POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE - DEI
Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione (D.M. 270/04)

Tesi di laurea in CALCOLO NUMERICO

ANALISI DI IMMAGINI CON RETI NEURALI CONVOLUZIONALI PER LA CLASSIFICAZIONE DEI CETACEI NEL GOLFO DI TARANTO

Relatore:
Chiar.mo Prof. Ing. Tiziano POLITI

Correlatore:
Dott. Ing. Vito RENÒ

Laureando:
Tommaso MONOPOLI
Matricola 568581

Anno Accademico 2018-2019

Sommario

Il sistema terrestre è sempre stato soggetto alle conseguenze delle attività umane, e la biodiversità degli ecosistemi acquatici e marini è fortemente a rischio. Diversi studi cercano di capire in che modo la perdita della biodiversità possa alterare l'integrità e il funzionamento di tali ecosistemi. Una risposta a questa domanda può essere ricercata negli studi effettuati sulla distribuzione e sullo stato di conservazione dei cetacei, oggetto di numerose ricerche negli ultimi anni.
Un'attività mirata alla raccolta di informazioni rilevanti allo studio dei cetacei è la *foto-identificazione degli individui* di una specie, che prevede il riconoscimento - automatico o manuale - di uno stesso individuo in diverse immagini collezionate nel tempo, mediante l'analisi di particolari segni distintivi (*feature*) presenti nell'immagine.
Questa attività può essere effettuata manualmente, ma con un grande costo in termini di tempo per i ricercatori, che spesso hanno a disposizione diverse migliaia o milioni di fotografie, scattate nel corso di anni. L'evidente difficoltà nell'approccio manuale alla foto-identificazione dei cetacei (tutt'oggi ancora ampiamente operata) suggerisce l'applicazione di metodologie di *Computer Vision* per automatizzare tale attività. L'obiettivo del presente lavoro di tesi è la creazione di classificatori binari che ricevano in input un dataset di immagini bidimensionali collezionate nei pressi delle isole Azzorre (Oceano Atlantico settentrionale) e sappiano suddividere lo stesso dataset in due classi di immagini, a seconda che in ciascuna immagine sia rilevata o meno una *feature* utile ad una successiva foto-identificazione. Nel caso dei cetacei, il criterio di classificazione è la presenza nell'immagine della pinna dorsale dell'individuo.
Le metodologie impiegate sono quelle del *machine learning*; in particolare, si è scelto di utilizzare la tecnica del *transfer learning* per il riuso e l'adattamento di modelli pre-addestrati, usati per risolvere task di classificazione diversi da quello in esame. Gli esperimenti condotti su dati reali acquisiti in mare dimostrano l'utilità di tali tecniche di Computer Vision nel campo della foto-identificazione dei cetacei.

Indice

Capitolo 1

Introduzione

La foto-identificazione è una tecnica largamente impiegata per l'identificazione dei singoli individui a partire da una o più immagini. Il principale vantaggio di questa tecnica è la sua non invasività che la rende particolarmente utile per studiare sia la dinamicità che i movimenti di ogni specie. Tale metodologia risulta essere uno strumento affidabile quando viene applicato nella comprensione dei comportamenti dei cetacei (migrazioni e spostamenti). Tra i delfini, vi sono due specie più adatte a tali studi. Il primo delfino riguarda la specie "Tursiops truncatus" (tursiope) o delfino dal naso a bottiglia, mentre la seconda specie riguarda la specie "Grampus griseus" (Grampo, o delfino di Risso), avente numerose cicatrici su tutto il corpo, entrambi appartenenti alla famiglia dei Delfinidi.[1]

Capitolo 2

Metodologie

Come già anticipato, l'approccio più efficace alla risoluzione del problema della classificazione delle immagini consiste nell'impiego delle reti neurali convoluzionali (CNN, *convolutional neural networks*). In questo capitolo saranno introdotti progressivamente i presupposti teorici matematici e informatici su cui si fondano le reti neurali, partendo dalle definizioni preliminari fino a costruire il modello generale di una CNN.

2.1 Reti neurali

TODO: parlare del neurone e dell'idea di replicarlo nella funzione (dopo aver parlato dei classificatori lineari, perché bisogna parlare anche

2.2 Immagini digitali

Caratterizziamo intuitivamente il concetto di "immagine" dal punto di vista informatico.

Un'**immagine digitale** è una rappresentazione binaria di un'immagine (in generale a colori) a due dimensioni[1]; essa può essere definita matematicamente come un tensore $\mathcal{I} \in \mathbb{R}^{h \times w \times c}$, dove h e w sono rispettivamente dette **altezza** e **larghezza** dell'immagine, la coppia (w, h) **risoluzione** mentre c è il numero di *canali di colore*[2]. Nello spazio di colore RGB, ampiamente adoperato, i canali di colore sono rosso (R, Red), verde (G, Green) e blu (B, Blue), quindi $c = 3$. In mancanza di diverse indicazioni, ci si riferirà nel seguito allo spazio di colore RGB.

Un **pixel** $p(i, j)$ è definito come la funzione vettoriale

$$p(i, j) = [r(i, j), g(i, j), b(i, j)]$$

essendo $r, g, b : \{0, \dots, h\} \times \{0, \dots, w\} \to \{0, \dots, 255\}$ le funzioni scalari che associano ad ogni posizione bidimensionale i, j dell'immagine un valore intero di *intensità luminosa* compreso tra 0 e 255, uno per ciascuno dei tre canali RGB. Ogni pixel

[1] Ci riferiamo in questa sede solo alle immagini di tipo raster, tipiche ad esempio delle fotografie digitali in formato jpg.

[2] Spesso si scrive che $\mathcal{I}$ è un immagine $w \times h \times c$, o più semplicemente $w \times h$ (assumendo $c = 3$)

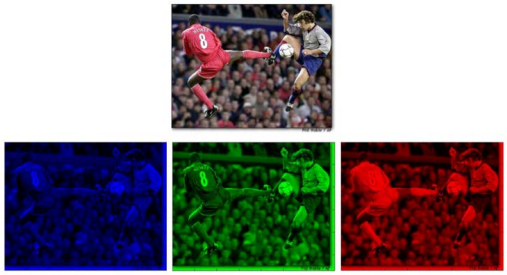

Figura 2.1: Canali RGB di un'immagine

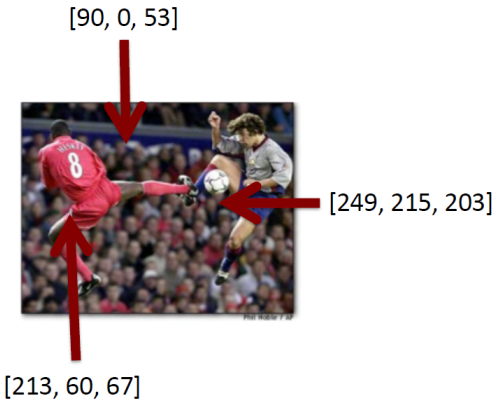

Figura 2.2: Pixel di un'immagine

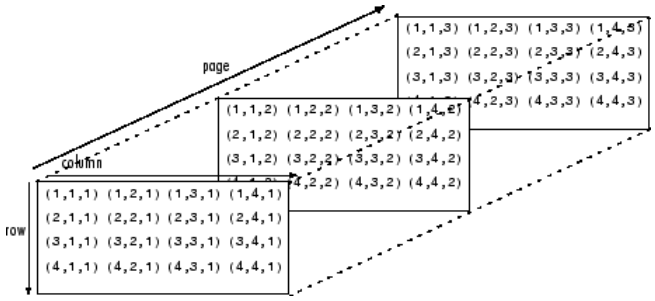

Figura 2.3: Rappresentazione grafica di un tensore tridimensionale; in ogni posizione compaiono gli indici del tensore

definisce univocamente un colore nello spazio RGB, il quale può rappresentare in tutto 256^3 colori diversi, cioè circa 17 milioni.

Si può immaginare il tensore immagine $\mathcal{I}$ come una "pila" di tre matrici, una per ogni canale di colore, come mostrato in figura 2.3.

TODO inserire problema dell'object detection pag. 43 Gianvito

2.2.1 Supervised Learning

Il paradigma dell'**apprendimento supervisionato** (*supervised learning*) si basa sulla creazione di un algoritmo in grado di apprendere una funzione che mappi un input all'output corretto, sulla base di una serie di esempi ideali costituiti da coppie di input e dei relativi output attesi, che gli vengono inizialmente forniti per addestrarlo [1]. Un algoritmo di apprendimento supervisionato analizza i dati di addestramento e inferisce una funzione che può essere usata per mappare nuovi input ai corretti output. Ciò richiede all'algoritmo la capacità di trovare una funzione che sappia generalizzare efficacemente dai dati di training, al fine di adattarsi bene a nuovi dati (per poterne mappare correttamente quanti più possibile).

Molti problemi pratici, come ad esempio la regressione e la classificazione, possono essere formulati ricorrendo ad una funzione matematica

$$\mathcal{F} : X \to Y$$

che associa ad ogni elemento nello spazio degli input X (dataset) uno ed un solo elemento dello spazio degli output. Il concetto di funzione implica l'esistenza di un solo elemento di Y a cui ogni elemento di X è correttamente associato. Il problema consiste allora nel cercare una funzione $\mathcal{F}$ in grado di ottenere esattamente tale associazione, per quanti più elementi di X possibile.

È evidente che questo tipo di problemi ben si presta ad essere approcciato con algoritmi di apprendimento supervisionato.

Prima di analizzare in dettaglio il problema di classificazione delle immagini oggetto della presente tesi, è necessario inquadrare il problema partendo da alcune definizioni preliminari.

Un **dataset** X è una generica collezione di N dati

$$X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$$

Ogni dato $\mathbf{x}^{(i)}$ è chiamato **esempio** (o **data point**). I data point possono essere anche non omogenei tra loro (cioè avere dimensioni differenti). Ciascun esempio si può caratterizzare come un vettore $\mathbf{x}^{(i)} \in \mathbb{R}^D$, in cui ciascun elemento x_i è detto **feature** e rappresenta una caratteristica di un oggetto o un evento misurato. D è il numero di feature in ogni esempio, o **dimensione** dell'esempio. In caso di esempi omogenei (cioè aventi stessa dimensione D) un dataset può essere descritto attraverso una matrice detta **design matrix**, in cui ogni riga corrisponde ad un particolare esempio e ogni colonna corrisponde ad una precisa feature. Un dataset di cardinalità N e in cui ogni esempio ha D feature ha quindi una design matrix di dimensione $N \times D$.

In un problema di classificazione delle immagini orientato all'*object recognition* (riconoscimento di un oggetto in un immagine), sussiste la seguente caratterizzazione:

- X: un insieme di N immagini digitali
- Y: un insieme di K classi predefinite di oggetti che possono essere individuati all'interno di un'immagine (possono essere dei "descrittori" testuali o, equivalentemente, dei numeri interi)

Un elemento di Y è solitamente chiamato **etichetta** o **categoria** (in inglese **label** o **class**); si dice quindi che ogni immagine $\mathbf{x}^{(i)} \in X$ può essere *descritta da un'etichetta* (o *associata ad una categoria*) $\mathbf{y}^{(i)} \in Y$ tramite una funzione di associazione f.[3]
Nella pratica, TODO f non può essere trovata esattamente. (vd gianvito)

TODO: scrivere ora o in un paragrafo a parte i tipi di dato per gestire le immagini messi a disposizione da matlab.

[3]Teoricamente una stessa immagine potrebbe essere descritta da più di un'etichetta o addirittura da nessuna, coerentemente col fatto che in essa potrebbero essere presenti più oggetti o nessun oggetto tra quelli previsti in Y. Tuttavia nella presente tesi questa ambiguità non può sussistere: la classificazione riduce qualsiasi immagine ad una di due categorie mutualmente esclusive e di cui almeno una deve essere ammessa, cioè la presenza o meno di una pinna nell'immagine.

2.3 Classificatore lineare

Il classificatore lineare è una tra le più semplici funzioni di classificazione.[4] Ipotizziamo di avere un insieme di N immagini $\mathbf{x}^{(i)}$ (*data points*), ciascuna con risoluzione fissa $w \times h$ e in formato RGB ($c = 3$), e un insieme di K distinte categorie di oggetti (*labels*). Un **classificatore lineare** è definito dalla funzione

$$f(\mathbf{x}^{(i)}; \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x}^{(i)} + \mathbf{b} \tag{2.1}$$

In questa espressione stiamo assumendo che $\mathbf{x}^{(i)}$ sia un vettore colonna di dimensione $D = hwc$ ottenuto incolonnando una ad una le righe dell'i-esima immagine di tutti e tre i canali di colore, $\mathbf{W}$ una matrice detta **matrice dei pesi** (*weights matrix*) di dimensione $K \times D$ e $\mathbf{b}$ un vettore colonna detto **vettore dei bias** (*bias vector*) di dimensione K. I pesi e i bias sono parametri della funzione f.

Ogni riga j-esima di W e il relativo j-esimo valore di $\mathbf{b}$ serve a calcolare la combinazione (lineare a meno del bias) $\mathbf{w}_j \cdot \mathbf{x}^{(i)} + b_j$. Ognuna delle K combinazioni calcolate è un numero reale che si può interpretare come un "punteggio" registrato dall'i-esima immagine in ogni classe di oggetti in Y (*class score*): l'i-esima immagine è classificata con l'etichetta $\mathbf{y}_j \in Y$ se l'elemento j-esimo del vettore output $f(\mathbf{x}^{(i)}; \mathbf{W}, \mathbf{b})$ è il massimo del medesimo vettore.

L'esempio in figura 2.4 mostra la classificazione di un'immagine di un gatto con $|Y| = 3$ classi (*gatto*, *cane*, *barca*). Per semplicità, l'immagine input è ipotizzata 2×2 e composta da un unico canale di colore ($c = 1$) (quindi $\mathbf{x}$, scritta come vettore colonna, è 4×1).

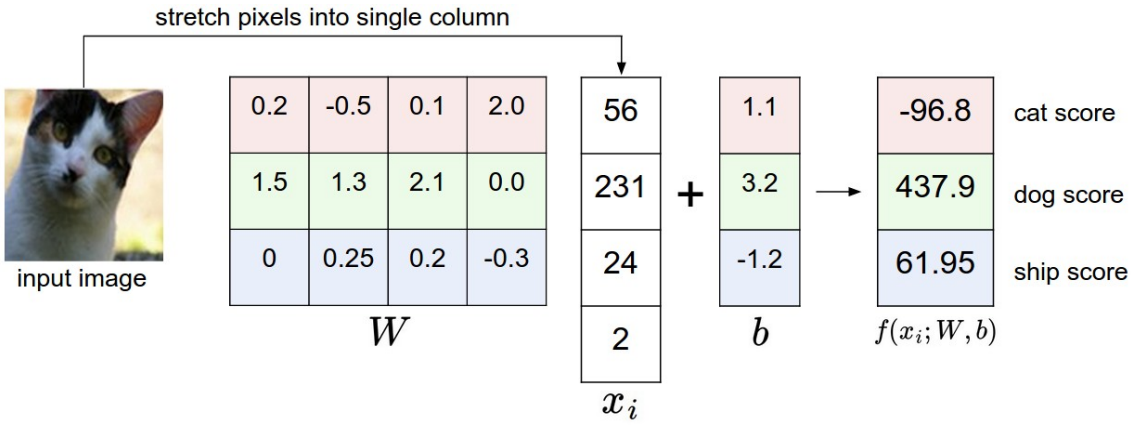

Figura 2.4: Mappatura di un'immagine ai punteggi di ogni classe mediante un classificatore lineare. Si noti che i pesi di $\mathbf{W}$ non costituiscono un buon set di parametri: il punteggio assegnato alla classe "cane" (sbagliata) è alto e quello totalizzato dalla classe "gatto" (corretta) è basso. Il classificatore "è convinto" di aver classificato l'immagine di un cane.

[4] La fonte principale per gli argomenti trattati in questo paragrafo è [2]

Interpretare un classificatore lineare

Poiché le immagini possono essere memorizzate come vettori colonna hwc-dimensionali, si possono immaginare le immagini di un dataset come dei punti nello spazio $\mathbb{R}^{hwc}$. Di conseguenza, il dataset può essere pensato come una collezione di punti multidimensionali. Ovviamente non possiamo visualizzare spazi con più dimensioni di $\mathbb{R}^3$, ma se immaginiamo di "comprimere" tutte le hwc dimensioni in sole due dimensioni otteniamo una visualizzazione del tipo in figura 2.5.

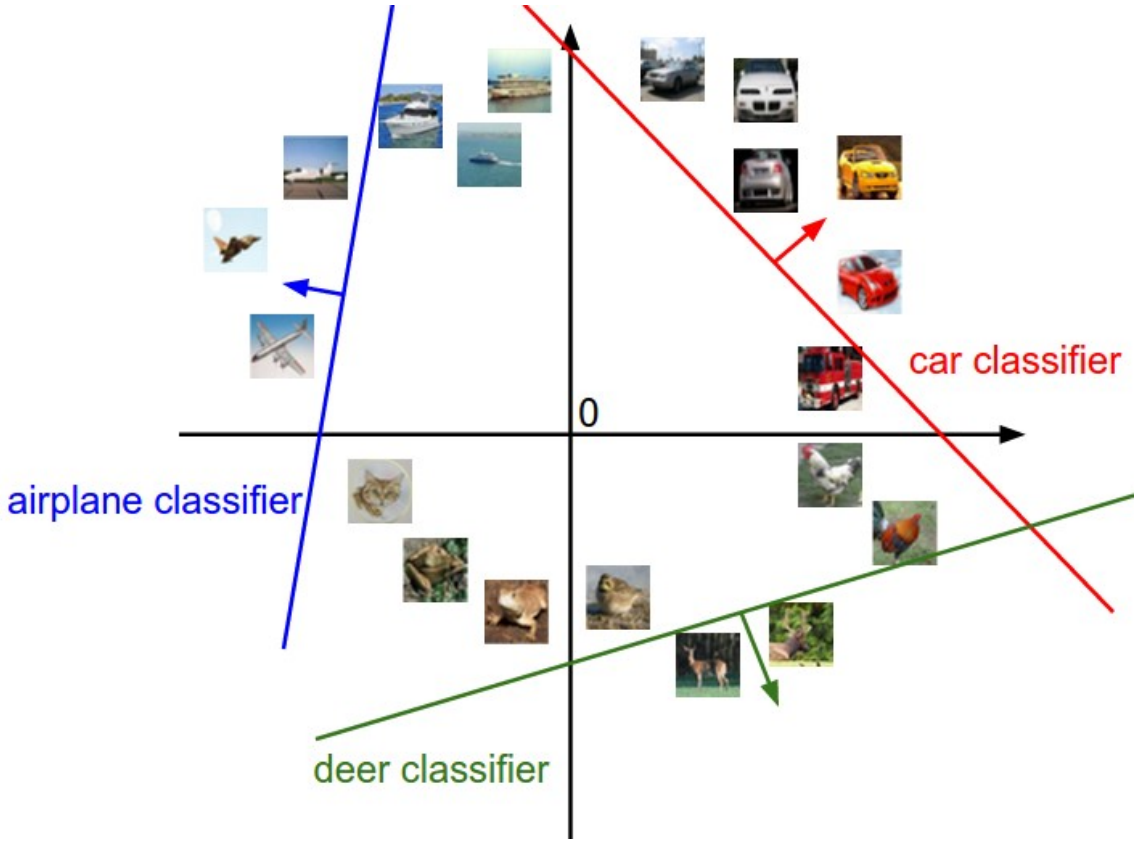

Figura 2.5: Visualizzazione di tre righe di un classificatore lineare, una per ciascuna delle classi "aereo", "auto", "cervo".

Le rette in figura devono in realtà essere pensate come degli iperpiani $(hwc-1)$-dimensionali, associati a ciascuna classe di Y (cioè a ciascuna riga di $\mathbf{W}$ e $\mathbf{b}$), e il piano come lo spazio $\mathbb{R}^{hwc}$. Sussistono le seguenti interpretazioni geometriche:

- Le immagini sono dei punti nel piano. Ogni retta è il luogo dei punti che totalizzano un punteggio nullo per la classe associata a quella retta (la classe è scritta in figura accanto ad ogni retta). La freccia nella figura indica la direzione seguendo la quale i punti del piano aumentano (linearmente) il punteggio realizzato per quella classe.
- Modificare i pesi di $\mathbf{W}$ significa regolare l'inclinazione delle rette (cioè ruotarle rispetto al punto di intercetta).
- Modificare i bias di $\mathbf{b}$ significa regolare l'intercetta delle rette (cioè traslarle verticalmente).

Un altro modo di interpretare i pesi $\mathbf{W}$ può essere quello di far corrispondere ogni riga di $\mathbf{W}$ a un **prototipo** (in inglese **template**) per una delle classi. In questa interpretazione, il punteggio realizzato per ogni classe da un'immagine è ottenuto attraverso l'operazione di prodotto matriciale tra il prototipo della classe j ($\mathbf{w}_j$) e l'immagine da classificare ($\mathbf{x}^{(i)}$). Usando la terminologia introdotta, possiamo affermare che ciò che sta facendo il classificatore lineare è un'operazione di *template matching*, dove i *templates* sono oggetto di apprendimento da parte del classificatore[5].

Ad esempio, analizziamo il dataset *CIFAR-10* [3]. Esso contiene immagini 32×32 ciascuna appartenente ad una di 10 classi. Visualizzando[6] i pesi (e quindi i 10 templates) di un classificatore lineare addestrato su CIFAR-10 si ottengono i risultati in figura seguente:

Figura 2.6: Visualizzazione dei templates di un classificatore addestrato sul dataset CIFAR-10

Si possono fare alcune interessanti osservazioni.

Ad esempio, il prototipo della classe "barca" è composto da molti pixel blu disposti perlopiù lungo i margini, come ci si potrebbe aspettare dal momento che molte immagini di barche in CIFAR-10 raffigurano queste in mare aperto. Questo template allora assegnerà un punteggio alto quando l'immagine che si vuole classificare (cioè *raffrontare al template*) è una barca in mare aperto. In altre parole, un'immagine realizzerà un punteggio tanto più alto in una certa classe quanto più essa è *simile* al template che il classificatore lineare *ha imparato* per quella classe.

Il prototipo per la classe "cavallo" sembra essere l'immagine di un cavallo a due teste; similmente, quello per la classe "auto" sembra una miscela di rappresentazioni di un'auto vista da più direzioni diverse. Ciò è coerente col fatto che il classificatore lineare è stato addestrato su immagini di cavalli visti rispetto a entrambi i profili e su immagini di auto raffigurate in tante direzioni diverse. Inoltre, il template per l'auto sembra rappresentare un'auto di colore rosso: evidentemente in CIFAR-10 la maggior parte delle automobili rappresentate sono di quel colore.

Come si vedrà nel seguito, questa operazione di *template matching* presenta una forte analogia con il funzionamento di un *Fully Connected Layer* di una rete neurale convoluzionale.

[5]Si introdurranno gli algoritmi di apprendimento (supervisionato) nel capitolo **??**.

[6]TODO Per i dettagli su come "visualizzare" i pesi si veda `https://it.mathworks.com/help/deeplearning/examples/visualize-activations-of-a-convolutional-neural-network.html`.

Bias trick

Concludiamo questo capitolo menzionando un "trucco" matematico molto utilizzato per rappresentare **W** e **b** come un'unica matrice, semplificando la notazione 2.1. Possiamo aggiungere il vettore dei bias in coda alla matrice dei pesi e aggiungere un "1" in coda al vettore che rappresenta l'immagine. In questo modo, il classificatore lineare è rappresentato dalla funzione di associazione

$$f(\mathbf{x}^{(i)}; \mathbf{W}) = \mathbf{W}\mathbf{x}^{(i)} \tag{2.2}$$

In questa maniera, f calcola solo combinazioni lineari (un singolo prodotto matriciale), poiché il vettore dei bias è stato eliminato. Tale utile passaggio, noto come *bias trick*, è visualizzato nella seguente figura

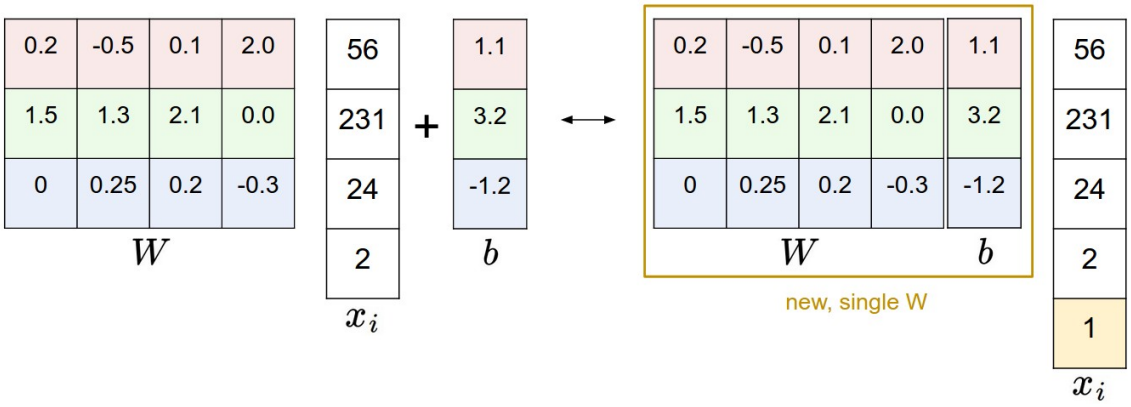

Figura 2.7: Bias trick

TODO: loss functions.
TODO: valutazione delle prestazioni di una rete neurale (matrice di confusione)

2.4 AlexNet

AlexNet è una CNN creata tra il 2011 e il 2012 da Alex Krizhevsky, in collaborazione con Ilya Sutskever e Geoffrey Hinton [4]. La vittoria di AlexNet nella ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [**imagenet**], ottenuta con un netto distacco nei confronti degli altri concorrenti, ha segnato l'inizio dell'enorme successo ottenuto dalle reti neurali profonde in svariati domini di applicazione [6]

Il risultato principale di AlexNet, così come dichiarato dai suoi creatori nell'articolo originale, è il fatto che la profondità del modello è stato essenziale per conferirgli prestazioni così alte. L'alto costo computazionale dell'addestramento di AlexNet, reso oneroso appunto dalla profondità del modello (e quindi dal grande numero di parametri - circa 62.3 milioni) è stato affrontato con l'impiego di schede grafiche (GPU), che cominciavano in quegli anni a raggiungere notevoli potenze di calcolo.

2.4.1 Architettura di AlexNet

L'architettura di AlexNet è riportata schematicamente nella figura 2.8 e con maggiore dettaglio in tabella 2.1. La rete accetta in input immagini 227×227. Essa si compone di otto layer con parametri - cinque convoluzionali e tre completamente connessi. L'output dell'ultimo layer completamente connesso passa per un softmax layer a 1000 vie, il quale fornisce la distribuzione di probabilità per le 1000 classi del dataset ImageNet.

Tra ognuno degli otto strati parametrizzati sono interposti alcuni strati intermedi: ReLU layer, Local Response Normalization layer, Max Pooling layer, Dropout layer. Ognuno di questi sarà analizzato in maggiore dettaglio nei paragrafi successivi.

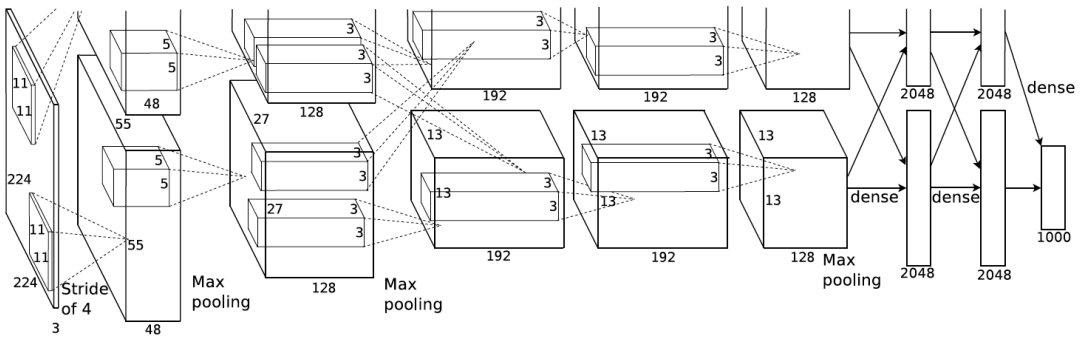

Figura 2.8: Architettura originale di AlexNet [4]

Come si evince dalla figura 2.8, la rete è composta da due "*pipeline*" parallele. Si scelse infatti di "estendere" la rete su due GPU NVIDIA® GeForce® GTX 580 3GB in fase di training, per raddoppiare la memoria massima disponibile (6GB in totale) per conservare la rete e i suoi parametri.

Queste GPU si prestano bene a lavorare in parallelo, poiché possono leggere e scrivere l'una sull'altra direttamente, senza passare dalla memoria della macchina host. Lo schema di parallelizzazione a due vie prevede che su ogni GPU risieda la

metà dei kernel (o dei neuroni) di ciascuno strato parametrizzato. Le GPU possono comunicare tra loro solo in certi strati. In particolare, i kernel del layer convoluzionale 1 e 3 hanno in input l'intero output volume rispettivamente del layer di input e del layer convoluzionale 2, mentre i kernel dei rimanenti strati convoluzionali hanno in input la sola metà dell'output volume presente nella stessa GPU (*grouped convolution*[7].

Sono di seguito passate in rassegna le principali scelte architetturali introdotte in AlexNet, ed alcuni dettagli relativi al suo addestramento.

Funzione di attivazione ReLU

Dopo ogni strato parametrizzato, i valori delle attivazioni sono passati alla funzione attivatrice "rettificatore": $f(x) = x^+ = \max(0, x)$ [7]. Questa funzione attivatrice non-lineare e non soggetta a saturazione permette un addestramento molto più veloce delle reti convoluzionali profonde, in confronto a funzioni attivatrici fino ad allora più utilizzate come la funzione sigmoidea $f(x) = (1 + \exp^{-x})^{-1}$ e la funzione tangente iperbolica $f(x) = \tanh(x)$.

Local Response Normalization

È stato verificato che la seguente normalizzazione delle attivazioni, *Local Response Normalization*, aumenta lievemente la capacità di generalizzazione del modello:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

dove $a^i_{x,y}$ è l'attivazione del neurone ottenuto applicando il kernel i-esimo alla posizione (x, y) e applicando in seguito la funzione ReLU, $b^i_{x,y}$ l'attivazione normalizzata, N il numero totale di kernel del layer corrente, k, n, α, β sono iperparametri; sono stati usati i valori $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$.

Questa normalizzazione è adoperata solamente nel primo e nel secondo layer convoluzionale.

Overlapping Max Pooling

La funzione di max pooling in AlexNet è stata caratterizzata dalla scelta di una dimensione del filtro di pooling 3×3 e uno stride di 2. È stato osservato durante la fase di training che questa funzione di *max pooling con sovrapposizione* ha attenuato lievemente l'*overfitting* della rete.

[7] La scelta di questo pattern di connettività fra le due GPU parallele è il risultato di un problema di cross-validation.

Data Augmentation

Una delle difficoltà che si incontrano spesso quando si vuole addestrare una rete neurale con moltissimi parametri avendo a disposizione un dataset relativamente piccolo è il rischio del sovradattamento (*overfitting*) della rete al training set, che compromette anche seriamente le prestazioni della rete quando le vengono presentati nuovi dati. In AlexNet l'overfitting è stato ridotto grazie a tecniche di *data augmentation*. In particolare, dopo aver ridimensionato a 256×256 tutte le immagini del training set, quest'ultimo è stato "arricchito" con le seguenti immagini:

- Estrazione casuale di ritagli 224×224 dalle immagini
- Riflessione orizzontale ("a specchio") delle immagini
- Somma di un'immagine e le sue componenti principali (PCA)[8]

Dropout

Un altro modo per ridurre il problema del sovradattamento è l'impiego di tecniche di regolarizzazione dei parametri. AlexNet utilizza la tecnica del *dropout* [9]. Questa tecnica consiste nel settare a zero l'attivazione di ciascun neurone di un layer intermedio con probabilità p (AlexNet impiega un dropout con $p = 0.5$. I neuroni "azzerati" sono essenzialmente eliminati dalla rete e non contribuiscono né alla propagazione all'indietro del gradiente né al calcolo delle attivazioni nello strato finale (in fase di addestramento). Questa tecnica riduce il *co-adattamento* tra neuroni: ogni neurone non può fare affidamento sulla presenza di altri neuroni, ed è costretto ad apprendere feature utili in congiunzione con diversi sottoinsiemi casuali degli altri neuroni, e non con un solo particolare sottoinsieme, migliorando la generalizzazione su nuovi dati.

In AlexNet, il dropout dei neuroni è utilizzato nei primi due layer completamente connessi. In fase di test, i neuroni di questi due strati sono moltiplicati per 0.5 per tenere conto dell'impiego del dropout in addestramento.

Addestramento di AlexNet

Nella sua forma originale, AlexNet fu addestrato usando la discesa stocastica del gradiente con momento= 0.9, mini-batch= 128 e decadimento dei pesi (weight decay) = 0.0005. Dettagli più specifici sulla fase di addestramento di AlexNet possono essere trovati nel paper originale [4].

[8] L'*analisi delle componenti principali* (PCA, principal component analysis) è una tecnica per la semplificazione dei dati utilizzata nell'ambito della statistica multivariata. In questa sede ci limitiamo a specificare che il suo utilizzo nell'ambito della data augmentation è di evidenziare una importante proprietà delle immagini naturali, e cioè che l'identità di un oggetto è invariante rispetto ai cambi d'intensità e di colori nella sua illuminazione. Si rimanda ad esempio a [8] per approfondimenti sulla PCA.

Tabella 2.1: Architettura originale di AlexNet

N	Layer	Attivazioni	Parametri
1	INPUT	$(227 \times 227 \times 3)$	
2	CONVOLUTION	$(55 \times 55 \times 96)$	Pesi: $(11 \times 11 \times 3) \times 96$ Bias: (96)
3	RELU	–	–
4	NORMALIZATION	–	–
5	MAX POOLING	$(27 \times 27 \times 96)$	–
6	GROUPED CONVOLUTION	$(27 \times 27 \times 256)$	Pesi: $(5 \times 5 \times 48) \times 128 \times 2$ Bias: $(128) \times 2$
7	RELU	–	–
8	NORMALIZATION	–	–
9	MAX POOLING	$(13 \times 13 \times 256)$	–
10	CONVOLUTION	$(13 \times 13 \times 384)$	Pesi: $(3 \times 3 \times 256) \times 384$ Bias: (384)
11	RELU	–	–
12	GROUPED CONVOLUTION	$(13 \times 13 \times 384)$	Pesi: $(3 \times 3 \times 192) \times 192 \times 2$ Bias: $(192) \times 2$
13	RELU	–	–
14	GROUPED CONVOLUTION	$(13 \times 13 \times 256)$	Pesi: $(3 \times 3 \times 192) \times 128 \times 2$ Bias: $(128) \times 2$
15	RELU	–	–
16	MAX POOLING	$(6 \times 6 \times 256)$	–
17	FULLY CONNECTED	4096	Pesi: 4096×9216 Bias: 4096
18	RELU	–	–
19	DROPOUT	–	–
20	FULLY CONNECTED	4096	Pesi: 4096×4096 Bias: 4096
21	RELU	–	–
22	DROPOUT	–	–
23	FULLY CONNECTED	1000	Pesi: 2×4096 Bias: 2
24	SOFTMAX	–	–
25	CROSS-ENTROPY LOSS	–	–

2.5 GoogLeNet

Capitolo 3

Esperimenti e risultati

In questo capitolo vengono presentati gli esperimenti condotti e si analizzano i risultati ottenuti.

3.1 Descrizione dei dataset utilizzati

Per la conduzione degli esperimenti sono stati adoperati due distinti dataset di immagini, di seguito descritti (TODO descrivere meglio)

- Il primo, usato per la fase di addestramento delle reti neurali, è una collezione di fotografie di tursiopi e grampi scattate tra il 2017 e il 2018 nel **Golfo di Taranto** (mar Ionio Settentrionale). Le fotografie sono state scattate e messe a disposizione dalla *Jonian Dolphin Conservation*, un'associazione di ricerca scientifica privata finalizzata allo studio dei cetacei nel Mar Ionio Settentrionale. Il dataset contiene immagini acquisite in un'area di 14000 km^2 percorsa su un catamarano e seguendo rotte prestabilite. Le macchine fotografiche utilizzate consistono di diversi corpi macchina (reflex) e diversi obiettivi ad essi associati. Il dataset acquisito contiene in totale TODO immagini, suddivise in cartelle in base alla data degli scatti.

- Il secondo, usato per testare le prestazioni dei classificatori binari precedentemente addestrati, consiste in un insieme di fotografie di tursiopi e grampi scattate nel mese di giugno 2018 nei pressi delle **Isole Azzorre** (Oceano Atlantico settentrionale) dall'associazione TODO. Questo dataset contiene in totale 5793 immagini, anche questa volta suddivise in cartelle in base alla data degli scatti.

Entrambi i dataset contengono fotografie con una notevole risoluzione TODO. Tuttavia, prendendo visione delle immagini in ciascuno dei due dataset ci si rende subito conto che non tutte contengono pinne dorsali di cetacei: in alcune foto sono totalmente assenti, rendendo lo scatto totalmente privo di contenuto informativo per i biologi. Anche laddove le pinne sono presenti, esse possono risultare sfocate o di bassa risoluzione se molto lontane. Infine, in tutte le fotografie sono inevitabilmente ritratti oggetti che non sono "informativi" ai fini dello studio delle sole pinne dorsali quali barche, persone, uccelli, terraferma (paesaggi), porzioni di cielo, boe,

lo specchio d'acqua ma anche parti dei cetacei diversi dalla loro pinna dorsale, quali pinne caudali e laterali e la testa degli esemplari.

Si rende perciò necessario filtrare in qualche modo le sole immagini che raffigurano al loro interno pinne dorsali di cetacei; è utile inoltre ritagliare da queste immagini filtrate le sole regioni in cui è effettivamente presente una pinna (si vuole cioè isolare l'informazione utile dal resto dal dato originale). Per far questo, i due dataset sono stati rielaborati attraverso un **algoritmo di riconoscimento e cropping** delle pinne dorsali, di seguito descritto nelle sue caratteristiche salienti.

TODO spiegare che allora il classificatore binario classifica i ritagli e non le foto intere.

3.2 CropFin v1: pre-processing e estrazione di feature dai dataset

Per ritagliare ed estrarre dalle immagini originali le sole pinne dorsali, è stata utilizzata la routine *CropFin v1* in linguaggio MATLAB sviluppata dall'ing. Gianvito Losapio [11] sulla base di un precedente lavoro dell'ing. Flavio Forenza [10]. Si può descrivere la routine in due fasi:

1. Segmentazione, filtraggio e ritaglio adattivo delle regioni delle immagini che possono verosimilmente contenere una pinna

2. Classificazione di ogni ritaglio ottenuto in due classi 'Pinna' e 'No Pinna', mediante una rete neurale artificiale creata *ad-hoc*.

La novità introdotta dal presente lavoro di tesi in merito al problema di estrazione delle pinne da un'immagine riguarda l'utilizzo di un metodo di classificazione basato sul *transfer learning*. In pratica, quindi, la principale differenza rispetto a CropFin v1 è nella seconda fase della routine: la classificazione avviene con l'utilizzo non più di una rete artificiale creata da zero per il problema in analisi, bensì riutilizzando un insieme di reti neurali profonde addestrate su un diverso problema di classificazione e adattate al nostro task. Questo nuovo modello è descritto dettagliatamente nel par. 3.3.

Al fine di ottenere i ritagli delle pinne, la routine adoperata attua una sequenza di operazioni di preprocessing su ciascuna immagine per poi individuare ed infine ritagliare e salvare separatamente le sole porzioni di immagini che possono eventualmente contenere pinne. Tale sequenza è implementata mediante un ciclo `for` che cicla su ogni immagine del dataset. Di seguito sono descritte sinteticamente le operazioni, nell'ordine in cui vengono applicate.

TODO inserire immagini in ogni subsection per visualizzare le funzioni utilizzate

3.2.1 Descrizione di CropFin v1

Ridimensionamento

L'immagine è innanzitutto ridimensionata mediante la funzione MATLAB `imresize`, al fine di ottenere una nuova immagine di risoluzione più bassa (800×1200). Questa operazione di preprocessing è stata adottata per diminuire il costo computazionale delle operazioni successive.[1]

CLAHE

L'immagine ridimensionata è sottoposta ad una equalizzazione adattiva dell'istogramma a contrasto limitato (CLAHE). Questa operazione consente un miglioramento del contrasto dell'immagine, proprietà utile per migliorare l'efficienza della successiva operazione, la sogliatura dell'immagine secondo il metodo di Otsu.

Segmentazione

L'immagine viene segmentata (cioè ogni pixel viene assegnato ad una di due classi: *background* e *foreground*) mediante il metodo di Otsu per la sogliatura automatica [**otsu**]. Il metodo di Otsu viene usato nella sua versione classica a due livelli, rispetto agli istogrammi dei canali L e b. In particolare, viene applicata la sogliatura secondo Otsu separatamente al canale L e b, cioè calcolate le soglie di Otsu per i due canali, mediante la funzione `multithresh`. Avendo a disposizione tali soglie, l'ipotesi avanzata è che le pinne dorsali possono essere isolate considerando le regioni di immagine che siano contemporaneamente:

- nella regione più scura del canale L, cioè a sinistra della soglia sul canale L
- nella regione contenente il grigio del canale b, cioè a destra della soglia sul canale b

L'immagine segmentata (binarizzata) finale è ottenuta quindi annerendo quei pixel dell'immagine che non verificano le seguenti condizioni (o, equivalentemente, rendendo bianchi i pixel che le verificano)[2]

- valore della componente L minore della soglia di Otsu sul canale L
- valore della componente b maggiore della soglia di Otsu sul canale b

[1] La risoluzione di partenza delle immagini utilizzate è stata 6000×4000, ottenendo una riduzione drastica di pixel del 96%, da 24 milioni a 960 mila.

[2] L'idea alla base di questo approccio nasce da una precisa conoscenza del dominio e da alcune ipotesi a priori riguardanti il contenuto delle immagini. In particolare, si suppone che esse contengano generalmente solo mare (background) e cetacei (foreground), e che queste due classi di oggetti contribuiscano alla creazione di due aree distinte e separabili degli istogrammi dei canali L e b. Volendo dare un'interpretazione intuitiva, si tratta di separare ciò che è grigio e più scuro da ciò che è blu e più chiaro. La scelta dello spazio di colori Lab è motivata proprio dalla possibilità di automatizzare questo tipo intuitivo di segmentazione.

Filtraggio delle regioni connesse

L'immagine binaria ottenuta in seguito alla segmentazione viene filtrata in modo che siano scartate quelle regioni binarie connesse (anche dette *blob*) che non presentano caratteristiche tali da poter rappresentare, verosimilmente, una pinna dorsale. In particolare vengono utilizzati, consecutivamente due filtri:

1. il primo è applicato all'intera immagine binarizzata e serve a migliorare il risultato della sogliatura secondo Otsu. Il filtro è configurato per mantenere, nell'ordine, le regioni connesse con le seguenti proprietà:

   - prime 15 in ordine decrescente di `Area` (n. di pixel che compongono la regione connessa)
   - `Area` nel range `[1600, 40000]`
   - `Extent` nel range `[-Inf, 0.55]` (rapporto tra `Area` e il n. di pixel del più piccolo rettangolo che racchiude l'intera regione connessa, con i lati paralleli a due a due paralleli ai bordi dell'immagine)

2. il secondo è applicato come segue

   (a) Si ritaglia la foto originale in corrispondenza delle regioni mantenute in seguito all'applicazione del primo filtro, sulla base delle coordinate dei bounding box. Per ottenere ritagli leggermenti più larghi rispetto ai blob, al fine di non perdere eventuali parti della pinna erroneamente anneriti dopo la binarizzazione, ogni dimensione è aumentata del 20%.

   (b) Si applica nuovamente, a ciascun ritaglio ottenuto, la sogliatura basata sul metodo di Otsu. In questo caso è omesso il miglioramento del contrasto mediante CLAHE prima del calcolo dei valori di soglia.

   (c) Si introduce a questo punto il secondo filtro, applicato alle regioni binarie ottenute per ciascun ritaglio. L'unico parametro utilizzato in questo caso è il seguente:

   - `Area` nel range `[20000, 1000000]`

   con lo scopo di isolare l'eventuale pinna (che rappresenta sicuramente la regione di area maggiore all'interno di ciascun ritaglio) in modo che possa essere sottoposta all'algoritmo di ritaglio adattivo, descritto nella sezione successiva.

Ritaglio adattivo

Le regioni binarie mantenute in seguito alla fase di filtraggio sono sottoposte ad un algoritmo che consente di ottenere un ritaglio preciso in corrispondenza delle pinne. Tale operazione si può definire "adattiva" nella misura in cui la regione di ritaglio è ottenuta a partire da precisi punti geometrici calcolati per ciascuna regione binaria. Evitando di scendere nei dettagli implementativi e numerici (riportati nel par. 5.1 in [11]), si descrivono nell'ordine le operazioni effettuate sulle singole regioni binarie dall'algoritmo di ritaglio:

1. Si sottopone la regione binaria al riempimento dei cosiddetti *holes*, cioè "buchi" anneriti racchiusi in una regione connessa, mediante la funzione `imfill` con opzione `'holes'`

2. Si individuano quattro punti di interesse; nell'ordine: punto più in alto, punto medio tra questo ed il centroide, punti di estrema sinistra e destra della regione connessa all'altezza del punto medio

3. Si identifica un rettangolo che racchiuda i punti precedentemente trovati

4. Si trasla e si estende il rettangolo trovato in modo che contenga l'intera pinna, a seconda della sua orientazione.

L'output di questa prima fase della routine sono i ritagli di quelle regioni dell'immagine originale che, verosimilmente, ritraggono una pinna dorsale. Questa ipotesi sul contenuto dei ritagli è sostenuta solamente sulla base del processo di segmentazione e filtraggio appena descritto.

La routine CropFin v1, nella sua prima fase di ritaglio adattivo, è stata applicata ai dataset degli scatti di Taranto e delle Azzorre. In tabella 3.1 è riportato il numero di ritagli (*crops*) prodotti da CropFin v1 con input i dataset sopracitati.

Dataset	N. foto	N. crop	di cui 'Pinna'	di cui 'No Pinna'
Taranto	10194	15228	4033	11195
Azzorre	11290	20395	TODO	TODO

Tabella 3.1: Output della prima fase di CropFin v1

Classificazione mediante rete ad hoc

È evidente da una rapida ispezione dell'output che la quantità di regioni estratte che però non contengono pinne risulta, su larga scala, superiore a quello che contiene effettivamente pinne. Numericamente questo fatto è evidenziato in tab. 3.1, dopo una fase di etichettatura a mano dei ritagli prodotti, nelle classi 'Pinna' e 'No Pinna' (spiegata nel seguito del paragrafo).

Questa osservazione è ciò che primariamente motiva l'introduzione di una fase di classificazione finale in CropFin v1, che consenta di automatizzare completamente la procedura di object detection.

Come anticipato, in CropFin v1 si decide di effettuare la classificazione binaria 'Pinna'/'No Pinna' per mezzo di una rete neurale creata ad-hoc. In particolare, CropFin v1 prevede l'utilizzo di cinque classificatori binari, addestrati con la tecnica della *5-fold cross-validation*[3] sui ritagli restituiti da CropFin v1 sul solo dataset con gli scatti di Taranto (descritto in 3.1).

Per consentire l'addestramento del classificatore si è reso necessario un lavoro di etichettatura manuale dei 15228 ritagli, attribuendo a ciascuno la classe 'Pinna' e 'No Pinna'. I risultati di questa etichettatura manuale sono presenti nella tab. 3.1. Si precisa che, nella fase di etichettatura manuale, sono stati attribuiti alla

[3]Si rimanda al par. **??** per i dettagli implementativi della tecnica *k-fold cross-validation*

classe 'Pinna' tutti e soli i ritagli contenenti una sola pinna in primo piano, intera o leggermente tagliata, escludendo invece quelli con pinne multiple e quelli con una presenza preponderante del dorso dei delfini. I ritagli con tali caratteristiche, infatti, sono considerati maggiormente affidabili ai fini di una successiva foto-identificazione automatica delle pinne (ad esempio con il metodo basato sul metodo *SIFT* sviluppato e descritto in [**emanuele**]). Inoltre, questa scelta è stata anche motivata dall'intenzione di creare un "concetto univoco" utile a semplificare sia la selezione manuale sia l'apprendimento del classificatore.

Le cinque reti risultanti lavorano in sinergia per classificare ciascun ritaglio, utilizzando un metodo di *major voting* (par. **??**) "ibrido", che rende la classificazione finale ternaria: se le classificazioni 'Pinna' prodotte dalle cinque reti

- sono maggiori o uguali a 4, la classificazione finale è 'Pinna'
- sono pari a 2 o 3, la classificazione finale è 'Incerta'
- sono minori o uguali a 1, la classificazione finale è 'No Pinna'

L'analisi delle prestazioni di questo tipo di classificazione è descritta nel par. 4.5 di [11].[4]

TODO dopo quando faccio il confronto con la rete di Gianvito sui 500 campioni devo scrivere "Per consentire il confronto del classificatore *ensemble* di CropFin descritto nel par. **??** e il classificatore *ensemble* creato nel presente lavoro di tesi si è scelto di ritenere le pinne 'Incerta' come 'No Pinna'."

3.3 Classificazione mediante CNN e Transfer Learning

Nel par. **??** sono stati descritti molteplici motivazioni per le quali per risolvere un problema di classificazione (in particolare di *image captioning*) può essere meglio usare la tecnica del *transfer learning*, adattando al task in esame una rete neurale pre-addestrata piuttosto che creare una rete da zero. Il nucleo principale di questo lavoro di tesi è quindi dedicato alla creazione di un nuovo modello di classificazione, basato su *transfer learning*, che possa migliorare la fase di classificazione di CropFin v1. Questi "miglioramenti" sono da valutare con rigore ingegneristico sulla base di alcuni parametri, che consentono un confronto di prestazioni con il classificatore di CropFin v1; l'analisi delle prestazioni e quindi il confronto è svolto nel par. **??**.

3.3.1 Creazione del dataset

Il dataset utilizzato per l'addestramento del nuovo classificatore binario è lo stesso usato in [11] per l'addestramento del classificatore di CropFin v1 composto dai ritagli, opportunamente etichettati a mano, prodotti a partire dagli scatti collezionati nel Golfo di Taranto.

[4] Qualora fosse necessario attenersi a un problema di classificazione strettamente binario, ad esempio per effettuare un confronto con altri metodi di classificazione binaria per il problema in esame, si possono ad esempio ricondurre i ritagli di classe 'Incerta' alla classe 'No Pinna'. Questa è la scelta effettuata nel par. TODO per il confronto

3.3.2 Addestramento

Sono state riutilizzate ed adattate mediante la tecnica del *transfer learning* quattro reti neurali convoluzionali (*CNN*) sviluppate nell'ambito della *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* ed addestrate sul dataset ImageNet (par. **??**). Esse sono di seguito elencate e, per ciascuna, ne viene motivata la scelta.

- **AlexNet** (par. 2.4)
  La sua vittoria nella *ILSVRC 2012* con un grado di accuratezza del 16.4% ha di fatto dimostrato alla comunità scientifica la straordinaria efficienza delle reti neurali convoluzionali nell'ambito dei problemi di *computer vision*.

- **GoogLeNet** (par. **??**)
  Grazie all'introduzione del modulo *Inception*, GoogLeNet è una rete profonda ma incredibilmente leggera e semplice da addestrare, se paragonata alle precedenti reti fino ad allora esistenti (tra tutte, AlexNet).

- **ResNet-18** (par. **??**)
  ResNet rappresenta lo stato dell'arte nell'ambito delle reti neurali convoluzionali; l'introduzione dei *residual blocks* ha permesso di avere reti con un grandissimo numero di layer, attenuando di molto i problemi legati all'estrema profondità dell'architettura.

- **ResNet-50** (par. **??**)
  Una variante di ResNet-18, più profonda e con migliori prestazioni sul dataset *ImageNet*.

Bibliografia

[1] Stuart Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2009.

[2] Andrej Karpathy. *Lectures of Stanford class CS231n: Convolutional Neural Networks for Visual Recognition*. Appunti del corso Stanford CS class C231n. 2019.

[3] Alex Krizhevsky. «Learning multiple layers of features from tiny images». In: (2009).

[4] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems 25*. 2012.

[5] *ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)*. 2012. URL: `http://image-net.org/challenges/LSVRC/2012/results`.

[6] Md Zahangir Alom et al. «The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches». In: ().

[7] Vinod Nair e Geoffrey E. Hinton. «Rectified Linear Units Improve Restricted Boltzmann Machines». In: *ICML*. 2010.

[8] Sergio Bolasco. *Analisi multidimensionale dei dati. Metodi, strategie e criteri d'interpretazione*. Carocci Editore, 1999.

[9] Geoffrey E. Hinton et al. «Improving neural networks by preventing co-adaptation of feature detectors». In: (2012).

[10] Flavio Forenza. *Tecniche innovative di Computer Vision per la Foto-Identificazione dei cetacei*. Tesi di laurea triennale, a.a. 2017/2018, rel. prof. Giovanni Dimauro, correl. dr. Vito Renò. 2017.

[11] Gianvito Losapio. «Tecniche di Deep Learning e Object Detection per la foto-identificazione dei cetacei». Tesi di laurea triennale, a.a. 2018/2019, rel. prof. Tiziano Politi, correl. dr. Vito Renò. Tesi di laurea mag. Politecnico di Bari, 2019.

[12] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.