Università degli Studi di Firenze
Scuola di Ingegneria

Corso di Laurea in

Ingegneria Informatica Magistrale

Riconoscimento Visuale con Alberi di Decisione Rilassati su Tassonomie multiple

Relaxed Decision Trees over multiple Taxonomies for Visual Recognition

Tesi di Laurea di

Claudio Tortorici

5 Dicembre 2013

Relatori:

Prof. Alberto Del Bimbo
Prof. Marco Bertini

Correlatori:

Lamberto Ballan
Svebor Karaman

Anno Accademico 2012/2013

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La *Visione Computazionale* è quel ramo dell'informatica che si occupa dell'acquisizione, processazione, analisi e comprensione di immagini. Lo sviluppo in quest'area dell'informatica mira a riprodurre le abilità umane per le macchine, con particolare attenzione alla percezione e comprensione delle immagini [3].

Le tecniche di visione computazionale hanno molte applicazioni pratiche. Uno delle più importanti è in ambito medico, nel quale l'acquisizione e l'analisi di immagini hanno lo scopo di formulare diagnosi mediche per i pazienti. Un'altra area di applicazione della visione computazionale è quella industriale, nella quale lo scopo è quello di supportare, e in molti casi automatizzare completamente, il processo produttivo. Infine i campi militare ed aerospaziale sono le aree in cui, negli ultimi anni, la visione computazionale ha dato il maggior contributo, poiché l'utilizzo di robot più o meno indipendenti, consente operazioni proibitive per l'uomo.

Ciascuna di queste applicazioni svolge un insieme di compiti. I più importanti sono:

Motion Analysis Studia sequenze di immagini per stimare il moto di una scena, sia in casi di camera fissa e scena in movimento, sia per scene più o meno fisse riprese da una camera mobile. Alcuni esempi di motion analysis sono

- Egomotion, che determina il movimenti di roto-traslazione della camera, analizzando sequenze di immagini riprese dalla camera stessa.
- Tracking, che mira a seguire il movimento di piccoli insiemi di interesse o oggetti (ad esempio persone o veicoli), all'interno di un'immagine.
- Optical Flow, che determina per ciascun punto nell'immagine, come questo si muove sul piano dell'immagine (moto apparente).

Scene Reconstruction Dato un'insieme di immagini, o un video, questa tecnica mira a generare un modello 3D della scena.

Image Restoration Si occupa del problema della rimozione del "rumore" dall'immagine.

Recognition È il più classico dei problemi della visione computazionale. Si possono evidenziare tre varianti di questo problema:

1. Object recognition
2. Identification
3. Detection

Questi problemi possono essere facilmente risolti da un computer per situazioni non complesse, ovvero il riconoscimento di figure geometriche, identificazioni di volti umani o di impronte digitali, ma ancora non siamo in gradi di risolvere il problema nel caso più generale, ovvero il riconoscimento di oggetti arbitrari in situazioni arbitrarie.

La classificazione di immagini e scene è un'abilità fondamentale dell'uomo ed un importante obbiettivo nella ricerca in *computer vision*.

I recenti progressi sulla classificazione di immagini sono stati impressionanti, e hanno prodotto un vasto numero tra features, modelli, classificatori e frameworks. Ma cosa succede se non ci limitiamo solo a piccole arie di ricerca, in cui il numero di classi è limitato, ma si cerca di estendere la ricerca ad ordini di grandezza maggiore?
La classificazione su larga scala è una delle sfide della computer vision. Per problemi su larga scala si intende sia l'elevata dimensione dello spazio fisico delle immagini, sia l'ampiezza dello spazio semantico che l'uomo usa per descrivere gli stimoli visivi. In particolare, gli psicologhi hanno postulato che l'essere umano è in grado di riconoscere decine di migliaia tra oggetti e scene [4], ed è proprio su questi ordini di grandezza che il divario tra uomo e macchina diventa evidente.
Studi empirici provano che sfruttare algoritmi e tecniche utilizzate per l'identificazione di un ristretto numero di categorie su larga scala, porta ad un notevole deterioramento delle prestazioni, sia computazionali che di precisione. Questo perdita di efficienza ed efficacia è dovuto all'introduzione di criticità all'aumentare del numero di categorie da riconoscere. Aumentare la varietà tra le categorie, implica un notevole aumento di classificatori da addestrare, inoltre l'incremento di densità tra le classi rende la loro distinzione più complicata. È infine da non sottovalutare il costo computazionale di queste operazioni se si opera realmente su larga scala, ovvero per grandi dataset si necessita di un enorme quantitativo di memoria.
L'evoluzione della classificazione si sviluppa quindi su tecniche più avanzate, che sfruttano strutture complesse per ottimizzarne la velocità e la precisione degli algoritmi.

In questo lavoro si presenta una tecnica per la classificazione di immagini

che prevede l'utilizzo di più strutture ad albero generate in modo non supervisionato a partire da caratteristiche visive differenti, unite insieme in una unica "Foresta"[1]. Inoltre presentiamo un nuovo modello per l'esplorazione degli alberi binari di ricerca, chiamato *Relaxed Decision Tree*, che introduce una forma di *soft backtracking*, al fine di migliorare i risultati della foresta di tassonomie.

[1]Una foresta risulta costituita da una unione di alberi (da cui deriva il nome) disgiunti tra loro. Questi alberi costituiscono le sue componenti connesse massimali.

Parte I

Stato dell'Arte

Capitolo 1

Classificazione di Immagini

In questo capitolo si presentano alcune delle tecniche di classificazione di immagini presenti in letteratura, focalizzando l'attenzione su quelle che sono state utilizzate o sono state fonti di ispirazione per il lavoro presentato.

La classificazione è il problema di identificare a quale insieme di categorie appartiene una nuova osservazione, sulle basi di conoscenze a priori fornite da un insieme di training per il quale sono conosciute le categorie di appartenenza. Le osservazioni individuali sono analizzate in un insieme di proprietà quantificabili, come variabili esplicative o features. Queste proprietà possono essere categoriche (come ad esempio il gruppo sanguigno A, B, AB...), ordinali, (i.e. grande, medio, piccolo), numeri interi (i.e. il numero di occorrenze di una parola in un documento), o numeri reali (i.e. un qualche tipo di misurazione di pressione, distanza etc...).
Molti algoritmi lavorano con quantità discrete di dati, pertanto spesso è necessario effettuare una discretizzazione dei valori ottenuti in gruppi.
Un algoritmo che implementa una classificazione, viene chiamato *classificatore*, e svolgerà il compito di assegnare una categoria secondo i dati forniti.

Nell'apprendimento automatico, la classificazione è considerata un'istanza dell'apprendimento supervisionato, ovvero quando è disponibile un insieme di training di osservazioni correttamente identificate in categorie. La procedura non supervisionata corrispettiva, è il *clustering*, il quale prevede il raggruppamento dei dati in categorie basandosi su determinate misure di similarità.
Nell'apprendimento automatico le osservazioni vengono anche chiamate *istanze*, le variabili esplicative che descrivono l'istanza vengono chiamate *feauters* (e raggruppare in vettori di features) ed infine le categorie che possono essere predette dal classificatore sono chiamate *classi*.
La classificazione può essere distinta in due problemi separati, la classificazione binaria e quella multi-classe. Nella classificazione binaria sono coinvolte solo due classi, mentre nella classificazione multi-classe consiste nell'assegnazione per un oggetto di una su un numero variabile di classi. Mentre per la classificazione binaria sono state sviluppate molte tecniche, per l'equivalente multi-classe si necessita di combinare un insieme di classificatori binari.

1.1 Descrittori di Features

I modelli *features based* sono, ad oggi, tra i modelli più popolari per la descrizione delle immagini. Essi hanno come scopo l'individuazione di parti rilevanti e fortemente caratterizzanti di un'immagine, che la descrivano in modo adeguato. Tali features vengono rappresentate come vettori numerici, fornendoci un adeguato strumento per operazioni di classificazione.
Quando si parla di features si deve distinguere tra il processo di *feature extraction*, che produce la feature stessa attraverso operazioni su intorni di punti applicate all'intera immagine, e quello di *feature detection*, che consente di

decidere quando una determinata feature è presente o meno in determinati punti dell'immagine.
Le features di un'immagine di possono classificare in due distinte categorie:

Globali che descrivono l'immagine nella sua interezza

Locali che descrivono, singolarmente, piccole porzioni dell'immagine

Le features globali mirano a catturare le caratteristiche visive, strutturali e le relazioni spaziali tra i componenti dell'immagine. Le features logali, invece, usano singoli punti o zone, per descrivere l'intera immagine. Il loro calcolo si effettua in due passaggi fondamentali: la ricerca delle caratteristiche salienti, e la loro descrizione. Per quanto riguarda la ricerca si possono utilizzare due tecniche di campionamento:

- Sparsa
- Densa

1.1.1 Modello Bag of Words

Attraverso tecniche di clustering[1] si può ottenere un modello di descrizione di alto livello di un oggetto, questa tecnica è denominata *Bag of Word*. Nel modello Bag of Word un "documento" viene descritto attraverso la frequenza delle occorrenze di un certo numero di "parole" al suo interno. Questo concetto si può facilmente estendere alle immagini utilizzando al posto delle parole, le features. Nella visione computazionale una *bag of visual words*, più comunemente chiamata *Bag of Features* (BoF), è un vettore sparso del numero di occorrenze di visual words appartenenti ad un vocabolario. Il vocabolario è un insieme di features locali di immagini.

[1] Insieme di tecniche di raggruppamento di elementi omogenei in un insieme di dati.

Rappresentare un'immagine tramite il modello Bag of Word significa trattare un'immagine come un documento di testo. Per ottenere questo risultato si devono effettuare tre operazioni:

1. Rilevazione delle features (*features detection*)
2. Features description (*features description*)
3. Generazione del dizionario di riferimento

Un dizionario è composto di *codeword* che possono essere considerate rappresentative per un insieme di parole, che compongono il documento, simili tra loro. Un metodo semplice per costruire il vocabolario è il clustering k-means[2]. Questa tecnica di quantizzazione consente di definire le codeword come i centri dei clusters costruiti sui documenti.
La rappresentazione tramite *bag of words*, o più correttamente *bag of features* è l'istogramma della distribuzione delle features dell'immagine rispetto alle *visual words* che compongono il dizionario.

Una volta in possesso della Bag of Features di ciascuna immagine, possiamo utilizzarla per addestrare un classificatore.

1.2 Classificatori

Un gran numero di algoritmi per la classificazione possono essere espressi come funzioni lineari che assegnano un punteggio a ciascuna delle possibili classi, combinando il vettore di features dell'istanza in esame con un vettore di pesi attraverso un semplice prodotto matriciale. La classe predetta sarà

[2]L'algoritmo K-Means è un algoritmo di clustering partizionale che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi.

quella che presenta il punteggio più alto.
Questo tipo di punteggio viene assegnato da una funzione che presenta la seguente forma generale:

$$\text{score}(X_i, k) = \beta_k \cdot X_i \tag{1.1}$$

dove X_i è il vettore di features dell'istanza i, β_k è il vettore di pesi corrispondente alla classe k, e $\text{score}(X_i, k)$ è il punteggio associato all'istanza i per la classe k.
I classificatori che presentano questa forma sono comunemente chiamati classificatori lineari. Un tipico esempio, comunemente usato, di classificatore lineare è la *Support Vector Machine*.

1.2.1 Support Vector Machines

Le Support Vector Machines, o *SVM*, sono strumenti di apprendimento supervisionato per la classificazione e la regressione di pattern[3]. La support vector machine è originariamente un metodo per la classificazione binaria sviluppata da Vladimir N. Vapnik [5].
Per un problema binario necessitiamo di un insieme di training composto da coppie $\{x_i, y_i\}$ con $i = 1, \ldots, l$, $y_i \in \{-1, 1\}$ e $x_i \in \mathbb{R}^d$. Supponiamo di avere un iperpiano che separa gli esempi positivi da quelli negativi. Il punto x che si trova sull'iperpiano soddisfa l'equazione

$$w \cdot x + b = 0 \tag{1.2}$$

dove w è normale all'iperpiano, $\frac{|b|}{\|w\|}$ è la distanza perpendicolare dall'iperpiano all'origine. Sia d_+ la distanza minore tra l'iperpiano e il più vicino

[3]Pattern è un termine inglese che può essere tradotto, a seconda del contesto, con disegno, modello, schema, schema ricorrente e, in generale, può essere utilizzato per indicare una regolarità che si riscontra all'interno di un insieme di oggetti osservati.

esempio positivo (rispettivamente d_- per il più vicino esempio negativo). Si possono a questo punto definire i *margini* dell'iperpiano come $d_+ + d_-$. Nel caso di una separazione lineare, SVM cerca l'iperpiano con i margini più grandi.
Questo può essere formulato come segue: sia l'insieme di training tale che soddisfi i vincoli

$$x_i \cdot w + b \geq +1 \qquad \text{con } y_i = +1 \tag{1.3}$$

$$x_i \cdot w + b \leq -1 \qquad \text{con } y_i = -1 \tag{1.4}$$

che può essere riassunto nella seguente equazione

$$y_i \left(x_i \cdot w + b\right) - 1 \geq 0 \qquad \forall i \tag{1.5}$$

Consideriamo ora un punto per il quale vale (1.3), questo equivale alla scelta di una scala per w e b, questo punto poggia sull'iperpiano $H_1 : x_i \cdot w + b = 1$ con normale w e distanza dall'origine $\frac{|1-b|}{\|w\|}$. Stessa cosa per (1.4), si trova l'iperpiano $H_2 : x_i \cdot w + b = -1$ con normale w e distanza dall'origine $\frac{|-1-b|}{\|w\|}$. Quindi otteniamo

$$d_+ = d_- = \frac{1}{\|w\|} \tag{1.6}$$

Si nota che gli iperpiani H_1 e H_2 sono paralleli e non contengono tra loro punti di training. Si possono quindi trovare coppie di iperpiani che forniscano i margini massimi minimizzando $\|w\|^2$ in (1.5) [6].

Anche se SVM è stato originariamente creato come classificatore binario, esistono approcci che risolvono il problema della classificazione con più classi, anche se questi sono computazionalmente più onerosi rispetto a risolvere

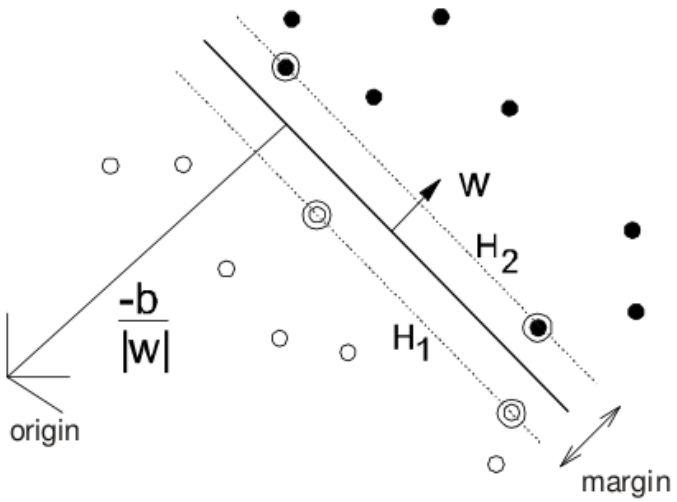


Figura 1.1: Iperpiano di separazione lineare, i vettori di supporto sono cerchiati.

problemi binari [7].
Ci sono varie tecniche per la decomposizione di problemi multi classe in più problemi binari, utilizzando SVM come classificatore binario.

1.2.2 One against All (OvA)

Per problemi di classificazione ad N classi, si costruiscono N classificatori binari. L'i-esimo SVM viene addestrato impostando l'etichetta della classe i-esima come positiva, ed il restante insieme di esempi con etichetta negativa. In fase di riconoscimento, si sottopone l'esempio di test a tutti gli N classificatori SVM e si etichetta con la classe il quale SVM presenta la risposta maggiormente positiva.
Lo svantaggio di questo metodo è la complessità nella fase di training, proporzionale alla grandezza del numero di esempi, poiché ciascuno degli N classificatori viene addestrato usando tutti i possibili esempi.

1.2.3 One against One (OvO)

A differenza della tecnica OvA (sottosezione 1.2.2), questo algoritmo prevede l'utilizzo di un SVM binario per ciascuna coppia di classi, si costruiscono quindi $\frac{N(N-1)}{2}$ classificatori. Ogni classificatore viene addestrato usando gli esempi della prima classe come positivi, e quelli della seconda classe come negativi. Per combinare i risultati di tutti questi classificatori si adotta la politica del massimo valore positivo, ovvero si seleziona la classe che riceve il "voto" massimo tra tutti gli $N(N-1)/2$ classificatori [6].

Il vantaggio rispetto a OvA, è che il numero di esempi utilizzati per il training di ciascun classificatore è piccolo, poiché soltanto due delle originali N classi vengono considerate per ciascun addestramento, portando così a tempi di addestramento notevolmente minori.

Lo svantaggio, invece, è che ciascun esempio di test deve essere confrontato con grande numero di classificatori $\left(\frac{N(N-1)}{2}\right)$. Questo elevato numero di confronti rallenta il test degli esempi al crescere del numero di classi N.

Capitolo 2

Classificatori Strutturati

L'evoluzione delle tecniche di classificazione ha portato non tanto al miglioramento delle singole tecniche esistenti, ma piuttosto all'utilizzo di suddette tecniche, associate a strutture che fornissero un supporto ai classificatori stessi. L'idea basilare è di utilizzare classificatori semplici su strutture più o meno complesse.

Grafi Diretti Aciclici di SVM (DAGSVM)

L'algoritmo per l'addestramento degli $\frac{N(N-1)}{2}$ SVM del grafo diretto aciclico (DAG), è uguale a quello utilizzato da OvO (sottosezione 1.2.3). In fase di riconoscimento la decisione dipende dal grafo. DAGSVM crea un modello per ciascuna coppia di classi, quando uno di questi modelli è in grado di distinguere tra la classe c_1 e la classe c_2, classificando l'esempio con la classe c_1, in realtà non "vota per" la classe c_1, ma piuttosto "vota contro" la classe c_2. Pertanto da questo punto in avanti del grafo, l'algoritmo ignorerà tutti i modelli di SVM che coinvolgono la classe c_2. Ciò significa che per ciascuna classificazione effettuata all'interno del grafo, si può escludere una classe dai possibili candidati, pertanto, dopo $N-1$ passi, rimarrà una sola classe can-

didata.

Questa tecnica presenta risultati prossimi alla modello One-against-One ma è notevolmente più veloce in fase di test.

2.1 SVM su Strutture ad Albero Binario (BTS)

Negli ultimi anni sempre di più sono state sviluppate tecniche per la classificazione che sfruttano strutture ad albero. Queste tecniche mirano o alla velocizzazione del processo di classificazione [8], diminuendo il numero di confronti con gli SVM, o all'aumentare l'accuratezza della classificazione stessa [6].

La Figura 2.1 mostra il funzionamento di questo tipo di tecnica. Per effettuare la classificazione si confronta, inizialmente, il campione con l'SVM presente sulla "radice" dell'albero, il risultato di questo SVM consente di scegliere in quale sotto-albero proseguire con la classificazione. Il processo continua fino al raggiungimento di una "foglia".

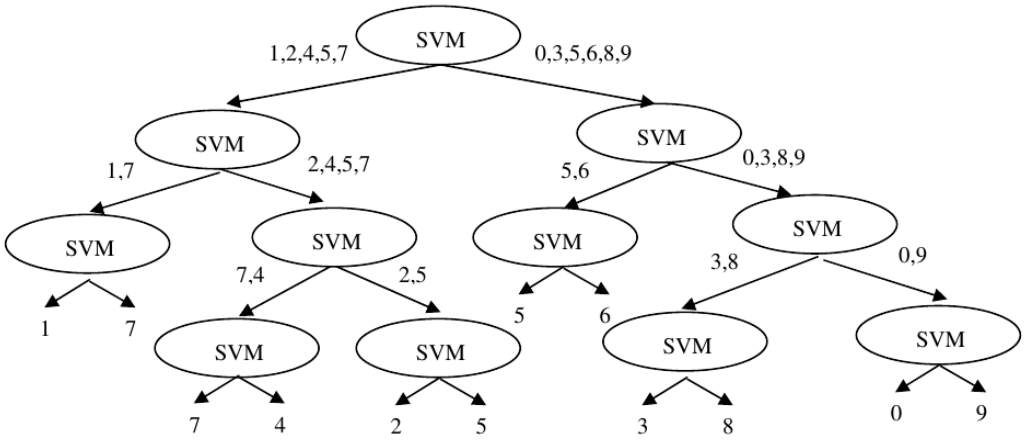


Figura 2.1: SVM con architettura ad albero binario per la classificazione di 10 classi.

Mentre con le tecniche *One-vs-All* e *One-vs-One* il numero di confronti con gli SVM sono rispettivamente N e $\frac{N(N-1)}{2}$, con la struttura ad albero binario si può scendere ad un numero di confronti pari a $\lceil \log_2 N \rceil$.
In questo modo si riesce a sfruttare i vantaggi sia di efficienza computazionale dell'architettura ad albero, sia l'alta accuratezza nella classificazione degli SVM.
Esistono molti modi per separare le N classi in due gruppi, consentendo così di costruire l'albero, ma raggruppare le classi in modo appropriato è di fondamentale importanza per ottenere buone prestazioni da questa tecnica.
In [6], propongono un metodo che suddivide ricorsivamente le classi in due gruppi disgiunti per ciascun nodo dell'albero, ed addestrando un SVM per decidere a quale gruppo assegnare il campione da analizzare.
In [9], invece, ciascun nodo presenta un classificatore SVM addestrato con due tra le classi appartenenti a quel nodo. Successivamente l'algoritmo implementa una misura probabilistica di similarità tra le restanti classi e le due utilizzate per il training di ciascun nodo. Ciascun nodo suddivide gli esempi in uno dei due sotto-nodi. Questo procedimento viene ripetuto fino ad ottenere una foglia contenente solamente gli esempi di una classe.
Il problema maggiore di questo metodo è che gli esempi di ciascuna classe devono essere testati su ciascun nodo per selezionare a quale sotto-nodo l'esempio appartiene. Questo procedimento, nella fase di costruzione dell'albero, può rallentare la fase di addestramento per dataset contenenti un numero elevato di classi.

L'utilizzo di strutture gerarchiche, a supporto di un sistema di SVM binari, diminuisce i costi computazionali e spesso migliora la precisione nella classificazione. Nonostante questo il divario uomo-macchina è ancora

eccessivo.

2.2 Evoluzione della Classificazione

Nella classificazione di un numero elevato di categorie, l'uso di tecniche One-vs-One (sottosezione 1.2.3) o One-vs-All (sottosezione 1.2.2) risulta essere poco efficiente, pertanto, sempre più spesso, si utilizzano strutture ausiliare (generalmente grafi) per migliorare le prestazioni.
L'uso di alberi binari di decisione, combinati con Support Vector Machines (come descritto in anche in sezione 2.1) portano già notevoli miglioramenti. Ma è possibile utilizzare strutture più complesse, o informazioni supplementari, per migliorare le tecniche di classificazione?
La ricerca negli ultimi anni ha sviluppato un considerevole numero di tecniche e strumenti con lo scopo di appiattire tale divario. Le nuove tecniche utilizzano approcci diversi, ciascuno con i propri pregi e difetti. Si va dall'introduzione di nuove tecniche per il calcolo delle features [10] [2] [11], l'introduzione di strutture ausiliarie (tassonomie, grafi, alberi o foreste) [1] [12] [13] [2], l'utilizzo di metriche "ad-hoc" per percorrere le strutture ausiliarie [14] [13] fino ad utilizzare informazioni di tipo semantico [14] [13] [2] per migliorare i risultati.

A seguire saranno presentati alcuni approcci che utilizzano le tecniche di classificazione viste in sezione 1.2.

2.2.1 Measuring Image Distances via Embedding in a Semantic Manifold

In "*Measuring Image Distances via Embedding in a Semantic Manifold*" [13] viene presentato un modello che utilizza un grafo ausiliario per generare relazioni semantiche tra immagini, a partire da un database composto da immagini ed etichette descrittive ad esse associate. Questo approccio prevede una fase offline, nella quale viene costruito un grafo che cattura le relazioni semantiche fornite dalle etichette associate alle immagini. In questa riorganizzazione del database sotto forma di grafo, i nodi rappresentano le immagini etichettate, mentre gli archi corrispondono a collegamenti tra immagini molto simili, sia dal punto di vista visivo che semantico. La relazione semantica tra immagini viene determinata comparando le annotazioni, mentre quelle visive confrontando le features. Successivamente è prevista una fase di test nella quale si può calcolare la distanza semantica[1] attraverso l'inserimento delle nuove immagini nel grafo. Questo fase calcola le distanze visive delle due nuove immagini separatamente, con le immagini all'interno del grafo, in questo modo è possibile posizionare le immagini di test nel grafo, consentendo così il calcolo della distanza semantica tra di esse.

Questa tecnica utilizza non solo caratteristiche visive, ma anche informazioni semantiche grazie all'introduzione di una nuova metrica. Percorrendo gli archi del grafo, questa metrica consente di sfruttare le correlazioni tra le immagini del grafo per sfruttare tecniche di classificazione semplici come la *Nearest-Neighbor*.

[1] Con distanza semantica si intende la misura di quanto due immagini siano semanticamente correlate.

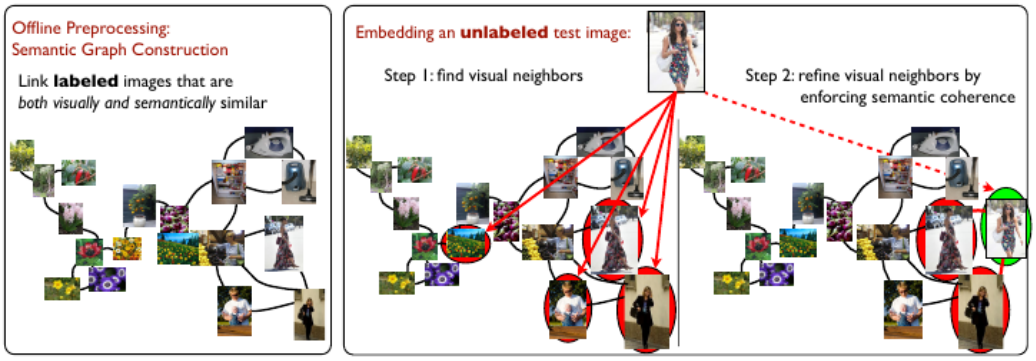


Figura 2.2: Illustrazione concettuale del metodo di inserimento di nuove immagini per la classificazione.

2.2.2 Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition

Nel lavoro presentato da Jia Deng si osserva come le gerarchie semantiche siano costituite da diversi livelli di astrazione. Questa osservazione porta allo studio di un metodo che ottimizzi il rapporto tra accuratezza e specificità, soprattutto per problemi di classificazione su larga scala.

Spesso le tecniche che sfruttano strutture ausiliare, come gli alberi, hanno il difetto che un eventuale errore a livelli superiori, impedisca il corretto funzionamento dell'algoritmo, portando a livelli di accuratezza molto basso. L'idea presentata in [12] è che spesso risulta sbagliato percorrere l'albero fino alle sue foglie, pertanto è da valutare la possibilità di fermarsi ad un determinato livello della gerarchia, diminuendo la specificità della classificazione, ma aumentando così l'accuratezza del sistema. Lo scopo di Deng è di creare un sistema di classificazione che massimizzi l'informazione ottenuta mantenendo un livello prefissato, arbitrariamente basso, di errore.

Questo problema corrisponde alla risoluzione di un problema di programmazione lineare. In particolare nel loro lavoro utilizzano un algoritmo (chiamato

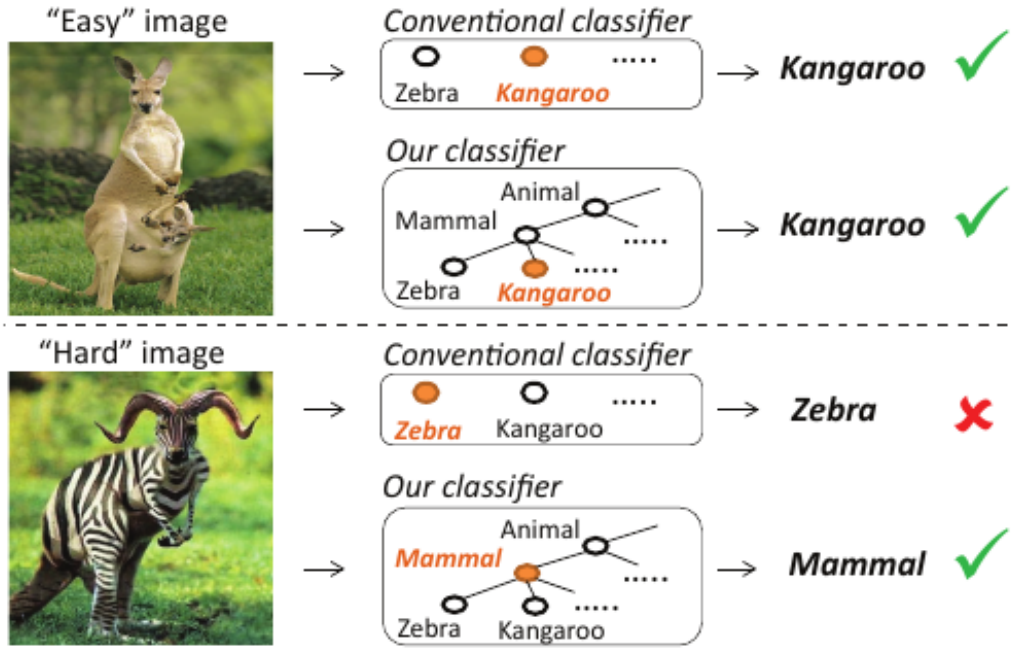


Figura 2.3: Confronto tra il metodo di ottimizzazione accuratezza-specificità, con metodi convenzionali.

DATRS) primale-duale basato sul metodo generalizzato dei moltiplicatori di Lagrange. Hanno dimostrato che DARTS tende all'ottimo, ottimizzando il divario tra accuratezza e specificità.
I test mostrano risultati interessanti soprattutto per database di grandi dimensioni, con elevato numero di categorie.

2.2.3 Semantic Kernel Forests from Multiple Taxonomy

"*Semantic Kernel Forests from Multiple Taxonomy*" è l'articolo a cui saranno fatti più riferimenti, sia per l'idea dell'utilizzo di una foresta di tassonomie, sia per il confronto dei risultati finali.

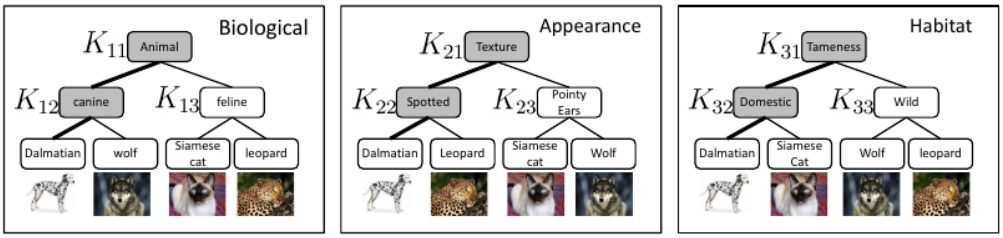


Figura 2.4: Presentazione di 3 diverse tassonomie per 4 classi.

Il termine *tassonomia* è nato nell'ambito delle scienze naturali, ma più recentemente è adoperato in modo più preciso per indicare lo studio teorico della classificazione, attraverso la definizione esatta dei principi, delle procedure e delle norme che la regolano.

A differenza di molti altri lavori che sfruttano il concetto di tassonomia per operare una classificazione gerarchica, nel lavoro della professoressa Grauman [2], si introduce l'idea di non limitarsi ad una singola tassonomia, ma di utilizzarne più di una.

Ciascuna tassonomia ha un proprio modo di descrivere l'insieme di classi che rappresenta, si può affermare che diverse tassonomie forniscono diversi *punto di vista*. Possono esistere tassonomie che descrivono bene determinate categorie, ma non sono soddisfacentemente accurate per altre, pertanto il loro lavoro mira ad utilizzare appunto una "foresta di tassonomie"[2]. L'idea di utilizzare tassonomie diverse consente di avere una visione di insieme delle classi più ampia, infatti utilizzare la foresta di tassonomia consente di analizzare le classi da più punti di vista, aumentando così il potere descrittivo, con un conseguente miglioramento delle prestazioni.

Il lavoro della Grauman è diviso in due fasi principali: una prima fase in cui

[2]Il termine "foresta di tassonomie" deriva dal fatto che una tassonomia è rappresentata sotto forma di albero, pertanto un insieme di più tassonomie si può chiamare foresta.

vengono addestrati i kernel di base (chiamati Semantic Kernel Forest), ed una seconda in cui viene generata una combinazione tra le tassonomie per operare con algoritmi di Multiple Kernel Learning (MKL).
Il database di riferimento, di cui parleremo in seguito in Capitolo 5, presenta 50 categorie di animali. Le tassonomie usate sono 4: una che descrive l'apparenza, una il comportamento, una l'habitat ed infine l'ultima utilizzata è WordNet[3]. I kernel di ciascuna tassonomia sono addestrati utilizzando SIFT (*Scale-Invariant Feature Transform*).
Come molti metodi che utilizzano le tassonomie, anche questo richiede una supervisione, seppure parziale, da parte dell'uomo. Infatti le tassonomie, in quanto tali, corrispondono a concetti insiti nella mente dell'uomo, ovvero descrivono il punto di vista umano delle cose. È pertanto possibile che una tassonomia, concepita sui medesimi criteri, risulti diversa se creata da persone di cultura diversa. L'uso della tassonomia quindi, non solo richiede supervisione, ma risulta essere soggettiva. Con il lavoro presentato in questa tesi si cerca di eliminare questo vincolo.

2.2.4 Creazione Non Supervisionata di Tassonomie

Ci si chiede, a questo punto, se sia effettivamente necessaria la supervisione dell'uomo per creare una buona tassonomia.
Alcuni studi dimostrano che, spesso, categorie semanticamente accoppiate, presentano caratteristiche visive simili. In particolare in [14], data una tassonomia, hanno analizzato le variazioni delle caratteristiche visive, al variare della profondità dell'albero, ovvero è stato testato se a ristretti domini seman-

[3]WordNet è un database semantico-lessicale per la lingua inglese elaborato dal linguista George Armitage Miller presso l'Università di Princeton, che si propone di organizzare, definire e descrivere i concetti espressi dai vocaboli.

tici, corrisponde una piccola variazione nello spazio visivo. È stato inoltre dimostrato che insiemi di categorie semanticamente correlate tra loro, condividono dei prototipi visivi che le descrivono con elevata accuratezza.

È quindi possibile creare tassonomie in modo non supervisionato?
In "*Unsupervised organization of image collections: taxonomies and beyond*" [1], viene introdotto un modello Bayesiano non parametrico, chiamato TAX, che organizza collezioni di immagini in tassonomie, a forma di albero, senza bisogno di alcuna supervisione. Questo modello, ispirato ad un processo stocastico a tempo discreto[4], associa a ciascuna immagine un percorso attraverso la tassonomia. Immagini simili condividono i segmenti iniziali di questo percorso, quindi condividono alcuni aspetti della loro rappresentazione grafica. Ciascun nodo interno della tassonomia rappresenta informazioni che sono comuni a più immagini.
Senza perdersi in dettagli implementativi, in [1] si evidenzia la possibilità di creare tassonomie definendole a partire da come l'informazione è condivisa tra più immagini. L'idea principale è che l'informazione in comune ad insiemi di immagini, è rappresentata solamente dai nodi interni comuni a queste immagini (Figura 2.5). Il limite di TAX è l'incredibile lentezza nell'addestramento della tassonomia, gli autori riportano che l'addestramento con 10 000 immagini impiaga circa 7 giorni. È di importanza cruciale, se si intende utilizzare tassonomie non supervisionate, valutare metodi più efficienti.

2.2.5 Meta-Classi

Nell'articolo di Torresani, *Meta-class features for large-scale object categorization on a budget* [10], si introduce il concetto di "Meta-classe". Mentre

[4]In particolare a *Nested Chinese Restaurant Process* (NCRP).

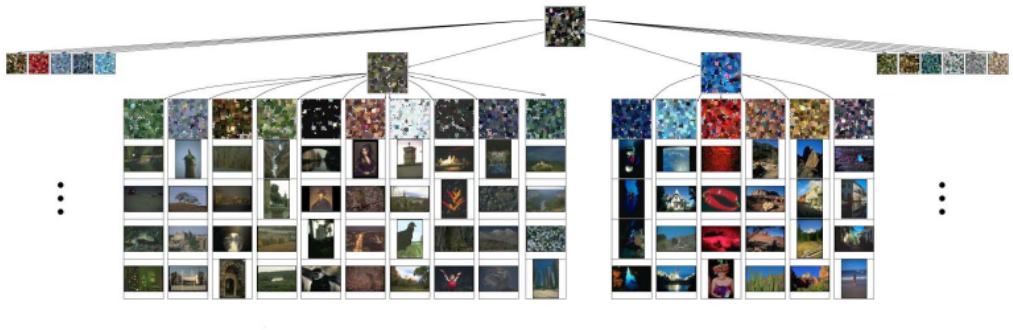
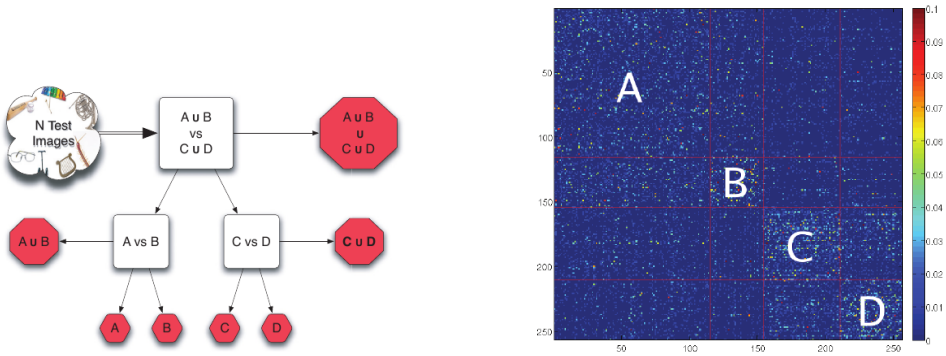


Figura 2.5: Esempio di tassonomia creata con il metodo non supervisionato descritto in [1].

nello stato dell'arte, solitamente, si utilizzano attributi tradizionali che descrivono classi selezionate a mano dall'uomo, e proprietà visive predefinite, le features descritte in questo lavoro sono addestrate automaticamente e corrispondono a classi "astrette", chiamate appunto *Meta-classi*.
Ciascuna meta-classe è una "super-categoria" ottenuta raggruppando un insieme di classi tali che, collettivamente, sono facilmente di distinguibili da altri insiemi di classi. Si ottengono così insiemi di attributi che codificano proprietà visive generiche, condivise da più classi.
La meta-classe consente di creare una nuova classe che descrive e consente di riconoscere anche nuove categorie che non fanno parte del database iniziale di riferimento. Viene dimostrato la l'introduzione delle meta-classi non solo aiuta a descrivere classi non presenti nell'insieme iniziale di addestramento, ma anche a distinguere insiemi di classi diversi tra loro.

Un'idea simile a quella di Torresani si trova in [15], dove, anche in questo caso, la tassonomia viene creata in modo automatico e non supervisionato. Il principio che sta dietro all'architettura è semplice e viene illustrato in Figura 2.6. Invece di addestrare un singolo classificatore One-vs-All (sotto-



(a) Rappresentazione gerarchica (b) Matrice di confusione

Figura 2.6: Raggruppamento Top-Down. Classi facilmente confondibili sono raggruppate per addestrare un nuovo classificatore.

sezione 1.2.2), la classificazione viene raggiunta attraverso una separazione ricorsiva di insiemi di possibili classi, all'interno di sottoinsiemi approssimativamente uguali.

Come abbiamo già accennato in sezione 2.1, l'utilizzo di classificatori binari è meno complicato che utilizzare nodi addestrati con tecnica One-vs-All.

In [15] vengono utilizzati due metodi per la creazione automatica di tassonomie, entrambe basate sullo studio della matrice di confusione (Figura 2.6b). Il primo metodo, approccio Top-Down, divide la matrice di confusione in due gruppi utilizzando *Self-Tuning Spectral Clustering* [16]. Questa è una variante dell'algoritmo di Spectral Clustering che, dato che sono utilizzati alberi binari, utilizza due per numero di cluster. Il secondo, invece, genera l'albero in modo Bottom-Up. Ad ogni step, due gruppi di categorie, che presentano la maggiore mutua confusione, sono unite insieme. Questo processo prosegue fino ad avere una unica super-categoria che racchiude al suo interno tutte le classi iniziali.

In Figura 2.7 viene mostrato un confronto tra queste due tecniche di creazio-

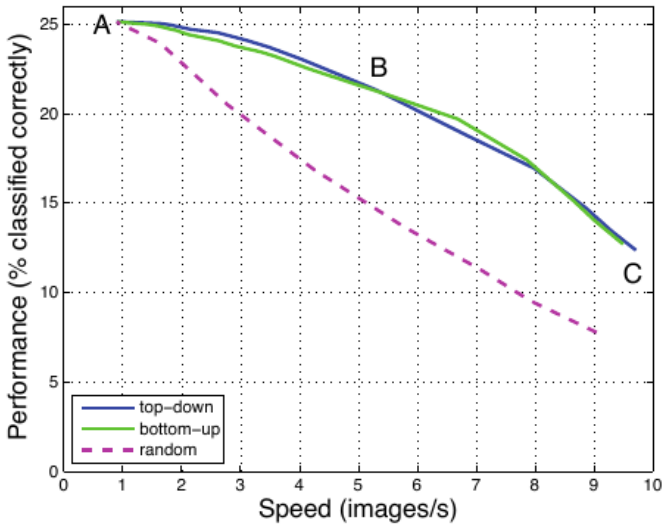


Figura 2.7: Comparazione tra 3 diversi metodi per la generazione della tassonomia.

ne automatica di tassonomie. Si nota che i risultati ottenuti con questi due metodi sono migliori rispetto ad una qualsiasi altra tassonomia. Inoltre la scelta tra il metodo Top-Down e quello Bottom-Up non pregiudica i risultati.

Parte II

Approccio Proposto

Capitolo 3

Creazione e Addestramento delle Tassonomie

In questo capitolo viene esposta la tecnica per la creazione non supervisionata delle tassonomie (sezione 3.1) adottata nel lavoro presentato. Di seguito, tali tassonomie verranno addestrate con due modelli, il modello di decisione e quello di acceptance (sezione 3.2), discriminando tra nodi interni all'albero e foglie.

3.1 Creazione Tassonomie

Il metodo utilizzato per la creazione delle tassonomie in modo non supervisionato, utilizza il concetto di meta-classe, introdotto e descritto in sottosezione 2.2.5. Per ciascuna tassonomia, la radice racchiuderà tutte le classi, le foglie rappresenteranno le categorie iniziali, mentre i nodi interni rappresenteranno le meta-classi, ovvero classi astratte che descrivono caratteristiche comuni a più classi reali.

Nonostante l'approccio utilizzato in [10] sia di tipo Top-Down, in questo lavo-

ro le meta-classi vengono utilizzate per creare la tassonomia con un approccio "greedy" Bottom-Up. Come si può vedere in Figura 2.7, l'utilizzo dell'approccio Bottom-Up non degrada le prestazioni rispetto a quello Top-Down, pertanto è preferibile in quanto più semplice ed intuitivo.
L'algoritmo di creazione della tassonomia è tanto semplice quanto efficace. Prevede quattro passaggi principali:

1. Addestramento di Multiclass-SVM
2. Validazione del modello appena addestrato
3. Analisi della matrice di confusione
4. Fusione delle classi (o meta-classi) con la mutua confusione maggiore.

Analizziamo in dettaglio questi quattro passaggi.
Per l'utilizzo di SVM è stata adoperata la libreria matlab `libsvm v. 3.17`. L'addestramento di Multiclass-SVM genera $\frac{N(N-1)}{2}$ SVM binari di tipo One-vs-One (sottosezione 1.2.3), dove N è il numero di classi per l'addestramento. Senza entrare nei dettagli di come SVM lavora (argomento già affrontato nel sottosezione 1.2.1), possiamo proseguire con la sua validazione. La validazione viene eseguita su un insieme disgiunto rispetto a quello utilizzato per l'addestramento di Multiclass-SVM, che viene chiamato insieme di validazione (o *validation set*). Analizzando l'output di Multiclass-SVM sull'insieme di validation, è possibile creare una matrice di confusione.

3.1.1 Analisi della Matrice di Confusione

La matrice di confusione restituisce una rappresentazione dell'accuratezza di classificazione. Ogni colonna della matrice rappresenta i valori predetti dal Multiclass-SVM, mentre ogni riga rappresenta ciascuna classe. L'elemento

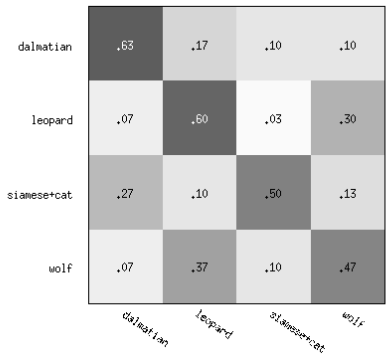
sulla riga i e sulla colonna j è il numero di casi in cui il classificatore ha classificato la classe vera j come classe i. Attraverso questa matrice è osservabile se vi è "confusione" nella classificazione delle diverse classi.
Una volta generata la matrice di confusione (Figura 3.1a), trascurando gli elementi presenti sulla diagonale, si può procedere al calcolo della mutua confusione tra le classi. Data $\mathcal{C}$ la matrice di confusione, si ricava la Matrice di Mutua Confusione, indicata con $\mathcal{M}$ (Figura 3.1b):

$$\mathcal{M} = \frac{(\mathcal{C} - \text{diag}(\mathcal{C})) + (\mathcal{C} - \text{diag}(\mathcal{C}))^T}{2} \tag{3.1}$$

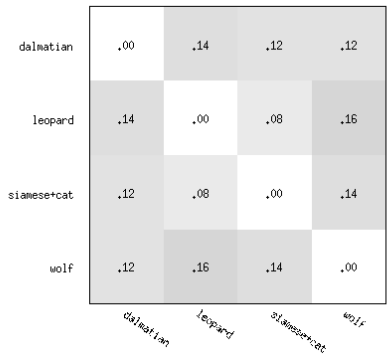
Infine si selezionano i e j tali che

$$\max \mathcal{M}_{ij} \qquad \forall i, j \in (N \times N) \tag{3.2}$$

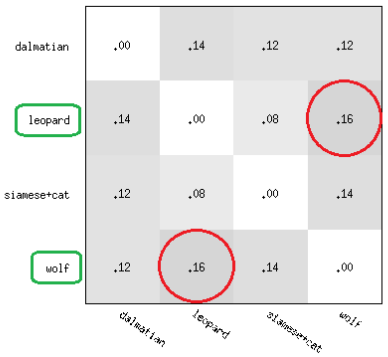
ovvero le due classi che presentano la mutua confusione massima. La classe i e la classe j vengono fuse per generare una nuova meta-classe. La meta-classe appena creata andrà a costituire un nodo interno nell'albero che definisce la tassonomia, ed avrà come nodi figli, le due classi (o meta-classi), da cui è stata generata.
Successivamente verrà addestrato un nuovo Multiclass-SVM con le precedenti classi, escludendo le due che sono state fuse insieme, e considerandole come una unica classe astratta.
Questo processo itera fino ad ottenere una unica meta-classe – la "radice" della tassonomia $\mathcal{T}$ – che racchiude al suo interno le caratteristiche di tutte le classi del database.



(a) Matrice di confusione a 4 classi



(b) Mutua confusione tra classi



(c) Generazione di nuova Meta-classe

Figura 3.1: In figura è mostrato il processo di analisi della matrice di confusione per il caso a 4 classi. In Figura 3.1b viene mostrata la mutua confusione tra le classi, in Figura 3.1c viene selezionata la mutua esclusione massima per generare la nuova meta-classe.

3.1.2 Creazione Foresta di Tassonomie

Abbiamo descritto, in sezione 3.1, come creare una tassonomia $\mathcal{T}$, ma l'idea iniziale di usare tassonomie diverse, come descritto in sottosezione 2.2.3, non è ancora stata sfruttata.

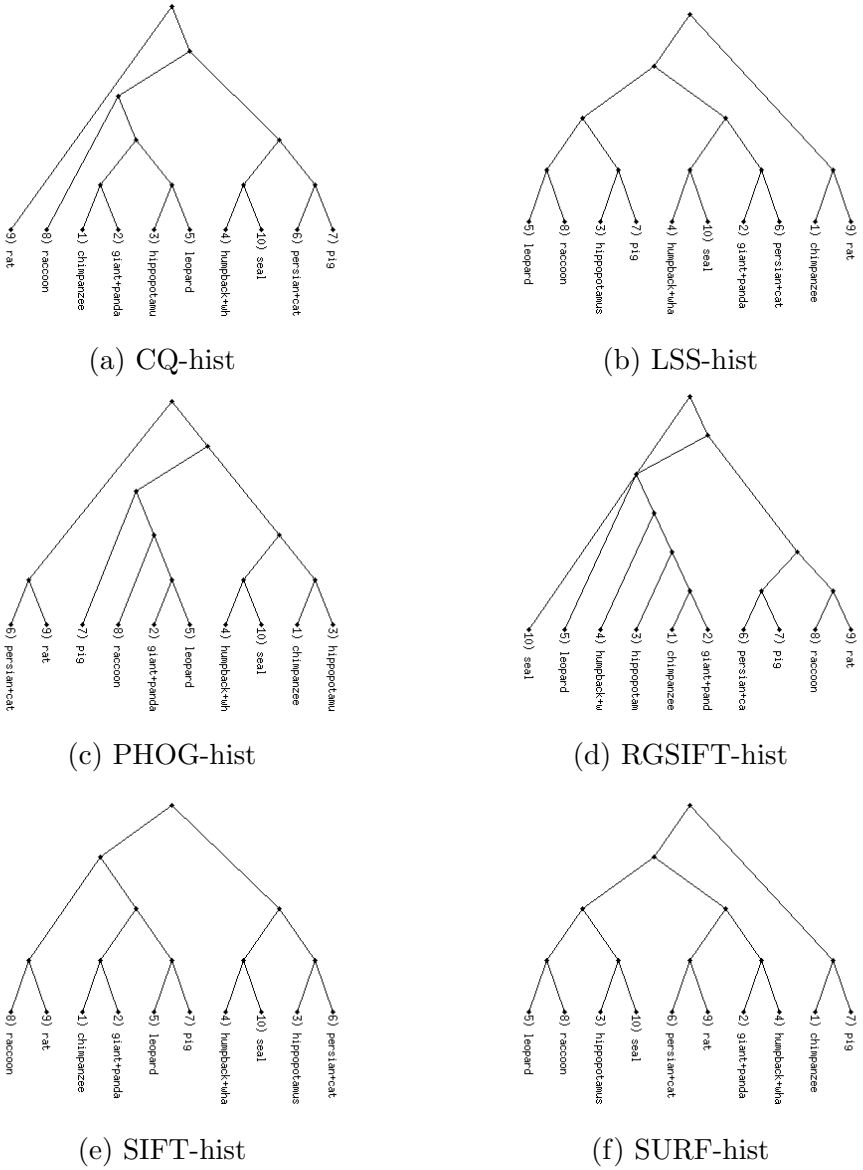
Nell'articolo su "*Semantic Kernel Forest*" [2], viene introdotta l'idea di rappresentare le stesse classi a partire da punti di vista semantici differenti. Vengono infatti distinte tassonomie per l'habitat, il comportamento e altre caratteristiche di tipo semantico. Ma come è stato spiegato in [14], le caratteristiche visive sono spesso comuni per oggetti semanticamente vicini tra loro. Quindi possiamo pensare di utilizzare solo caratteristiche visive per creare le tassonomie, e successivamente la foresta. In particolare si possono utilizzare descrittori di features diversi per ciascuna immagine.

Ciascun descrittore di features rappresenta caratteristiche visive differenti pertanto fornisce un diverso punto di osservazione, caratterizzato da aspetti visivi, per ciascuna classe. È quindi sensato creare una tassonomia per ciascuna feature $\mathcal{T}_f$. La foresta $\mathcal{F}$ sarà quindi definita dall'unione di tutte le tassonomie

$$\mathcal{F} = \bigcup^{n} \mathcal{T}_f \tag{3.3}$$

dove n è il numero di features utilizzate per creare le tassonomie.

Il database di riferimento per questo lavoro (Capitolo 5), fornisce 6 descrittori di features per ciascuna immagine, come mostrato in Figura 3.2. Pertanto, a partire da questi, è stato possibile creare 6 tassonomie diverse, che descrivono le classi secondo le caratteristiche del descrittore utilizzato.



(a) CQ-hist (b) LSS-hist

(c) PHOG-hist (d) RGSIFT-hist

(e) SIFT-hist (f) SURF-hist

Figura 3.2: Illustrazione delle sei tassonomie create a partire da sei descrittori di features diversi.

3.2 Addestramento Tassonomie

Una volta creata la foresta di tassonomie $\mathcal{F}$, si può procedere con l'addestramento delle stesse.
Ciascuna tassonomia $\mathcal{T}_f$ viene addestrata con il descrittore f utilizzato per la sua creazione.
Ciascun nodo verrà addestrato con due SVM caratterizzati da modelli diversi, che, per comodità e chiarezza di esposizione, distingueremo chiamandoli rispettivamente **modello di decisione** e **modello di *acceptance***.

3.2.1 Modello di Decisione

Il modello di decisione $\mathcal{D}$ è un modello di SVM binario che consente di decidere su quale percorso, all'interno dell'albero, proseguire con la classificazione. Si distingue tra modello di decisione per

- Nodi interni ($\mathcal{D}_{node}$)
- Nodi foglie ($\mathcal{D}_{leaf}$)

Il modello di decisione per i nodi interni (mostrato in Figura 3.3) mostra come esso viene addestrato. Per ciascun nodo si addestra un modello capace di distinguere tra le classi che sono descritte dalla meta-classe del figlio destro da quelle descritte dalla meta-classe del figlio sinistro. Escludendo eventualmente tutte le classi che non sottostanti al nodo che stiamo addestrando[1].
Per quanto riguarda invece le foglie della tassonomia, il modello di decisione prevede, per ciascuna di esse, di addestrare un classificatore One-vs-All (sottosezione 1.2.2), come mostrato in Figura 3.4.

[1]Per il nodo radice, l'addestramento viene fatto su tutte le classi del database.

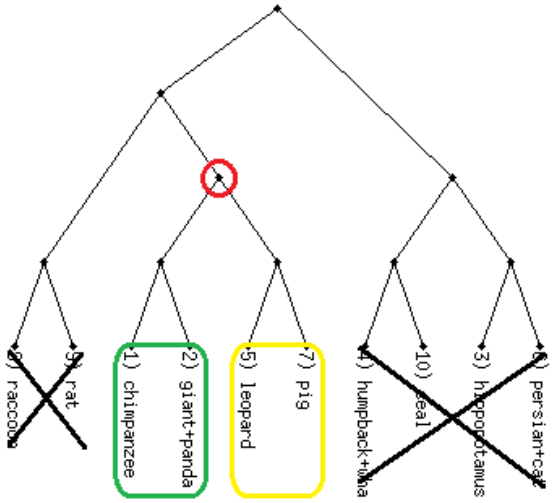


Figura 3.3: Rappresentazione grafica dell'addestramento del modello di decisione di un generico nodo interno $\mathcal{D}_{node}$ di una generica tassonomia $\mathcal{T}_f$.

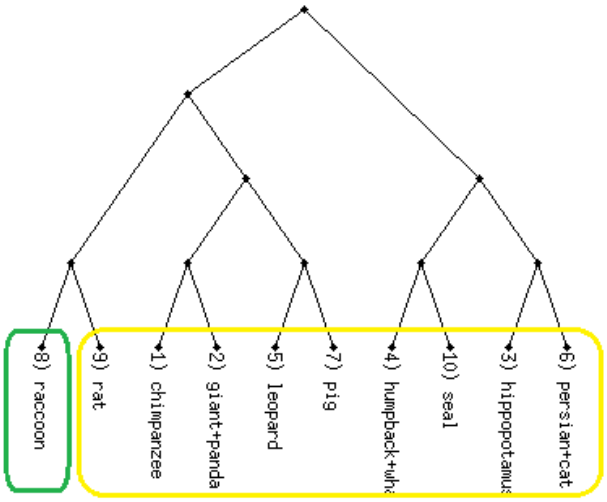


Figura 3.4: Rappresentazione grafica dell'addestramento del modello di decisione di una generica foglia $\mathcal{D}_{leaf}$ di una generica tassonomia $\mathcal{T}_f$.

3.2.2 Modello di *Acceptance*

Il modello di decisione appena descritto non introduce niente di nuovo rispetto allo stato dell'arte. Ma esso viene combinato con il modello di acceptance, che è il maggior contributo del lavoro presentato, in funzione del successivo utilizzo della foresta di tassonomie.

Come da nome, il modello di acceptance $\mathcal{A}$ viene utilizzato dai nodi figli per decidere se la scelta effettuata dal modello di decisione del padre, è corretta o meno. L'idea è quella di inserire una forma di *backtracking rilassato*, all'interno dell'albero, per sopperire ad eventuali errori di valutazione da parte del modello di decisione del padre.

Anche in questo caso si tratta di un SVM binario che conferma o meno la decisione del padre, ovvero fornisce un valore positivo se l'SVM riconosce l'immagine come appartenente alle classi descritte dalla meta-classe su cui si costruisce il modello di acceptance, o negativo altrimenti. Come mostrato in

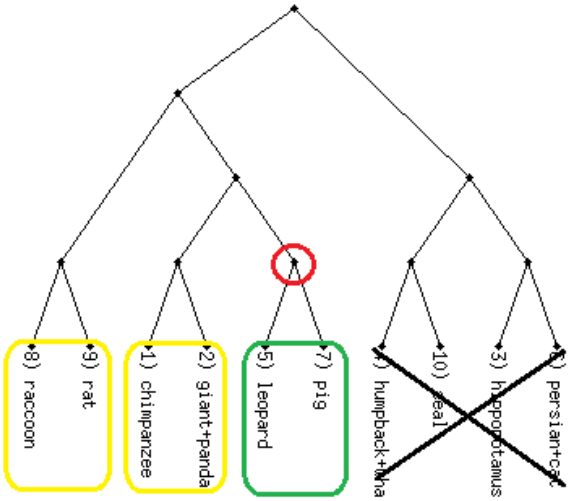


Figura 3.5: Rappresentazione grafica dell'addestramento del modello di acceptance $\mathcal{A}$ per un generico nodo della tassonomia $\mathcal{T}_f$.

Figura 3.5, le classi coinvolte sono quelle descritte dalla meta-classe su cui si sta generando il modello (considerate come esempi positivi) e quelle descritte dalle meta-classi del "fratello" e dello "zio" (considerate come esempi negativi). L'analogia dei termini fratello e zio, sta ad indicare esattamente i gradi di parentela con il nodo su cui si sta generando il modello di acceptance. Con fratello e zio si intende, rispettivamente l'altro figlio del padre, ed il figlio del nonno[2] che non è padre per il nodo in esame. In questo modello non vengono considerate quelle classi che non sono descritte dalle meta-classi sopra elencate.

Tutti i classificatori SVM di entrambi i modelli (decisione ed acceptance), sono classificatori lineari, addestrati su due classi (o meta-classi), pertanto sono classificatori binari. Inoltre per ciascuno di essi è previsto un algoritmo greedy per la ricerca del miglior valore di cross-validazione, e ciascuna meta-classe è addestrata assegnando un peso w che dipende dal numero di classi che descrive:

$$w = \frac{n_{tot}}{2 \cdot n} \tag{3.4}$$

dove n è il numero di classi descritte dalla meta-classe a cui è associato il peso, e n_{tot} è il numero totale di classi utilizzate per l'addestramento dell'SVM.

[2]Con nonno, proprio come i gradi di parentela, si intende il padre del padre del nodo in esame.

Capitolo 4

Metodi di Classificazione

In questo capitolo verranno descritte le tecniche di classificazione utilizzate, distinguendo tra classificazione gerarchica classica, utilizzando un albero di decisione binario (sezione 4.1), classificazione delle foglie (sezione 4.2), classificazione tramite modello di acceptace *utilizzato da un* Relaxed Decision Tree *(sezione 4.3), ed infine la classificazione che sfrutta il concetto di* Foresta di Tassonomie *(sezione 4.4).*

4.1 Albero di Decisione Binario

La tassonomia è un tipo di organizzazione gerarchico che modella un certo tipo di conoscenza. Costruire una tassonomia (come abbiamo fatto nel Capitolo 3) fornisce di per se una struttura gerarchica per la classificazione degli oggetti che compongono la tassonomia stessa. Come abbiamo visto in Parte I, lo stato dell'arte è stato inondato di questo genere di tecniche di classificazione, e di loro varianti. Se siamo in possesso di una struttura gerarchica che ben definisce l'insieme di categorie da classificare, possiamo percorrer-

la, ovvero prendere delle decisioni sul percorso dalla radice alle foglie. Tali decisioni ci consentono di migliorare le performance di riconoscimento, e/o le prestazioni in termini computazionali, in confronto ad una classificazione di tipo *flat*, che pone tutte le classi sul medesimo livello. È proprio questo concetto di stratificazione delle classi che rende la classificazione gerarchica possibile. Nel percorso dalla radice alle foglie, la decisione, presa in questo caso da un classificatore binario SVM, diventa sempre più specifica. In questo modo il numero di confronti con i classificatori, diventa proporzionale all'altezza dell'albero, piuttosto che al numero di categorie da classificare.

$$O(N) \qquad \text{per classificatori One-vs-All} \tag{4.1}$$

$$O(N^2) \qquad \text{per classificatori One-vs-One} \tag{4.2}$$

$$O(\log_2 N) \qquad \text{per classificatori gerarchici} \tag{4.3}$$

Nel lavoro qui presentato, piuttosto che percorrere semplicemente la tassonomia per ottenere una singola classe di appartenenza, si è preferito optare per un sistema di voti, per avere una classifica delle possibili classi di appartenenza per ciascun esempio in esame. Questa scelta è stata presa per due motivi. Il primo è dato dal fatto che una classifica di possibili soluzioni è preferibile rispetto ad una decisione univoca, in quanto, in caso di errore nella classificazione, è possibile valutare le decisioni successive, consentendo una stima sull'errore commesso. Infatti un errore può essere considerato più o meno grave in base alla distanza che intercorre tra il risultato della classificazione e l'effettiva classe di appartenenza.

Il secondo invece è dato dalla necessità di un voto per ciascuna classe, da parte di ogni tassonomia, per valutare il risultato secondo la tecnica di classificazione che sfrutta la foresta di tassonomie, come sarà spiegato successivamente in sezione 4.4.

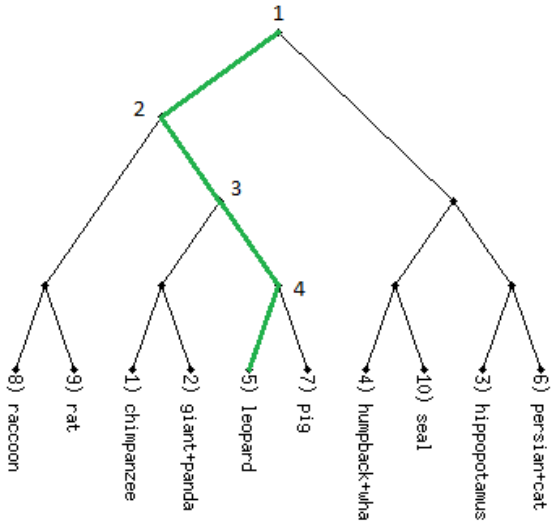


Figura 4.1: Rappresentazione grafica dell'esplorazione della tassonomia seguendo le decisioni degli SVM di ciascun nodo.

Per assegnare un valore a ciascuna classe si utilizzano gli output degli SVM del modello di decisione $\mathcal{D}$ (descritto accuratamente in sottosezione 3.2.1), che contribuiscono alla scelta del percorso da esplorare. Essendo l'SVM binario, oltre a stabilire su quale ramo dell'albero proseguire, fornisce anche un valore di *decisione*. Questo valore di decisione definisce la distanza dell'istanza testata, dall'iperpiano definito dall'SVM, maggiore sarà la distanza, minore sarà l'indecisione del classificatore per quella scelta.
L'idea è quella di esplorare il percorso sulla tassonomia, seguendo le decisioni degli SVM, ma assegnando a ciascun arco il valore di decisione restituito dall'SVM, positivo per il percorso deciso dall'SVM, e negativo altrimenti. La decisione viene propagata fino alle foglie, così da ottenere, per ciascuna di esse, un valore decisionale v. Siano N le classi di $\mathcal{T}_f$ e $\mathcal{D}_{node}(i)$ la decisione del modello per l'i-esimo nodo appartenente al percorso σ dalla radice alla

foglia, il voto v assegnato a ciascuna classe sarà:

$$v(l) = \sum \mathcal{D}_{node}(i) \qquad i \in \sigma, \quad l \in N \tag{4.4}$$

Maggiore sarà il voto $v(l)$, maggiore sarà il rank per quella foglia[1] l.
Il valore assegnato a ciascuna classe ci consente di dare un ordine di classificazione, dove il valore massimo indica la classe più simile, il minimo quella maggiormente differente.

4.2 Classificazione Foglie

La classificazione delle foglie, viene così chiamata per poterla comparare con il resto della tassonomia e delle classificazioni descritte in questo lavoro. In realtà si tratta della più semplice delle classificazioni, ovvero la One-vs-All (sottosezione 1.2.2). Nel modello di decisione descritto in sottosezione 3.2.1, abbiamo mostrato come le foglie della tassonomia, ovvero le classi che compongono il dataset, vengano addestrate secondo il modello $\mathcal{D}_{leaf}$.
Questo tipo di classificazione, spesso utilizzata per la classificazione di più di due classi, presenta notevoli problematiche se applicata su larga scala. Nel nostro caso è stata effettuata al solo scopo di valutare le prestazioni delle tassonomie create in modo non supervisionato.
Anche in questo caso, per poter comparare i risultati con la classificazione gerarchica, non otteniamo N risultati indipendenti[2] tra loro. Ma vengono valutati, i valori di decisione forniti in output dall'SVM del modello di decisione delle foglie. Gli SVM di ciascuna foglia andranno a votare per la propria classe, fornendo un rank comparabile con quello restituito dalla classificazione

[1] Si ricorda che ad ogni foglia è assegnata una classe originale, e ad ogni nodo interno una meta-classe astratta.

[2] Uno per ciascuna foglia/classe della tassonomia.

gerarchica (sezione 4.1).

$$v(l) = \mathcal{D}_{leaf}(l) \qquad l \in N \tag{4.5}$$

Essendo effettivamente il risultato di questa classificazione comparabile con quello precedentemente ottenuto, è stata anche valutata una combinazione delle due tecniche. Ovvero ai valori ottenuti su ciascuna classe dalla classificazione gerarchica, per ciascuna tassonomia percorrendo σ dalla radice alle foglie secondo le decisioni prese su $\mathcal{T}_f$, vengono aggiunti quelli forniti dai voti delle foglie, aumentando così il potere informativo della decisione associata a ciascuna categoria.

$$v(l) = \left(\sum \mathcal{D}_{node}(i)\right) + \mathcal{D}_{leaf}(l) \qquad i \in \sigma, \quad l \in N \tag{4.6}$$

4.3 Relaxed Decision Tree (RDT)

L'albero di decisione rilassato, o *Relaxed Decision Tree* (RDT), è stato studiato in funzione del suo utilizzo per la classificazione con foresta di tassonomie. L'idea di base espande il concetto di decisione fino ad ora concepito. Selezionare un percorso all'interno di una gerarchia, può introdurre un errore non più correggibile, a meno che non si utilizzino tecniche di *backtracking* sulla tassonomia. Il modello di decisione fino ad ora utilizzato dall'albero di decisione binario, è da considerarsi come "monodirezionale": ovvero la decisione viene sempre presa dall'alto (radice) verso il basso (foglie).
La tecnica di classificazione che qui presentiamo introduce il concetto di *acceptance*, ovvero una sorta di approvazione data dal basso verso l'alto. Questa idea prevede che la decisione del nodo padre, generata dal modello di decisione, come per la classificazione gerarchica, venga accettata o respinta dai

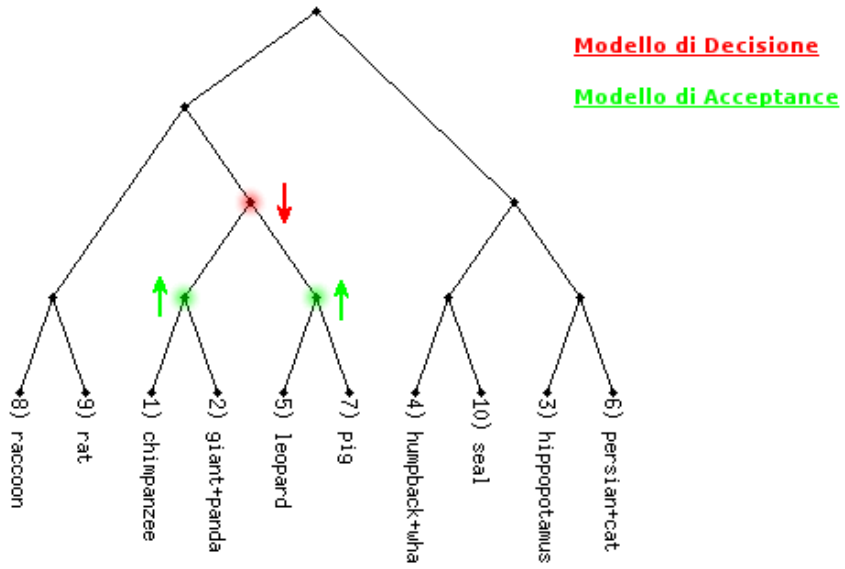


Figura 4.2: In figura vengono mostrati i due differenti modelli di decisione $\mathcal{D}$ (in rosso) e di acceptance $\mathcal{A}$ (in verde).

suoi due figli utilizzando l'SVM addestrato secondo il *modello di acceptance*, che è stato descritto nella sottosezione 3.2.2.

Come spiegato precedentemente il modello di acceptance differisce da quello di decisione non solamente per il metodo di utilizzo, ma anche per le classi su cui viene effettuato l'addestramento. Il modello di acceptance tiene in considerazione non solo le classi descritte dal nodo su cui il modello è presente, ma tiene traccia anche di quelle descritte dai nodi "fratello" e "zio".

In caso di scelta corretta da parte del modello di decisione del padre, i modelli di acceptance dei figli andranno a confermare tale decisione, aumentando così il valore assegnato alle classi descritte dalla meta-classe selezionata. In caso contrario, se la scelta del padre è errata, il voto di acceptance dei figli andrà a ridurre tale errore.

Altra caratteristica fondamentale di questa tecnica, che viene valorizzata soprattutto con l'utilizzo della foresta di tassonomie, è la possibilità di correg-

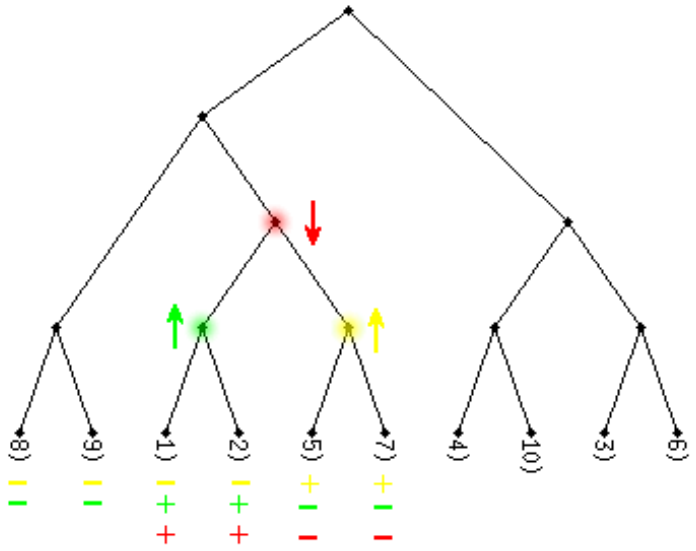


Figura 4.3: In figura vengono rappresentati i voti assegnati dal modello di decisione e da quello di acceptance, considerando con il verde il figlio selezionato dal padre come classe corretta, e con il giallo quella errata. I segni + e − rappresentano rispettivamente l'accettazione o il rifiuto da parte del nodo con il corrispettivo colore.

gere errori commessi a livelli superiori. Se entrambi i modelli di acceptance dei figli rifiutano l'esempio di test, significa che il percorso intrapreso è probabilmente non corretto. Pertanto il voto di rifiuto verrà assegnato alle classi descritte dal nodo zio.

Questo meccanismo consente, in caso di indecisione, di distribuire i voti su nodi non presenti sul percorso esplorato. Mentre nel caso in cui la decisione è sicura, il voto sarà paragonabile a quello generato dall'albero di decisione binaria (sezione 4.1).

Vediamo quindi ora nel dettaglio come questo modello si comporta in una

situazione di decisione corretta da parte del padre, e successivamente nel caso di decisione errata.

4.3.1 Distribuzione Voti

Prendiamo come riferimento la figura Figura 4.4. Supponiamo di voler valutare il modello con una immagine di test appartenente alla classe 1, supponiamo inoltre di trovarci sul nodo indicato con il colore rosso, che da ora in poi chiameremo nodo padre, e che il modello di decisione $\mathcal{D}$ (indicato in figura con $\downarrow$) prenda la decisione corretta, ovvero di proseguire verso il nodo verde. In questo scenario il nodo padre darà un voto positivo per le classi sottostanti al nodo verde, e negativo per quelle sottostanti al nodo giallo. I due modelli di acceptance, $\mathcal{A}_L$ per il figlio sinistro e $\mathcal{A}_R$ per il destro, escludendo

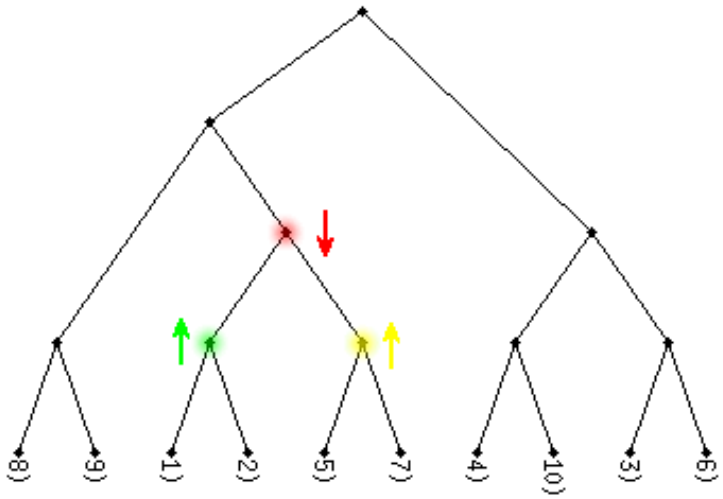


Figura 4.4: Rappresentazione grafica dei modelli chiamati in causa nel RDT ad uno step intermedio. Con $\downarrow$ viene indicato il modello di decisione del padre, mentre con $\uparrow$ quello di acceptance dei figli.

eventuali errori, forniranno un voto positivo e negativo, rispettivamente se guardiamo prima il nodo verde e poi quello giallo. Se le decisioni saranno sicure allora i valori saranno elevati, altrimenti saranno molto bassi (quindi influiranno poco sulla decisione complessiva).
Come mostrato in Figura 4.3, il voto del nodo verde, se affermativo[3], andrà a sommarsi alle foglie sottostanti al nodo verde e a sottrarsi per le foglie del di lui fratello e zio. Viceversa il voto del nodo giallo, se non riconoscerà la classe 1 come a lui sottostante, produrrà un voto negativo, che andrà a sommarsi, decrementandone così il voto, per le classi descritte da quel nodo, e a sottrarsi per le classi del di lui fratello e zio, andando in questo caso ad aumentarne il voto.
Infine un altro caso di studio è quando l'errore è stato commesso a monte, ovvero se qualsiasi sia la scelta del nodo rosso, sia sbagliata poiché la classe corretta non viene descritta da quel nodo. Supponiamo, a questo proposito, che la classe di test sia la numero 8, quindi non descritta dal nodo rosso, ma che comunque le decisioni prese precedentemente ci abbiano portato a trovarci proprio su quel nodo. In questo caso il modello di decisione del nodo rosso sceglierà su quale dei due nodi figli proseguire, ma, plausibilmente, il voto sarà molto basso, poiché fortemente affetto da indecisione. Per quanto riguarda i modelli di acceptance del nodo verde e giallo, entrambi rigetteranno l'immagine di test, poiché entrambi addestrati anche con la classe 8[4], restituendo un valore per entrambi negativo. In questo caso questo valore negativo verrà sottratto per le classi sottostanti al nodo zio, aumentando così

[3]Ovvero che approva la decisione del padre.

[4]Come spiegato in sottosezione 3.2.2 il modello di acceptance viene addestrato anche con le classi descritte dallo zio del nodo addestrato.

il voto per quelle classi.

$$d = \begin{cases} \mathcal{D} + \mathcal{A}_L - \mathcal{A}_R & \text{per le foglie del nodo sinistro} \\ \mathcal{D} - \mathcal{A}_L + \mathcal{A}_R & \text{per le foglie del nodo destro} \\ -\mathcal{A}_L - \mathcal{A}_R & \text{per le foglie del nodo zio} \end{cases} \tag{4.7}$$

dove $\mathcal{D}$, $\mathcal{A}_L$ e $\mathcal{A}_R$ rappresentano rispettivamente le decisioni del modello di decisione, e quello di acceptance per il figlio sinistro e destro.
Il voto $v(l)$ per ciascuna foglia dato dall'RDT sarà quindi la somma di tutte le singole decisioni d prese lungo il percorso esplorato σ che va dalla radice ad una foglia, seguendo le decisioni prese al livello superiore della tassonomia $\mathcal{T}_f$

$$v(l) = \sum d_i \qquad i \in \sigma, \quad l \in N \tag{4.8}$$

Questa forma di rivisitazione delle decisioni passate, introduce una sorta di *soft backtracking*[5], un modo per valutare e votare anche classi che sono state escluse da decisioni precedenti, consentendo di correggere errori eventualmente commessi nell'esplorare la tassonomia.

Utilizzare tre classificatori differenti, oltre ad introdurre elasticità nella classificazione, consente anche di diminuire gli errori di decisione dei singoli classificatori: se un classificatore commette un errore, può essere corretto dagli altri due.

[5]Non può essere considerata una tecnica di backtracking in quanto non ripercorre a ritroso la tassonomia

4.4 Classificazione tramite Foresta di Tassonomie

Le tassonomie, sia che siano generate in modo supervisionato o non supervisionato, o che rappresentino caratteristiche semantiche o visive, non descrivono in modo uniforme le categorie. Pertanto avremo tassonomie che consentono una facile distinzione tra determinate classi, ma scarsa per altre. Nel lavoro della Grauman [2] le tassonomie che compongono la foresta sono generate in modo supervisionato e distinguono le classi secondo caratteristiche semantiche. Nel nostro lavoro, invece, si vuole studiare come questo metodo funziona per tassonomie generate in modo completamente automatiche e che distinguono le categorie per caratteristiche puramente visive.

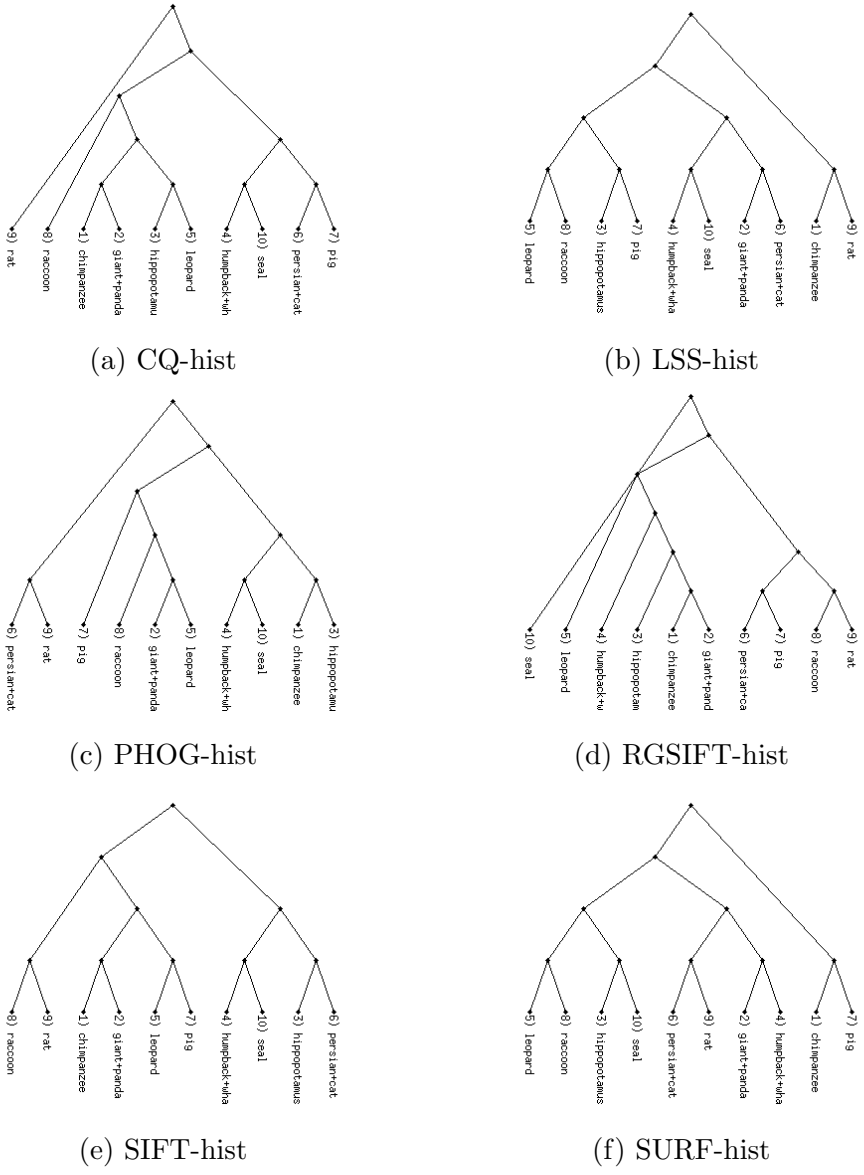
Ogni descrittore di features (cq-hist, lss-hist, phog-hist, rgsift-hist, sift-hist e surf-hist dei quali parleremo in seguito) utilizzato per la creazione delle singole tassonomie, descrive in modo diverso caratteristiche visive diverse delle medesime categorie (Figura 4.5). Questo può essere visto come un analisi da più "punti di vista" dello stesso oggetto. Vedere un oggetto da punti di vista differenti, ci consente di descriverlo in modo più preciso, e quindi di riconoscerlo con maggior accuratezza.

L'idea di usare più tassonomie per meglio descrivere tutte le classi è tanto semplice quanto efficace.

Data una successione di tassonomie $\{\mathcal{T}_f\}_{f=1}^{n}$, dove n è il numero di tassonomie (nel nostro caso $n = 6$) e $\mathcal{V}_f(l)$ il voto assegnato dalla singola tassonomia a ciascuna classe l,

$$v(l) = \sum_{f=1}^{n} \mathcal{V}_f(l) \qquad l \in N \tag{4.9}$$

Il metodo di classificazione con foresta di tassonomie, utilizza i risultati di ogni singola tassonomia, per combinarli e restituire una nuova classifica di



(a) CQ-hist (b) LSS-hist

(c) PHOG-hist (d) RGSIFT-hist

(e) SIFT-hist (f) SURF-hist

Figura 4.5: Illustrazione della foresta composta dalle sei tassonomie.

possibili soluzioni, dove anche in questo coso, la classe con punteggio più alto sarà la classe che con maggior probabilità, mentre quella con punteggio più basso sarà la classe meno probabile.
Questa tecnica consente di trascurare errori commessi da singole tassonomie. Se una o più tassonomie commettono errori nella classificazione, le altre tassonomie, se concordano sulla classificazione, possono imporsi sulla decisione, correggendo l'errore. Il metodo funziona poiché le tassonomie forniscono un voto per ciascuna classe, che è indipendente dalle altre tassonomie, pertanto se la minoranza di queste non descrive accuratamente determinate classi, i loro voti verranno trascurati.

La foresta di tassonomie può utilizzare i voti di ciascuna delle tecniche presentate in questo capitolo, sia i voti dell'albero di decisione binario (sezione 4.1) che quelli della classificazione One-vs-All (che abbiamo chiamato classificazione delle foglie in sezione 4.2). Ma i voti che, utilizzati dalla foresta, forniscono i risultati migliori sono quelli generati dal modello RDT (sezione 4.3). Questo perché il modello di acceptance genera voti maggiormente distribuiti sulle classi limitrofe[6] nel caso di indecisione. Tale elasticità e distribuzione di voto, combinata con le decisioni delle altre tassonomie, consente una miglior valutazione, ed un sistema di voto collettivo più preciso.

[6]Ovvero quelle classi che, per una tassonomia, presentano caratteristiche visive strettamente collegate tra loro.

Parte III

Risultati

Capitolo 5

Database

Il database utilizzato per gli esperimenti è Animals with Attributes [17]. AwA fornisce una piattaforma per il benchmark di algoritmi di classificazione. Consiste di 30475 immagini di 50 categorie di animali, con sei descrittori di feature pre-estratti per ciascuna immagine. A causa delle eccessive dimensioni del database, viene distribuito in tre forme:

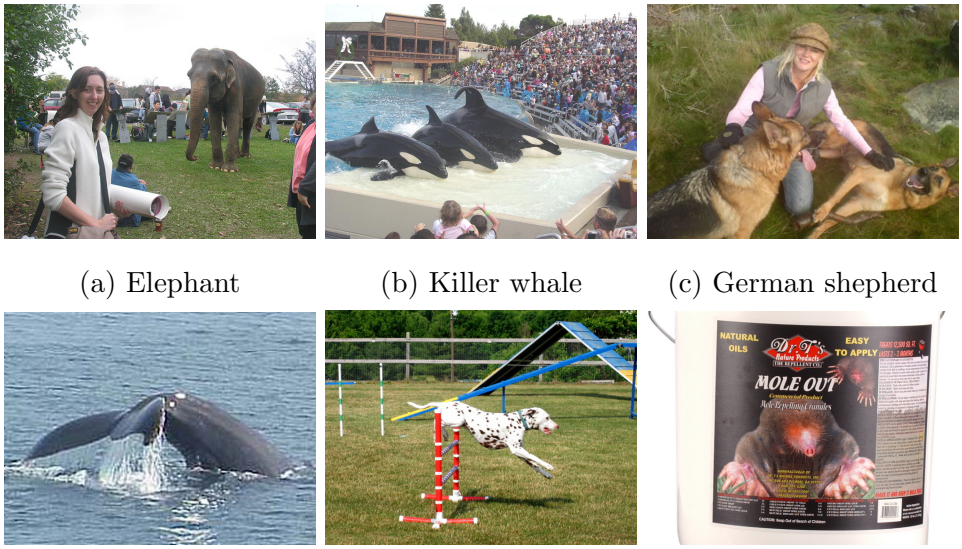
AwA-4 contenente immagini di quattro diverse categorie (dalmatian, leopard, siamese cat, wolf).

AwA-10 contenente immagini di dieci diverse categorie (chimpanzee, hippopotamus, leopard, pig, rat, giant panda, humpback whale, persian cat, raccoon, seal).

AwA-All contiene tutte le 50 categorie che compongono il database.

I database è stato scelto poiché utilizzato anche nell'articolo di riferimento [2], e perché fornisce una versione pre-estratta di sei differenti feature per ciascuna immagine, consentendoci di concentrarci direttamente sulla creazione ed addestramento delle tassonomie, trascurando la scelta, la ricerca e l'estrazione delle features.

AwA è un database di tipo *fine-grained*, infatti presenta razze diverse, di uno stesso animale, come classi differenti (i.e. persian cat e siamese cat). Inoltre le immagini che lo compongono non sono stilizzate, ma sono scatti fotografici raffiguranti scene di vita reale, pertanto sono fortemente affette da rumore e confusione (alcuni esempi sono presentati in Figura 5.1). Pertanto questo database risulta essere una vera sfida per la classificazione.



(a) Elephant (b) Killer whale (c) German shepherd

(d) Blue Whale (e) Dalmatian (f) Mole

Figura 5.1: Esempi di immagini che compongono il database AwA-All.

5.1 Features

Le features fornite con il database sono sei: *RGB color histograms* (CQ), *local self-similarity histograms* (LSS) [18], *Pyramid Histogram of Oriented Gradients* (PHOG) [19], *Scale-Invariant Feature Transform* (SIFT) [20], rg-SIFT [21], *Speeded Up Robust Features* (SURF) [22].

L'istogramma del colore (CQ-hist) è una rappresentazione della distribuzione dei colori in un'immagine. Per un'immagine digitale, l'istogramma del colore rappresenta il numero di pixel che ha colore uguale ad un insieme quantizzato di possibili colori presenti nell'immagine stessa [3]. Tale istogramma può essere costruito sia sullo spazio RGB (Red Green Blue) che su quello HSV (Hue Saturation Brightness). Per il database AwA l'istogramma è stato creato sullo spazio di colore RGB.

LSS, o *Local Self-Similarity*, è un descrittore che misura la somiglianza tra due entità visive, basandosi sulla corrispondenza delle auto-somiglianze[1] interne [18]. Ciò si basa sul fatto che la correlazione tra due immagini dipende dallo schema interno delle auto-somiglianze locali (fino ad un certo fattore di distorsione), nonostante queste auto-somiglianze generino modelli leggermente differenti per ciascuna immagine. Misurando queste auto-somiglianze è possibile creare correlazioni tra immagini diverse.

PHOG è l'acronimo di *Pyramid Histogram of Oriented Gradients*. È la versione piramidale, ovvero che utilizza la piramide spaziale, di HOG. L'istogramma dei gradienti orientati è un descrittore che conta il numero delle orientazioni dei gradienti in una porzione di immagine [3]. Queste porzioni sono generate utilizzando una griglia densa di celle di dimensione uniforme.

Gli *Scale-Invariant Feature Transform* (SIFT) comprendono sia il rilevatore che il descrittore di features. Il rilevatore estrae da un'immagine un numero di frame (o regioni) coerenti per determinate caratteristiche, come variazioni

[1] Un oggetto si dice *auto-simile* se è esattamente o approssimativamente simile ad una parte di se stesso.

di illuminazione, punto di vista o altre caratteristiche visive. Il descrittore, successivamente, associa a ciascuna regione una firma che la identifichi in modo compatto e robusto.
Il descrittore RGSIFT, sfrutta la tecnica presentata da SIFT utilizzando i colori, per incrementare l'invarianza alla luminosità e il potere discriminante del descrittore.

Infine SURF, o *Speeded Up Robust Features*, è un rilevatore robusto di feature locali. Parzialmente ispirato a SIFT, SURF è più robusto e veloce. È basato sulle risposte della forma d'onda di Haar 2D e fa un uso efficiente di immagini integrali.

I vettori di features CQ e PHOG sono estratti separatamente per tutte le 21 celle di 3 livelli della piramide spaziale[2]. Per ciascuna cella vengono estratte e concatenate istogrammi di colore a 128 dimensioni, per formare un vettore di features da 2688 dimensioni. Per PHOG, il procedimento è il medesimo, ma si utilizzano istogrammi base da 12 dimensioni. Per quanto riguarda i vettori di features degli altri descrittori, questi sono istogrammi da 2000 *bag of visual words*.
Come è già stato detto nel capitolo precedente, ciascun descrittore rappresenta in modo diverso la stessa immagine, fornisci un punto di vista, puramente visuale, dell'immagine.

Per la valutazione della tecnica qui presentata si fa riferimento ad AwA-10, il

[2]Quella della piramide spaziale (spatial pyramid) è una tecniche che prevedere il partizionamento dell'immagine in sotto-regioni sempre più piccole e con una descrizione dei dettagli maggiore, per le quali si calcolano le feature localmente al loro interno.

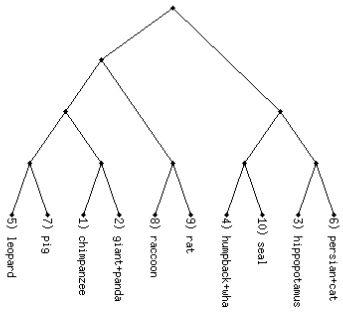
quale è composto da 6180 immagini, distribuite tra le classi precedentemente elencate.

Capitolo 6

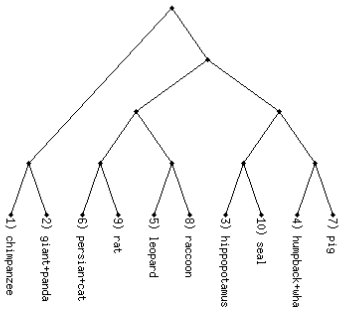
Analisi dei Risultati

In questo capitolo verranno presentati i risultati ottenuti sul database di riferimento (Capitolo 5) mostrando e confrontando gli output delle tecniche presentate nel Capitolo 4.

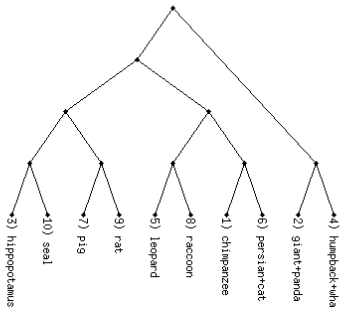
Gli esperimenti sono stati eseguiti con 30 immagini per l'insieme di train, 30 per il validation e infine 30 per il test. I risultati presentati sono stati calcolati come la media dei singoli risultati ottenuti su 5 differenti split del dataset. Per ciascuno split vengono generate nuovamente le tassonomie nel modo non supervisionato indicato in sezione 3.1, ed addestrate di conseguenza con le immagini appartenenti all'insieme di train. Come si può verificare in Figura 6.1, la struttura della tassonomia non dipende esclusivamente dal tipo di descrittore utilizzato per generarla (vedi Figura 4.5), ma anche dall'insieme di train e validation utilizzato. Utilizzando un numero di immagini relativamente basso, la confusione tra le classi, già alta a causa della natura delle immagini presenti nel dataset originale, risulterà molto elevata. Pertanto si è verificato che la precisione di una tassonomia nella classificazione, è maggiore se si utilizza il nuovo insieme di train/validation non solo per l'addestramento, ma anche per la creazione della tassonomia stessa. Questo perché, creare



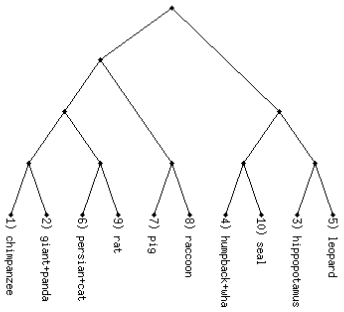
(a) 1° split



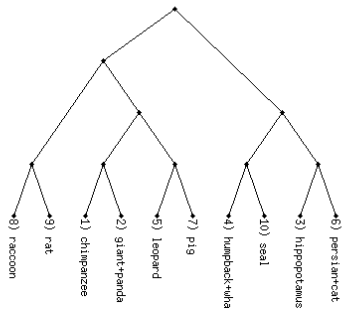
(b) 2° split



(c) 3° split



(d) 4° split



(e) 5° split

Figura 6.1: Le immagini mostrano le differenti tassonomie, create dal descrittore SIFT, per ciascuno dei cinque split.

una nuova tassonomia, per ciascuno split, consente di generare una struttura che consente la distinzione tra le classi che, per quel determinato split e quel determinato descrittore, presentano la maggior confusione. In questo modo l'addestramento degli SVM binari sarà maggiormente mirato alla risoluzione di questa confusione, con un notevole incremento dell'accuratezza.

6.1 Analisi delle Tassonomie

Analizzare i risultati della classificazione dell'albero di decisione binario, ovvero quello che utilizza il solo modello di decisione $\mathcal{D}$, per ciascuna tassonomia consente di effettuare una prima valutazione sulla struttura della tassonomia. Si può verificare che l'utilizzo di tassonomie, oltre a diminuire notevolmente i costi computazionali, in quanto il numero di confronti con gli SVM è proporzionale a $\lceil \log_2 N \rceil$, presenta un accuratezza che è comparabile a quella fornita dal metodo standard One-vs-All. Questo confronto ha il solo scopo di valutare se la gerarchia definita dalla tassonomia creata automaticamente è buona o meno.

Se evidenziamo solo i risultati della classificazione di tipo gerarchico in Tabella 6.1, si osserva come feature diverse forniscono risultati molto diversi. Infatti si nota che, per questo database, alcune feature, come SURF e SIFT, sono migliori di altre, come ad esempio PHOG. Questo fatto prova che descrittori diversi si comportano in modo molto diverso sulle medesime immagini, il che è uno svantaggio se le tassonomie sono prese singolarmente, ma un potenziale vantaggio in una visione più ampia, se si utilizza il concetto di foresta di tassonomie. Questo concetto verrà approfondito successivamente quando analizzeremo i risultati della foresta di tassonomie (sezione 6.3).

Descrittore	**AwA-4**		**AwA-10**	
	H	**L**	**H**	**L**
CQ-hist	37,83%	39,83%	26,93%	29,26%
LSS-hist	48,83%	47,83%	29,13%	29,20%
PHOG-hist	36,00%	35,83%	21,67%	19,80%
RGSIFT-hist	49,33%	44,33%	31,33%	22,80%
SIFT-hist	51,50%	51,99%	30,60%	33,46%
SURF-hist	58,17%	59,33%	36,67%	38,86%

Tabella 6.1: Confronto tra i risultati ottenuti misurando l'accuratezza dell'albero di decisione binaria standard **H** (sezione 4.1) con la classificazione delle foglie **L** (sezione 4.2). I risultati sono stati calcolati singolarmente per ciascuna tassonomia creata ed addestrata con il rispettivo descrittore di features.

Un altra possibile analisi, è quanta informazione aggiuntiva possono dare i voti degli SVM calcolati dalle foglie, ovvero la classificazione One-vs-All data dalle foglie della tassonomia descritta in sezione 4.2, rispetto ai voti dati dalla gerarchia. Si nota che l'aggiunta dei voti degli SVM delle foglie a quelli dati dalla gerarchia, non influisce positivamente, almeno non sufficientemente da giustificare il numero di confronti aggiuntivi con gli SVM da fare per fornire i voti anche da parte delle foglie della tassonomia.

Infine si confrontano le tassonomie create in modo supervisionato nel lavoro della professoressa Grauman [2] con le nostre. Infatti se i risultati delle tassonomie semantiche loro e quelle automatiche qui presentate, sono comparabili, sarà poi possibile fare un confronto tra la tecnica di Semantic Kernel Forest con quelle presentate in sezione 4.3 e sezione 4.4.

Descrittore	**AwA-4**		**AwA-10**	
	H	**H+L**	**H**	**H+L**
CQ-hist	37,83%	36,83%	26,93%	29,73%
LSS-hist	48,83%	49,66%	29,13%	29,73%
PHOG-hist	36,00%	36,33%	21,67%	21,46%
RGSIFT-hist	49,33%	47,33%	31,33%	32,40%
SIFT-hist	51,50%	52,33%	30,60%	31,33%
SURF-hist	58,17%	58,83%	36,67%	37,13%

Tabella 6.2: Confronto tra l'accuratezza data dai voti della gerarchia (**H**) e quelli dati dalla somma tra quelli della gerarchia e i voti dati dagli SVM delle foglie (**H**+**L**).

In Tabella 6.3 è mostrato il confronto delle medie delle quattro tassonomie utilizzate in [2], e le sei create automaticamente con il metodo da noi presentato. Si nota che la differenza, sia per AwA-4 che per AwA-10, si attesta intorno al punto percentuale. L'assenza di supervisione e l'utilizzo di tassonomie non propriamente adatte alla classificazione per questo tipo di

	AwA-4	**AwA-10**
Non-Supervised Tax	46,94%	29,39%
Supervised Semantic Tax	47,67%	30,80%

Tabella 6.3: Confronto tra le medie delle accuratezze dei risultati ottenuti con le tassonomie semantiche utilizzate dalla professoressa Grauman in [2], e quelle da noi create.

database[1], giustifica questa leggera perdita di accuratezza. Infatti i risultati sono assolutamente paragonabili a quelli ottenuti dalla Grauman nonostante le tassonomie vengano create in modo totalmente automatico.
Poiché tale confronto ha portato esiti positivi, si può affermare che è possibile comparare i due diversi approcci, consentendoci di proseguire con la sperimentazione ed il confronto con il suddetto articolo.

[1]Alcuni dei descrittori distribuiti insieme al database non sono considerabili ottimali per la distinzione delle varie classi di animali qui presenti.

6.2 Analisi Modello RDT

Per analizzare il modello RDT, Relaxed Decision Tree, abbiamo studiato il suo comportamento nell'assegnazione e la distribuzione dei voti in sottosezione 4.3.1. Adesso valuteremo la sua accuratezza complessiva, come in precedenza per la classificazione gerarchica (sezione 6.1), separando i voti per ciascuna tassonomia.

Nei precedenti capitoli abbiamo introdotto alcuni concetti, come quello di *acceptance*, che prevede una sorta di verifica della decisione presa all'interno della gerarchia da parte dei componenti in causa. Abbiamo introdotto il modello di acceptance, ovvero un modello addestrato per distinguere, all'interno di una tassonomia, non solo le classi descritte da un nodo e quelle descritte dal suo fratello, ma anche da quelle dello zio. Ed infine abbiamo spiegato come questo modello viene utilizzato per classificare gli esempi di test.

Inizialmente era stata implementata una tecnica di backtracking effettivo[2], ovvero nel duplice modello di decisione e acceptance ($\mathcal{D} + \mathcal{A}$), se i figli rifiutavano entrambi la decisione del padre, di conseguenza la tassonomia veniva percorsa all'indietro, selezionando in seconda istanza il nodo zio come step successivo della classificazione. Questa tecnica avrebbe, in teoria, dovuto migliorare le accuratezze delle singole tassonomie, ma questo non si è verificato, anzi questo spesso portava a problemi come loop nelle decisioni. Pertanto si è preferito optare per un approccio più soft, che garantisse comunque le ottime prestazioni del metodo se utilizzato per la foresta di tassonomie.

Valutare i risultati su AwA-4 non presenterebbe risultati interessanti, in quanto le tassonomie, per quel dataset, non sono sufficientemente articolate da giustificare l'utilizzo del modello di acceptance.

[2]Diverso dal *soft backtracking* implementato e spiegato in sezione 4.3

Descrittore	**AwA-10**	
	L	**RDT**
CQ-hist	29,73%	26,93%
LSS-hist	29,73%	28,06%
PHOG-hist	21,46%	22,13%
RGSIFT-hist	22,80%	16,40%
SIFT-hist	33,46%	29,60%
SURF-hist	38,86%	34,93%

Tabella 6.4: Confronto tra i risultati ottenuti con la tecnica del Relaxed Decision Tree (**RDT**) e la classificazione delle foglie (**L**), calcolati sincolarmente per ciascuna tassonomia.

Come si può vedere in Tabella 6.4 i risultati ottenuti sono tutti inferiori o uguali rispetto alla classificazione One-vs-All delle foglie delle tassonomie. Questo decremento della precisione è dovuto all'elasticità nella distribuzione dei voti, introdotta da RDT. Essendo il database affetto da confusione molto elevata, la distribuzione dei voti, per casi di classificazione non sicura può portare alla diminuzione dell'accuratezza.
Come, però, è già stato detto in sezione 4.3, il modello del Relaxed Decision Tree, è stato ideato per essere eseguito in una foresta di più tassonomie, piuttosto che per migliorare le prestazioni della tassonomia stessa. Il concetto è quello di rilassare le decisioni singolarmente per ottenere dei vantaggi (come vedremo successivamente utilizzando la foresta di tassonomie sezione 6.3) nel risultato collettivo finale.

Una considerazione interessante da fare è sul comportamento dell'RDT nel caso in cui l'esplorazione dell'albero risulta errata. In Tabella 6.5 si mostra

	$\mathcal{D}_P$	$\mathcal{A}_L$	$\mathcal{A}_R$	**Totale**
Predicted	$+0,17$	$-0,81$	$+1,10$	$+0,46$
Sibling	$-0,17$	$+0,81$	$-1,10$	$-0,46$
Uncle	np	$+0,81$	$+1,10$	$+1,91$

Tabella 6.5: Esempio di comportamento del modello RDT in caso di decisione errata agli step precedenti. Sulle righe sono presenti i nodi interessati, mentre sulle colonne sono riportati i voti dei modelli. Immagine di riferimento leopard_0022.

come la tecnica di *soft backtracking* da noi implementata possa correggere gli eventuali errori presentatisi negli step precedenti.
Supponiamo che ci sia stato un errore nel selezionare un nodo all'interno della tassonomia $\mathcal{T}_f$, e chiamiamo questo nodo padre indicandolo con P. Essendo la classe dell'immagine di test non presente nel modello di decisione $\mathcal{D}_P$ del padre, la sua decisione risulta relativamente bassa $|0,17|$. Per l'esempio in questione (immagine del dataset AwA-10 leopard_0022) il nodo predetto dal padre[3] è il figlio sinistro che indicheremo con L. Il modello di *acceptance* $\mathcal{A}_L$ rifiuta la decisione del padre, dando voto negativo per se stesso, e voto positivo per il nodo *sibling* (l'altro figlio del padre) e per il nodo *uncle* (ovvero lo zio del nodo predetto). Anche il modello di *acceptance* del figlio destro $\mathcal{A}_R$ (ovvero il sibling) non riconoscerà come propria l'immagine di test. Pertanto il suo voto è negativo per se stesso e positivo per il nodo predetto dal padre, anche se non corretto, e per il nodo zio. Sommando i risultati, visibili in Tabella 6.5, si vede come il voto assegnato al nodo zio sia superiore ad entrambi i nodi presenti nel percorso intrapreso lungo la tassonomia. Questo voto verrà propagato fino a ciascuna foglia sottostante al nodo zio confe-

[3]Qualsiasi sua decisione sarebbe stata errata.

	$\mathcal{D}_P$	$\mathcal{A}_L$	$\mathcal{A}_R$	**Totale**
Predicted	$+0,48$	$-0,77$	$-0,98$	$-1,27$
Sibling	$-0,48$	$+0,77$	$+0,98$	$+1,27$
Uncle	np	$+0,77$	$-0,98$	$-0,21$

Tabella 6.6: Esempio di comportamento del modello RDT in caso di errore nel modello di decisione del padre $\mathcal{D}_P$. Sulle righe sono presenti i nodi interessati, mentre sulle colonne sono riportati i voti dei modelli. Immagine di riferimento persian+cat_0008.

rendogli un voto relativamente più alto in contrapposizione con la decisione presa dai modelli di decisione $\mathcal{D}$ per esplorare la tassonomia $\mathcal{T}_f$.
La stessa considerazione si può fare nel caso in cui l'errore sia commesso dal modello di decisione del nodo padre $\mathcal{D}_P$. Ovvero se il figlio predetto è il sinistro L, quando la decisione corretta sarebbe stata di percorrere il percorso verso il nodo figlio destro R. In Tabella 6.6 si nota come i modelli di *acceptance* $\mathcal{A}_L$ e $\mathcal{A}_R$ correggano l'errore di decisione del padre assegnando al nodo figlio destro un voto maggiore rispetto al nodo figlio predetto. Anche in questo caso, propagando i voti dati da questa iterazione verso le foglie della tassonomia, l'RDT terrà conto dell'errore verificatosi, penalizzando le foglie che si trovano lungo il percorso intrapreso.

6.3 Analisi Foresta di Tassonomie

Analizzeremo di seguito i risultati ottenuti utilizzando le tassonomie, non singolarmente come fatto in sezione 6.1 e sezione 6.2, ma utilizzandole in una collettività che per comodità è stata chiamata foresta di tassonomie.
Come già spiegato in sezione 4.4, ciascuna tassonomia descrive un differente punto di vista per ciascuna immagine, pertanto ognuna voterà in modo differenze per tutte le classi del dataset. Utilizzare i voti di tutte le tassonomie consentirà al sistema di classificazione, di avere più informazioni per ciascuna immagine di test, aumentando così la precisione nell'assegnare la giusta categoria.
Analizzeremo i risultati per AwA-10 mostrando l'incremento di accuratezza, sia confrontando quelli ottenuti utilizzando il modello dell'albero binario di ricerca, sia con RDT. Successivamente saranno paragonati i risultati ottenuti con quelli da "Semantic Kernel Forest" [2].
Come si può vedere dalla Tabella 6.7, confrontandola con Tabella 6.1 e Tabella 6.4, l'utilizzo del voto condiviso delle sei tassonomie migliora notevolmente la media dei risultati delle singole. Per il metodo che utilizza gli alberi di decisione binari, il risultato della foresta è leggermente migliore rispetto al miglior risultato ottenuto dalle sei singole tassonomie, in particolare la tassonomia creata ed addestrata con SURF raggiunge il $36,67\%$ (Tabella 6.1). Se, invece, si confrontano i risultati ottenuti per il modello RDT, si vede che si passa da una media ed un massimo rispettivamente di $26,34\%$ e $34,93\%$, al risultato della foresta che raggiunge il $39,88\%$ di accuratezza.
Analizzando questi risultati si evince che, nonostante RDT presentasse inizialmente risultati peggiori, il voto collettivo della foresta premia il modello del Relaxed Decision Tree. La distribuzione dei voti perpetrata da RDT, se pur peggiorando le accuratezze di classificazione delle singole tassonomie,

Metodo	**Accuratezza**
Semantic Kernel Forest	35,87%
Late Fusion	35,16%
Foresta H	36,96%
Foresta RDT	**39,88**%

Tabella 6.7: Risultati ottenuti utilizzando la foresta di tassonomie, per i 3 metodi di classificazione spiegati nel Capitolo 4 e confrontati con quelli rilevati nell'articolo della professoressa Grauman [2].

migliora notevolmente le prestazioni per la foresta di tassonomie. Questo è possibile poiché la formula del soft backtracking descritto in sottosezione 4.3.1, garantisce in caso di errore nell'esplorazione della tassonomia, di valutare alcune delle classi scartate, come possibili candidate. Questo consente, nel caso di errore da parte di una minoranza delle tassonomie, la sua correzione con il voto delle restanti. In Figura 6.2, si nota come la gestione del voto della foresta di tassonomie, sia robusta agli errori delle singole. In particolare nell'esempio solo le tassonomie create ed addestrate con i descrittori LSS, PHOG e RGSIFT restituiscano il massimo punteggio per la classe corretta. Si evidenzia come, tra le tassonomie che classificano erratamente l'istanza, quella creata ed addestrata da CQ fornisca addirittura un voto negativo. Altra considerazione da fare è come l'utilizzo della foresta di tassonomie sia migliore rispetto alla selezione della classe che presenta il valore maggiore tra tutti i voti delle tassonomie. Infatti, in questo caso il valore massimo è assegnato ad una categoria errata, ma poiché le altre tassonomie presentano voti molto bassi, se non addirittura negativi, il voto dato della foresta esclude la classe non corretta.

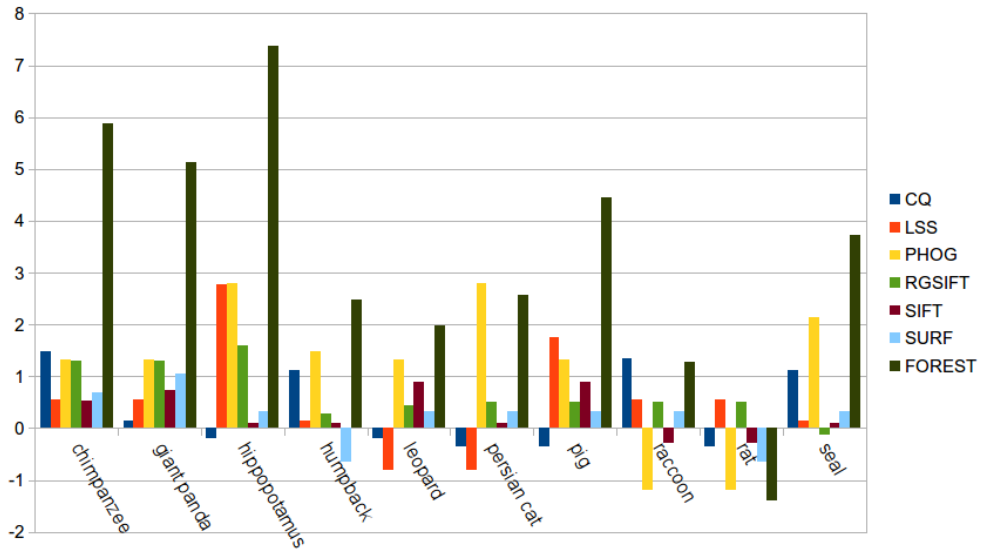


Figura 6.2: Illustrazione grafica del comportamento della foresta in caso di errore da parte di alcune tassonomie, l'esempio riguarda l'immagine hippopotamus_0606.

Si può quindi affermare che la foresta è robusta agli errori delle singole tassonomie, e che il voto collettivo delle tassonomie riesce a risolvere situazioni di decisioni critiche.

Infine sono stati confrontati i risultati da noi ottenuti con quello presentato nell'articolo della professoressa Grauman [2]. Come da Tabella 6.7, i risultati del modello RDT sono nettamente superiori rispetto a quelli ottenuti dalla Semantic Kernel Forest.

6.3.1 Analisi del Comportamento della Foresta RDT

Soddisfatti dei risultati ottenuti e confrontati con quelli esistenti, abbiamo cercato di analizzare più nel dettaglio il comportamento della foresta RDT,

	6 Tassonomie	**5 Tassonomie**		**4 Tassonomie**	
	Acc	**Acc**	**Inc**	**Acc**	**Inc**
Media tax	25,49%	28,33%	+1,99%	29,88%	+3,54%
Foresta RDT	39,88%	39,98%	+0,10%	40,40%	+0,52%

Tabella 6.8: Confronto tra le medie delle accuratezze calcolate sulle singole tassonomie e la foresta RDT, ed i rispettivi incrementi (Inc), al variare del numero di tassonomie utilizzate, escludendo incrementalmente le tassonomie che presentavano i risultati peggiori.

valutando il suo comportamento al variare del numero di tassonomie.
Se si guarda la Tabella 6.4, si può notare come alcune delle tassonomie utilizzate siano nettamente peggiori rispetto ad altre. Pertanto sono stati effettuati alcuni esperimenti che, utilizzando sempre la foresta RDT, escludessero le tassonomie meno performanti. L'utilizzo della foresta RDT è robusta in presenza di tassonomie molto poco performanti. Infatti, come si vedere in Tabella 6.8, mentre le medie tra le accuratezze delle singole tassonomie, migliorano notevolmente, i risultati per le foreste, subiscono variazioni minime. In particolare, analizzando i risultati del modello RDT per le singole tassonomie, si utilizzano le cinque e le quattro migliori tassonomie, eliminando rispettivamente RGSIFT-hist e PHOG-hist. Mentre le medie mostrano un incremento del $3,54\%$, la foresta RDT guadagna appena mezzo punto percentuale. Questo fenomeno è giustificato dal fatto che il modello RDT, in caso di forte indecisione, distribuendo notevolmente il voto sulle classi, rende i valori pressoché uniformi, facendoli diventare insignificanti nell'analisi della foresta.
Abbiamo inoltre provato ad addestrate le tassonomie create da un singolo descrittore, con tutti i descrittori disponibili, ovvero, utilizzando una unica

	Accuratezza
Foresta CQ	32,60%
Foresta LSS	39,08%
Foresta PHOG	37,68%
Foresta RGSIFT	30,78%
Foresta SIFT	39,14%
Foresta SURF	37,66%
Foresta Random	31,26%
Foresta RDT Standard	**39,88%**

Tabella 6.9: Confronto tra le accuratezze date dalle foreste distinguendo tra: foreste composte da una singola tassonomia (creata dal descrittore indicato) addestrata con descrittori diversi, foresta composta di tassonomie casuali addestrate con i sei descrittori, e infine la foresta RDT standard.

tassonomia (creata da un descrittore qualsiasi) si addestrano gli SVM con tutti i descrittori forniti dal database. Questo ha consentito di valutare la costruzione delle tassonomie per ciascun descrittore indipendentemente dal descrittore utilizzato per la sua creazione.
Per ogni tassonomia, e per ogni immagine di test, vengono esplorati sei differenti percorsi, dipendenti dal descrittore utilizzato per l'addestramento della tassonomia. Questo non può essere considerata una vera e propria foresta di tassonomie, in quanto la struttura dell'albero rimane la medesima, varia solamente il percorso usato per la classificazione.
Ciascuna foresta crea una singola tassonomia a partire da un descrittore, e la addestra con tutti i descrittori, i voti forniti dai sei percorsi esplorati vengono valutati come in sezione 6.3. In questo modo è possibile effettuare una valutazione sulla costruzione delle tassonomie. Si nota come, nonostante il

miglior descrittore per la classificazione era il SURF, le tassonomie migliori vengono generate a partire dal descrittore LSS e SIFT, mentre RGSIFT si dimostra il descrittore con le peggiori performance sia nella creazione della tassonomia che nella classificazione.

Oltre a questo è stata anche testata una foresta composta da tassonomie le cui strutture sono state generate in modo casuale, ed addestrate con tutti i descrittori.
I risultati in Tabella 6.9 mostrano il comportamento delle differenti foreste. Si nota che il risultato migliore rimane quello della foresta RDT standard, nella quale ciascuna tassonomia viene creata ed addestrata con il medesimo descrittore. Poiché le tassonomie così create ed addestrate possono essere considerate un sotto-insieme di tutte le tassonomie possibili, confrontando questo risultato con la foresta di tassonomie casuali, si evince che il miglior risultato è ottenuto se le tassonomie sono create su misura per ciascun descrittore.

Per concludere è stato effettuato un test utilizzando tutte le 36 diverse tassonomie viste in Tabella 6.9, ovvero le sei tassonomie create da ciascun descrittore ed addestrate ognuna con tutti i descrittori. Anche in questo caso (Tabella 6.10) si vede come il risultato, nonostante utilizzi un numero superiore di alberi sia leggermente inferiore ai risultati dalla foresta RDT standard. Si può concludere che la creazione di della tassonomia in funzione del descrittore utilizzato per l'addestramento, è la combinazione migliore per massimizzare l'accuratezza nell'utilizzo della foresta con il modello dei Relaxed Decision Trees.

	Accuratezza
Foresta 36 tax	38,34%
Foresta RDT Standard	39,88%

Tabella 6.10: Risultato ottenuto dalla foresta composta dalle 36 tassonomie create per ciascun descrittore ed addestrate con tutti i descrittori del dataset.

6.3.2 Esperimenti su AwA-All

Presentiamo adesso i risultati ottenuti sul dataset AwA-All, composto da 50 classi e più di 30 000 immagini. Come già esposto nel Capitolo 5, il dataset AwA-All, oltre ad essere composto da immagini prese dalla vita reale, quindi raffiguranti scene complesse e spesso poco chiare, presenta molte categorie facilmente confondibili tra loro anche per l'uomo (Figura 6.3). Nella valutazione dei risultati è dunque da considerare anche questo aspetto del dataset.

In Tabella 6.11 si può verificare come i risultati ottenuti sul dataset AwA-All, siano paragonabili a quelli ottenuti precedentemente per AwA-10. Ovvero come i rapporti tra le varie tecniche rimangano stabili anche con l'aumentare del numero, e della densità di classi.

(a) Hamster (b) Mouse (c) Rat

Figura 6.3: Esempi di tre classi facilmente confondibili anche per l'uomo, in particolare sono raffigurate la classe criceto, topo e ratto.

Descrittore	**AwA-10**			**AwA-All**		
	H	**L**	**RDT**	**H**	**L**	**RDT**
CQ-hist	26,93%	29,73%	26,93%	7,28%	9,20%	6,88%
LSS-hist	29,13%	29,73%	28,06%	7,87%	9,33%	7,25%
PHOG-hist	21,67%	21,46%	22,13%	5,12%	6,32%	4,71%
RGSIFT-hist	31,33%	22,80%	16,40%	8,96%	9,69%	2,50%
SIFT-hist	30,60%	33,46%	29,60%	7,95%	9,57%	7,39%
SURF-hist	36,67%	38,86%	34,93%	9,48%	13,42%	9,43%

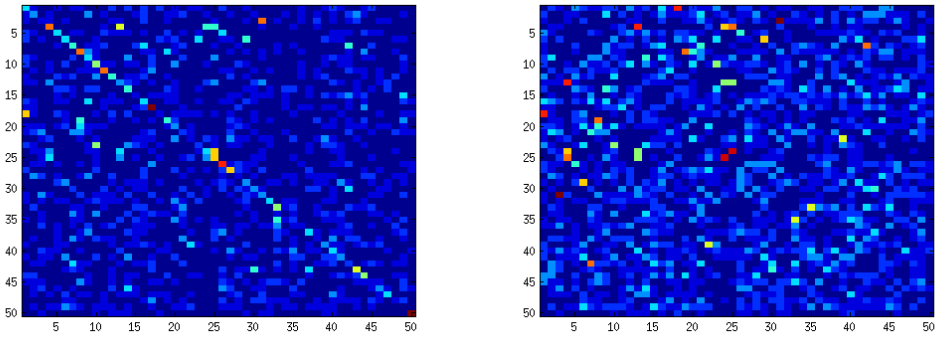
Tabella 6.11: Confronto tra i risultati ottenuti misurando l'accuratezza su AwA-10 e AwA-All, distinguendo tra albero di decisione binario (**H**), OvA sulle foglie (**L**) e Relaxed Decision Tree (**RDT**).

Metodo	**Accuratezza**
Foresta H	8,66%
Foresta RDT	10,50%

Tabella 6.12: Confronto tra i risultati, per AwA-All, della foresta di tassonomie utilizzando sia il metodo gerarchico (H) che quello RDT.

Infine presentiamo i risultati con l'utilizzo della foresta di tassonomie, mostrando il confronto tra la foresta calcolata con l'albero di decisione binario prima, e con Relaxed Decision Trees poi. Come per AwA-10 i risultati mostrano un sensibili miglioramento utilizzando il modello di acceptance presente in RDT.

Purtroppo non esistono nello stato dell'arte risultati per il dataset AwA-All confrontabili con quelli da noi ottenuti, pertanto l'unico confronto che si può fare è con il metodo più comune One-vs-All che, analizzando la media per i sei

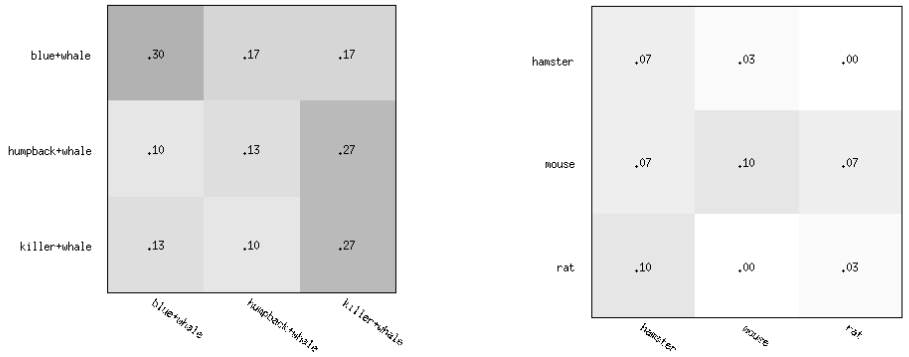


(a) Matrice di confusione $\mathcal{C}$ (b) Matrice di mutua confusione $\mathcal{M}$

Figura 6.4: Rappresentazione grafica delle due matrici $\mathcal{C}$ e $\mathcal{M}$. Nella prima si evidenzia come, eccetto siano presenti alti valori anche esternamente alla diagonale. Nella seconda i valori massimi evidenziano le classi maggiormente confondibili tra loro.

descrittori testati è di $9,59\%$, mentre sommando i voti dei modelli One-vs-All di tutte le tassonomie si ottiene un'accuratezza del $12,92\%$. Si nota come la *late fusion* tra i modelli One-vs-All delle tassonomie, è migliore rispetto alla foresta RDT. Questo è un problema che si è verificato in questo dataset, e non nei precedenti, perché presenta un'elevata specificità (vedi Figura 6.3), ovvero in AwA-All, non solo si distinguono molteplici tipi di animali, ma per ciascuno sono presenti più razze.

In Figura 6.4 si evidenzia questo problema. Come da norma la matrice di confusione $\mathcal{C}$ (Figura 6.4a), presenta valori relativamente alti sulla diagonale. Si nota però che anche in zone della matrice che non coincidono con la diagonale, sono presenti valori abbastanza alti. Se si analizza la matrice di mutua confusione tra le classi (Figura 6.4b), e se ne selezionano i massimi si nota come essi coincidano con le classi che presentano il numero maggiore di differenziazioni tra razze. Ad esempio si nota come balene (*blue whale*, *hum-*



(a) Balene (b) Topi

Figura 6.5: Rappresentazione grafica delle matrici di confusione per le due classi che presentano i valori più alti esternamente alla diagonale di $\mathcal{C}$.

pback whale, *killer whale*), topi (*hamster*, *mouse*, *rat*) e gatti (*persian cat*, *siamese cat*) che presentano un'elevata specificità, presentino nella matrice $\mathcal{M}$ i valori più alti all'esterno della diagonale.
Infine dobbiamo considerare che eseguire la *late fusion* tra i risultati dei modelli One-vs-All di tutte le tassonomie, per dataset grandi come AwA-All, risulta essere computazionalmente molto più oneroso rispetto alla foresta RDT, sia in fase di addestramento, dato che ciascun One-vs-All SVM deve essere addestrato con le immagini di tutte le classi in questione, sia in fase di test poiché i confronti tra l'immagine e gli SVM è proporzionale al numero delle classi N, mentre nel nostro caso dell'RDT è $3 \cdot \lceil \log_2 N \rceil$.

6.4 Considerazione su Tempi e Costi

Considerato che la classificazione che sfrutta strutture gerarchiche serve per diminuire il carico computazionale, è necessario presentare anche una valutazione sui tempi impiegati per la creazione e l'addestramento delle tassonomie,

	AwA-10	**AwA-All**
Late Fusion	$0,18$ s	$34,69$ s
Foresta H	$0,10$ s	$5,56$ s
Foresta RDT	$0,32$ s	$17,13$ s

Tabella 6.13: Tempi computazionali per i test su i due dataset AwA-10 e AwA-All, distinguendo tra le 3 tipologie di foreste calcolate.

e successivamente per i diversi tipi di classificazione utilizzati.
I tempi presentati di seguito si riferiscono all'esecuzione del programma su una macchina laptop, con processore Intel® Core™ i5 CPU M 460 @ 2.53GHz × 4, e 3,7 GB di memoria, lavorando su sistema operativo Linux Ubuntu 13.04, e si riferiscono all'esecuzione di un singolo esperimento nella forma descritta all'inizio del Capitolo 6. Si aggiunge inoltre che, per quanto riguarda l'addestramento, sono stati selezionati i migliori valori per per l'SVM, utilizzando la cross-validazione su 5 sottoinsiemi dell'originale insieme di training.
L'addestramento delle singole tassonomie è un procedimento offline, ovvero deve essere eseguito una sola volta. I tempi per AwA-10 per la creazione e l'addestramento sono di $17,21$ secondi. Mentre per il dataset AwA-All abbiamo fatto una distinzione tra l'addestramento delle tassonomie escludendo prima, e considerando poi, l'addestramento delle foglie con classificatori One-vs-All.
Nel primo caso i tempi sono di 16 minuti, mentre considerando l'addestramento delle 50 foglie[4] è di quasi 30 minuti. Questa separazione è stata fatta poiché il risultato fornito dalla foresta che utilizza i valori restituiti dagli SVM One-vs-All delle foglie per il calcolo combinato del voto finale, è puramente indicativo per la valutazione del risultato della foresta RDT.

[4]Il numero di foglie è pari al numero di classi presenti nel dataset.

Per quanto riguarda invece i test, il timing rispecchia la scalabilità della foresta RDT che, nonostante sia superiore alla foresta H (come si vede in Tabella 6.13), scalano notevolmente rispetto alla foresta che usa tutti i confronti One-vs-All di tutte le tassonomie. Paragonando infatti i tempi di AwA-10 e AwA-All, si nota che, all'aumentare del numero di classi, il divario tra i tempi delle due foreste aumenta a favore della foresta RDT.

Conclusioni

In questo lavoro di tesi è stato studiato il comportamento dell'uso delle tassonomie, combinato in una foresta di alberi di decisione. L'idea di non limitarsi all'utilizzo di una singola tassonomia per la classificazione delle immagini, ha portato un incremento delle prestazioni, poiché ciascuna tassonomia, sia essa costruita in modo supervisionato o non supervisionato, costituisce un diverso punto di osservazione dell'oggetto esaminato. Utilizzare più tassonomie si è dimostrato efficace, poiché l'amento di informazioni, dato dalla combinazione delle diverse tassonomie, aumenta le capacità di classificazione del sistema.
Il sistema di tassonomie è stato generato in modo completamente automatico, dimostrando come queste tassonomie, create a partire da features puramente visive, possono essere confrontate con quelle create in modo supervisionato (o semi-supervisionato) dall'uomo. La creazione di tassonomie generate analizzando la matrice di confusione per ciascun descrittore fornito dal database, ha consentito la creazione di una foresta di tassonomie che delinea le classi secondo i diversi modelli di rappresentazione forniti dai descrittori di features.
Inoltre è stato presentato un nuovo modello per l'esplorazione delle tassonomie chiamato Relaxed Decision Tree. RDT introduce un concetto di approvazione (*acceptance*) da parte dei nodi oggetti della decisione. Questo modello, oltre a fornire una valutazione sulla decisione presa dall'albero binario di

decisione, consente di correggere eventuali errori commessi nell'esplorazione della tassonomia. Questo introduce una forma di *soft backtracking* che non prevede di ripercorrere la gerarchia a ritroso, ma, nei casi necessari, di assegnare un voto positivo anche alle classi precedentemente scartate.
Infine sono stati presentati i primi risultati per il database AwA-All – composta da 50 classi di animali – ottenuti sfruttando il concetto di foresta di tassonomie.

Possibili sviluppi futuri potrebbero comprendere lo studio dei descrittori, al fine di massimizzare le prestazioni della foresta di tassonomie. Inoltre, sempre per quanto riguarda la creazione delle tassonomie, si può valutare altre tecniche per la formazione della struttura che definisce la tassonomia. La tecnica utilizzata in questo lavoro è di tipo Bottom-Up, ed è molto semplice ed intuitiva. L'utilizzo di tecniche più elaborate potrebbe accrescere il potere descrittivo di ciascuna tassonomia.
Il modello di acceptance (presentato in sottosezione 3.2.2), che al momento consente solo di effettuare un soft backtracking, potrebbe essere utilizzato anche per correggere effettivamente l'eventuale errore commesso. L'implementazione del backtracking, nella sua forma originale, potrebbe essere un interessante caso di studio.

Bibliografia

[1] E. Bart, M. Welling, and P. Perona, "Unsupervised organization of image collections: taxonomies and beyond," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2302–2315, 2011.

[2] S. J. Hwang, K. Grauman, and F. Sha, "Semantic kernel forests from multiple taxonomies," in *Advances in Neural Information Processing Systems*, pp. 1727–1735, 2012.

[3] "Wikipedia, the free encyclopedia."

[4] I. Biederman, "Recognition-by-components: a theory of human image understanding.," *Psychological review*, vol. 94, no. 2, p. 115, 1987.

[5] Vapnik, "The nature of statistical learning theory," 1999.

[6] G. Madzarov, D. Gjorgjevikj, I. Chorbev, *et al.*, "A multi-class svm classifier utilizing binary decision tree.," *Informatica (Slovenia)*, vol. 33, no. 2, pp. 225–233, 2009.

[7] J. Weston and C. Watkins, "Multi-class support vector machines," tech. rep., Citeseer, 1998.

[8] M. Arun Kumar and M. Gopal, "A hybrid svm based decision tree," *Pattern Recognition*, vol. 43, no. 12, pp. 3977–3987, 2010.

[9] B. Fei and J. Liu, "Binary tree of svm: a new fast multiclass training and classification algorithm," *Neural Networks, IEEE Transactions on*, vol. 17, no. 3, pp. 696–704, 2006.

[10] A. Bergamo and L. Torresani, "Meta-class features for large-scale object categorization on a budget," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3085–3092, IEEE, 2012.

[11] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[12] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei, "Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3450–3457, IEEE, 2012.

[13] C. Fang and L. Torresani, "Measuring image distances via embedding in a semantic manifold," in *Computer Vision–ECCV 2012*, pp. 402–415, Springer, 2012.

[14] T. Deselaers and V. Ferrari, "Visual and semantic similarity in imagenet," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1777–1784, IEEE, 2011.

[15] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[16] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in neural information processing systems*, pp. 1601–1608, 2004.

[17] "Animals with attributes, a dataset for attribute based classification," 2009.

[18] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[19] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 401–408, ACM, 2007.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] K. E. Van De Sande, T. Gevers, and C. G. Snoek, "Evaluating color descriptors for object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1582–1596, 2010.

[22] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.