindi un numero tra 0 e $+\infty$

bera

Listato 5.2: Estratto di Kb che rappresenta una coppia di oggetti con una bassa entropia.

```
"(`mouse', `tv')": {
      "below": 0.789,
      "side-down": 0.174,
      "side": 0.013,
      "hanging": 0.015,
      "inside": 0.002,
      "side-up": 0.003,
      "above": 0.003,
      "sup": 608,
      "entropy": 0.950
}
```

In questo lavoro, l'entropia è stata utilizzata per calcolare la distribuzione di probabilità degli istogrammi. Più il valore dell'entropia risulta basso, più la relazione tra la coppia di oggetti e la loro posizione può essere considerata significativa. Come si può vedere nel Listato 5.2, l'istogramma relativo alla coppia di oggetti *(mouse, tv)* ha un valore entropico sufficientemente basso, pari a 0.95.

Nel prossimo capitolo verranno mostrati i risultati ottenuti dall'architettura proposta analizzando la *Knowledge Base* estratta.

Capitolo 6

Validazione sperimentale

In questo capitolo verranno spiegate le motivazione che hanno permesso di scegliere il dataset utilizzato, fornendo i dettagli implementativi circa la creazione del *ground truth*. Successivamente verranno spiegate le tecniche di validazione del metodo e di conseguenza le motivazioni che hanno permesso la scelta del modello di classificazione utilizzato per la predizione delle posizioni per ogni coppia di *subject* e *reference*. Infine, verranno definite le scelte funzionali di estrazione dalla *Knowledge Base* in base alle analisi statistiche degli istogrammi calcolati.

6.1 Preparazione del dataset

In questo lavoro di tesi, è stato scelto di utilizzare il dataset di *Microsoft*, MS COCO, discusso nella sezione 3.7.1.

MS COCO, che contiene circa 200.000 immagini, è relativamente piccolo rispetto ad altri dataset ma ciò che lo contraddistingue sono le informazioni aggiuntive che contiene. Il dataset, infatti, invece di contenere solamente i dati che permettono la delimitazione degli oggetti tramite *bounding box*, fornisce informazioni sulla semantica per ogni immagine. Ogni istanza di oggetto, contenuta all'interno di ogni

immagine, è stata assegnata ad una specifica categoria seguendo un approccio gerarchico di etichettatura, in modo da poter essere verificata ed infine segmentata. Ad esempio un oggetto etichettato come "televisore" è contenuto all'interno di un insieme di oggetti catalogati con l'etichetta "elettronica". Grazie alla segmentazione degli oggetti, dunque, il classificatore sarà in grado di fornire delle prestazioni migliori, aumentando così l'accuratezza nella fase di addestramento rispetto all'utilizzo dei semplici delimitatori di oggetti (*bounding-box*).

MS COCO, contrariamente ad altri dataset molto popolari come ImageNet [64], PASCAL VOC [65] e SUN [66], contiene meno categorie di oggetti ma più istanze di oggetto per ogni categoria. Questo comporta di migliorare l'apprendimento dei vari algoritmi di classificazione e di conseguenza la localizzazione degli oggetti.

La Tabella 6.1 mostra il numero medio di categorie etichettate per immagine per ciascun dataset, mentre la Tabella 6.2 mostra il numero medio di istanze etichettate per immagine per ogni dataset.

Tabella 6.1: Numero di categorie etichettate per ogni immagine rispettivamente per i dataset MS COCO, ImageNet, PASCAL VOC, SUN

Dataset	*Categorie per immagine*
MS COCO	3.5
ImageNet	1.7
PASCAL VOC	1.4
SUN	9.8

Etichettatura dei dati

Dal *training set* del dataset MS COCO è stato selezionato un sottoinsieme di immagini, dove per ognuna di queste sono stati individuati e selezionati, in maniera casuale tramite la funzione *random.choice()* di Python, due oggetti qualsiasi:

Tabella 6.2: Numero medio di istanze di oggetto etichettate per immagine rispettivamente per i dataset MS COCO, ImageNet, PASCAL VOC, SUN

Dataset	*Istanze per immagine*
MS COCO	7.7
ImageNet	3.0
PASCAL VOC	2.3
SUN	17.0

l'oggetto *Subject* e l'oggetto *Reference* che sono stati contrassegnati con due colori diversi, il primo in blu, mentre il secondo evidenziato con il colore giallo (vedi Figura 6.1).

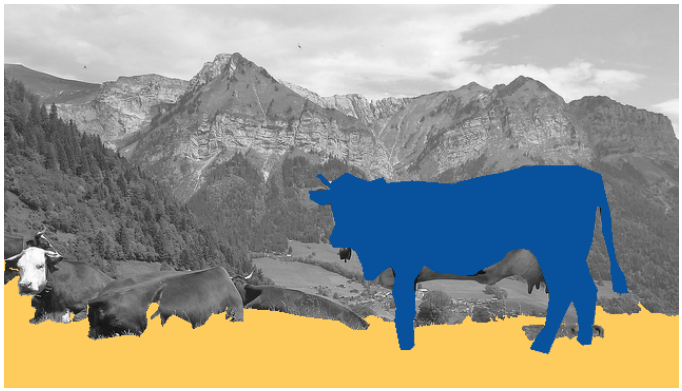

Figura 6.1: Immagine con *subject* (blu) e *reference* (giallo) etichettati come "on"

Dopo aver estratto i due oggetti, l'immagine è stata analizzata, considerando il contesto e la posizione tra i due elementi, in modo da etichettare manualmente la relazione che intercorre tra il *subject* ed il *reference*.

Come si vede dalla Figura 6.1, raffigurante una scena di pascolo in una zona montuosa, sono stati selezionati due oggetti in modo casuale, un bovino che rappresenta il *subject* mentre il prato sottostante che rappresenta l'oggetto a cui fa riferimento. Analizzando questa immagine e considerando le categorie di posizioni viste nel paragrafo 5.1.1, i due oggetti si presentano in una posizione adiacente

tra loro condividendo parte delle coordinate orizzontali, per questo motivo è stato scelto di assegnargli l'etichetta "*ON*".

Per eseguire l'operazione di etichettatura, è stato sviluppato un *Tool* (vedi Figura 6.2) che permette di assegnare le etichette in maniera semplice e senza causare errori. La finestra sviluppata presenta, sulla parte sinistra, l'immagine con gli og-

Figura 6.2: Esempio di GUI per l'operazione di etichettatura delle immagini

getti evidenziati, mentre sulla parte destra viene presentato un menu a tendina, contenente tutte le possibili classi che possono assere assegnate per ogni immagine. Una volta scelta l'etichetta, per l'immagine mostrata, è possibile confermare tramite il bottone "Conferma" la scelta effettuata in modo da passare all'immagine successiva se presente, altrimenti il processo viene terminato.

Creazione del *Ground truth*

L'etichetta assegnata con il metodo appena descritto, viene memorizzata all'interno di un file *CSV* (vedi Figura 6.3) contenente quattro colonne: l'identificativo dell'immagine (*Image ID*), gli identificativi dei due oggetti (*Subject ID* e *Reference ID*) e l'etichetta relativa alla loro posizione relativa (*Position*), assegnata tramite il *tool* appena descritto.

Image ID	Subject ID	Reference ID	Position
78	6377340	6380166	**hanging**
81	6577491	13352369	**inside**
86	7697781	15132390	**side**
89	2964820	4670280	**side-up**
92	1979989	2645885	**hanging**
94	6449259	15659505	**hanging**
109	1658173	4738898	**above**
110	8034733	12111822	**below**
208	5203807	9346959	**side**
241	4342857	8025782	**side**
247	6061717	12426363	**below**
250	5525068	11701877	**below**
257	4024963	6581116	**on**
260	2436658	3224631	**side**
263	5267530	11447475	**on**
283	3223599	4605509	**below**
294	2898501	3033192	**below**
307	5072984	11844011	**below**
308	7312259	10064924	**side**
309	1656640	2918699	**side**
312	6589368	7502716	**inside**
315	5069938	5529460	**hanging**
321	6193541	6846341	**hanging**
322	926994	3418660	**around**

Figura 6.3: Estratto del file contenente le informazioni relative agli oggetti per i quali è stata assegnata un etichetta manuale

Per ogni immagine etichettata sono state estratte le feature corrispondenti, secondo le tecniche viste nel capitolo 5.1.2. Tali informazioni sono state memorizzate all'interno di un ulteriore file *CSV* (Figura 6.4). Il file contiene, anche in questo caso, l'identificativo dell'immagine (*Image ID*), gli identificativi dei due oggetti (*Subject ID* e *Reference ID*), le feature estratte relative alla posizione (*i on j*, *j on i*, *i above j*, *j above i*, *i around j*, *j around i*) e le feature relative alla distanza tra i *bounding-box* degli oggetti in esame (*deltaY1*, *deltaY2*, *deltaX1* e *deltaX2*).

Nel prossimo capitolo vedremo come i file ottenuti, contenenti il *ground truth*, verranno utilizzati per addestrare l'algoritmo di classificazione in modo da estrarre i pattern frequenti di posizione.

Image ID	Subject ID	Reference ID	i on j	j on i	i above j	j above i	i around j	j around i	deltaY1	deltaY2	deltaX1	deltaX2
9	2005197	2271460	0.0	0.0	0.0	0.094771	0.0	0.0	0.2166	-0.90625	-0.81718	-0.2296
25	4162714	14932431	0.0	0.0	0.0	0.37812	0.0	0.0	0.63145	-1.0	-1.0	-0.37812
30	7045770	11777973	0.1294	0.0	0.023529	0.0	0.29411	0.0	-0.7476	-0.4649	-0.31093	-0.34687
34	2589299	5069153	0.0	0.0	0.0	0.0	0.53906	0.0	-0.93882	-0.95294	-0.69062	-0.998
36	4018997	9275200	0.48024	0.0	0.04573	0.0	0.0	0.0	-0.17968	-0.0968	-1.0	-1.0
42	7896231	12633045	0.0	0.2766	0.0	0.33141	0.0	0.0	-0.18619	-0.59623	-0.66406	-0.8781
49	4540741	14079445	0.0	0.0	0.0	0.79734	0.0	0.0	0.167999	-0.756	-0.79002	-1.0

Figura 6.4: Feature di alcune immagini estratte dal training set

6.2 Classificazione della posizione tra oggetti

Una volta definito il *ground truth*, rappresentato dai due file *CSV* appena visti, è possibile addestrare il classificatore in modo da ottenere il modello finale.

Prima di effettuare tale operazione, per ottenere delle prestazioni migliori, è stato utile bilanciare il dataset (*ground truth*), in modo addestrare il classificatore con lo stesso numero di oggetti per classe. Infatti, in caso di dataset sbilanciato il processo di apprendimento può essere compromesso perché mira a focalizzare la propria attenzione sulla classe dominante ed ignorare le altre. Come si vede dalla Figura 6.5 sono stati selezionati circa 60 *sample* per ogni classe di categoria, in modo da avere un dataset sufficientemente grande.

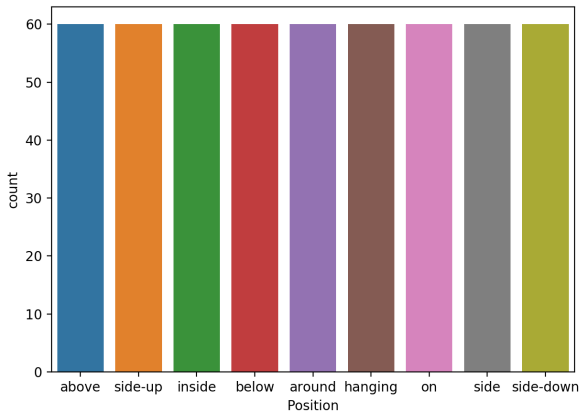

Figura 6.5: Istogramma che mostra il numero di *sample* per ogni classe di posizione

In questo lavoro di tesi sono stati analizzati diversi algoritmi di classificazione in modo da valutare, tramite le metriche che abbiamo visto nel capitolo 3.4, quale algoritmo, tra quelli contemplati nel capitolo precedente, restituisce una prestazione migliore per il calcolo dei pattern di posizione.

Le performance dei vari modelli, solitamente, vengono valutate utilizzando un insieme diverso di dati (test set) rispetto a quello utilizzato per l'addestramento del modello; in questo modo la misura delle prestazioni risulta priva di *bias* perché eseguita su dati non utilizzati in fase di *training*. In questo lavoro è stata utilizzata la tecnica della **Cross Validation** che permette di suddividere il dataset (nel nostro caso il *ground truth*) in k partizioni in modo da costruire il *test set* iterativamente con i dati a disposizione. Nello specifico è stato utilizzato un **Leave One Out** (LOO) che permette di addestrare il modello utilizzando tutti i *sample* del dataset tranne uno, che viene utilizzato come campione per il *test set*. Questo processo viene ripetuto per un numero di volte pari al numero degli elementi del dataset di addestramento. La procedura di validazione appena descritta non spreca molti dati perché viene escluso un solo campione alla volta, ma ha lo svantaggio di essere molto costoso in termini di tempo, per questo motivo viene utilizzato per dataset non molto grandi.

Per ogni classificatore è stata calcolata la matrice di confusione in modo da valutare le prestazioni dell'algoritmo in esame. Per interpretare la bontà delle matrici e quindi capire quale tra esse restituisce delle performance soddisfacenti, sono stati calcolati i due indici che permettono tale interpretazione: *Precision* e *Recall*. Il calcolo dell'*F1 Score*, che restituisce la media armonica degli indici appena menzionati, permette di decodificare al meglio i risultati ottenuti.

La Tabella 6.3 riporta i risultati ottenuti dalle valutazioni dei classificatori analizzati per ogni tipo di classe. La voce *Macro-average* rappresenta la media calcolata tra tutti i valori *F1 Score* per ogni categoria, separatamente per ogni algoritmo di classificazione. Come si può notare il *Random Forest* è l'algoritmo che ha ottenuto

Tabella 6.3: Valore medio F1 Score per ogni algoritmo di classificazione

Classe	*KNN*	*RBF SVM*	*Decision Tree*	*Random Forest*
Above	*0.89*	*0.89*	*0.92*	*0.93*
Around	*0.93*	*0.94*	*0.95*	*0.96*
Below	*0.95*	*0.92*	*0.94*	*0.98*
Hanging	*0.90*	*0.84*	*0.91*	*0.92*
Inside	*0.92*	*0.94*	*0.93*	*0.95*
On	*0.78*	*0.82*	*0.86*	*0.93*
Side	*0.82*	*0.75*	*0.78*	*0.85*
Side-down	*0.89*	*0.88*	*0.92*	*0.94*
Side-up	*0.92*	*0.88*	*0.92*	*0.88*
Macro-average	*0.89*	*0.87*	*0.89*	***0.93***

il valore medio di F1 più alto rispetto agli altri con un'ottima matrice di confusione (vedi Figura 6.6). Per questo motivo è stato scelto come modello finale di per predire i pattern di posizione sulla totalità delle immagini del training set come spiegato nel paragrafo 5.2.1.

Ottenuto il modello di classificazione finale è stato possibile proseguire con la generazione della KB contenente gli istogrammi per ogni coppia di oggetti di tutte le immagini contenute nel training set.

Nel prossimo paragrafo verrà discusso e analizzato il contenuto della KB costruita con informazioni ricavate fino ad ora.

6.3 Estrazione della Knowledge Base

Dopo aver generato la KB, è stato possibile analizzarne il contenuto tramite l'utilizzo di due grafici (Figura 6.7 e Figura 6.8), i quali rappresentano rispettivamente i valori di supporto ed i valori di entropia di tutti gli 8000 istogrammi nella

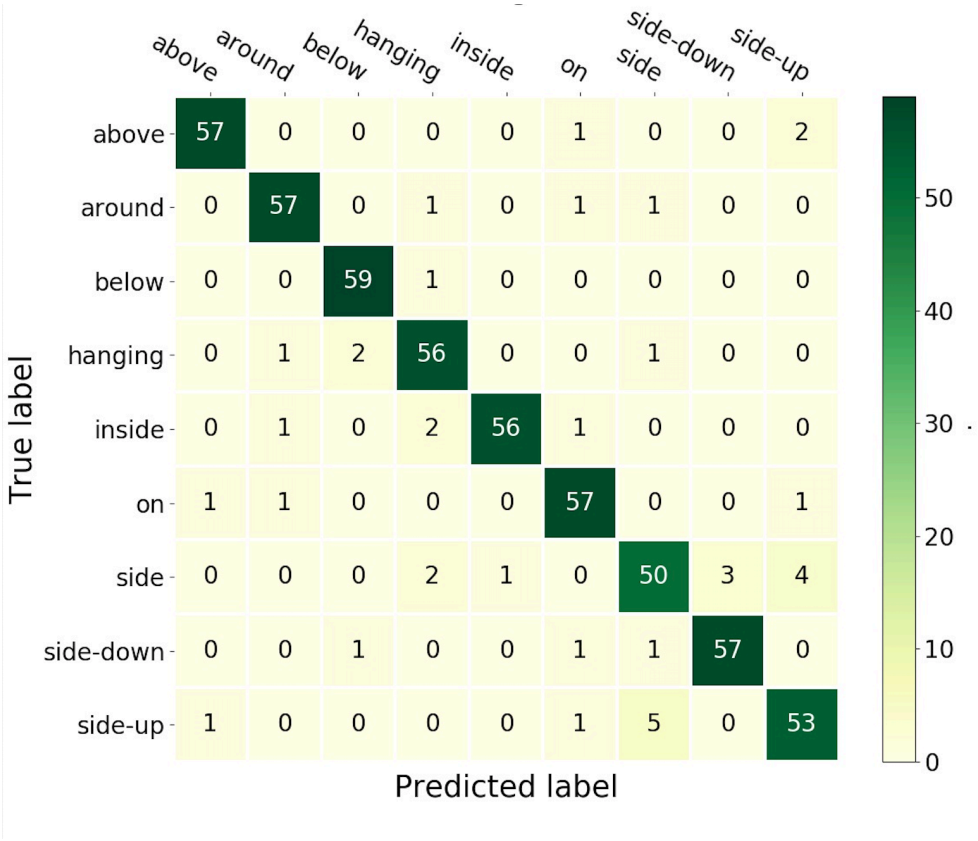

Figura 6.6: Matrice di confusione del Random Forest

Knowledge Base. Oltre a questi è stato realizzato uno *scatterplot* che permette di raffigurare la relazione tra supporto ed entropia (Figura 6.9).

Analizzando i valori del supporto (Figura 6.7) è possibile notare come solo 2000 degli 8000 istogrammi abbiano un basso supporto (< 10), quasi mille istogrammi hanno supporto fra 10^3 e 10^5.

Analizzando la distribuzione dell'entropia (Figura 6.8) si possono notare dei cambi di densità per i valori: 1.0 e 1.6.

Ciò è confermato dallo *scatterplot* in Figura 6.9 si può notare una nuvola molto densa per valori di entropia approssimativamente maggiori di 1.6 che individuano informazioni poco significative perché, come abbiamo visto nel capitolo 5.2.1, un valore di entropia alto indica un istogramma che contiene molto rumore. È possibile

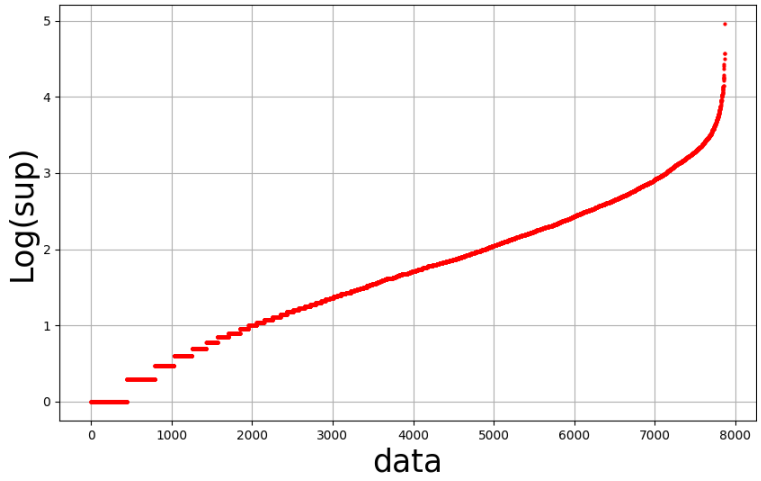

Figura 6.7: Distribuzione dei valori di supporto in scala logaritmica in base 10

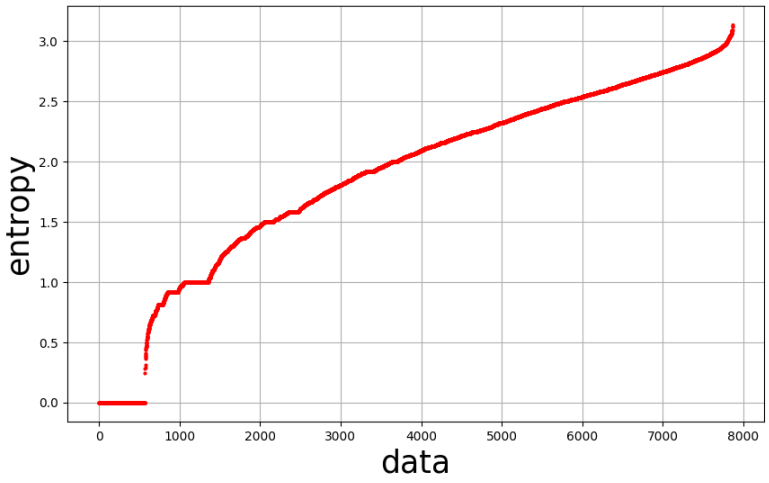

Figura 6.8: Distribuzione dei valori di entropia

notare l'altro cambiamento della densità anche per i valori di entropia compresi nell'intervallo tra 1.0 e 1.6 rispetto all'area compresa tra 0.0 e 1.0. Si può notare inoltre la presenza di alcuni istogrammi con entropia pari a 0 che però hanno un basso supporto e quindi risultano essere poco significativi.

Alla luce di quanto evidenziato si è scelto di eliminare dalla *Knowledge Base*

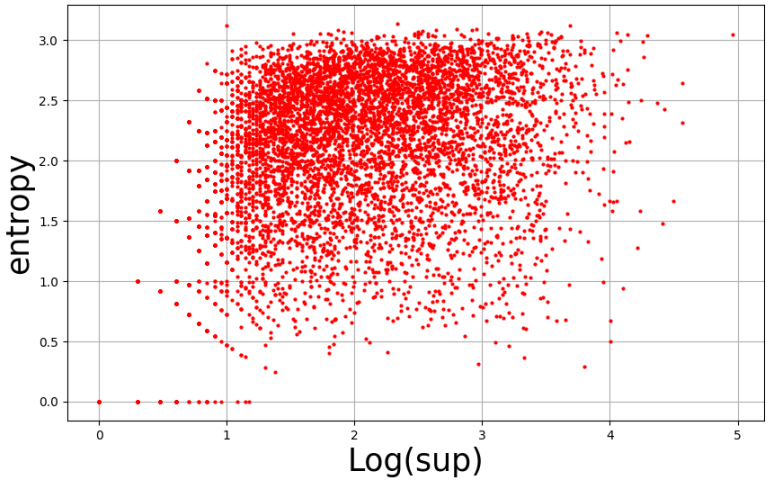

Figura 6.9: Relazione tra i valori di supporto ed entropia

i dati poco rilevanti, ovvero tutte quelle informazioni che non aiutano il sistema ad ottenere dei risultati ottimali. Per far ciò e stato trovato un compromesso per stabilire il valore massimo di entropia ed il valore minimo di supporto da considerare per eliminare gli istogrammi che non forniscono alcuna informazione utile per gli eventuali utilizzi della KB. È stato applicato il seguente filtro:

$$entropia \leq 1 \wedge supporto > 50$$

dove dalla KB vengono esclusi tutti gli istogrammi poco significativi con un entropia minore di 1 ed un supporto maggiore di 50. Gli istogramma così ottenuti avranno un indice di entropia abbastanza basso con dei supporti significativi.

La Tabella 6.4 mostra qualche esempio di coppie di oggetti con la relativa posizione restituiti dalla KB dopo aver eliminato gli istogrammi poco significativi. La prima colonna indica la coppia di oggetti (*subject* e *reference*), la seconda e la terza colonna mostrano rispettivamente i valori del supporto e quelli di entropia, mentre la quarta colonna mostra le probabilità più significative delle posizioni. Per esempio, la prima riga della colonna specifica che la strada si trova al di sotto del

cielo con una probabilità del 92%, la seconda riga descrive l'87% dei casi l'orologio si trova sopra al pavimento.

Tabella 6.4: Esempi di posizione tra varie coppie di oggetti

Coppia di Oggetti	*Supporto*	*Entropia*	*Istogramma*
strada, cielo	*10145*	*0.50*	*below=0.92, side-down=0.06*
orologio, pavimento	*596*	*0.72*	*above=0.87, side-up=0.10*
soffitto, pavimento	*3809*	*0.67*	*above=0.85, side-up=0.13*
montagna, cielo	*6257*	*0.29*	*hanging=0.96, below=0.03*
erba, cielo	*12692*	*0.94*	*below=0.81, hanging=0.11*
neve, albero	*3041*	*0.99*	*hanging=0.83, below=0.10*
bottiglia, soffitto	*1609*	*0.74*	*below=0.87, side-down=0.08*
strada, cielo	*12692*	*0.94*	*below=0.92, side-down=0.06*
mouse, tv	*608*	*0.95*	*below=0.79, side-down=0.18*
auto, cielo	*8790*	*0.99*	*below=0.80, hanging=0.08*
toaster, piastrelle	*62*	*0.94*	*hanging=0.84, side=0.05*
cellulare, finestra	*81*	*0.85*	*below=0.85, inside=0.07*
mela, soffitto	*200*	*0.67*	*below=0.88, side-down=0.10*
coltello, cielo	*89*	*0.58*	*below=0.86, side-down=0.14*

Come si vede, le relazioni trovate rispecchiano le aspettative e ciò implica che il processo di estrazione della Knowledge Base è in grado di reperire una descrizione fedele della realtà.

L'istogramma in Figura 6.10 mostra la percentuale delle posizioni relative agli istogrammi più rilevanti di tutte le triplette contenute all'interno della KB finale. Come si può notare le classi *above*, *hanging* e *below* sono quelle che tra tutte le

coppie di oggetti hanno una frequenza maggiore.

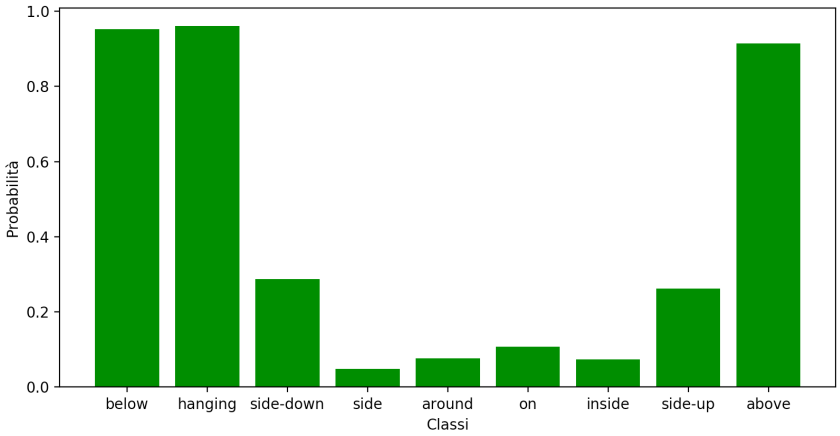

Figura 6.10: Istogramma che mostra la frequenza delle diverse classi all'interno della *Knowledge Base* finale

Capitolo 7

Conclusioni

Lo scopo di questa tesi era quello di analizzare e sviluppare una nuova architettura per l'estrazione delle informazioni dalle immagini, in modo da specificare le relazioni che legano le diverse coppie di oggetti contenute in esse.

La soluzione proposta in questa tesi si basa sullo sviluppo di un'architettura che può essere suddivisa in due macro-sezioni: l'addestramento del classificatore nel riconoscere la posizione relativa di due oggetti e la successiva generazione della Knowledge Base. Per addestrare il classificatore è stato costruito un dataset di *ground truth* ad hoc contenente sia le feature di alcune delle immagini segmentate provenienti dal training set del dataset MS COCO, sia le coppie di oggetti contenute all'interno delle suddette immagini, la cui posizione relativa è stata etichettata manualmente attraverso l'uso di un tool appositamente realizzato. Per il rilevamento e la rappresentazione delle posizioni relative sono stati utilizzati un insieme di valori discreti, così da avere una migliore rappresentazione semantica relativa all'effettiva forma dell'oggetto in esame. Successivamente, applicando l'algoritmo di classificazione sulla totalità del training set, è stato possibile generare la Knowledge base contenente le triplette e gli istogrammi relativi a tutte le coppie di oggetti presenti in tutte le immagini. I risultati così ottenuti sono stati valutati tramite la produzione di grafici basati su analisi statistiche, da cui sono state estratte informazioni utili

alla creazione di regole di assegnazione semantica valide per gli oggetti analizzati.

7.1 Sviluppi futuri

Uno dei possibili sviluppi futuri potrebbe riguardare l'utilizzo della Knowledge Base per validare e supportare i risultati dei metodi di deep learning applicati alla segmentazione semantica degli oggetti. Si potrebbe pensare di verificare se gli oggetti con una bassa *likelihood* possono essere considerati dei potenziali errori (Falsi Positivi) derivanti da una errata classificazione della rete neurale.

Bibliografia

[1] Caso Alessandro, *Deep learning per l'analisi dei dati*

[2] Vitrano Arianna, *Deep Learning per Image Analysis*

[3] Gaurav Kumar, Pradeep Kumar Bhatia, *A Detailed Review of Feature Extraction in Image Processing Systems*, 2014 Fourth international conference on advanced computing & communication technologies. IEEE, 2014.

[4] Maisto Assunta, *Feature Extraction tramite Deep Neural Network*

[5] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. *Deep learning*. nature, 2015, 521.7553: 436-444.

[6] Piccolo Elio & Leonardo Giannantoni, *Classificazione di dati biologici attraverso l'uso di molecole di ncRNA*, 2016

[7] F. Giannessi, *Teoremi di separazione e problemi di classificazione*

[8] Weinberger, Kilian Q., John Blitzer, and Lawrence K. Saul, Distance metric learning for large margin nearest neighbor classification. In: *Advances in neural information processing systems*. 2006. p. 1473-1480.

[9] Gandin Silvia, *Object Detection e Visual Servoing per applicazioni robotiche di grasping e manipolazione*, 2017

[10] He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*, 2017

[11] Garza, P., & Amara, P. *Mask R-CNN per la segmentazione di oggetti destinati alla vendita al dettaglio*

[12] Mauriello, M. *Classificazione del traffico di applicazioni mobili*

[13] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. *Panoptic segmentation* In Proceedings of the IEEE conference on computer vision and pattern recognition. 2019. p. 9404-9413.

[14] Lin, Maire, Belongie, Bourdev, Girshick, Hays, Perona, Ramanan, Zitnick, Dollar *Microsoft COCO: Common Objects in Context*

[15] Zhou, Zhao, Puig, Fidler, Barriuso, Torralba *Scene Parsing through ADE20K Dataset*, University of Toronto, Canada

[16] Torresin, L. (2019) *Sviluppo ed applicazione di reti neurali per segmentazione semantica a supporto della navigazione di rover marziani*

[17] Strat, T. M. *Employing contextual information in computer vision*, (1993), DARPA93, 217-229.

[18] Galleguillos, C., Rabinovich, A., & Belongie, S.*Object categorization using co-occurrence, location and appearance*, In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). IEEE.

[19] DIGIORGIO, Pasquale. *Location learning and prediction in Social Networks*, 2019. PhD Thesis. Politecnico di Torino.

[20] LEE, Yunjin, et al. *Overfitting control for surface reconstruction* In: Symposium on Geometry Processing. 2006. p. 231-234.

[21] MINAEE, Shervin, et al. *Image Segmentation Using Deep Learning: A Survey.* arXiv preprint arXiv:2001.05566, 2020.

[22] AHMED, Mohiuddin; MAHMOOD, Abdun Naser; HU, Jiankun. *A survey of network anomaly detection techniques* Journal of Network and Computer Applications, 2016, 60: 19-31.

[23] LEONDES, Cornelius T. (ed.). *Intelligent Systems: Technology and Applications*, Six Volume Set. CRC Press, 2018.

[24] PASINI A., BARALIS E. *Detecting Anomalies in Image Classification by means of Semantic Relationships.*

[25] SEGRETO T. *Modelli e Tecniche Computazionali Intelligenti nei Processi di Lavorazione.*

[26] Gareth James et al.*An Introduction to Statistical Learning* With Applications in R. Springer Publishing Company, Incorporated, 2014

[27] LAURA, Luigi; FRANCESCO, Giuseppe; DE ROSA, Pasquale.*Reti neurali e collaborative filtering: un approccio avanzato al marketing one-to-one*

[28] LECUN, Yann, et al.*Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998, 86.11: 2278-2324.

[29] NIELSEN, Michael A.*Neural networks and deep learning.* San Francisco, CA, USA: Determination press, 2015.

[30] KHAN, Asifullah, et al.*A survey of the recent architectures of deep convolutional neural networks.* arXiv preprint arXiv:1901.06032, 2019.

[31] Simonyan, K., Zisserman, A.*Imagenet classification with deep convolutional neural networks.* In: NIPS (2012)

[32] Krizhevsky, A., Sutskever, I., Hinton, G.E.*Very deep convolutional networks for large-scale image recognition.* ICLR (2015)

[33] Y. LeCun et al., *Learning algorithms for classification: A comparison on handwritten digit recognition,‖ Neural networks Stat.* Mech. Perspect., vol. 261, p. 276, 1995.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov *A Simple Way to Prevent Neural Networks from Overfitting* J. Mach. Learn. Res., vol. 1, no. 60, p. 11, 2014.

[35] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks* arXiv Prepr. arXiv1311.2901v3, vol. 30, no. 1–2, pp. 225–231, 2013

[36] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition Multimed.* Tools Appl., vol. 77, no. 9, pp. 10437–10453, Dec. 2015

[37] K. Simonyan and A. Zisserman, *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION* ICLR, vol. 75, no. 6,

pp. 398–406, 2015.

[38] Ross B. Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation* In: CoRR abs/1311.2524 (2013). arXiv: 1311.2524. url: http://arxiv.org/abs/1311.2524.

[39] Olga Russakovsky et al.*ImageNet Large Scale Visual* In: Int. J. Comput. Vision 115.3 (dic. 2015), pp. 211–252. issn: 0920-5691. doi: 10.1007/s11263-015-0816-y. url: http://dx.doi.org/10.1007/s11263- 015-0816-y.

[40] Ross B. Girshick. *Fast R-CNN.* In: CoRR abs/1504.08083 (2015). arXiv: 1504.08083. url: http://arxiv.org/abs/1504.08083.

[41] J.R.R. Uijlings et al. *Selective Search for Object Recognition.* In: International Journal of Computer Vision (2013). doi: 10.1007/s11263-013-0620-5

[42] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* In: CoRR abs/1506.01497 (2015). arXiv: 1506.01497.

[43] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection* (2016)

[44] BRILLI, GIANLUCA. *Implementazione e Testing di Reti Neurali Convoluzionali su Architetture Embedded per Automotive.* 2018.

[45] Kaiming He et al. *Mask R-CNN.* In: CoRR abs/1703.06870 (2017). arXiv: 1703.

[46] Farhana Sultana,Abu Sufiana Paramartha Duttab *Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey* (2020)

[47] ZORZAN, Riccardo. *Sviluppo di un algoritmo basato su tecniche di deep learning per la segmentazione di cellule dendritiche in immagini della cornea.* (2020).

[48] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, *Pyramid scene parsing network* CoRR abs/1612.01105. arXiv:1612.01105.

[49] Jeremy Jordan *Evaluating image segmentation models* (2018) url: https://www.jeremyjordan.me/evaluating-image-segmentation-models

[50] Ekin Tiu *Metrics to Evaluate your Semantic Segmentation Model* (2019) url: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

[51] ZHANG, Wei Emma; SHENG, Quan Z. *Managing Data From Knowledge Bases: Querying and Extraction.* Springer International Publishing, 2018.

[52] LEHMANN, Jens, et al. *DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia.* Semantic Web, 2015, 6.2: 167-195.

[53] KASNECI, Gjergji; SUCHANEK, Fabian; WEIKUM, Gerhard. *Yago-a core of semantic knowledge* 2006.

[54] GENE ONTOLOGY CONSORTIUM. *The gene ontology project in 2008. Nucleic acids research*, 2008, 36.suppl-1: D440-D444.

[55] MILLER, George A., et al. *Introduction to WordNet: An on-line lexical database.* International journal of lexicography, 1990, 3.4: 235-244.

[56] HAYES, Jonathan; GUTIERREZ, Claudio. *Bipartite graphs as intermediate model for RDF.* In: International Semantic Web Conference. Springer, Berlin, Heidelberg, 2004. p. 47-61.

[57] SIGNORE, Oreste. *RDF per la rappresentazione della conoscenza.* In: Knowledge Management (V edizione)–Forum Proceedings on the knowledge management in the organizations. 2002. p. 35-46.

[58] Lars Hulstaert *A Beginner's Guide to Object Detection* (2018)

[59] Sari, Arif. *A Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications. Journal of Information Security.* (2015).6. 142-154. 10.4236/jis.2015.62015.

[60] J.Han, M.Kamber, J.Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc., 2005, ISBN: 1558609016

[61] GARZA, Paolo; TEDESCO, Alessandro. *Anomaly Detection sui Sistemi Monitorati dall'Oracle Enterprise Manager.*

[62] GALLEGUILLOS, Carolina; BELONGIE, Serge. *Context based object categorization: A critical survey. Computer vision and image understanding* 2010, 114.6: 712-722.

[63] BIEDERMAN, Irving. *Perceiving real-world scenes.* Science, 1972, 177.4043: 77-80.

[64] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*in CVPR,2009.

[65] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL visual object classes (VOC) challenge* IJCV, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[66] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, *SUN database: Large-scale scene recognition from abbey to zoo* in CVPR, 2010

[67] ZHANG, Quanshi; NIAN WU, Ying; ZHU, Song-Chun. *Interpretable convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018. p. 8827-8836.

[68] Galleguillos, C., Rabinovich, A., Belongie, S. *Object categorization using co-occurrence, location and appearance.* In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). IEEE.