



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica
Curriculum: Data Science

Tesi di Laurea

TECNICHE DI INCREMENTO DEI DATI PER
LA RICERCA DI IMMAGINI BASATA SU
CONTENUTI CON CNN

DATA AUGMENTATION TECHNIQUES FOR
CONTENT-BASED IMAGE RETRIEVAL
USING CNNS

IRENE DINI

Relatore: *Marco Bertini*

Anno Accademico 2019-2020

Irene Dini: *Tecniche di incremento dei dati per la ricerca di immagini basata su contenuti con CNN*, Corso di Laurea Magistrale in Informatica
Curriculum: Data Science, © Anno Accademico 2019-2020

INDICE

1	Introduzione	7
2	Image Retrieval	11
2.1	Storia dell'Image retrieval	12
2.2	Descrittori SIFT (Scale-Invariant Feature Transform)	14
2.2.1	Creazione dello scale space	14
2.2.2	Approssimazioni LoG	15
2.2.3	Ricerca di massimi e minimi nella DoG	16
2.2.4	Rimozione di key point non significativi	17
2.2.5	Invarianza per rotazioni	18
2.2.6	Generazione delle features locali	19
2.3	Reti neurali convoluzionali	20
2.3.1	Immagine in input	20
2.3.2	Convoluzione	21
2.3.3	Receptive field	22
2.3.4	Pooling	23
3	Strumenti utilizzati	25
3.1	Framework di valutazione	25
3.1.1	Distanza del coseno	26
3.1.2	Mean Average Precision	26
3.2	Transfer Learning	27
3.3	VGG16	28
3.4	Triplet Loss	29
3.5	NetVlad	32
3.6	Data Augmentation	33
3.6.1	AutoAugment	35
3.6.2	RandAugment	37
3.7	PCA	38
4	Dataset	41
4.1	INRIA Holidays	41
4.2	Oxford Buildings	43
4.3	Paris	47
5	Esperimenti	49
5.1	Esperimenti su Holidays	49
5.2	Esperimenti su metodi di pooling	51
5.3	Esperimenti su data augmentation	53

2 Indice

5.4 Esperimenti sulla dimensione delle immagini	57
6 Conclusioni	61
6.1 Sviluppi futuri	62

ELENCO DELLE FIGURE

Figura 1.1	Rappresentazione schematica di un neurone e di un neurone artificiale	7
Figura 1.2	Rappresentazione di una rete neurale artificiale	8
Figura 2.1	Storia delle tecniche utilizzate nell'ambito della computer vision	13
Figura 2.2	SIFT: Scale space	15
Figura 2.3	SIFT: Differenza di Gaussiane	16
Figura 2.4	SIFT: Ricerca dei key points	17
Figura 2.5	SIFT: Key points selezionati	18
Figura 2.6	Calcolo dell'orientamento in un punto chiave	19
Figura 2.7	Immagine RGB	21
Figura 2.8	Convoluzione	22
Figura 2.9	CNN: Receptive Field	23
Figura 2.10	Esempio di max-pooling	24
Figura 3.1	Differenza apprendimento tradizionale e transfer learning	28
Figura 3.2	Struttura di VGG16	29
Figura 3.3	Triplet loss	30
Figura 3.4	Esempi di negativi per la triplet loss	31
Figura 3.5	Struttura di NetVLAD	33
Figura 3.6	Esempi di data augmentation	34
Figura 3.7	Algoritmo di ricerca delle policy	36
Figura 3.8	Esempio di policy	37
Figura 4.1	Esempi di immagini di INRIA Holidays	42
Figura 4.2	Esempi immagini di Oxford	44
Figura 4.3	Immagine di query di Oxford	44
Figura 4.4	Risposte alla query della classe good	45
Figura 4.5	Risposte alla query della classe ok	46
Figura 4.6	Risposte alla query della classe junk	46
Figura 4.7	Esempi immagini di Paris	47
Figura 5.1	Confronto tra immagini originali e immagini generate utilizzando policy di RandAugment ottimizzate su CIFAR-10	55

4 Elenco delle figure

- Figura 5.2 Confronto tra immagini originali e immagini generate utilizzando policy di RandAugment 56
- Figura 5.3 Query Keble 3: AP = 98% 58
- Figura 5.4 Query Pitt Rivers 2: AP = 94% 58
- Figura 5.5 Query Hertford 5: AP = 72% 59
- Figura 5.6 Query Cornmarket 5: AP = 51% 59
- Figura 5.7 Query Bodleian 4: AP = 44% 59
- Figura 5.8 Query Christ Church 5: AP = 18% 59

1

INTRODUZIONE

Una rete neurale artificiale è un modello matematico che consente di ricreare all'interno di un computer il funzionamento del cervello umano in modo da imparare dall'esperienza e poter prevedere un risultato a partire dal contesto fornito.

A livello biologico, l'elemento fondamentale del sistema nervoso centrale, di cui fa parte il cervello, è il neurone. Ogni neurone è collegato ad altri neuroni e comunica con loro tramite impulsi elettrici. Da solo infatti non è in grado di svolgere quasi alcun compito, ma, combinati assieme, i neuroni formano un rete in grado di svolgere compiti estremamente complessi. Allo stesso modo, le reti neurali artificiali, sono composte da neuroni.

In figura 1.1 è riportata una rappresentazione schematica di un neurone cerebrale affiancata a quella di un neurone artificiale.

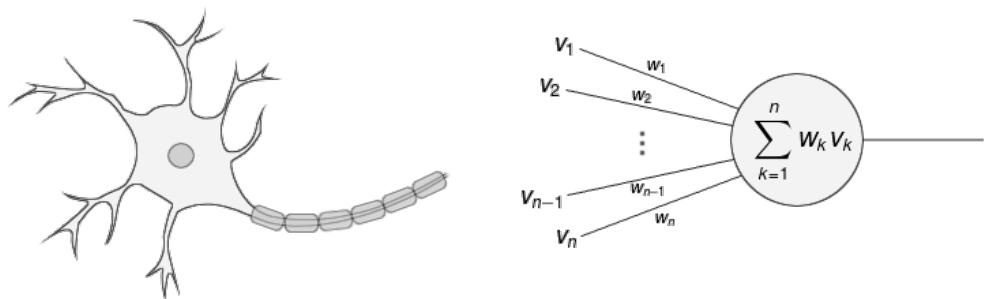


Figura 1.1: Rappresentazione schematica di un neurone e di un neurone artificiale

In questa rappresentazione v_1, v_2, \dots, v_n rappresentano i dati in input, che nel caso del cervello umano sarebbero gli stimoli procurati dai sensi; la linea a destra rappresenta l'output, la reazione a questi stimoli. Il

suo valore può essere continuo $[0, 0.2, 0.5, 1]$, binario $[0, 1]$ o categorico [rosso, giallo, verde, ...]. La parte centrale rappresenta invece il neurone vero e proprio: questo trasforma gli stimoli in reazione, cioè applica una funzione, detta funzione di attivazione, alla somma pesata dei valori ricevuti in input. I pesi, che rappresentano le intensità dei segnali elettrici scambiati tra neuroni, svolgono un ruolo fondamentale: dicono al neurone quali segnali sono rilevanti e quali no.

Una rete neurale risulta quindi formata da 3 tipi di livelli: il livello di input (input layer), i livelli nascosti (hidden layers) e il livello di output (output layer).

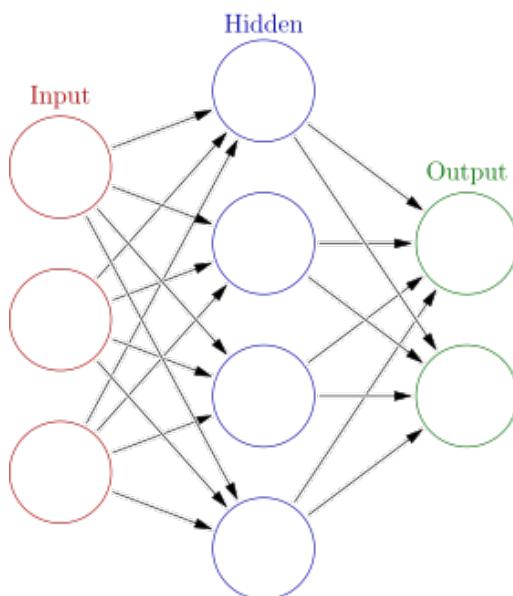


Figura 1.2: Rappresentazione di una rete neurale artificiale

Durante l'addestramento della rete neurale, un dato la attraversa interamente producendo un'etichetta. In base alla correttezza o meno di quest'ultima i pesi di tutti i livelli vengono aggiornati, tramite un meccanismo chiamato *back-propagation*.

La struttura di una rete neurale è mostrata in figura 1.2.

Quando si utilizzano reti neurali in cui i livelli sono molti (indicativamente almeno 10) si parla di apprendimento profondo, *deep learning*. In particolare per apprendimento profondo si intende un insieme di tecniche basate su reti neurali artificiali organizzate in diversi strati, dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa. Grazie alla disponibilità di tecnologie hardware più avanzate che riescono a gestire l'enorme potenza

di calcolo richiesta dalle reti neurali artificiali, esso sta prendendo sempre più campo nella ricerca.

Sono tantissimi gli ambiti studiati dall'apprendimento profondo, noi ci concentreremo sulla *computer vision*, che si occupa di sviluppare sistemi in grado di comprendere la composizione strutturale e semantica di immagini e video. A sua volta essa si articola in più discipline come l'*image classification* il cui scopo è quello di stabilire a quale classe appartiene un'immagine, dato l'insieme delle classi possibili; e l'*object detection* che cerca di rilevare e classificare istanze di oggetti all'interno delle immagini. In questa tesi ci occuperemo invece del *content-base image retrieval* (CBIR) il cui scopo è, data un'immagine detta *query*, trovare all'interno di un database tutte e sole le immagini che raffigurano lo stesso scenario e gli stessi oggetti raffigurati nella query.

In particolare abbiamo studiato come creare un descrittore per le immagini, ovvero una sequenza di numeri che rappresentino il contenuto strutturale e semantico dell'immagine stessa, a partire da una rete neurale. Abbiamo effettuato quindi diversi esperimenti sui metodi di *pooling*, che consentono di ridurre questi descrittori a una lunghezza fissata, cercando di mantenere tutte le informazioni importanti. Infine, abbiamo affrontato problematiche relative alla scarsa quantità di dati disponibili, utilizzando tecniche di *Data Augmentation*.

La tesi è dunque strutturata come segue:

- il Capitolo 1 presenta un'introduzione all'image retrieval, fornendo una panoramica dell'evoluzione delle tecniche sviluppate fino ad oggi per risolvere questo problema;
- nel Capitolo 2 è descritta l'architettura utilizzata, con una spiegazione dettagliata delle tecniche e degli strumenti con cui abbiamo lavorato per lo sviluppo della tesi;
- il Capitolo 3 contiene la descrizione dei principali dataset utilizzati per condurre gli esperimenti;
- infine, il Capitolo 4 riporta gli esperimenti che abbiamo effettuato.

2

IMAGE RETRIEVAL

Il *Content-Based Image retrieval*, abbreviato con CBIR o semplicemente con Image Retrieval diventò un ambito di ricerca molto popolare nei primi anni '90, quando si diffuse la pratica di salvare le immagini in basi di dati all'interno dei computer. Le prime tecniche di memorizzazione erano essenzialmente repliche dei sistemi non digitali. La maggior parte delle basi di dati era indicizzata tramite parole chiave: a ciascuna immagine venivano associate delle parole chiave, che venivano poi utilizzate per ritrovarla quando necessario. Questa tecnica si presta benissimo nell'ambito dei documenti di testo, in cui è stata sviluppata, mentre risulta molto problematica con le immagini perché, contrariamente ad archivi formati da file di testo, non esiste un modo semplice per estrarre informazioni sul contenuto che viene archiviato. Le immagini non contengono informazioni sui loro autori, non hanno titoli o parole chiave. Inoltre la procedura di generazione di parole chiave, è difficile da standardizzare, perché dipende dalla persona che sta osservando l'immagine, ed è una procedura molto lunga. Quindi, da quando le persone hanno iniziato ad immagazzinare grandi quantità di immagini digitali, gli esperti hanno iniziato a studiare metodi per rendere questa procedura più semplice.

Questa ricerca ha portato al CBIR, in cui invece di cercare immagini tramite parole chiave, si utilizza il loro contenuto. Ovviamente per contenuto si intende una codifica matematica che riesca a catturare il valore semantico e strutturale dell'immagine. Il punto focale del CBIR diventa quindi l'estrazione di caratteristiche rappresentative dell'immagine, dette *features*. Come è spiegato nel capitolo successivo, queste features possono essere estratte in modo ingegneristico (metodi SIFT) o automatico, attraverso l'utilizzo di reti neurali convoluzionali.

2.1 STORIA DELL'IMAGE RETRIEVAL

Il campo dell'analisi delle immagini è progredito molto negli ultimi 20 anni. Che si tratti di *image retrieval*, *image tagging* o *image classification*, la sfida che gli studiosi si sono trovati ad affrontare è stata quella di costruire una codifica delle immagini, detta descrittore, che riesca a catturare il valore semantico in esse contenuto, in modo efficace ed efficiente.

Idealmente il descrittore dovrebbe contenere caratteristiche peculiari dell'immagine, dette features: particolari pattern, informazioni sul colore, confini degli oggetti, oggetti stessi, ecc.

Formalmente un'immagine I è rappresentata da un'insieme di features $\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$. L'obiettivo è quello di combinarle in una rappresentazione globale $\Psi(I)$ tale che le rappresentazioni di due immagini contenenti lo stesso oggetto o la stessa scena siano più simili (secondo una misura di distanza) rispetto a quelle di due immagini non correlate.

Dal 2003 al 2012 questa ricerca si è concentrata sui metodi SIFT (*scale-invariant feature transform*, spiegati in dettaglio più avanti), che cercano di localizzare punti di interesse all'interno dell'immagine, che siano stabili a particolari variazioni come cambi di luce, rotazioni o deformazioni, cioè trasformazioni affini. I descrittori SIFT soffrono però del problema della *burstiness*, un fenomeno secondo cui una struttura si ripete molte volte all'interno dell'immagine. Questo fenomeno impatta in modo negativo il calcolo delle similarità fra immagini. Inoltre, il fatto che siano calcolati solamente sulla scala bianco e nero, e che utilizzino solo features locali fa sì che alcuni pattern vengano riconosciuti come simili anche quando appartengono a contesti completamente diversi. Per questo, dal 2012, hanno preso sempre più piede i descrittori basati su Convolutional Neural Networks (CNN – anche queste spiegate meglio in seguito), ottenendo prestazioni migliori in quasi tutti i casi di applicazione [1].

Per creare un descrittore di un'immagine a partire da una rete convoluzionale ci sono 3 fattori da tenere in considerazione:

- Come identificare le features;
- Quali features tenere e quali scartare;
- Come aggregare le features scelte.

Per quanto riguarda l'identificazione delle features si è fatto grande uso di algoritmi di *region proposal* come Selective Search o Edge Boxes, come in [2] e [3]. Il loro problema è che sono molto costosi e tendono a

restituire molte regioni non di interesse. In [4] e [5] vengono quindi proposte delle *region proposal networks* i cui pesi vengono allenati a dare più importanza a regioni di maggiore interesse, mentre in [6] si propongono delle maschere che applicate ai descrittori, selezionino solo le regioni più significative. In [7] invece viene rifiutato del tutto l'approccio delle region proposals e viene sostituito da operazioni di *stochastic random crop pooling* che selezionino in modo molto più rapido, molte zone dell'immagine. Spesso, le reti neurali ad oggi utilizzate, non si basano più su questo tipo di approccio ma sulla capacità di una sequenza di livelli convoluzionali di individuare caratteristiche fondamentali delle immagini sia a basso che ad alto livello.

Un altro fattore rilevante riguarda i livelli della rete neurale da cui estrarre le features. Inizialmente, venivano quasi sempre estratte dall'ultimo livello denso della rete, ottenendo però descrittori globali, senza attenzione al particolare. Si è notato infatti che i dettagli, come texture, spigoli ed angoli, vengono catturati dai primi livelli convoluzionali della rete e sono altrettanto importanti. In [8] e [9] vengono proposti approcci che consentano di unire e valorizzare entrambi i livelli di granularità.

Infine, una volta estratte e selezionate le features è necessario aggregarle in una rappresentazione a lunghezza fissa e compatta. Per far questo si utilizzano solitamente meccanismi di *pooling*, che possono essere quelli classici (max-pooling e avg-pooling) [2] oppure tecniche più elaborate come sum-pooling [10], Fisher Vector [3] o VLAD.

L'evoluzione storica delle tecniche di computer vision è riportata in figura 2.1.

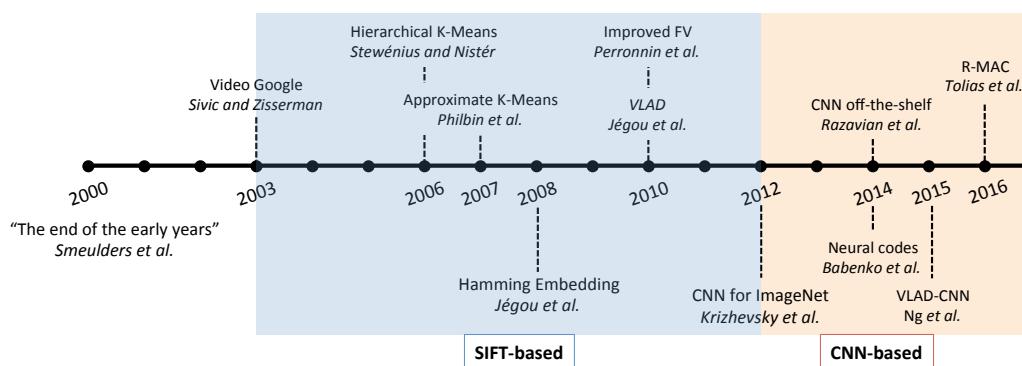


Figura 2.1: Storia delle tecniche utilizzate nell'ambito della computer vision

2.2 DESCRITTORI SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

SIFT è un algoritmo che consente di identificare e descrivere le cosiddette feature locali di un'immagine: caratteristiche relative ad un'area poco estesa dell'immagine come particolari pattern, angoli o spigoli. Per questo scopo identifica alcuni punti di interesse, detti *key points* e gli assegna un descrittore, ovvero un vettore di numeri reali. È importante che questo descrittore sia invariante per alcune trasformazioni (come traslazione, rotazione e cambi di illuminazione) per evitare che immagini che rappresentano gli stessi oggetti, siano classificate come diverse solo perché catturate sotto diverse condizioni. Il processo di costruzione dei descrittori SIFT è articolata nei seguenti passi.

2.2.1 Creazione dello scale space

Il primo passo è la creazione dello *scale space* (spazio in scala) ottenuto sfocando e ridimensionando l'immagine di partenza. Lo scopo dello scale space è quello di simulare il fatto che nella realtà, un oggetto può essere visto a distanze diverse, e a seconda di quanto siamo distanti riusciamo a vedere più o meno dettagli. Per creare questo spazio per prima cosa l'immagine viene resa in bianco e nero, poi viene progressivamente sfocata n volte applicando ai suoi punti la convoluzione di Gauss, illustrata qui di seguito.

$$\begin{aligned} L(x, y, \sigma) &= G(x, y, \sigma) * I(x, y) \\ G(x, y, \sigma) &= \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \end{aligned}$$

dove:

- L è l'immagine sfocata;
- G è l'operatore di Gauss;
- I è l'immagine;
- x e y sono le coordinate del punto considerato;
- σ è il parametro di scala. Più grande è σ maggiore è la sfocatura;
- $*$ è l'operatore di convoluzione in x e y .

Ogni immagine così ottenuta viene dimezzata di dimensione e sfocata nuovamente n volte (l'insieme delle immagini con la stessa dimensione viene indicato come ottava): questa operazione viene ripetuta finché non si ottiene un'immagine troppo piccola per poter essere significativamente processata. Spesso, prima di costruire lo scale space l'immagine di partenza è raddoppiata di dimensioni.

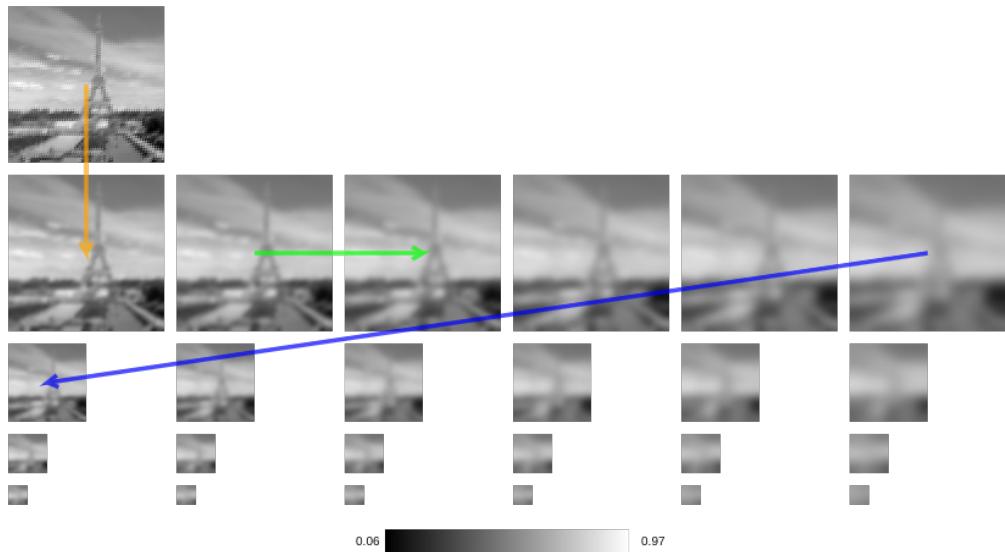


Figura 2.2: SIFT: Scale space

2.2.2 Approssimazioni LoG

Le immagini dello scale space vengono utilizzate per creare un altro insieme di immagini detto "Differenza di Gaussiane" (DoG), che verranno poi utilizzate per identificare i punti chiave. Per ottenerle si utilizza il Laplaciano della funzione Gaussiana (LoG), che è la seguente:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Tale operatore, prima sfoca l'immagine, poi calcola su di essa il Laplaciano, ovvero le derivate seconde. Il primo passo è necessario per ridurre il rumore, a cui la derivata seconda è molto sensibile. Questa operazione consente di trovare gli angoli e gli spigoli dell'immagine, che sono ottimi per identificare i punti chiave, ma calcolare esattamente tutte queste derivate è molto costoso quindi si usa un'approssimazione.

Per calcolare il LoG velocemente si utilizza lo scale space: si calcolano le differenze tra due scale consecutive (appartenenti alla stessa ottava) ottenendo le cosiddette DoG. Esse sono infatti un'ottima approssimazione del LoG, sono molto più semplici da calcolare e sono invarianti per cambiamenti in scala.

Purtroppo però i LoG così ottenuti sono ancora dipendenti dalla quantità di *blur* (sfocatura) utilizzata per costruire lo scale space, il parametro σ . Per ottenere l'indipendenza dobbiamo riuscire a "toglierlo" dalla formula di G: se il LoG è della forma $\nabla^2 G$, allora quello indipendente dalla scala è della forma $\sigma^2 \nabla^2 G$. Fortunatamente i risultati ottenuti tramite le DoG sono già moltiplicati per σ^2 .

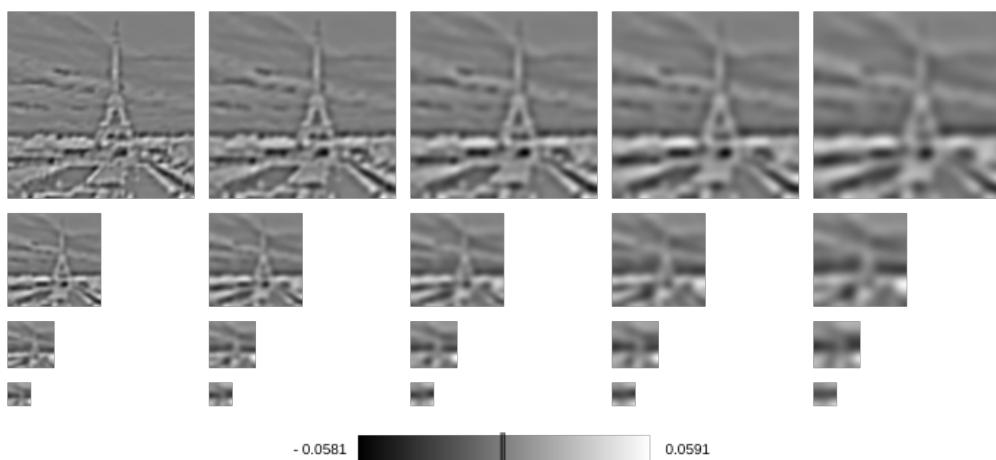


Figura 2.3: SIFT: Differenza di Gaussiane

I LoG sono anche moltiplicati per $(k - 1)$ dove k indica il fattore di sfocatura (quanto blur si applica tra un'immagine e l'altra della stessa ottava). In realtà questo non è un problema perché stiamo cercando la posizione dei massimi e dei minimi, non il loro valore.

2.2.3 Ricerca di massimi e minimi nella DoG

Per trovare i massimi e i minimi nella DoG si itera su tutti i pixel e per ciascuno di essi si controllano i vicini, considerando anche l'immagine dell'ottava precedente e quella successiva, per un totale di 26 punti. Se un punto è più grande o più piccolo di tutti i suoi vicini, allora viene marcato come key point. In realtà gli estremi non giacciono mai esattamente in un pixel, si utilizza quindi l'espansione in serie di Taylor per generare dei

subpixel che consentano di avvicinarsi alla posizione esatta. La formula di Taylor è la seguente:

$$\mathbf{D}(\mathbf{x}) = \mathbf{D} + \frac{\delta \mathbf{D}^T}{\delta \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\delta^2 \mathbf{D}}{\delta \mathbf{x}^2} \mathbf{x}$$

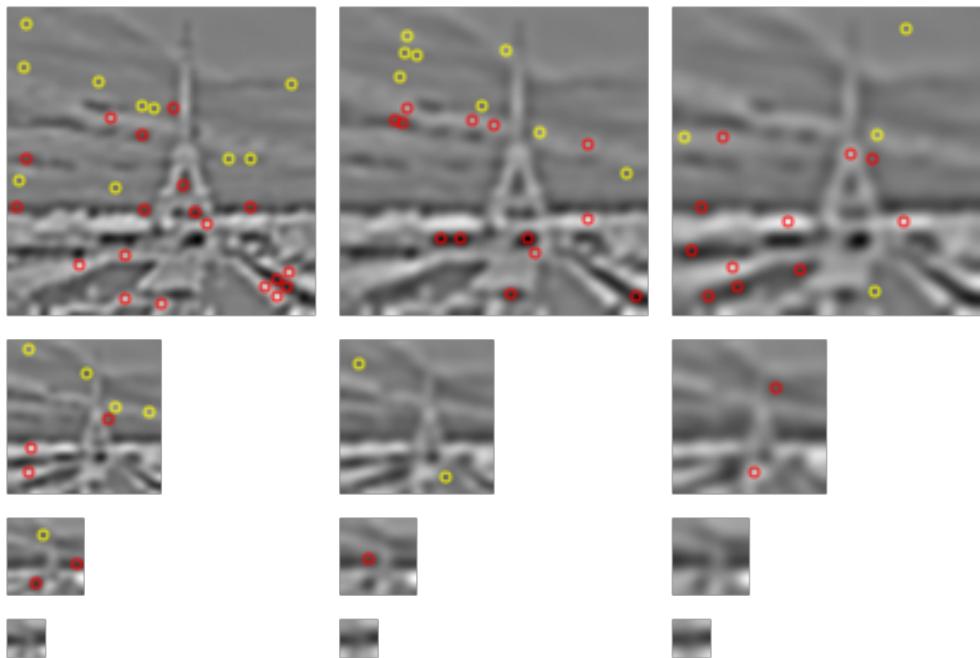


Figura 2.4: SIFT: Ricerca dei key points

2.2.4 Rimozione di key point non significativi

Il passo successivo è quello di escludere i key point non interessanti: alcuni di essi infatti non hanno abbastanza contrasto oppure si trovano negli spigoli degli oggetti, zone poco significative.

Rimuovere i primi è molto semplice: se il loro valore assoluto nel DoG (o nell'espansione di Taylor nel caso dei subpixel) è minore di un certo valore soglia, vengono scartati.

Per identificare i punti lungo gli spigoli l'idea è quella di calcolare due gradienti perpendicolari nel key point. L'intorno del punto chiave può essere di 3 tipi:

- una regione piatta: in questo caso entrambi i gradienti sono piccoli;

- uno spigolo: in questo caso il gradiente perpendicolare allo spigolo è grande, quello parallelo è piccolo;
- un angolo: entrambi i gradienti sono grandi.

Gli unici punti che vengono conservati sono quelli che si trovano negli angoli, sempre che abbiano un valore più alto della soglia. Matematicamente questa operazione viene fatta utilizzando l’Hessiana.

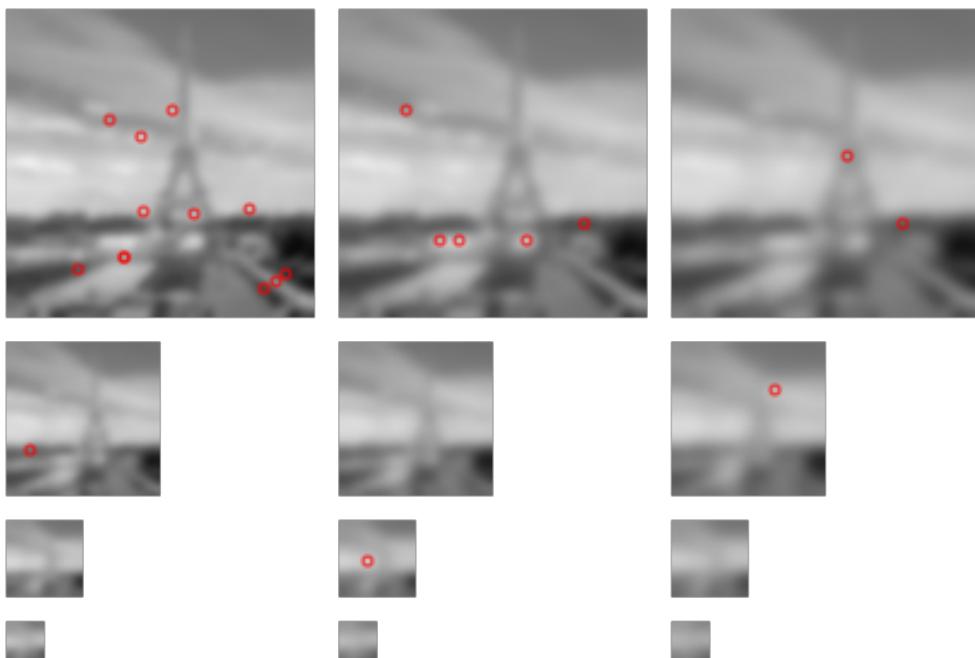


Figura 2.5: SIFT: Key points selezionati

Adesso tutti i key point rimasti si trovano in zone significative dell’immagine, sono stabili e conosciamo la scala a cui sono stati trovati (il livello dell’ottava in cui si trova l’immagine a cui appartengono, all’interno dello scale space) e sono quindi invarianti per variazioni in scala. Nel prossimo passo si ottiene anche l’invarianza per rotazioni.

2.2.5 Invarianza per rotazioni

Per ottenere anche l’invarianza per rotazioni bisogna associare un orientamento a ciascun punto: si calcolano le direzioni θ e i valori m del gradiente intorno a ciascun punto chiave utilizzando le seguenti formule:

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

Questi valori sono calcolati per tutti i pixel intorno al punto chiave e raccolti in un istogramma pesandoli secondo il valore del gradiente. Una volta fatta questa operazione per tutti i punti, l'istogramma evidenzierà un picco in corrispondenza di un valore che sarà l'orientamento scelto. Se ci sono altri picchi con valori maggiori dell'80% del valore del picco massimo, vengono scelti come nuovi punti chiave, con la stessa posizione e scala di quello originale, ma con l'orientamento di quel picco. In pratica l'orientamento può dividere un punto chiave in più punti.

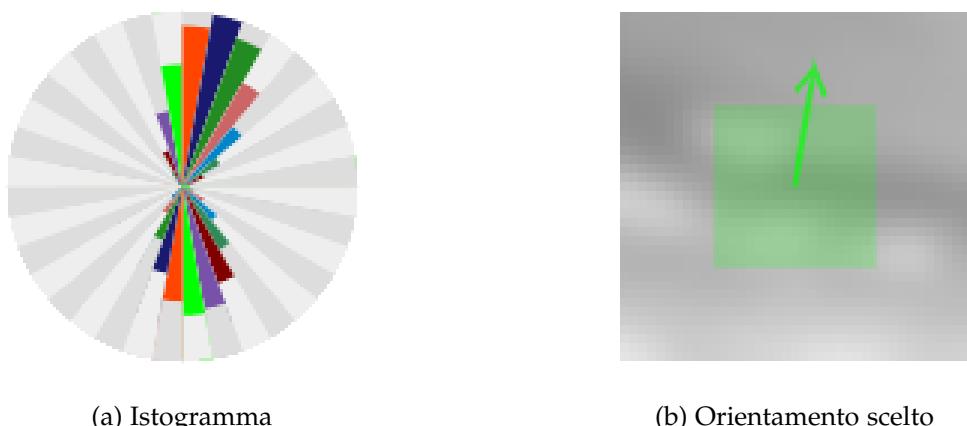


Figura 2.6: Calcolo dell'orientamento in un punto chiave

2.2.6 Generazione delle features locali

L'ultimo passo rimasto è quello di generare una firma per ogni punto chiave che permetta di identificarlo, ovvero di capire su che oggetto giace. La firma deve essere facile da calcolare e non troppo "stringente" poiché immagini diverse che rappresentano lo stesso oggetto non produrranno mai punti chiave identici. Si prende quindi una finestra di dimensione 16×16 divisa in 16 finestre 4×4 intorno al punto chiave, sulla quale vengono calcolate valore e orientamento del gradiente. Gli orientamenti vengono poi inseriti in un istogramma suddiviso in 8 intervalli (ciascuno ampio 45 gradi) in modo proporzionale al loro valore e alla distanza dal

punto chiave. Fatta questa operazione per tutte le regioni si ottengono 128 numeri. Essi vengono normalizzati e formano il vettore delle feature che identifica il punto chiave. In realtà il punto chiave non giace esattamente su un pixel ma a metà tra essi quindi bisogna interpolare l'immagine per generare orientamento e intensità dei gradienti nei punti che si trovano tra più pixel.

Il vettore delle feature ha ancora due problemi:

1. Dipendenza dalla rotazione: ruotando l'immagine, tutti i valori e gli orientamenti calcolati cambiano. Per ottenere l'indipendenza, l'orientamento del gradiente viene calcolato relativamente a quello dei punti chiave;
2. Dipendenza dall'illuminazione: se stabiliamo una soglia per i valori molto grandi otteniamo l'indipendenza per l'illuminazione. Solitamente la soglia scelta è 0.2.

2.3 RETI NEURALI CONVOLUZIONALI

Le reti neurali convoluzionali, anche dette CNN, sono un tipo di rete neurale specializzata nel processare dati che hanno un'organizzazione a griglia. Esempi di questi dati sono le serie temporali, che possono essere viste come griglie 1 – D composte da campionamenti eseguiti a intervalli regolari di tempo, e le immagini, che sono rappresentate da un insieme di griglie 2 – D di pixel. Il nome "reti neurali convoluzionali" implica che esse facciano uso di un'operazione matematica detta convoluzione, al posto della semplice moltiplicazione tra matrici, all'interno di almeno un livello. Grazie all'addestramento di alcuni filtri, queste reti acquisiscono la capacità di rilevare quali sono gli aspetti più importanti di un'immagine, e quella di distinguere un'immagine dalle altre. Inoltre, il pre-processing richiesto da una CNN è molto minore rispetto a quello necessario ad altri algoritmi, e mentre nei metodi precedenti, come SIFT, i filtri erano costruiti in modo ingegneristico, qui vengono imparati in modo automatico.

2.3.1 *Immagine in input*

Solitamente le immagini vengono fornite alle CNN sotto forma di matrici della forma ($\text{larghezza} \times \text{altezza} \times \text{canali}$), dove i canali contengono le intensità della rappresentazione bianco e nero o RGB o una equivalente.

Se le immagini sono molto grandi diventa computazionalmente molto costoso elaborarle. Lo scopo delle reti neurali convoluzionali è quello di ridurre l'immagine in una forma più rapida e semplice da elaborare, preservando tutte informazioni importanti.

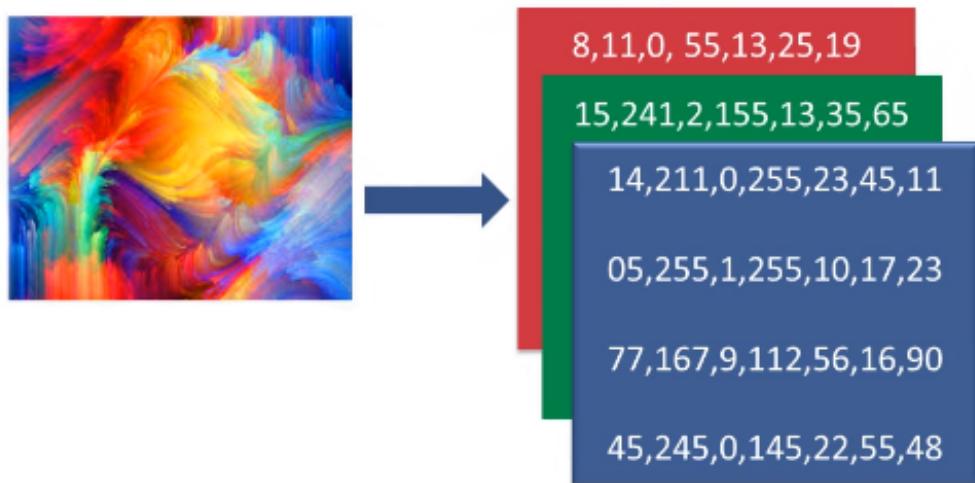


Figura 2.7: Immagine RGB

2.3.2 Convoluzione

In matematica la convoluzione è un integrale che calcola quanto due funzioni si sovrappongono quando una passa sopra l'altra. Nell'ambito dell'analisi di immagini, le "funzioni" tra cui è applicata la convoluzione sono l'immagine presa in input (rappresentata da una matrice) e un filtro (rappresentato anch'esso da una matrice) che viene fatto scorrere su di essa.

In figura 2.8 è mostrata in dettaglio l'operazione, considerando come filtro la matrice

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Il filtro scorre sull'immagine verso destra e verso il basso, muovendosi di un numero fissato di celle detto *stride*, in questo caso pari a 1, fino a che non ha attraversato tutta l'immagine. Il risultato della convoluzione viene

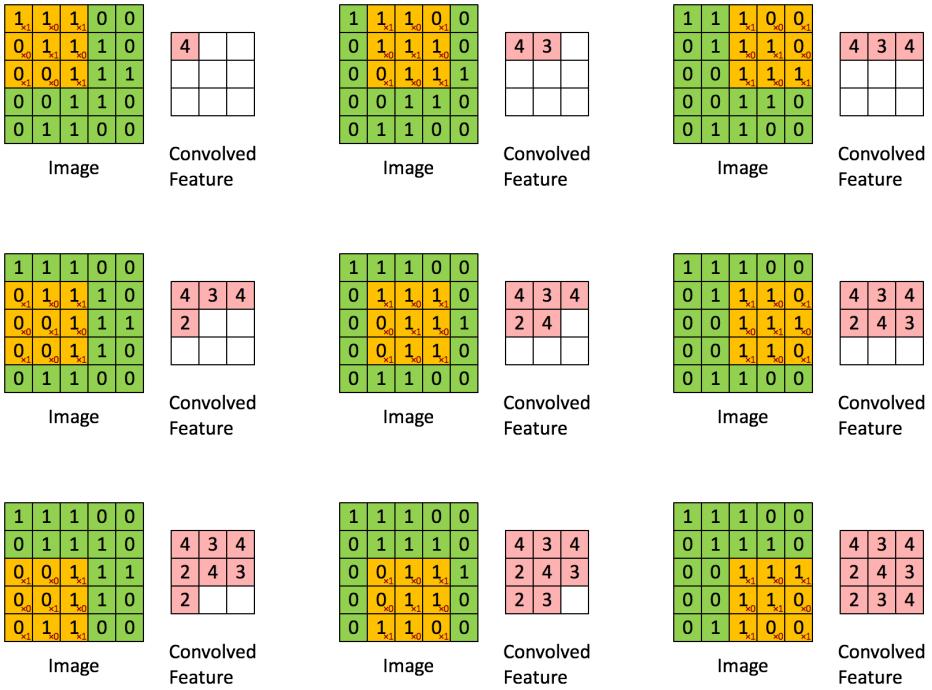


Figura 2.8: Convoluzione

detto *feature map*. Nel caso di immagini con più canali (per esempio RGB), il filtro ha lo stesso numero di canali dell'immagine. Il valore della convoluzione è ottenuto facendo prima un prodotto puntuale tra gli elementi del filtro e della porzione di immagine a cui esso si sovrappone, poi la somma dei valori ottenuti. Se entrambe le matrici hanno valori alti nelle stesse posizioni, questo prodotto sarà grande. Altrimenti, sarà piccolo. In questo modo sappiamo se l'area d'immagine sovrapposta al filtro, contiene il pattern che esso rappresenta. I primi livelli convoluzionali hanno il compito di catturare feature di basso livello, come bordi degli oggetti, colore, orientamento del gradiente, ecc. Man mano che si aggiungono livelli, la ricerca si concentra su feature più di alto livello, ottenendo una rete neurale che riesce ad avere una comprensione globale dell'immagine. Questa operazione riduce anche la dimensione dell'immagine.

2.3.3 Receptive field

Il *receptive field*, in italiano *campo recettivo*, è definito come quella regione dell'input che viene utilizzata da un'unità di un livello convoluzionale

per calcolare il valore del suo output. La figura 2.9 aiuterà a comprendere meglio questo concetto. Supponiamo di utilizzare un solo filtro per ogni livello convoluzionale, che A, B e C siano 3 livelli di una CNN e che la dimensione del filtro sia 3×3 . Il campo recettivo di B(2, 2) è il quadrato A(1 : 3, 1 : 3); quello di B(4, 2) è il quadrato A(3 : 5, 1 : 3); quello di C(3, 3) è B(2 : 4, 2 : 4) che a sua volta riceve in input A(1 : 5, 1 : 5) e così via.

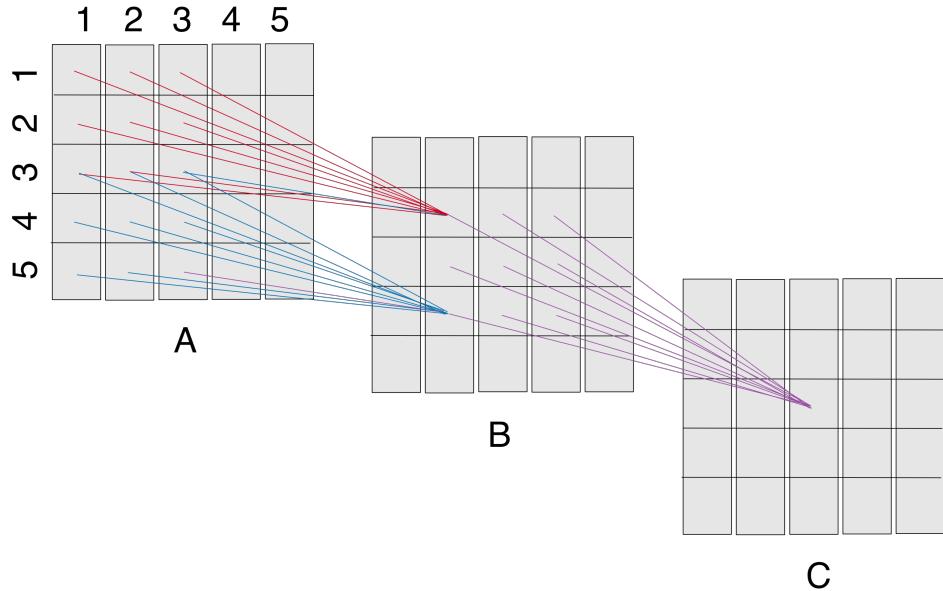


Figura 2.9: CNN: Receptive Field

I campi recettivi del livello B hanno dimensione 3×3 , mentre quelli del livello C sono di dimensione 5×5 . Se aggiungessimo un livello D, i suoi campi recettivi avrebbero dimensione 7×7 . Questo mette in evidenza il fatto che utilizzare una sequenza di livelli convoluzionali con filtri di dimensioni ridotte, porta ad avere campi recettivi che si otterrebbero altrimenti solo grazie all'utilizzo di filtri di grandi dimensioni.

Un aspetto fondamentale è che non tutti i pixel all'interno del campo recettivo sono ugualmente importanti: più un pixel è vicino al centro del campo, maggiore è il suo contributo al calcolo della feature map.

2.3.4 Pooling

Soltamente, dopo un livello convoluzionale, si trova un livello di sottocampionamento, *pooling*. Questo tipo di livello ha l'obiettivo di estrarre le

feature più importanti, diminuendo di conseguenza le dimensioni delle immagini e riducendo il costo della computazione.

In questo caso immaginiamo di far scorrere una matrice vuota sopra l'immagine, nello stesso modo in cui scorre il filtro in un livello convoluzionale: l'operazione di pooling viene applicata, ad ogni passo, alle celle dell'immagine a cui si sovrappone la matrice.

I due tipi di pooling più utilizzati sono:

- Max-pooling: ogni cella contiene il valore massimo della sottomatrice di partenza;
- Avg-pooling: ogni cella contiene la media dei valori della sottomatrice di partenza.

Un esempio di max-pooling è rappresentato in figura 2.10.

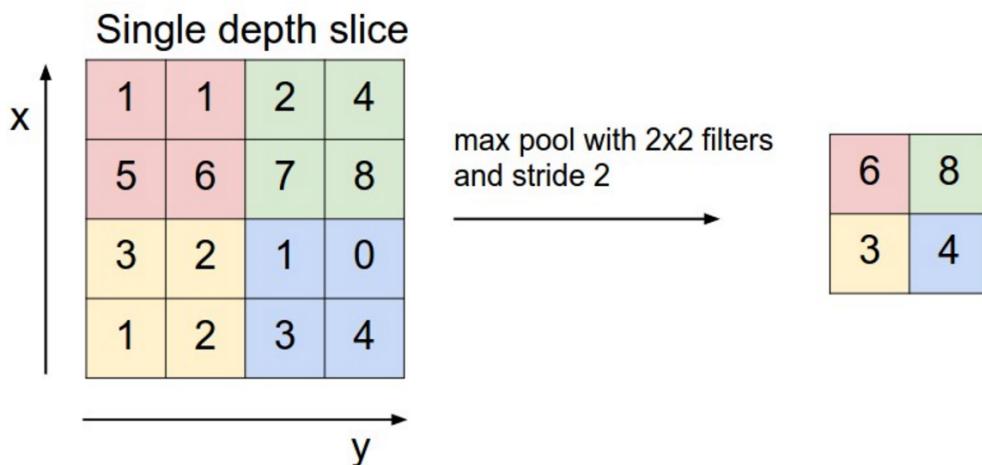


Figura 2.10: Esempio di max-pooling

Dopo aver attraversato un certo numero di livelli di convoluzione e di pooling, l'immagine è stata ridotta a un insieme più piccolo di matrici che possono essere utilizzate come descrittori, date in input ad una rete di classificazione, di object detection, ecc.

3

STRUMENTI UTILIZZATI

Prima di procedere con gli esperimenti è necessario introdurre alcuni concetti e strumenti che abbiamo utilizzato per condurli.

3.1 FRAMEWORK DI VALUTAZIONE

Abbiamo già detto come l’obiettivo dei sistemi di Image Retrieval sia quello di generare descrittori per le immagini, in modo che immagini che rappresentano gli stessi oggetti e scene risultino vicini secondo una funzione di distanza, mentre descrittori di immagini che rappresentano oggetti e scene differenti, risultino lontani. Vogliamo quindi costruire un framework che, dati tutti i descrittori di tutte le immagini, ci indichi quanto buone sono le prestazioni ottenute. Tutti i dataset che utilizzeremo saranno divisi in due gruppi:

- Immagini di query: immagini utilizzate per simulare una ricerca nel dataset;
- Immagini non di query: tutte le altre immagini del dataset, su cui viene effettuata la ricerca.

Sostanzialmente per ogni immagine query si effettua una ricerca nell’intero dataset e si genera una lista ordinata di immagini dove le prime sono le più vicine all’immagine di query e le ultime sono le più lontane.

Ma come si capisce quanto sono distanti due immagini? Dato che i descrittori non sono altro che vettori di numeri reali, tutti della stessa dimensione, per capire quanto siano vicine due immagini è sufficiente scegliere una misura di distanza. La metrica che abbiamo utilizzato noi è la distanza del coseno.

3.1.1 Distanza del coseno

La *distanza del coseno* è definita a partire dalla *similarità del coseno* che si calcola come segue. Dati due vettori non nulli \mathbf{X} e \mathbf{Y} ,

$$\text{cosine_similarity} = \frac{\mathbf{X} \cdot \mathbf{Y}}{\|\mathbf{X}\| \|\mathbf{Y}\|} = \frac{\sum_{i=1}^n X_i Y_i}{\sqrt{\sum_{i=1}^n X_i^2} \sqrt{\sum_{i=1}^n Y_i^2}}$$

Il valore varia tra -1 , che indica che \mathbf{X} e \mathbf{Y} sono esattamente opposti, a 1 che indica che \mathbf{X} e \mathbf{Y} sono identici. Semplicemente, la distanza del coseno si ottiene sottraendo ad 1 questa quantità:

$$\text{cosine_distance} = 1 - \text{cosine_similarity}$$

Essa varia tra 0 e 2 e a valori più grandi corrisponde una maggiore differenza.

Utilizzando questa formula si calcolano i vettori delle distanze tra ogni immagine di query e tutte le altre immagini, e lo si ordina in modo decrescente. In questo modo le prime immagini di ogni vettore saranno quelle considerate più simili alla query, le ultime quelle più diverse. Dopo aver simulato la ricerca serve una metrica che ci indichi quanto siano buone le prestazioni ottenute. Solitamente, nell'ambito dell'Image Retrieval viene utilizzata la *Mean Average Precision*.

3.1.2 Mean Average Precision

La *Mean Average Precision* (mAP) è una metrica che può essere utilizzata per valutare le prestazioni dell'image retrieval.

Prima di introdurre la mAP è necessario introdurre altre due misure: la *precision* P e la *recall* R. La precision misura la percentuale di predizioni corrette, mentre la recall misura la percentuale di predizioni corrette recuperate rispetto al totale. Le loro formule matematiche sono le seguenti:

$$P = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$R = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

Se si calcola il grafico della precision in funzione della recall, $P(R)$, la *Average Precision* (AP) indica il valore medio di $P(R)$ nell'intervallo $[0, 1]$ e si calcola quindi con l'integrale:

$$AP = \int_0^1 P(R) dR$$

A partire dalle precedenti espressioni, la mean average precision viene definita come la media delle AP delle varie classi:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i$$

3.2 TRANSFER LERANING

Capita molto raramente di addestrare un'intera rete neurale convoluzionale partendo da zero poiché è piuttosto difficile riuscire a procurarsi un dataset del dominio di interesse abbastanza grande da consentirlo. Solitamente si addestra la rete su di un dataset simile molto grande (per esempio su ImageNet che contiene 1.2 milioni di immagini di 1000 categorie diverse) e si utilizza questa rete neurale come punto di partenza o come estrattore di feature per il nostro scopo particolare.

Questo è un esempio di *Transfer Learning*, ovvero dell'utilizzo di conoscenze sviluppate nella risoluzione di un problema, per risolvere un altro problema. Formalmente la definizione di Transfer Learning viene data in termini di dominio e obiettivo (*task*). Il dominio \mathcal{D} è composto da uno spazio delle feature \mathcal{X} e da una distribuzione di probabilità marginale $P(X)$ con $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. Dato un dominio specifico $\mathcal{D} = \{\mathcal{X}, P(X)\}$, si definisce task una coppia $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$, dove \mathcal{Y} è lo spazio delle etichette, e $f(\cdot)$ è la funzione di predizione. Questa funzione viene imparata tramite l'addestramento dei dati organizzati in coppie $\{x_i, y_i\}$ dove $x_i \in \mathcal{X}$ è un dato (per esempio un'immagine) e $y_i \in \mathcal{Y}$ è un'etichetta (che, per esempio, può rappresentare la scena contenuta nell'immagine). La funzione $f(\cdot)$ può poi essere utilizzata per predire l'etichetta $f(x)$, di una nuova istanza. Dati un dominio sorgente \mathcal{D}_S , un task di apprendimento su di esso \mathcal{T}_S , un dominio target \mathcal{D}_T e un task di apprendimento su di esso \mathcal{T}_T , l'obiettivo del Transfer Learning è quello di migliorare l'apprendimento della funzione predittiva del target $f_T(\cdot)$ su \mathcal{D}_T , utilizzando la conoscenza su \mathcal{D}_S e \mathcal{T}_S .

Due esempi pratici di transfer learning utilizzati con le reti neurali convoluzionali sono i seguenti:

- Utilizzare una CNN come un estrattore di feature: si prende una CNN già addestrata su un grande dataset, si toglie l'ultimo livello, quello di classificazione, e si utilizza tutto il resto della rete come se fosse un estrattore di feature.

- Fare *Fine-Tuning* sulla CNN: in questo caso, oltre a togliere il livello di classificazione, si addestra nuovamente la rete utilizzando il nuovo dataset. In questo modo si possono modificare i pesi della rete (tutti o solo degli ultimi livelli) per adattarli al nuovo obiettivo. Questo metodo è motivato dal fatto che i primi livelli di una rete convoluzionale solitamente trovano feature più generali, utilizzabili per task diversi, mentre con l'aumentare della profondità, si trovano feature sempre più specifiche per il task su cui la rete è stata addestrata.

Durante lo sviluppo della tesi sono state utilizzate entrambe le tecniche. La differenza tra i metodi di apprendimento tradizionali e il transfer learning è illustrata in figura 3.1.

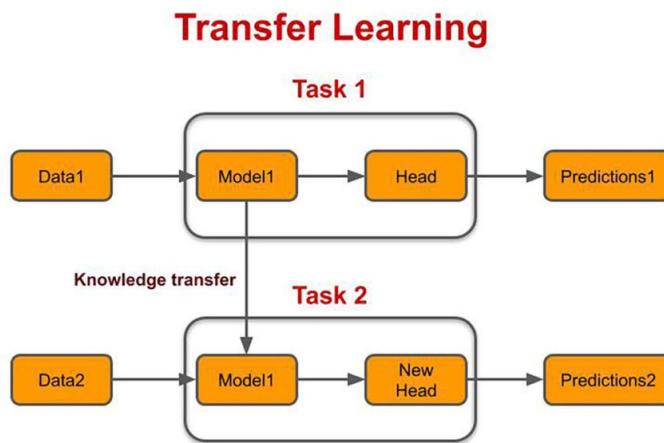


Figura 3.1: Differenza apprendimento tradizionale e transfer learning

3.3 VGG16

La rete neurale utilizzata per tutti gli esperimenti è l'implementazione in Keras di VGG16 [11]. VGG16 è una rete neurale convoluzionale sviluppata nel 2014 per l'*ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), una competizione annuale composta da due sfide: una di localizzazione degli oggetti di 200 tipi diversi all'interno delle immagini, e una di classificazione delle immagini tra 1000 possibili categorie. La rete vinse il primo ed il secondo posto.

Per quanto riguarda la sua struttura, illustrata in figura 3.2, la dimensione dell'input è fissata a $224 \times 224 \times 3$, dove il 3 rappresenta il numero

di canali della rappresentazione RGB. L'immagine passa attraverso una serie di livelli convoluzionali con filtri di dimensione 3×3 , la dimensione più piccola che consente di catturare il concetto di alto/basso, destra/sinistra e centro. Lo stride è sempre posto a 1. Dopo ogni gruppo di livelli convoluzionali, c'è un livello di max-pooling, dove l'operazione di pooling viene effettuata utilizzando finestre di dimensione 2×2 con stride pari a 2. In fondo alla rete ci sono 3 livelli densi: i primi due composti da 4096 nodi, il terzo con 1000 nodi per la classificazione di ImageNet. A tutti questi livelli è applicata la funzione di attivazione ReLu, eccetto alla fine dove troviamo un livello di Softmax. È inoltre stata la prima rete neurale a preferire l'utilizzo di una lunga serie di filtri convoluzionali di dimensione 3×3 , invece di pochi livelli convoluzionali composti da filtri più grandi. Questa scelta architettonica si è rivelata efficiente sia a livello di prestazioni che a livello computazionale, tanto che ad oggi si preferiscono in generale reti neurali "profonde" rispetto a reti neurali più "larghe".

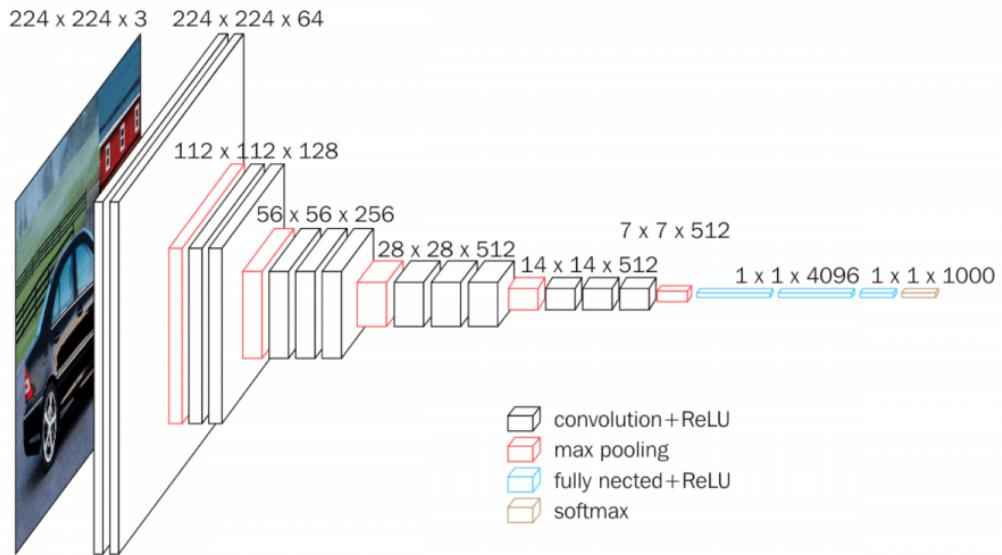


Figura 3.2: Struttura di VGG16

3.4 TRIPLET LOSS

Per addestrare una rete neurale tramite back-propagation è necessario utilizzare una funzione di *loss*, che dopo ogni epoca indichi alla rete in quale direzione spostare i valori dei pesi. Una funzione di loss che si

presta bene per l'Image retrieval è la *triplet loss* [12]. La triplet loss è una funzione che è stata ideata per imparare dei buoni descrittori per i volti per il task di *face recognition*. La sua struttura permette di lavorare con un numero di classi non prefissato e di avere ottime prestazioni anche nella gestione di immagini di classi mai viste. Il principio che segue è il seguente:

- Due immagini della stessa classe devono avere descrittori vicini nello spazio dei descrittori;
- Due immagini di classi diverse devono avere descrittori lontani tra di loro.

Questo risultato si ottiene addestrando la rete a discriminare delle triplettre formate da:

- un'immagine **ancora A**;
- un'immagine **positiva P**, della stessa classe di A;
- un'immagine **negativa N** di una classe diversa da quella di A.

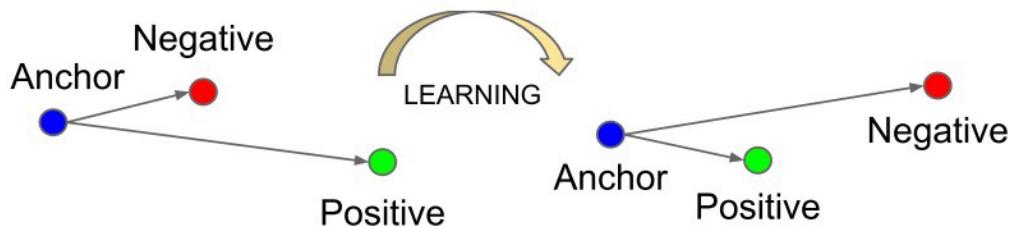


Figura 3.3: Triplet loss

La rete deve imparare a produrre dei descrittori tali che la distanza tra A ed N sia maggiore della distanza tra A e P di un certo margine. Formalmente

$$d(A, N) \geq d(A, P) + \alpha$$

dove d è una funzione di distanza e $\alpha > 0$ è un valore fissato detto margine. Sia quindi il contributo di ogni tripletta:

$$\mathcal{L}(A_i, P_i, N_i) = \max(d(A, P) - d(A, N) + \alpha, 0)$$

la triplet loss è data da:

$$\mathcal{L} = \sum_i \mathcal{L}(A_i, P_i, N_i)$$

Minimizzare questa funzione corrisponde a spingere $d(A, P)$ verso lo 0 e $d(A, N)$ a essere maggiore di $d(A, P) + \alpha$.

In base alle precedenti definizioni si distinguono 3 categorie di triplette:

- **triplette facili**: quelle per cui la loss vale 0 in quanto $d(A, P) + \alpha < d(A, N)$
- **triplette difficili**: quelle per cui N è più vicino ad A di quanto lo sia P, cioè $d(A, N) < d(A, P)$
- **triplette semi-difficili**: quelle per cui N non è più vicino ad A di P, ma per cui la loss è ancora positiva: $d(A, P) < d(A, N) < d(A, P) + \alpha$

Queste definizioni dipendono da dove si trova N relativamente ad A e P, quindi si possono estendere queste 3 definizioni anche ai negativi: negativi difficili, negativi semi-difficili e negativi facili.

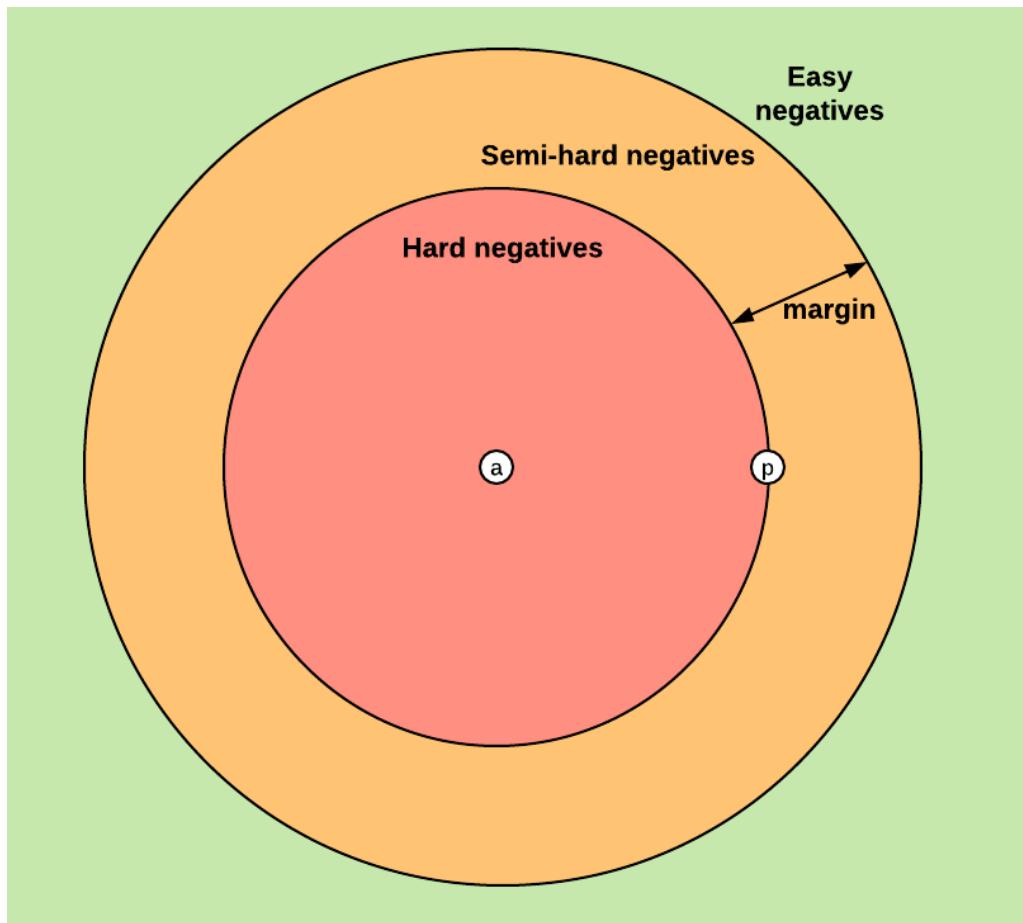


Figura 3.4: Esempi di negativi per la triplet loss

Ovviamente, per poter utilizzare questa formula è necessario avere almeno due immagini di ogni classe all'interno di ogni batch.

L'implementazione della triplet loss che abbiamo utilizzato è una versione molto semplice da utilizzare, ma che non consente di fare il *triplet mining*: la ricerca e l'utilizzo durante l'addestramento della rete di triplette difficili, in modo da migliorare le prestazioni.

3.5 NETVLAD

NetVLad [13] è un livello ispirato a VLAD, implementato in modo da poter essere collegato in fondo ad una rete neurale convoluzionale e addestrato in modo end-to-end tramite back-propagation. Come ogni livello di pooling, il suo scopo è quello di ridurre le dimensioni dei descrittori calcolati dalla rete ad una dimensione fissata.

Si considera quindi la rete neurale come un estrattore di feature globali: l'output dell'ultimo livello convoluzionale è una tabella di feature di dimensione $H \times W \times D$ che può essere vista come un insieme di descrittori D -dimensionali estratti da $H \times W$ posizioni. Presi in input N descrittori locali D -dimensionali $\{x_i\}$, e K centri di cluster $\{c_k\}$, l'output di VLAD è una rappresentazione dell'immagine $K \times D$ -dimensionale V , calcolata come segue:

$$V(j, k) = \sum_{i=1}^N a_k(x_i)(x_i(j) - c_k(j))$$

dove $x_i(j)$ e $c_k(j)$ sono rispettivamente la j -esima dimensione dell' i -esimo descrittore e il centro del k -esimo cluster. $a_k(x_i)$ vale 1 se il descrittore x_i appartiene al k -esimo cluster, 0 altrimenti. Intuitivamente, ciascuna colonna D -dimensionale k di V contiene le somme delle distanze $(x_i - c_k)$ dei descrittori che sono assegnati al cluster k , dal centro del cluster. V viene poi normalizzata (con norma L2) per colonne, convertita in un vettore, e normalizzata di nuovo globalmente.

A partire da questa formula è stato costruito un livello addestrabile in maniera end-to-end: questa condizione richiede che tutte le operazioni che esso compie siano derivabili rispetto a tutti i suoi parametri e rispetto all'input. La discontinuità in VLAD è data da $a_k(x_i)$, che assume due valori discreti. Per risolvere questo problema si rende il parametro continuo nel modo seguente:

$$\bar{a}_k(x_i) = \frac{e^{-\alpha \|x_i - c_k\|^2}}{\sum_{k'} e^{-\alpha \|x_i - c_{k'}\|^2}}$$

In questo modo il peso di x_i viene assegnato al centro c_k in modo proporzionale alla loro prossimità e tenendo conto degli altri cluster. $\alpha > 0$ è un parametro che controlla il decadimento del peso all'aumentare della distanza. Per $\alpha \rightarrow +\infty$ si ottiene il VLAD di partenza. Semplificando $e^{-\alpha \|x_i\|^2}$ tra numeratore e denominatore e sostituendo $w_k = 2\alpha c_k$ e $b_k = -\alpha \|c_k\|^2$ si ottiene:

$$V(j, k) = \sum_{i=1}^n \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_k^T x_i + b_{k'}}} (x_i(j) - c_k(j))$$

dove $\{w_k\}, \{b_k\} \{c_k\}$ sono insiemi di parametri addestrabili per ciascun cluster k . Sono 2 parametri in più rispetto alla versione originale di VLAD, e si ha quindi più flessibilità. È importante ricordare queste definizioni perché serviranno per inizializzare i pesi di NetVLAD.

NetVLAD può anche essere visto come un meta-livello poiché è composto da una serie di livelli convoluzionali collegati tra loro come mostrato in figura 3.5.

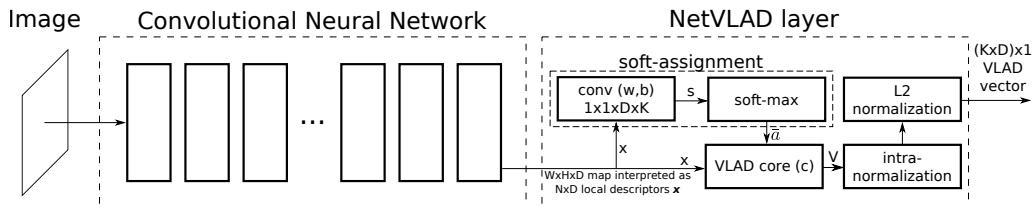


Figura 3.5: Struttura di NetVLAD

3.6 DATA AUGMENTATION

Per ottenere buoni risultati con l'addestramento di reti neurali è spesso necessario avere un dataset abbastanza grande. Il rischio principale che si corre quando si hanno poche immagini, è quello di non riuscire a capire quali siano le feature caratteristiche delle immagini che permettono di distinguere gli oggetti e le scene tra di loro. Un altro problema è che poche immagini, quindi pochi scatti dello stesso oggetto sotto diverse condizioni, non rendono il modello robusto nei confronti di trasformazioni come cambi di prospettiva e di luce. Uno strumento utile per ridurre questo problema è la *Data Augmentation*.

La più semplice forma di data augmentation consiste nell'applicare casualmente alle immagini in input delle trasformazioni geometriche come rotazione, inversione secondo l'asse orizzontale o verticale, modifiche sul colore.

In figura 3.6 sono riportate alcune delle trasformazioni più utilizzate:

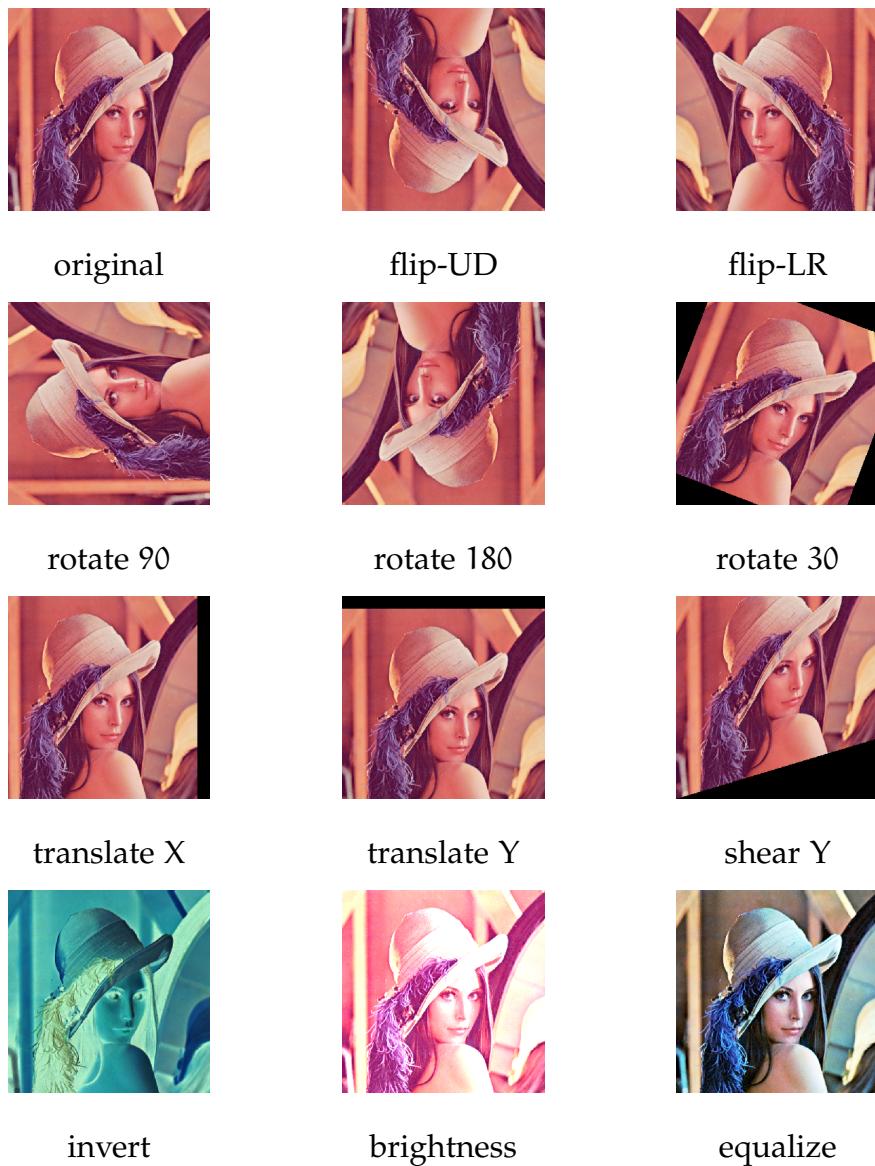


Figura 3.6: Esempi di data augmentation

Questa tecnica, che può sembrare semplice, ha in realtà un gran numero di iperparametri: quante trasformazioni applicare ad un'immagine, quali applicare e con quale intensità. I metodi più naïve, scelgono casualmente una trasformazione per ciascuna immagine e con intensità casuale o fissata a priori, portando spesso a risultati non soddisfacenti. Altre tecniche si basano invece sulla conoscenza del dominio su cui applicare la data augmentation: conoscendo bene il dataset utilizzato si può intuire quali siano le trasformazioni che possono effettivamente rivelarsi utili. Questo approccio non è però generalizzabile.

La ricerca si è quindi mossa verso lo sviluppo di algoritmi che riuscissero ad ottimizzare questi iperparametri per ottenere una data augmentation ottimale per il dataset considerato.

3.6.1 AutoAugment

AutoAugment [14] è una procedura che ha l'obiettivo di trovare l'insieme di operazioni ottimale da applicare ad un certo dataset per effettuare la data augmentation. Si cerca di identificare delle sequenze di operazioni dette *policy* che a loro volta sono suddivise in *sub-policy*. Ciascuna sub-policy è una coppia di operazioni a cui sono associate probabilità e intensità di applicazione. Formalmente:

- $\text{policy} = (\text{subpolicy}_1, \text{subpolicy}_2, \dots, \text{subpolicy}_n)$
- $\text{subpolicy}_i = (\text{op}_{i1}, \text{op}_{i2})$
- $\text{op}_j = (\text{trasformazione}_j, p_j, m_j)$

Una volta identificato l'insieme delle policy, quando andremo ad applicare la data augmentation, per ciascuna immagine verrà selezionata in modo casuale una sub-policy. Le operazioni tra cui scegliere per la loro costruzione sono:

- | | | |
|---------------|----------------|------------------|
| • invert | • autoContrast | • equalize |
| • rotate | • solarize | • color |
| • posterize | • contrast | • brightness |
| • sharpness | • shear-X | • shear-Y |
| • translate-X | • translate-Y | • sample pairing |

Il problema della ricerca delle policy viene formulato come problema discreto ed è caratterizzato da due componenti: un algoritmo di ricerca e uno spazio di ricerca. Le operazioni tra cui scegliere sono 16 e a ciascuna di esse è associato un intervallo diverso entro cui scegliere l'intensità di applicazione. Questo intervallo viene però discretizzato uniformemente in 10 valori, in modo da poter usare un algoritmo di ricerca discreta. Allo stesso modo le probabilità di applicare un'operazione vengono discretizzate in 11 valori (considerando anche i valori 0 e 1). Quindi lo spazio di ricerca per ogni sub-policy è composto da $(16 \times 10 \times 11)^2$ possibilità. Trovare una policy, che nel caso di AutoAugment è composta da 5 sub-policy, comporta quindi la ricerca in uno spazio di dimensione $(16 \times 10 \times 11)^{10} \approx 2.0 \times 10^{32}$. L'algoritmo di ricerca utilizzato è Reinforcement Learning. Questo algoritmo ha due componenti: un controllore, che è una RNN, e un algoritmo di addestramento, Proximal Policy Optimization Algorithm [15]. Ad ogni passo questo algoritmo genera una policy S , che viene poi utilizzata per addestrare il controllore con un segnale di ricompensa, che indica quanto è utile S per aumentare la generalizzazione del modello, calcolata in termini di accuracy nel validation set. Il meccanismo utilizzato è illustrato in figura 3.7. In media, il controllore genera 15.000 policy durante l'addestramento.

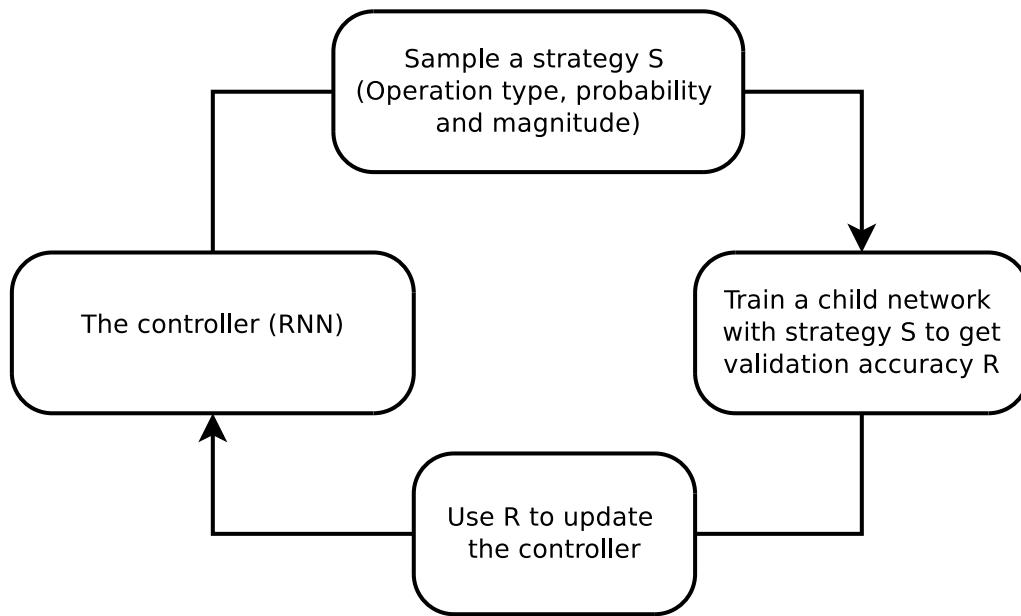


Figura 3.7: Algoritmo di ricerca delle policy

Un esempio di policy è riportato in figura 3.8. Per esempio la prima sub-policy, applica l'operazione ShearX con intensità 7 e probabilità 90%,

seguita dall'operazione `Invert` con probabilità 20% (in questo caso l'intensità non ha significato in quanto semplicemente l'immagine viene specchiata lungo l'asse Y). La seconda policy applica invece per prima cosa l'operazione `shearY` con probabilità 70% e intensità 6, poi applica `Solarize` con probabilità 40% ed intensità 8. Come si può notare la stessa policy, applicata alla stessa immagine può dare risultati diversi poiché potrebbero venire applicate entrambe le operazioni, una delle due, o nessuna.

	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						
	ShearX, 0.9, 7 Invert, 0.2, 3	ShearY, 0.7, 6 Solarize, 0.4, 8	ShearX, 0.9, 4 AutoContrast, 0.8, 3	Invert, 0.9, 3 Equalize, 0.6, 3	Invert, 0.9, 3 AutoContrast, 0.7, 3	ShearY, 0.8, 5 AutoContrast, 0.7, 3

Figura 3.8: Esempio di policy

Con questo metodo sono state generate le policy ottimali per ImageNet, SVHN e CIFAR-10, ma una delle tesi degli sviluppatori è che, oltre a poter generare delle policy addestrando AutoAugment direttamente sul dataset utilizzato, si possono anche utilizzare policy imparate su altri dataset ed ottenere comunque buone prestazioni.

3.6.2 RandAugment

Il principale problema di AutoAugment è che, essendo lo spazio di ricerca molto grande, il costo computazionale del suo addestramento non è alla portata di tutti gli hardware. Questo implica che se le policy da loro generate non sono ottimali per il dataset di interesse, è probabile che non si abbia modo di generare policy "ad hoc". In alcuni casi questo problema viene affrontato effettuando la ricerca in un dataset ridotto, che però porta a risultati non ottimali. L'obiettivo di RandAugment [16] è quello di ridurre notevolmente lo spazio di ricerca in modo da poter essere addestrato nell'intero dataset di interesse.

A questo scopo, i parametri del modello sono stati ridotti a 2:

- N: numero di trasformazioni da applicare ad ogni immagine;
- M: intensità di applicazione di tutte le trasformazioni.

Le trasformazioni possibili sono $K = 14$:

- | | | |
|---------------|----------------|--------------|
| • identity | • autoContrast | • equalize |
| • rotate | • solarize | • color |
| • posterize | • contrast | |
| • sharpness | • shear-X | • brightness |
| • translate-X | • translate-Y | • shear-Y |

Durante la procedura vengono scelte in modo casuale, ciascuna con probabilità $\frac{1}{K}$. Per ogni immagine si sceglie quindi tra K^N possibili policy. Per quanto riguarda M, ogni trasformazione ha una scala di intensità, che varia da 0 a 30, dove 30 indica il valore massimo. Solitamente gli algoritmi di data augmentation cercano di trovare un valore per ogni trasformazione possibile, mentre in questo approccio si ricerca un unico valore globale per M.

3.7 PCA

Durante lo sviluppo della tesi abbiamo usato più volte la PCA (analisi delle componenti principali). Essa è un metodo che rientra nei problemi di trasformazione lineare, utilizzata soprattutto per l'estrazione delle caratteristiche e la riduzione della dimensionalità. Permette infatti di trovare le direzioni di massima varianza nei dati ad alta dimensione e di proiettarle su un nuovo sottospazio con dimensioni uguali o inferiori a quello originale. Utilizzando la proiezione matematica, l'insieme originale dei dati, che potrebbe articolarsi in numerose variabili, viene descritto da poche variabili, dette componenti principali. Le componenti principali godono di 3 proprietà:

- sono combinazioni lineari delle variabili originali;
- sono ortogonali tra di loro;
- la variazione presente tra di esse diminuisce man mano che ci spostiamo dalla prima all'ultima.

Il calcolo della PCA può essere fatto in molti modi, in seguito è riportato il metodo matriciale. Sia A la matrice dei dati, in cui ogni riga rappresenta un elemento del dataset ed ogni colonna rappresenta una variabile. Per esempio, prendiamo

$$A = \begin{pmatrix} 6 & 7 & 8 \\ 5 & 7 & 6 \\ 7 & 8 & 6 \\ 9 & 6 & 5 \\ 7 & 7 & 7 \end{pmatrix}$$

La prima azione da svolgere è quella di sottrarre da ogni colonna la propria media. Nell'esempio le medie delle colonne sono rispettivamente 6.8, 7 e 6.4. Sottraendole da A si ottiene la matrice

$$B = \begin{pmatrix} -0.8 & 0 & 1.6 \\ -1.8 & 0 & -0.4 \\ 0.2 & 1 & -0.4 \\ 2.2 & -1 & -1.4 \\ 0.2 & 0 & 0.6 \end{pmatrix}$$

A questo punto si calcola la matrice di covarianza, che per l'esempio è

$$\text{Cov} = \begin{pmatrix} 2.2 & -0.5 & -0.9 \\ -0.5 & 0.4 & 0.2 \\ -0.9 & 0.2 & 1.04 \end{pmatrix}$$

Su di essa si calcolano gli autovalori e gli autovettori. Gli autovalori sono $\lambda_1 \approx 0.268$, $\lambda_2 \approx 0.563$ e $\lambda_3 \approx 2.809$, la matrice degli autovettori è invece

$$v = \begin{pmatrix} 3.247 & 0.477 & -1.862 \\ 10.748 & -0.237 & 0.469 \\ 1 & 1 & 1 \end{pmatrix}$$

Gli autovettori definiscono le direzioni dei nuovi assi, e quindi, per ridurre di dimensionalità è necessario rimuoverne alcuni. Nell'esempio ne rimuoveremo uno. Per decidere quali autovettori eliminare dobbiamo guardare gli autovalori a loro corrispondenti: gli autovettori con gli autovalori più bassi portano il miglior numero di informazioni sulla distribuzione dei dati e sono quindi quelli che vengono eliminati. Elimineremo quindi l'autovettore corrispondente a λ_3 . Più gli autovalori sono

piccoli, minore sarà la perdita di informazione. La matrice di autovettori rimanente è quindi

$$W = \begin{pmatrix} -1.862 & 0.477 \\ 0.469 & -0.237 \\ 1 & 1 \end{pmatrix}$$

L'ultimo passo è la formazione delle componenti principali $PC = W^T \times B^T$. Quindi

$$\begin{aligned} CP &= \begin{pmatrix} -1.862 & 0.469 & 1 \\ 0.477 & -0.237 & 1 \end{pmatrix} \times \begin{pmatrix} -0.8 & -1.8 & 0.2 & 2.2 & 0.2 \\ 0 & 0 & 1 & -1 & 0 \\ 1.6 & -0.4 & -0.4 & -1.4 & 0.6 \end{pmatrix} = \\ &= \begin{pmatrix} 3.0896 & 2.9516 & -0.3034 & -5.9654 & 0.2276 \\ 1.2184 & -1.2586 & -0.5416 & -0.1136 & 0.6954 \end{pmatrix} \end{aligned}$$

Quello che si ottiene sono i dati originali, ma scritti considerando gli autovettori come assi di riferimento. I 2 assi selezionati, indicati dalle righe della matrice CP, rappresentano gli autovettori scelti, o le componenti principali.

4

DATASET

I dataset utilizzati durante lo sviluppo della tesi sono i seguenti.

4.1 INRIA HOLIDAYS

Il dataset *INRIA Holidays* contiene principalmente fotografie scattate dai creatori durante le proprie vacanze, in una grande varietà di scenari (naturali, cittadini, ecc.). Le restanti immagini sono state invece inserite per testare la robustezza a vari cambiamenti come rotazioni, sfocatura, cambi di prospettiva e illuminazione. Esso è composto da 1491 immagini divise in 500 gruppi, ciascuno dei quali rappresenta un particolare scenario o oggetto. La prima immagine di ogni gruppo è l'immagine di query e le altre immagini del gruppo sono considerate le risposte corrette alla query. Oltre alle immagini sono forniti i descrittori SIFT di dimensione 128.

Esempi di immagini di questo dataset sono riportati in figura 4.1

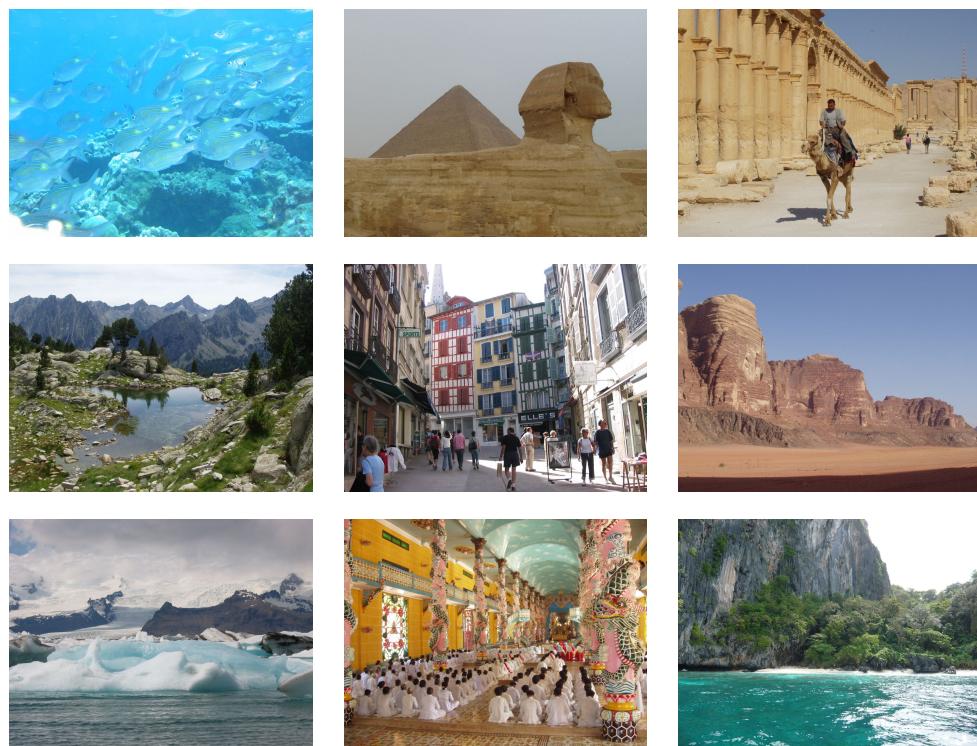


Figura 4.1: Esempi di immagini di INRIA Holidays

4.2 OXFORD BUILDINGS

Il dataset *Oxford Buildings* è composto da 5062 immagini raccolte da Flickr cercando particolari edifici o monumenti di Oxford (All Souls, Ashmolean, Balliol, Bodleian, Christ Church, Cornmarket, Hertford, Keble, Magdalen, Pitt Rivers, Radcliffe Camera). Per ciascuna di queste classi ci sono 5 immagini di query e 3 file che contengono le risposte corrette alla query:

- Good: contengono immagini che rappresentano chiaramente il luogo o l'edificio;
- Ok: contengono immagini che mostrano l'oggetto per una superficie maggiore al 25%;
- Junk: contengono immagini che mostrano meno del 25% della superficie dell'oggetto, oppure che hanno alti livelli di distorsione o occlusione.

Esempi di immagini di questo dataset sono riportati in figura 4.2. In realtà un grande numero di immagini al suo interno, non rappresentano il luogo o l'edificio della classe a cui appartengono, ma sono foto di persone. Per alcune classi queste foto sono persino la maggioranza.

Di seguito è riportato l'esempio di una query: la prima immagine (figura 4.3) è l'immagine di query, in questo caso appartenente alla classe cornmarket. I successivi gruppi di immagini contengono le fotografie che vengono considerate risposte corrette alla query, rispettivamente appartenenti ai gruppi good (figura 4.4), ok (figura 4.5) e junk (figura 4.6).



Figura 4.2: Esempi immagini di Oxford



Figura 4.3: Immagine di query di Oxford



Figura 4.4: Risposte alla query della classe good

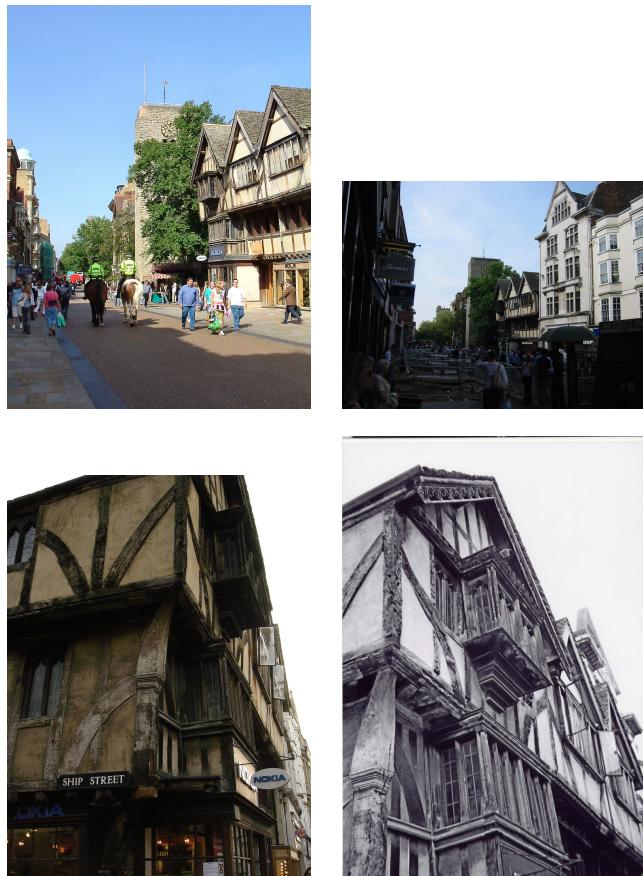


Figura 4.5: Risposte alla query della classe ok

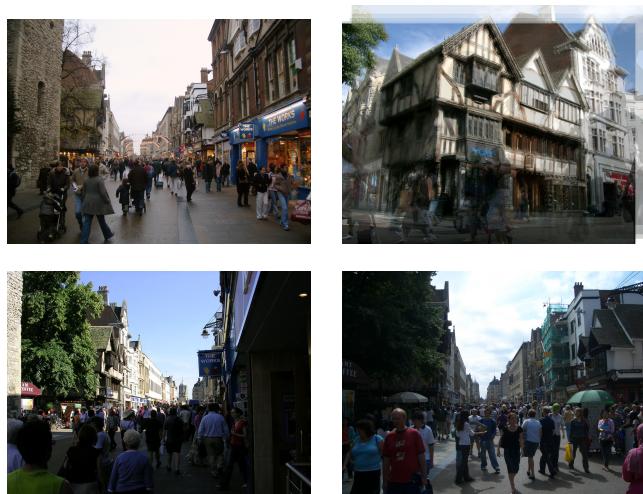


Figura 4.6: Risposte alla query della classe junk

4.3 PARIS

Il dataset *Paris* è composto da 6412 immagini raccolte da Flickr cercando particolari edifici o monumenti di Parigi (La Défense, Eiffel Tower, Hotel des Invalides, Louvre, Moulin Rouge, Musee d'Orsay, Notre Dame, Patntheon, Pompidou, Sacre Coeur, Arc de Triomphe). La struttura del dataset ricalca esattamente quella di Oxford.

Oltre che la struttura, Paris condivide con Oxford i problemi: sono presenti infatti molte fotografie di persone e grandi differenze tra immagini della stessa classe. In particolare, su questo dataset, ci sono molti edifici per cui una parte delle foto è stata scattata all'esterno, mentre un'altra parte all'interno, soprattutto a quadri e statue.

Esempi di immagini di Paris sono riportati in figura 4.7.

Durante l'addestramento di reti neurali utilizzando questo dataset ci siamo accorti che l'utilizzo delle immagini etichettate come Junk deteriorava molto le prestazioni, per questo le abbiamo tolte, rimanendo solamente con 1789 immagini.

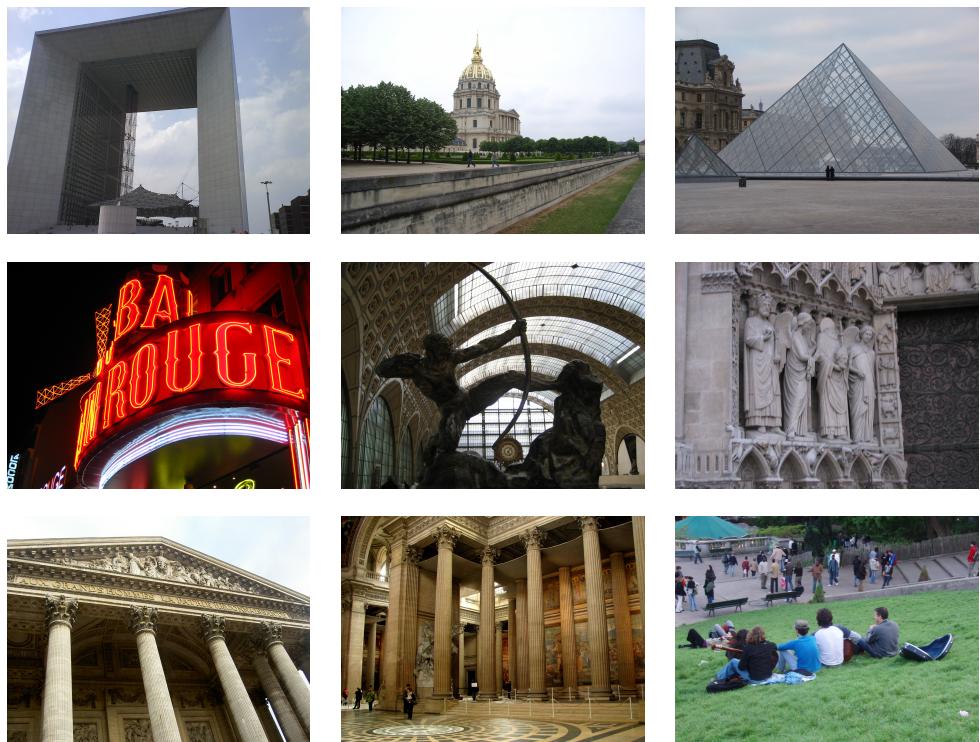


Figura 4.7: Esempi immagini di Paris

5

ESPERIMENTI

In questo capitolo sono descritti gli esperimenti effettuati su INRIA Holidays, Oxford e Paris. La prima parte degli esperimenti ha l’obiettivo di confrontare le prestazioni dei descrittori SIFT con i descrittori estratti da reti convoluzionali. Nella seconda parte sono stati confrontati 4 metodi di pooling diversi, per capire quale fosse il più efficiente. La terza parte degli esperimenti è invece concentrata sugli algoritmi di data augmentation: in un primo momento abbiamo utilizzato le policy di AutoAugment, ottimizzate su altri dataset. Poi abbiamo ottimizzato le policy di RandAugment per il nostro caso specifico, confrontando i risultati ottenuti. Infine, abbiamo testato le prestazioni dell’image retrieval al variare della dimensione delle immagini considerate.

5.1 ESPERIMENTI SU HOLIDAYS

Il primo esperimento effettuato è stato quello di verificare se, come suggerito dalla letteratura, descrittori per immagini estratti da reti neurali convoluzionali siano più performanti rispetto ai descrittori SIFT, nell’ambito dell’image retrieval. Il framework utilizzato per la comparazione è quello costruito da Yael per il dataset INRIA Holidays.

In questo framework vengono utilizzati dei descrittori SIFT per le immagini, di dimensione 128, precedentemente calcolati tramite un apposito tool. Questi vengono poi aggregati tramite Fisher-Vector [17] ed infine viene applicata una PCA che mantenga solo le 64 componenti più significative. Dopo aver costruito i descrittori per tutte le immagini del dataset, a partire dalle immagini di query, si simula la ricerca sul dataset, che porterà al recupero di 8 immagini. Per ogni query viene calcolato il vettore delle distanze tra essa e tutte le altre immagini, ordinato per distanza crescente: le prime 8 immagini saranno quindi il risultato della query, e su di esse viene calcolata la mAP. Il valore ottenuto con i descrittori SIFT è di 0.590.

Per quanto riguarda invece i descrittori convoluzionali, abbiamo utilizzato come rete neurale VGG16 preaddestrata su ImageNet. Prima di entrare nel dettaglio della loro estrazione è necessario mettere in luce un importante aspetto: mentre i descrittori SIFT sono stati calcolati su immagini di dimensione qualsiasi, VGG16 richiede in input immagini di dimensione fissa. Per questo motivo, abbiamo ripetuto gli esperimenti variando la dimensione delle immagini in input, a partire da 224×224 , il valore di default, fino a 1120×1120 .

VGG16 è una rete neurale costruita ed addestrata per la classificazione in modo che, presa in input un'immagine, ciò che predice è un'etichetta che indica il soggetto dell'immagine. Il dataset su cui è stata addestrata è ImageNet, composto da migliaia di fotografie di animali e oggetti di uso comune, molto diverso da INRIA Holidays che contiene fotografie di vacanze. Non è stato inoltre possibile effettuare un fine-tuning su di esso in quanto la natura sparsa di questo dataset (poche immagini e molte classi) rende impossibile l'utilizzo della triplet loss. Eliminando però gli ultimi 3 livelli di VGG16, quelli utilizzati per la classificazione, si ottiene una rete neurale che, presa in input un'immagine qualsiasi, produce un vettore di numeri che può essere utilizzato come descrittore. A questo output sono stati applicati due metodi di pooling: un layer di max-pooling, indicato generalmente come metodo migliore, e il Fisher-Vector calcolato tramite il framework di INRIA. Un ulteriore esperimento è stato quello di utilizzare Fisher-Vector con descrittori estratti non dal penultimo livello della rete, ma da quello ancora precedente.

I risultati di questi esperimenti sono riportati in tabella 5.1. F-V (1) indica l'esperimento effettuato con i descrittori estratti dal penultimo livello, mentre F-V (2) indica quello effettuato con i descrittori estratti dal livello precedente.

	224×224	448×448	672×672	896×896	1120×1120
Max-pooling	0.551	0.574	0.594	0.591	0.584
F-V (1)	0.585	0.700	0.724	0.657	0.680
F-V (2)	0.637	0.719	0.637	0.548	0.497

Tabella 5.1: mAP del retrieval con VGG16 su INRIA Holidays al variare delle dimensioni delle immagini

Per quanto riguarda il max-pooling, nonostante il metodo di aggregazione utilizzato sia molto più semplice, e nonostante gli svantaggi descritti in precedenza, si può notare come le prestazioni siano praticamente equi-

valenti a quelle ottenute con i descrittori SIFT. Utilizzando Fisher-Vector invece le prestazioni migliorano nettamente, fino a raggiungere una mAP di 0.724 con immagini di dimensione 672×672 .

I risultati di questo esperimento mettono in luce tre aspetti importanti:

- I descrittori convoluzionali sono molto più performanti rispetto ai descrittori SIFT;
- Il metodo di pooling utilizzato può fare molto la differenza in termini di prestazioni;
- Una rete neurale addestrata per un obiettivo e su di un dataset diversi da quelli considerati, è comunque in grado di raggiungere ottime prestazioni. Questo indica che i filtri convoluzionali della rete hanno imparato a catturare le caratteristiche importanti delle immagini, indipendentemente dalla loro natura.

5.2 ESPERIMENTI SU METODI DI POOLING

Dato che gli esperimenti precedenti hanno messo in luce l'importanza del metodo di pooling utilizzato, abbiamo condotto esperimenti che mettessero a confronto vari metodi. Oltre ai classici avg-pooling e max-pooling, abbiamo deciso di provare VLAD, un'evoluzione di Fisher-Vector.

Contrariamente agli esperimenti precedenti, dove il Fisher-Vector veniva calcolato "manualmente", abbiamo deciso di inserire VLAD come vero e proprio livello nella rete neurale, utilizzando NetVLAD, illustrato in precedenza. Questo livello deve essere collegato all'ultimo livello convoluzionale di VVG16, `block5_conv3` il cui output ha dimensione $(14, 14, 512)$. Utilizzando la notazione introdotta per spiegare NetVLAD, avremo $H = 14$, $W = 14$, $D = 512$, ponendo il numero dei cluster $K = 128$, avremo che l'output di questo livello sarà di dimensione $K \times B = 128 \times 512 = 66'536$.

Passaggio fondamentale per il suo corretto funzionamento è l'inizializzazione dei pesi, che sono 3: i centroidi dei K cluster dei tensori ricevuti in input \mathbf{c}_k , i pesi \mathbf{w}_k e i bias b_k .

Per inizializzare questi pesi, servono i tensori che NetVLAD riceve in input, ovvero quelli in uscita dal livello `block5_conv3`. Abbiamo quindi tagliato la rete a questo livello, processato l'intero dataset (se fosse stato di dimensioni più grandi sarebbe stato sufficiente processarne un campione

casuale) e calcolato i centroidi c_k tramite K-Means. Abbiamo quindi potuto calcolare gli altri due pesi:

$$\mathbf{w}_k = 2\alpha \mathbf{c}_k \quad b_k = -\alpha \|\mathbf{c}_k\|^2$$

Una volta scelto il metodo di pooling da utilizzare, abbiamo semplicemente collegato VGG16 con i pesi di ImageNet a questo livello e abbiamo eseguito un fine-tuning su Paris ripulito. Nella fase di testing abbiamo invece utilizzato il dataset Oxford e calcolato la mAP tramite uno script sviluppato dai creatori del dataset stesso. Essa viene calcolata con lo stesso metodo degli esperimenti su INRIA Holidays ma utilizzando la lista completa del retrieval, non solo le prime 8 immagini recuperate.

I risultati ottenuti sono i seguenti:

Metodo di pooling	mAp
Nessuno	0.360
AVG-pooling	0.411
MAX-pooling	0.507
NetVlad	0.485

Tabella 5.2: Test Paris-Oxford

Per prima cosa bisogna porre attenzione sul fatto che utilizzando max-pooling e avg-pooling si ottengono descrittori di dimensione 512, utilizzando NetVlad di dimensione 65'537, mentre non utilizzando nessun metodo, di dimensione 25'089. I risultati denotano come NetVlad e max-pooling abbiano prestazioni decisamente superiori rispetto agli altri due metodi. Il fatto che, nonostante le dimensioni maggiori, le prestazioni di NetVlad siano comunque, sebbene di poco, peggiori rispetto a max-pooling, può essere dovuto a alla *curse of dimensionality*, ovvero i dati risultano proiettati in uno spazio così grande che diventano sparsi. Per risolvere questo problema sarebbe sufficiente ridurre il numero di cluster calcolati all'interno di NetVlad, ma l'addestramento diventa molto instabile e porta ad un ulteriore peggioramento delle prestazioni.

Una nota su questi esperimenti è che per ottenere questi risultati abbiamo dovuto ripulire Paris dalle immagini junk, altrimenti la massima mAP ottenuta utilizzando max-pooling era pari solo a 0.443.

5.3 ESPERIMENTI SU DATA AUGMENTATION

In seguito sono illustrati gli esperimenti fatti per testare l'efficacia dei parametri di AutoAugment sull'image retrieval. La rete neurale utilizzata come base di partenza è VGG16 con max-pooling, preaddestrata su ImageNet e con fine tuning su Paris. Si cerca quindi di migliorare la mAP di 0.507 ottenuta su Oxford nell'esperimento precedente. Le policy testate sono 4:

- policy di Tensorflow
- policy ottenute su ImageNet
- policy ottenute su CIFAR-10
- policy ottenute su SVHN

Si effettua un nuovo fine-tuning sulla rete di partenza utilizzando queste policy: ogni immagine, prima di essere passata alla rete neurale viene modificata tramite una sub-policy scelta casualmente in modo uniforme.

I risultati ottenuti su Oxford con questo secondo fine-tuning sono i seguenti:

	Baseline	SVHN	ImageNet	Tensorflow	CIFAR-10
mAp	0.507	0.498	0.486	0.513	0.522

Tabella 5.3: mAp image retrieval su Oxford con policy di AutoAugment

Si può notare come le policy di Tensorflow e quelle di CIFAR-10 migliorino le prestazioni, mentre quelle di SVHN e di ImageNet le peggiorino: questo indica che probabilmente i dataset utilizzati per addestrare AutoAugment per i primi due gruppi di policy sono geometricamente più simili ad Oxford e Paris di quanto lo siano gli altri due. In ogni caso, il miglioramento di prestazioni, conferma l'ipotesi degli autori dell'articolo sul fatto che le policy possano essere utilizzate per altri dataset e per task diversi da quelli su cui erano state ottimizzate.

Per quanto riguarda invece RandAugment, per la ricerca dei parametri M ed N è stata utilizzata la piattaforma Comet.ml che mette a disposizione diversi algoritmi di ottimizzazione. Per i nostri esperimenti abbiamo utilizzato l'algoritmo bayesiano, che viene da loro definito come il più efficiente. Come parametro aggiuntivo abbiamo inserito il numero di

epoch da utilizzare per il fine-tuning con data augmentation. I risultati sono riportati in tabella 5.4.

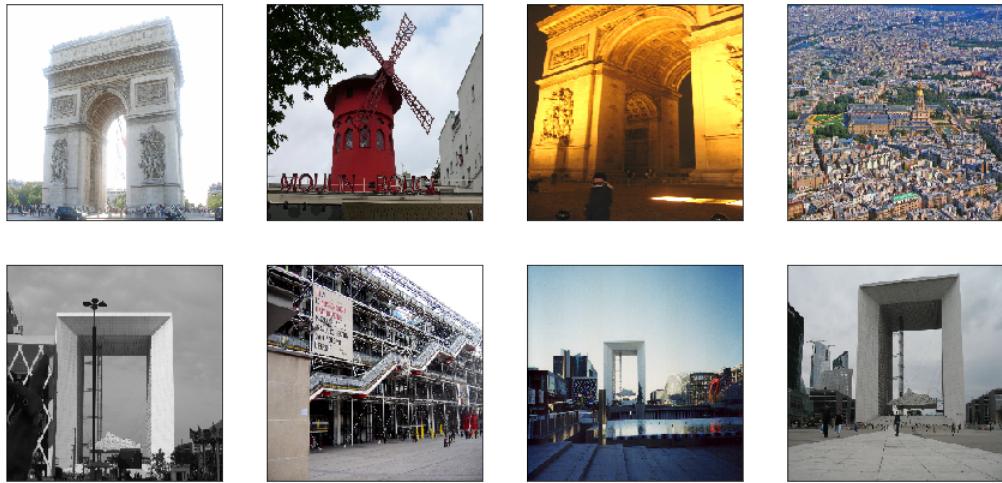
N	M	Epochs	mAP
1	8	15	0.5102
1	14	10	0.5263
1	16	11	0.5124
1	19	4	0.5148
1	24	15	0.5151
1	25	10	0.5139
2	5	13	0.5501
2	5	14	0.5196
2	16	11	0.5124
2	19	9	0.5240
3	6	7	0.5083
3	6	8	0.5148

Tabella 5.4: Risultati RandAugment

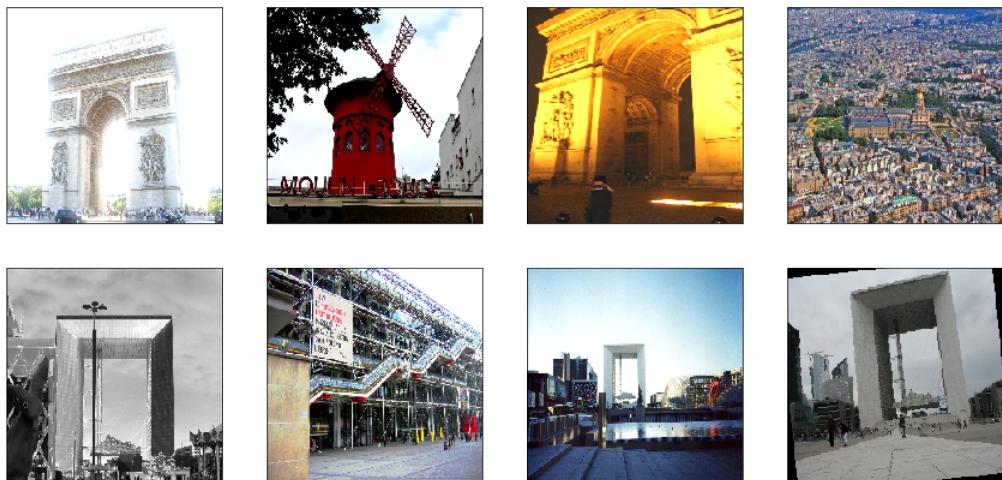
Come si può notare, essi sono nettamente superiori rispetto a quelli ottenuti con AutoAugment evidenziando il fatto che un sistema di data augmentation più semplice, addestrato sul dataset che verrà poi utilizzato, porta a prestazioni migliori rispetto ad un sistema molto più articolato ma addestrato su un dataset diverso e per un task diverso.

Di seguito sono riportati esempi di Data Augmentation applicato ad immagini di Paris. In figura 5.1 le prime 8 immagini sono quelle originali, le successive 8 sono state invece generate applicando delle policy di AutoAugment, ottenute con l'ottimizzazione su CIFAR-10.

In figura 5.2 invece, le prime 8 immagini sono originali e le successive 8 sono state generate applicando policy di RandAugment. Come si può notare, in entrambi i casi vengono applicate sia trasformazioni sul colore che sulla geometria dell'immagine, simulando cambi di prospettiva e di condizioni di luce. Le policy di RandAugment però applicano trasformazioni in modo molto più evidente.



Immagini originali

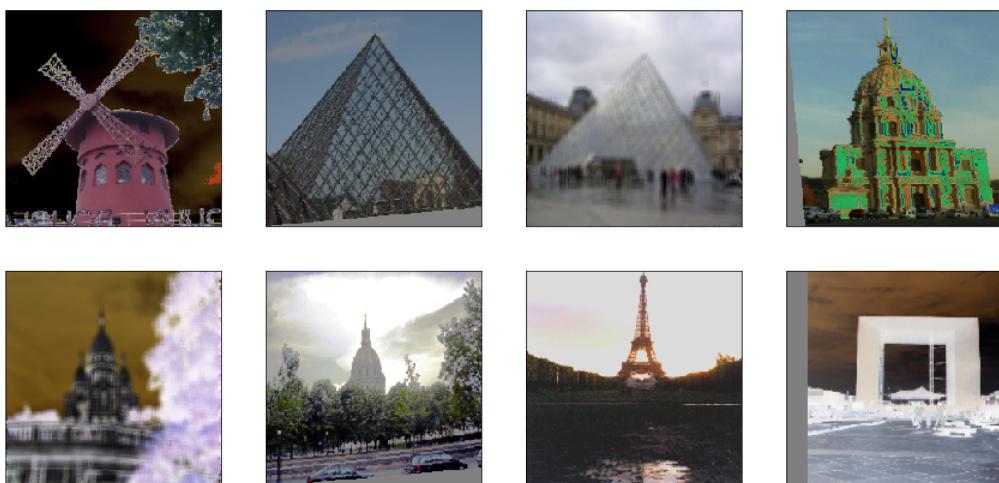


Immagini generate utilizzando policy di AutoAugment

Figura 5.1: Confronto tra immagini originali e immagini generate utilizzando policy di RandAugment ottimizzate su CIFAR-10



Immagini originali



Immagini generate utilizzando policy di RandAugment

Figura 5.2: Confronto tra immagini originali e immagini generate utilizzando policy di RandAugment

5.4 ESPERIMENTI SULLA DIMENSIONE DELLE IMMAGINI

In tutto il corso della tesi, abbiamo utilizzato immagini di dimensione 224×224 , sia durante le fasi di addestramento che durante le fasi di testing. Una dimensione così piccola consente un addestramento delle reti neurali veloce e permette di mantenere la *batch size* a valori più alti: indispensabile per utilizzare la triplet loss. L'architettura che avevamo a disposizione consentiva di utilizzare immagini fino alla dimensione di 448×448 per quanto riguarda il train set, ma la diminuzione della batch size portava ad un grande calo delle prestazioni. Per quanto riguarda invece la fase di test, abbiamo avuto la possibilità di sperimentare varie grandezze.

La rete neurale utilizzata per questi esperimenti è sempre VGG16 con max-pooling, addestrata su ImageNet e con fine-tuning su Paris utilizzando immagini di dimensione 224×224 . Il test è sempre effettuato su Oxford.

Per prima cosa abbiamo cercato di ottimizzare la rete per il testing su immagini di dimensione 448×448 utilizzando tecniche sviluppate in precedenza. Il primo esperimento è stato effettuato addestrando semplicemente la rete su Paris, con immagini di dimensione 224×224 , e poi testando su Oxford con immagini di dimensione 448×448 . La mAP ottenuta è di 0.5534, quasi un 5% migliore rispetto al testing con immagini di dimensione 224×224 . Abbiamo provato anche a migliorare questo risultato utilizzando la data augmentation ma senza successo.

A questo punto abbiamo 2 insiemi di pesi che raggiungono una mAP di circa 0.55 su Oxford: i pesi della rete addestrata su RandAugment e ottimizzata per immagini di dimensione 224×224 , che chiameremo `vgg_ra_224` e i pesi della rete addestrata senza RandAugment e ottimizzata per immagini di dimensione 448×448 , che chiameremo `vgg_448`.

Utilizzando i pesi `vgg_ra_224` per testare con immagini di dimensione 448×448 , si ottiene una mAP pari a 0.5687 (gli stessi pesi, ricordiamo, su immagini di dimensione 224×224 producevano una mAP di 0.5507).

Abbiamo quindi provato a combinare i descrittori estratti da VGG16 utilizzando entrambi pesi: prima concatenandoli semplicemente, poi applicando una PCA per vedere se l'incremento delle prestazioni era dovuto solamente alla dimensione doppia del descrittore. I risultati sono riportati in tabella 5.5, ed evidenziano come, per immagini di dimensione 448×448 , la combinazione di questi pesi aumenti notevolmente le prestazioni. Inoltre, l'utilizzo della PCA riduce abbastanza la mAP,

ma quest'ultima rimane comunque più elevata rispetto all'utilizzo dei descrittori utilizzati separatamente. Sarebbe molto interessante provare ad effettuare questa ottimizzazione anche per immagini più grandi e vedere qual è la dimensione migliore.

	224×224	448×448	672×672
Con PCA	0.5368	0.5848	0.5722
Senza PCA	0.5478	0.6055	0.5937

Tabella 5.5: mAP su Oxford al variare della dimensione delle immagini utilizzate durante la fase di test

Riportiamo di seguito i risultati di alcune query effettuate: la prima immagine rappresenta l'immagine di query, le successive sono le prime 5 immagini recuperate. Se l'immagine è bordata di verde, significa che è stata recuperata correttamente, ovvero appartiene alla stessa classe della query. In caso contrario è bordata di rosso. Come si può notare, anche le immagini che ottengono un punteggio molto basso (che ricordiamo essere calcolato su tutte le immagini recuperate, non solo sulle prime 5) recuperano principalmente immagini della classe corretta.



Figura 5.3: Query Kebble 3: AP = 98%

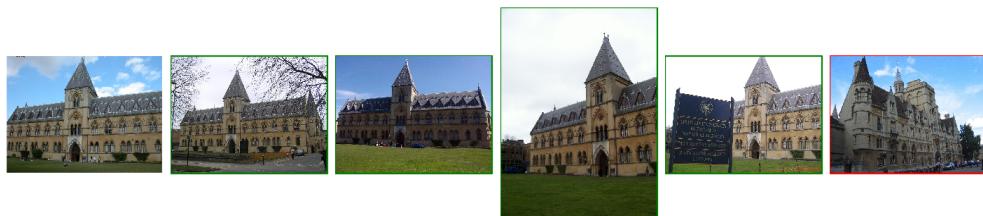


Figura 5.4: Query Pitt Rivers 2: AP = 94%



Figura 5.5: Query Hertford 5: AP = 72%



Figura 5.6: Query Cornmarket 5: AP = 51%

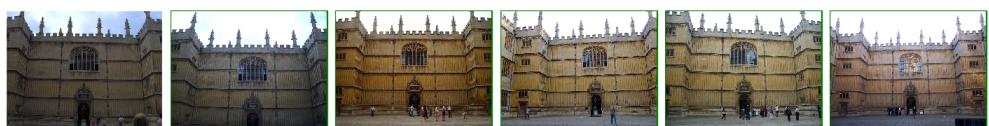


Figura 5.7: Query Bodleian 4: AP = 44%

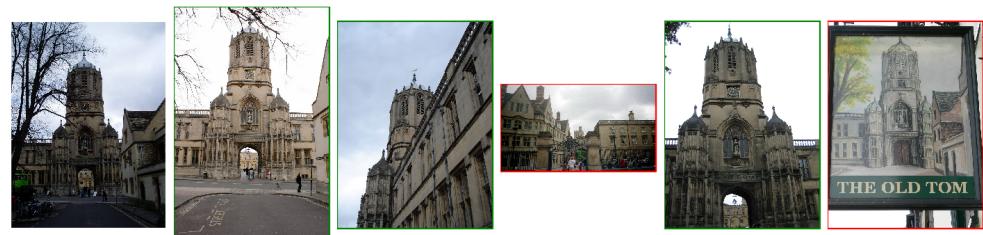


Figura 5.8: Query Christ Church 5: AP = 18%

6

CONCLUSIONI

Durante lo sviluppo di questa tesi è stata utilizzata una rete neurale convoluzionale, VGG16, per studiare tecniche che consentissero di migliorare le prestazioni dell’image retrieval sul dataset Oxford, noto per la sua difficoltà. All’atto pratico, questo obiettivo consiste nel cercare di addestrare la rete in modo che produca descrittori per le immagini, che catturino nel miglior modo possibile le caratteristiche fondamentali dei luoghi e degli edifici di Oxford. I punti di maggiore interesse sono stati la Data Augmentation e l’utilizzo di immagini di scale diverse durante la fase di testing.

Per quanto riguarda la Data Augmentation abbiamo studiato due algoritmi di ottimizzazione diversi: AutoAugment ricco di parametri ma molto costoso da utilizzare, e RandAugment con soli 2 parametri ma con tempi di esecuzione molto ridotti. Utilizzando policy di AutoAugment sviluppate per l’image classification su vari dataset, abbiamo dimostrato che queste possono essere utili per migliorare le prestazioni dell’image retrieval su Oxford. Le migliorie apportate non sono però paragonabili a quelle ottenute addestrando direttamente RandAugment per il nostro obiettivo.

Dopo lo studio della data augmentation abbiamo cercato di capire quali fossero gli effetti del cambiamento di scala delle immagini: ci siamo chiesti se fare image retrieval utilizzando immagini più grandi portasse ad un aumento delle prestazioni. Questo non avviene sempre, anzi aumentando troppo la dimensione delle immagini le prestazioni arrivano persino a peggiorare. Un importante risultato è stato raggiunto invece ottimizzando i pesi di VGG16 utilizzando immagini di dimensioni diverse e combinando i descrittori così ottenuti. In particolare, il risultato migliore è stato ottenuto combinando i descrittori estratti della rete ottimizzata su immagini di dimensione 224×224 con RandAugment, e quelli estratti dalla stessa rete ottimizzata su immagini di dimensione 448×448 senza Data Augmentation.

6.1 SVILUPPI FUTURI

Un possibile sviluppo interessante, potrebbe essere quello di verificare l'effetto nell'image retrieval di tecniche di Data Augmentation sviluppate per l'object detection, invece che per l'image classification. I dataset di object detection sono strutturati in modo che, in ogni immagine, gli oggetti di interesse siano identificati da delle *bounding box*. Durante la Data Augmentation quindi, le trasformazioni possono essere applicate a tutta l'immagine, solo agli oggetti all'interno della bounding box o solo allo sfondo. Ovviamente a 3 diversi tipi di applicazione, possono corrispondere 3 diversi insiemi tra cui scegliere la trasformazione da applicare.

Altro possibile sviluppo è il miglioramento del processo di creazione dei descrittori per le immagini, usando funzioni di loss contrastive che non richiedano l'uso del triplet mining, come la Contrastive Loss o ArcFace [18]. Questo genere di metriche, nato nell'ambito del riconoscimento di facce, ha il vantaggio di non dover creare triplette, operazione comunque costosa, ed evita anche il problema della creazione di triplette troppo difficili o troppo facili.

BIBLIOGRAFIA

- [1] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, 2018. (Cited on page 12.)
- [2] J. Luo, J. Li, J. Wang, Z. Jiang, and Y. Chen, "Deep attributes from context-aware regional neural codes," *CoRR*, vol. abs/1509.02470, 2015. (Cited on pages 12 and 13.)
- [3] T. Uricchio, M. Bertini, L. Seidenari, and A. D. Bimbo, "Fisher encoded convolutional bag-of-windows for efficient image retrieval and social image tagging," in *ICCV Workshops*, pp. 1020–1026, IEEE Computer Society, 2015. (Cited on pages 12 and 13.)
- [4] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," *CoRR*, vol. abs/1604.01325, 2016. (Cited on page 13.)
- [5] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *ICCV*, pp. 3476–3485, IEEE Computer Society, 2017. (Cited on page 13.)
- [6] T. Hoang, T.-T. Do, D.-K. Le Tan, and N.-M. Cheung, "Selective deep convolutional features for image retrieval," in *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, (New York, NY, USA), pp. 1600–1608, ACM, 2017. (Cited on page 13.)
- [7] M. Wang, C. Luo, R. Hong, J. Tang, and J. Feng, "Beyond object proposals: Random crop pooling for multi-label image recognition," *IEEE Transactions on Image Processing*, vol. PP, pp. 1–1, 09 2016. (Cited on page 13.)
- [8] W. Yu, K. Yang, H. Yao, X. Sun, and P. Xu, "Exploiting the complementary strengths of multi-layer cnn features for image retrieval," *Neurocomput.*, vol. 237, pp. 235–241, May 2017. (Cited on page 13.)
- [9] Z. Wu and J. Yu, "A multi-level descriptor using ultra-deep feature for image retrieval," *Multimedia Tools and Applications*, vol. 78, 05 2019. (Cited on page 13.)

- [10] A. Babenko and V. S. Lempitsky, "Aggregating deep convolutional features for image retrieval," *CoRR*, vol. abs/1510.07493, 2015. (Cited on page 13.)
- [11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. (Cited on page 28.)
- [12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. (Cited on page 30.)
- [13] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 32.)
- [14] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.09501, 2018. (Cited on page 35.)
- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. (Cited on page 36.)
- [16] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical data augmentation with no separate search," *CoRR*, vol. abs/1909.13719, 2019. (Cited on page 37.)
- [17] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image Classification with the Fisher Vector: Theory and Practice," *International Journal of Computer Vision*, vol. 105, pp. 222–245, Dec. 2013. (Cited on page 49.)
- [18] J. Deng, J. Guo, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *CoRR*, vol. abs/1801.07698, 2018. (Cited on page 62.)