

UNIVERSITÀ DI PISA

Scuola di Ingegneria
Corso di laurea in Ingegneria Informatica
A.A. 2017/2018

Analisi della percezione delle differenze cromatiche
al variare della risoluzione mediante l'uso di una
rete neurale artificiale

Candidato
Davide Cocomini

Relatori
Prof.ssa Beatrice Lazzerini
Dr. Francesco Pistolesi

A mia nonna Gemma, per non avermi mai fatto mancare il suo affetto, appoggio e fiducia, dandomi la forza di affrontare tutte le sfide che questo percorso mi ha riservato.

Indice

1	Introduzione	5
2	Colorimetria	6
2.1	Lo spettro visibile	6
2.1.1	Percezione del colore	7
2.2	Spazio colore	8
2.2.1	Modello CIE XYZ	8
2.2.2	Modello CIE L*a*b*	9
2.3	Risoluzione e compressione	10
2.3.1	Esempi di risoluzione	11
3	Reti Neurali	12
3.1	Reti neurali e cervello umano	12
3.1.1	Dal neurone biologico a quello artificiale	12
3.2	Perceptron	13
3.2.1	Addestramento del perceptron	14
4	Svolgimento	15
4.1	Creazione del dataset	15
4.1.1	Codice C++	16
4.1.2	Esempi di applicazione dei filtri	17
4.2	Feature Extraction	18
4.2.1	Preparazione degli input e degli output	18
4.2.2	Estrazione dei dati dalle immagini	18
4.2.3	Codice C++	20
4.2.4	Preparazione degli output	21
4.2.5	Esempio di risposta della rete	22
4.2.6	Esempi di valutazioni da parte dell'osservatore	23
4.3	Considerazioni preliminari	24
4.3.1	Analisi dei giudizi dell'osservatore	24
4.3.2	Calcolo del Lab-SSIM	25
4.3.3	Analisi dei risultati ottenuti con il Lab-SSIM	26
4.4	Addestramento della rete	27
4.4.1	Valutazione delle prestazioni	28
4.5	Feature selection	29
4.5.1	Analisi dei risultati	30
4.5.2	Undersampling e oversampling	31
4.6	Addestramento a seguito della feature selection	32
4.6.1	Classificatore Ensemble	33
4.6.2	Valutazione delle prestazioni del classificatore ensemble	35
5	Test	36

INDICE

6 Conclusioni	37
7 Ringraziamenti	38

1 Introduzione

L'individuazione delle differenze percettive, tra due o più immagini, è uno dei grandi problemi dell'industria della colorimetria moderna. È infatti spesso necessario riprodurre un materiale o un tessuto partendo da un originale, nel tentativo di replicarlo il più fedelmente possibile. Tuttavia, durante il processo produttivo si possono ottenere delle copie percettivamente diverse rispetto all'originale. Bisogna quindi trovare un modo per confrontare oggettivamente l'immagine originale con le copie, nella speranza di individuare eventuali differenze percettive e correggere ciò che non va nel processo produttivo. Sono stati quindi realizzati macchinari in grado di acquisire immagini ad alta risoluzione dei soggetti originali e di quelli riprodotti così da poterli confrontare attraverso indici matematici più o meno precisi, ad esempio SSIM. Ovviamente, sia per l'acquisizione che per il confronto tra immagini ad alta risoluzione, sono necessari tempi e costi non indifferenti che sarebbero abbattuti se questi processi potessero avvenire a risoluzioni più basse. Lo scopo di questa ricerca è quindi quello di realizzare una rete neurale che sia in grado di identificare le differenze tra le immagini e sfruttarla per capire qual è la minima risoluzione che le immagini devono possedere affinché queste differenze restino individuabili.

2 Colorimetria

La colorimetria è la disciplina che si occupa di normalizzare la misurazione del colore attraverso lo studio dei modelli di colore.

2.1 Lo spettro visibile

Tutte le colorazioni percepite dall'occhio umano compongono lo “spettro del visibile” che si trova nella parte centrale dello spettro della luce, il quale comprende anche i raggi infrarossi e quelli ultravioletti. Per descrivere il colore di un oggetto la colorimetria utilizza la percentuale della luce incidente che è stata riflessa, compresa nell'intervallo del visibile (400-700 nm). Ciascun oggetto colorato viene pertanto definito da una curva di riflettanza, similmente alle impronte digitali nell'uomo.

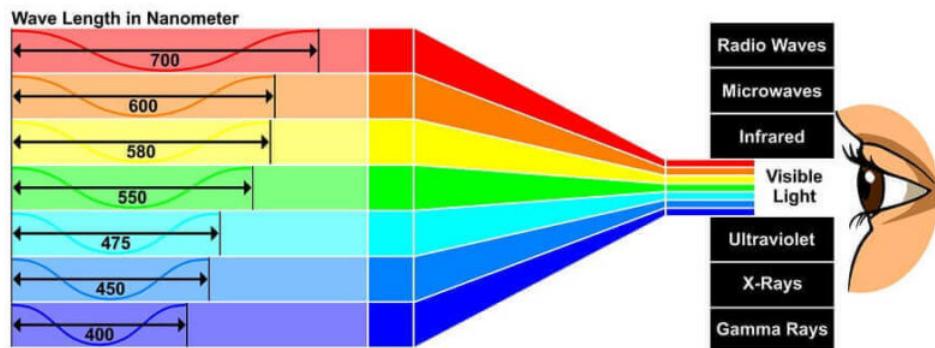


Figura 1: Lo spettro visibile

2.1 Lo spettro visibile

2.1.1 Percezione del colore

Il colore nasce dalla luce. La luce che colpisce un oggetto viene parzialmente assorbita a seconda del materiale che lo compone. La parte non assorbita viene riflessa e trasmessa ai recettori cromatici all'interno dell'occhio umano. Questi ultimi trasformano la luce assorbita in impulsi che percorrono le vie nervose fino a raggiungere il cervello, dove vengono interpretati: nasce così un'impressione cromatica.

Dal punto di vista prettamente biologico il colore si genera pertanto nell'occhio dell'osservatore e costituisce un'impressione sensoriale. Proprio perché nella percezione del colore vengono coinvolte componenti biologiche, ciascun individuo percepisce il colore in modo differente. Perfino la stessa persona può percepire differentemente il colore in momenti diversi ed in base allo stato d'animo. È quindi molto complesso definire in maniera oggettiva un colore.

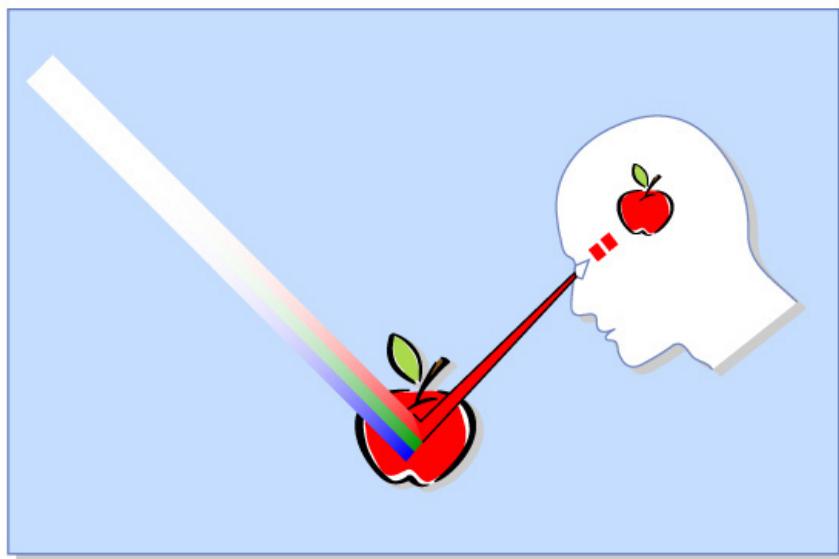


Figura 2: Percezione del colore

2.2 Spazio colore

Per rappresentare in maniera rigorosa i colori, sono nati gli spazi colore. Ognuno di essi si basa su alcuni parametri e facendoli variare si riesce a rappresentare un numero più o meno grande di colori. Esistono numerosi spazi colore che, in base alle loro peculiarità, si prestano meglio ad uno specifico utilizzo piuttosto che ad un altro.

2.2.1 Modello CIE XYZ

Un importante spazio colore è il modello CIE XYZ che rappresenta tutti i colori caratterizzati da tre parametri: luminosità, tinta e purezza, rappresentati attraverso un solido. Del solido viene solitamente rappresentata soltanto la sezione secondo il piano XY, in cui X ed Y indicano la cromaticità (tinta e purezza) mentre la luminosità non viene rappresentata.

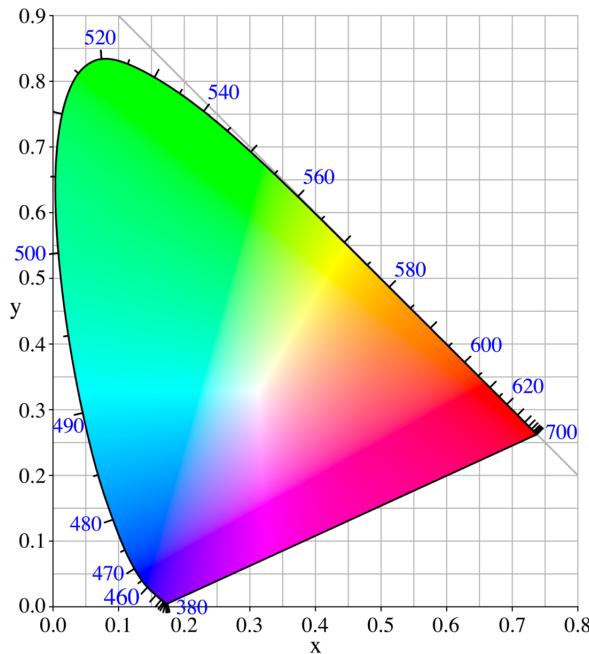


Figura 3: Spazio colore CIE XYZ

Questo spazio colore si basa sui valori di tristimolo, cioè dei parametri che definiscono il modo in cui l'essere umano percepisce i colori. I valori tristimolo di un colore con una distribuzione di potenza spettrale $I(\lambda)$ sono date in termini di un osservatore standard da:

$$\mathbf{X} = \int_0^{\infty} I(\lambda) \bar{x}(\lambda) d\lambda; \mathbf{Y} = \int_0^{\infty} I(\lambda) \bar{y}(\lambda) d\lambda; \mathbf{Z} = \int_0^{\infty} I(\lambda) \bar{z}(\lambda) d\lambda.$$

2.2.2 Modello CIE L*a*b*

A partire dallo spazio colore CIE XYZ si è poi sviluppato il modello CIE L*a*b*. Esso è un modello tridimensionale che si sviluppa lungo tre assi ortogonali. Due assi sul piano orizzontale riguardano la cromaticità: l'asse **a** si estende dal verde (-a) al rosso (+a) e l'asse **b** dal blu (-b) al giallo (+b); vi è poi un asse verticale che riguarda la luminosità o luminanza **L** che varia dal basso verso l'alto. Questo particolare spazio colore include tutti i colori percepibili dall'occhio umano e quindi anche tutto il gamut degli spazi RGB e CMYK ed è indipendente dal dispositivo che li rappresenta. Essendo quindi molto più simile al modo in cui l'essere umano riesce a percepire i colori, questo spazio colore si presta particolarmente bene agli scopi di questa ricerca.

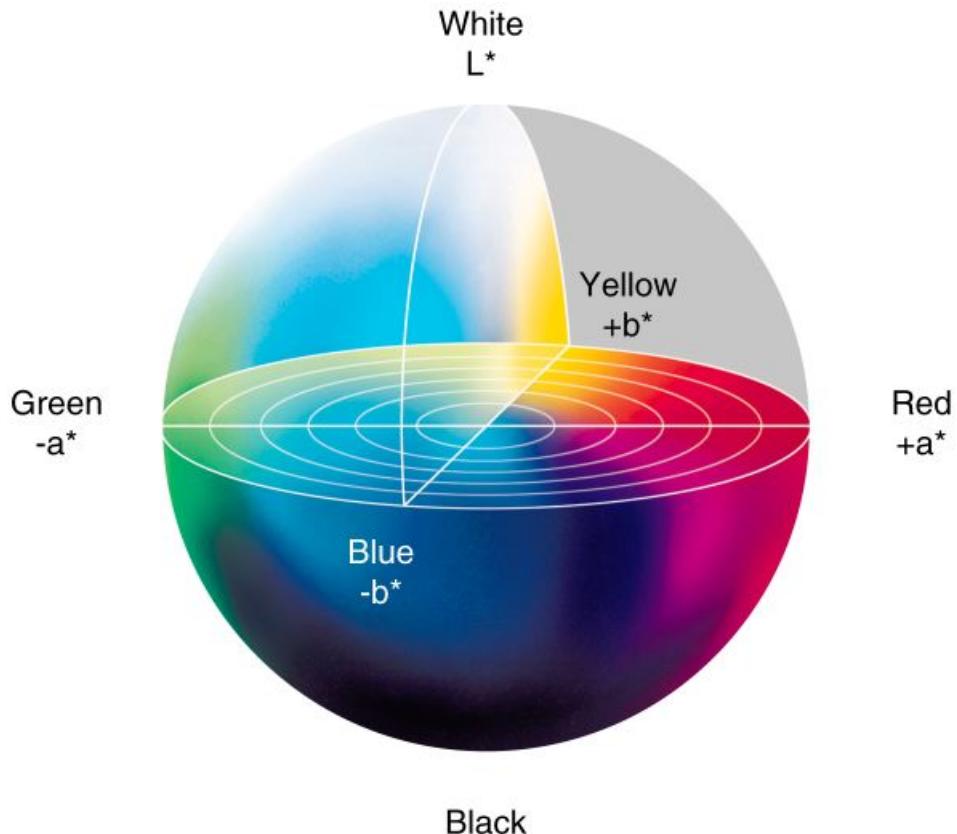


Figura 4: Spazio colore CIE L*a*b*

2.3 Risoluzione e compressione

La risoluzione di un'immagine corrisponde al numero di pixel per pollice che contiene, indicata con il termine in dpi (punti per pollice). Un maggior numero di dpi si traduce in una maggiore quantità di informazioni e quindi in una qualità dell'immagine superiore. Ovviamente, maggiore è il numero di informazioni che un'immagine contiene, maggiore sarà il peso del file. Per questa ragione, sono state sviluppate alcune tecniche di compressione tra cui quella JPEG. Questa tecnica stabilisce due metodi di compressione di base, uno basato sull'uso della trasformata discreta del coseno, con compressione di tipo "lossy" e cioè con perdita di informazione, e l'altro sull'uso di un metodo predittivo con compressione di tipo lossless, cioè senza perdita di informazione. La perdita di informazioni durante la compressione può compromettere drasticamente la qualità dell'immagine.

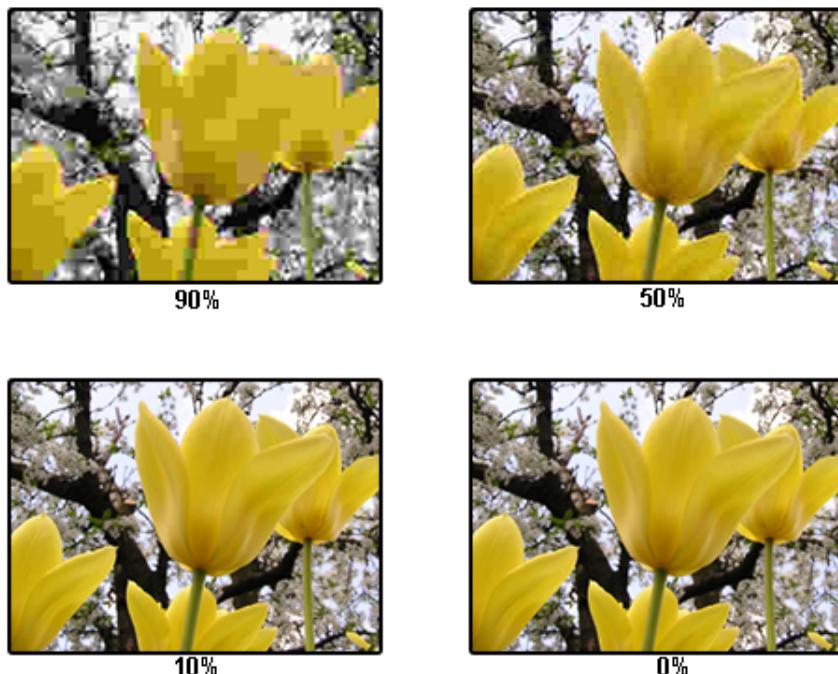


Figura 5: Esempio di compressione JPEG

2.3.1 Esempi di risoluzione

Ai fini della nostra ricerca, sono state prese in considerazione 5 risoluzioni diverse. Ad ogni risoluzione è associata una coppia di valori numerici rappresentativi della larghezza e dell'altezza dell'immagine:

- 4K → 3840x2160;
- 3K → 3000x2000;
- 1080p → 2048x1080;
- 720p → 1280x720;
- 480p → 544x480;

Il prodotto di questi valori rappresenta il numero di pixel presenti in un'immagine ad una data risoluzione. Ad esempio, a 4K avremo:

$$3840 * 2160 = 8.294.400$$

mentre a 480p il numero di pixel sarà:

$$544 * 480 = 261.120$$

Quindi un'immagine a 4K userà un numero di pixel decisamente superiore rispetto ad una a 480p e questo le consentirà di rappresentare l'immagine con un numero di dettagli superiore e quindi ad una qualità maggiore.

3 Reti Neurali

3.1 Reti neurali e cervello umano

Le reti neurali artificiali sono nate per emulare attività tipiche del cervello umano. Risultano quindi molto utili per risolvere problemi di riconoscimento delle immagini o del linguaggio umano. Per riuscire ad emulare questi comportamenti ci si è quindi ispirati al funzionamento del cervello umano. Nel sistema nervoso esistono miliardi di neuroni. Un neurone è formato da un corpo cellulare e da molti prolungamenti ramificati, detti dendriti, attraverso i quali il neurone riceve segnali elettrici da altri neuroni. Ogni neurone ha anche un prolungamento filamentoso chiamato assone. All'estremità l'assone si ramifica formando terminali attraverso i quali i segnali elettrici vengono trasmessi ad altre cellule. Tra un terminale di un assone e la cellula ricevente esiste uno spazio. I segnali superano questo spazio per mezzo di sostanze chimiche dette neurotrasmettitori. Il punto di connessione tra terminale e dendrite è detto sinapsi. Un neurone si “attiva”, cioè trasmette un impulso elettrico lungo il suo assone quando si verifica una differenza di potenziale elettrico tra l'interno e l'esterno della cellula. L'impulso elettrico provoca la liberazione di un neurotrasmettore dai terminali dell'assone, che a loro volta possono influenzare altri neuroni.

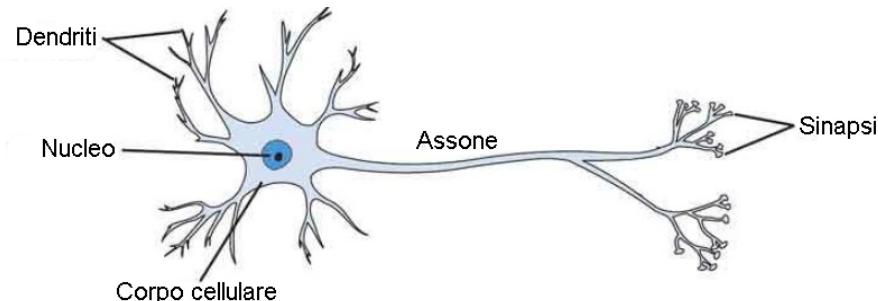


Figura 6: Neurone biologico

3.1.1 Dal neurone biologico a quello artificiale

Per riprodurre artificialmente il cervello umano occorre realizzare una rete di elementi molto semplici in grado di imparare e generalizzare. Tipicamente, il neurone artificiale ha molti ingressi ed una sola uscita. Per determinare la conducibilità e quindi l'importanza del canale di ingresso, ogni neurone ha associato un peso. L'attivazione del neurone è una funzione della somma pesata degli ingressi.

3.2 Perceptron

Il metodo più usato per addestrare una rete neurale consiste nel presentare in ingresso alla rete un insieme di esempi (training set). La risposta fornita dalla rete per ogni esempio viene confrontata con la risposta desiderata, si valuta la differenza (errore) fra le due e, in base a tale differenza, si modificano i pesi cercando di ottenere il risultato desiderato.

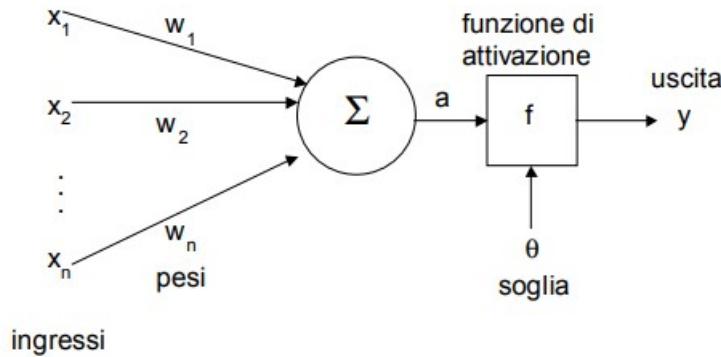


Figura 7: Modello di neurone

In generale si hanno quindi n ingressi x_1, \dots, x_n , a ciascuno dei quali è associato un peso. I pesi w_i sono numeri reali che riproducono l'amplificazione o lo smorzamento subito dal segnale nella comunicazione fra neuroni. Se $w_i > 0$, il canale è detto eccitatorio, se $w_i < 0$, il canale è inibitorio. Il valore assoluto di un peso rappresenta la forza della connessione. L'uscita, cioè il segnale con cui il neurone trasmette la sua attività all'esterno, è calcolata applicando la funzione di attivazione alla somma pesata degli ingressi:

$$y = f\left(\sum_{i=1}^n w_i x_i\right)$$

3.2 Perceptron

Un semplice esempio di rete neurale è il perceptron. Un perceptron è una rete neurale ad un singolo livello che si dimostra molto utile nei problemi di classificazione. In generale è composto da una serie x_1, \dots, x_n di input a cui sono associati dei pesi w_1, \dots, w_n . Vi è poi una funzione che si occupa di sommare i prodotti dei valori di input e dei rispettivi pesi. Il risultato passa infine per una funzione di attivazione che genera l'output.

3.2.1 Addestramento del perceptron

Per addestrare un perceptron bisogna realizzare un dataset adeguato per il problema che si vuole risolvere. Dopo aver predisposto il dataset è necessario seguire i seguenti passi:

1. si inizializzano i pesi w_i con valori casuali;
2. si presenta alla rete un ingresso x_k , si confronta col t_k desiderato in uscita;
3. si calcola la risposta y_k della rete e si aggiornano i pesi mediante la delta rule con l'obiettivo di ridurre l'errore;
4. si ripete il ciclo dal passo 2, finché la risposta della rete non risulta soddisfacente.

In particolare la delta rule è una regola usata per modificare i valori dei pesi di un neurone andando alla ricerca di quelli più adeguati. Dati t ed y , rispettivamente l'uscita desiderata e l'uscita neurale, l'errore δ è dato da:

$$\delta = t - y.$$

Fissato un numero reale η , compreso tra 0 e 1, detto learning rate, la delta rule stabilisce che la variazione del generico peso Δw_i è:

$$\Delta w_i = \eta \delta x_i.$$

Dopo l'addestramento la rete viene testata controllandone il comportamento su un insieme di dati, detto test set, non utilizzati durante la fase di training. La fase di test ha quindi lo scopo di valutare la capacità di generalizzazione della rete neurale.

4 Svolgimento

4.1 Creazione del dataset

Per poter realizzare la rete è necessario creare un dataset significativo affinché questa possa essere addestrata e testata. Per fare ciò sono state prese in considerazione 40 immagini, nello spazio colore L*a*b*, ad una risoluzione di 4K, sulle quali sono state successivamente fatte delle elaborazioni.

In particolare le elaborazioni sono state:

1. Per ogni immagine nel dataset iniziale è stata generata un'immagine ad una risoluzione minore o uguale (4K, 3K, 1080p, 720p, 480p).
2. Per ogni immagine generata al punto 1 sono stati applicati tre filtri cromatici ciascuno a tre intensità diverse.

Due dei filtri utilizzati coinvolgono indistintamente tutte le zone dell'immagine, mentre il terzo filtro altera solo le zone di una specifica tonalità di colore, in questo caso quelle del blu, in quanto maggiormente presente nel nostro dataset e le cui variazioni sono più complesse da individuare. Questo procedimento ha così generato 2000 immagini, utilizzabili per addestrare e testare la rete.

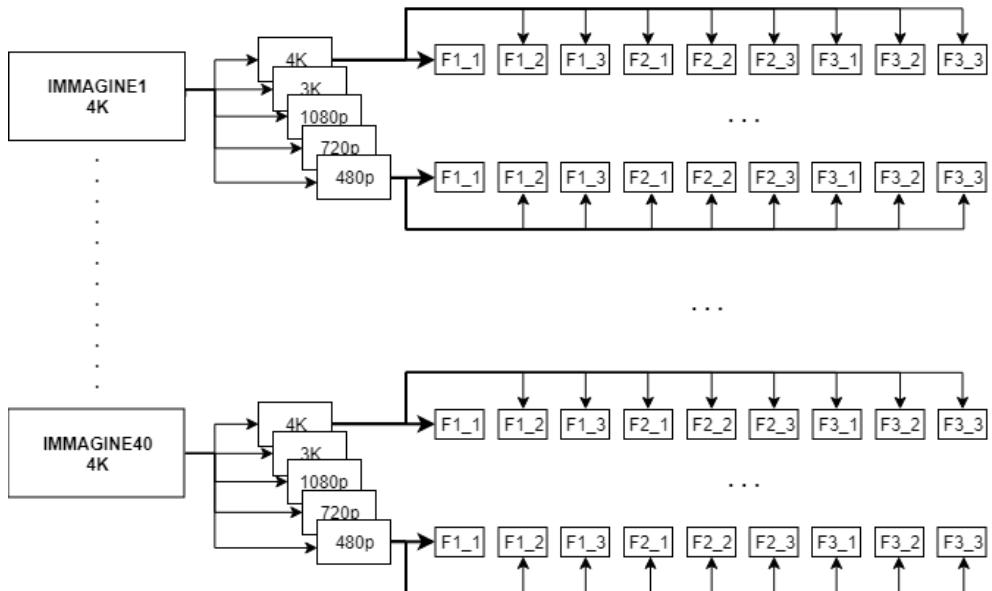


Figura 8: Generazione del dataset

4.1 Creazione del dataset

4.1.1 Codice C++

Per ottenere il risultato, precedentemente descritto nel paragrafo 4.1, è stato sviluppato un programma in C++ sfruttando la libreria OpenCV. Per ogni immagine in una specifica directory, vengono generate le immagini a risoluzioni più basse e, per ognuna di queste, vengono applicati i filtri:

```
1  for (dimension resolution : resolutions){
2      Size size(resolution.height, resolution.width);
3      Mat resizedImage;
4      resize(imageOriginal, resizedImage, size);
5      string newPath = basePath + "resized/" + resolution.name + "/" +
6          fileName + ".tif";
7
8      if (resolution.name.compare("4K") == 0)
9          copyFile(filePath, newPath);
10     else
11         imwrite(newPath, resizedImage);
12     for (int intensityF1 = 6, double intensityF2 = 1.1, int intensityF3 = 5;
13          intensityF1 <= 56, intensityF2 <= 1.5; intensityF3 <= 15;
14          intensityF1 += 25, intensityF2 += 0.2, intensityF3 += 5)
15     {
16         Mat imageFiltered = applyFilter(newPath, 1, intensityF1);
17         string newPathFiltered = basePath + "resized/" + resolution.name
18             + "/filters/" + explode(fileName, '_')[0] + "_B" + to_string(
19                 nameCounter) + ".tif";
20         imwrite(newPathFiltered, imageFiltered);
21
22         imageFiltered = applyFilter(newPath, 2, intensityF2);
23         newPathFiltered = basePath + "resized/" + resolution.name + "/"
24             + "filters/" + explode(fileName, '_')[0] + "_B" + to_string(
25                 nameCounter+3) + ".tif";
26         imwrite(newPathFiltered, imageFiltered);
27
28         nameCounter++;
29     }
30 }
```

4.1 Creazione del dataset

4.1.2 Esempi di applicazione dei filtri

Di seguito vengono riportati alcuni esempi di immagini generate con il precedente procedimento le cui differenze sono facilmente percepibili ad occhio nudo.

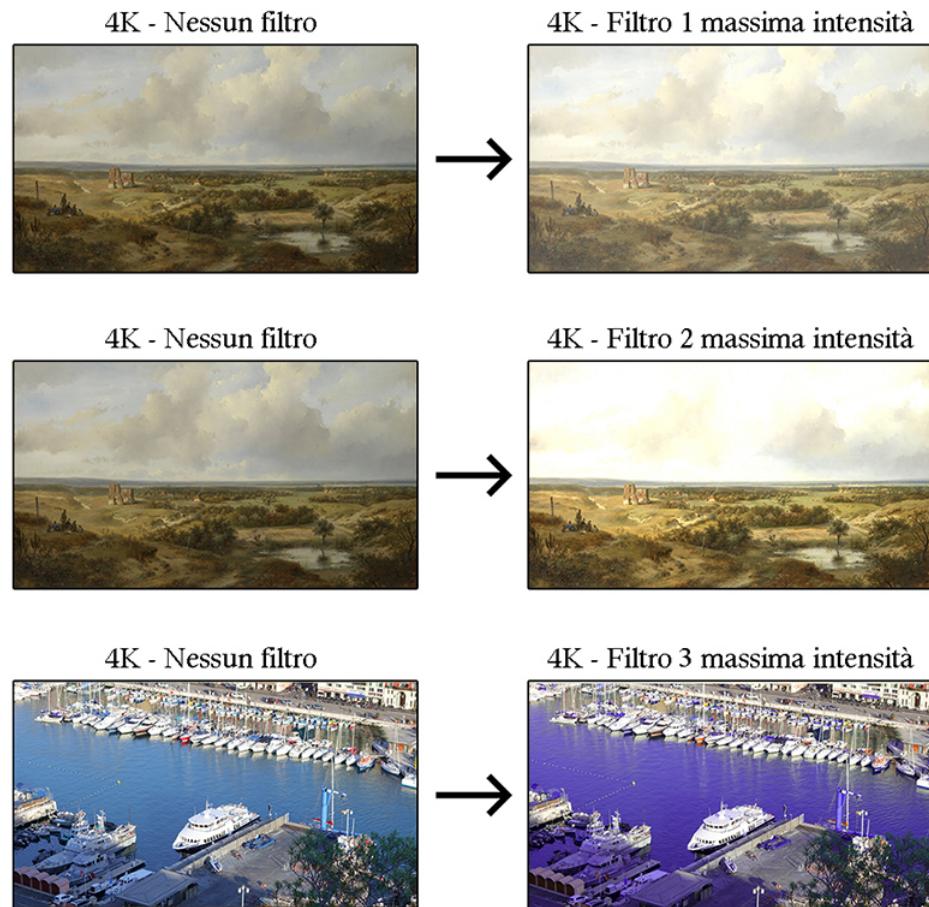


Figura 9: Esempi di applicazione dei filtri

4.2 Feature Extraction

Per addestrare la rete si è deciso di usare un tipo di addestramento detto "supervisionato". L'addestramento supervisionato è una tecnica di apprendimento automatico che mira a istruire una rete in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input, sulla base di una serie di esempi ideali costituiti da coppie di input e di output, che gli vengono inizialmente forniti. L'estrazione delle informazioni dal dataset da fornire come input alla rete viene detta Feature Extraction.

4.2.1 Preparazione degli input e degli output

Affinché si possa procedere con l'addestramento, è necessario fornire alla rete dei dati, in grado di rappresentare tutte le immagini da confrontare (input). Dobbiamo inoltre fornire delle valutazioni effettuate da un osservatore umano (output), che indichino la somiglianza tra le varie coppie di immagini da confrontare. Studiando questi dati la rete sarà in grado di emulare il comportamento umano nell'identificazione delle differenze percettive tra due immagini.

4.2.2 Estrazione dei dati dalle immagini

I dati di input vengono estratti dalle immagini nel nostro dataset, illustrato nel paragrafo 4.1. In particolare, questi dati faranno riferimento a zone delle immagini sufficientemente grandi da poter permettere alla rete di comprenderne non solo il colore, ma anche la struttura, distinguendone forme e dettagli generali. Per fare ciò, tutte le immagini sono state suddivise in 9 zone di dimensione uguale e per ognuna di esse sono stati calcolati i valori di media e varianza (dette feature). Questo procedimento viene effettuato per ognuno dei 3 canali (L, a, b). Per la media è stata usata la funzione mean() della libreria OpenCV, mentre per la varianza è stata implementata una funzione apposita:

```
1  double variance(Mat & m, int i, int j, int block_height, int block_width)
2  {
3      double variance = 0;
4      Mat m_tmp = m(Range(i, i + block_height), Range(j, j + block_width));
5      Mat m_squared(block_height, block_width, CV_64F);
6
7      multiply(m_tmp, m_tmp, m_squared);
8
9      double avg = mean(m_tmp)[0];
10     double avg_2 = mean(m_squared)[0];
11     var = sqrt(avg_2 - avg * avg);
12
13     return variance;
14 }
```

4.2 Feature Extraction

Media e varianza sono tipicamente utilizzati in colorimetria proprio per scopi simili, come ad esempio, nel calcolo dell'SSIM.

La rete dovrà avere in ingresso la lista dei valori di ogni immagine originale, associata ad ogni sua immagine filtrata. Dopo aver calcolato i valori, questi vengono inseriti in una matrice di dimensione 1800x108, dove 1800 è il numero di coppie $(Im_i, F_j(Im_i, in))$ con Im_i immagine i -esima, F_j funzione filtro j -esima e in intensità del filtro, mentre 108 è il numero di feature calcolate per ogni coppia.

In particolare, date X ed Y rispettivamente immagine originale e filtrata, n numero di zone su cui sono suddivise le immagini, una riga della matrice contiene:

- $\mu_{xi}(C)$: media di ogni zona dell'immagine X nei 3 canali;
- $\mu_{yi}(C)$: media di ogni zona dell'immagine Y nei 3 canali;
- $\sigma_{xi}^2(C)$: varianza di ogni zona dell'immagine X nei 3 canali;
- $\sigma_{yi}^2(C)$: varianza di ogni zona dell'immagine Y nei 3 canali;

con $i \in (1, n)$ e $C \in (L, a, b)$.

Le feature sono quindi: 3 canali * 9 finestre * 4 valori = 108, per riga. La matrice di questi valori rappresenterà quindi l'input della nostra rete ed ogni sua riga è detta sample.

4.2.3 Codice C++

Per estrapolare queste features è stata realizzata la seguente funzione C++:

```

1   for (int k = 0; k < nbBlockPerHeight; k++) {
2       for (int l = 0; l < nbBlockPerWidth; l++) {
3           {
4               int m = k * block_height;
5               int n = l * block_width;
6               // Media canale b
7               double avg_ob = mean(img_original[0](Range(k, k + block_height),
8                               Range(l, l + block_width)))[0];
9               double avg_cb = mean(img_filtered[0](Range(k, k + block_height),
10                             Range(l, l + block_width)))[0];
11              values.push_back(avg_ob);
12              values.push_back(avg_cb);
13              // Media canale a
14              double avg_oa = mean(img_original[1](Range(k, k + block_height),
15                             Range(l, l + block_width)))[0];
16              double avg_ca = mean(img_filtered[1](Range(k, k + block_height),
17                             Range(l, l + block_width)))[0];
18              values.push_back(avg_oa);
19              values.push_back(avg_ca);
20              // Media canale L
21              double avg_oL = mean(img_original[2](Range(k, k + block_height),
22                             Range(l, l + block_width)))[0];
23              double avg_cL = mean(img_filtered[2](Range(k, k + block_height),
24                             Range(l, l + block_width)))[0];
25              values.push_back(avg_oL);
26              values.push_back(avg_cL);
27              // Varianza canale b
28              double sigma_oa = variance(img_original[2], m, n, block_height,
29                             block_width);
30              double sigma_ca = variance(img_filtered[2], m, n, block_height,
31                             block_width);
32              values.push_back(sigma_oa);
33              values.push_back(sigma_ca);
34              // Varianza canale a
35              double sigma_ob = variance(img_original[1], m, n, block_height,
36                             block_width);
37              double sigma_cb = variance(img_filtered[1], m, n, block_height,
38                             block_width);
39              values.push_back(sigma_ob);
40              values.push_back(sigma_cb);
41              // Varianza canale L
42              double sigma_oL = variance(img_original[0], m, n, block_height,
43                             block_width);
44              double sigma_cL = variance(img_filtered[0], m, n, block_height,
45                             block_width);
46              values.push_back(sigma_oL);
47              values.push_back(sigma_cL);
48          }
49      }

```

4.2.4 Preparazione degli output

Affinché la rete possa esprimere un giudizio di somiglianza relativamente a una coppia di immagini da confrontare, si sono considerati 5 livelli crescenti di affinità: Low(L), Medium-Low(ML), Medium(M), Medium-High(MH), High(H). La rete è quindi un classificatore a cinque classi. Questi sono indicativi di quanto due immagini siano simili tra loro. Sono state quindi analizzate 1800 coppie di immagini (40 immagini * 9 immagini filtrate * 5 risoluzioni), e ad ognuna di queste è stato assegnato un giudizio in questa scala. Così come un essere umano riesce ad individuarne le differenze guardando una coppia di immagini, la rete neurale si comporterà esattamente allo stesso modo ma guardando alle immagini come un insieme di caratteristiche espresse da valori numerici. Guardando i valori rappresentativi di una coppia di immagini, la rete cercherà di estrapolare un giudizio basandosi sulla sua esperienza pregressa. Questa esperienza le viene fornita mostrandole degli esempi, forniti da un osservatore umano che dovrà effettuare tutti i confronti tra le varie coppie di immagini e dare una sua valutazione. Quando la rete sarà giunta ad un responso, dovrà comunicare all'esterno il suo risultato. Per farlo utilizzerà i neuroni di output, attivando quello corrispondente al giudizio che ha ottenuto. I neuroni di uscita saranno quindi 5, uno per ogni giudizio possibile, e il valore di uscita della rete sarà:

- 0,0,0,0,1 → L
- 0,0,0,1,0 → ML
- 0,0,1,0,0 → M
- 0,1,0,0,0 → MH
- 1,0,0,0,0 → H

Dove con 1 si indica il neurone attivo e con 0 il neurone non attivo.

4.2.5 Esempio di risposta della rete

Spieghiamo ciò che si vuole ottenere dalla rete neurale attraverso un esempio. Si considerino due immagini, una originale ed una a cui è stato applicato il primo filtro alla massima intensità. Da esse sono state estratte le feature, come visto nel paragrafo 4.2.2, che vengono poi passate in input ad una rete addestrata. Passando attraverso una serie di "livelli nascosti", che la rete crea per effettuare le elaborazioni necessarie all'ottenimento dell'output, la rete si accorge che le due immagini sono percettivamente molto diverse tra loro. Procede quindi attivando l'ultimo neurone di uscita.

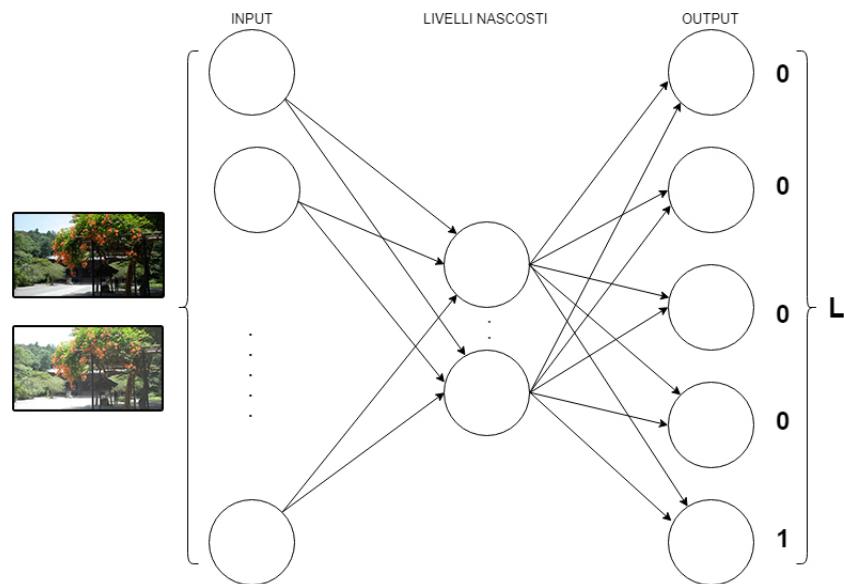


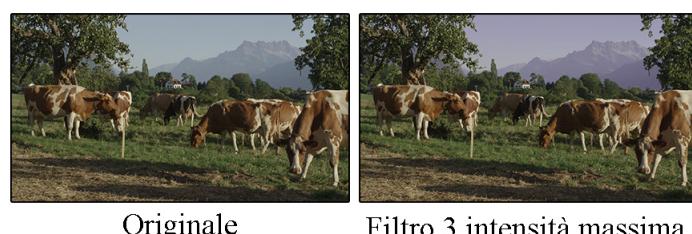
Figura 10: Esempio di risposta della rete

4.2.6 Esempi di valutazioni da parte dell'osservatore

L'osservatore umano, compirà lo stesso procedimento ma assegnando un valore simbolico ad ogni coppia di immagini, come illustrato nel paragrafo 4.2.4. Di seguito vengono riportati alcuni esempi di valutazioni effettuate dall'osservatore.



H



Filtro 3 intensità massima

M



Filtro 2 intensità massima

L

Figura 11: Esempi di risposta dell'osservatore umano

4.3 Considerazioni preliminari

Prima di procedere con l'addestramento della rete, analizziamo i dati ottenuti nei precedenti paragrafi.

4.3.1 Analisi dei giudizi dell'osservatore

Prendendo in considerazione tutte le 1800 coppie di immagini originali e filtrate e associandole ai rispettivi giudizi dati dall'osservatore nel paragrafo 4.2.4, si nota una tendente accentuazione delle differenze cromatiche, al diminuire della risoluzione. In particolare, prendendo come esempio il giudizio dato ad una coppia di immagini a 4K e confrontandolo con quello della stessa coppia a risoluzioni inferiori, per tutte le possibili coppie nel dataset, si ottengono i seguenti risultati:

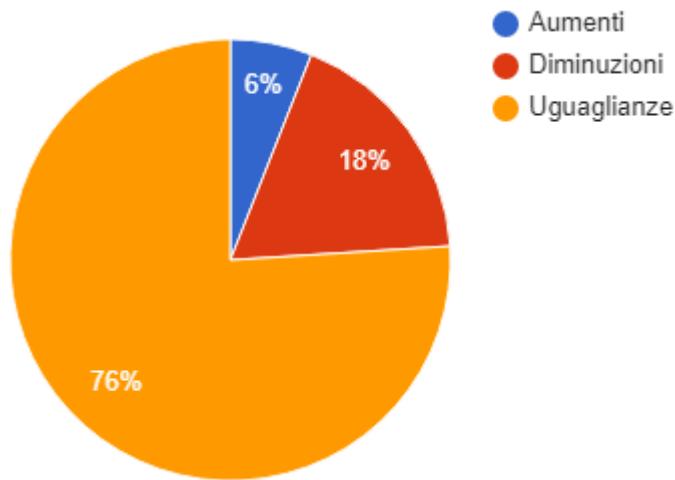


Figura 12: "Variazione della percezione umana"

Come si evince dal grafico, all'abbassarsi della risoluzione, le differenze percettive tra le immagini rimangono nella maggioranza dei casi invariate e tendono ad essere accentuate, causando così una valutazione più bassa dell'indice di similarità. Ciò accade nel 94% dei casi presi in considerazione.

4.3.2 Calcolo del Lab-SSIM

Per avvalorare questa tesi ed evitare che questo comportamento possa essere riconducibile solo alla percezione dell'osservatore umano preso in considerazione, si è deciso di effettuare lo stesso procedimento considerando però un indice matematico capace di esprimere le differenze percettive tra due immagini $L^*a^*b^*$, il Lab-SSIM. Questo indice è diverso dall'SSIM tradizionale e consiste nel calcolare, i valori di luminanza dei canali a e b, di contrasto e di struttura del canale L e fare la media tra tutti questi valori.

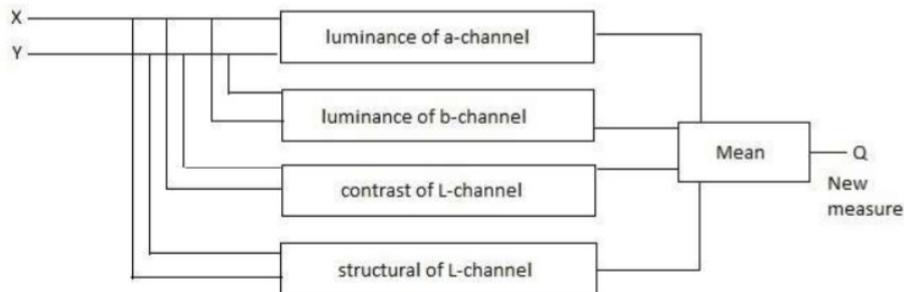


Figura 13: Lab-SSIM

Il Lab-SSIM si basa quindi su tre misurazioni di confronto tra i campioni di x e y : luminanza l , contrasto c e struttura s . Le singole funzioni di confronto sono:

$$1. \ l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$2. \ c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

$$3. \ s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

in cui si hanno:

- μ_x e μ_y sono, rispettivamente, la media di x e y ;
- σ_x^2 e σ_y^2 sono, rispettivamente, la varianza di x e y ;
- σ_{xy} è la covarianza di x e y ;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ sono due variabili per stabilizzare la divisione con il denominatore inadatto;
- c_3 è una costante pari a $c_2/2$.
- $k_1 = 0.01$ e $k_2 = 0.03$ sono costanti predefinite.

4.3.3 Analisi dei risultati ottenuti con il Lab-SSIM

Dopo aver calcolato il Lab-SSIM per tutte le coppie di immagini da confrontare, si passa ad analizzare i risultati, cercando di capire se, abbassando la risoluzione, le differenze continuano ad essere visibili. Si fanno quindi le stesse considerazioni del paragrafo 4.3, ma questa volta considereremo come giudizio l'indice matematico e non il giudizio dato dall'osservatore umano.

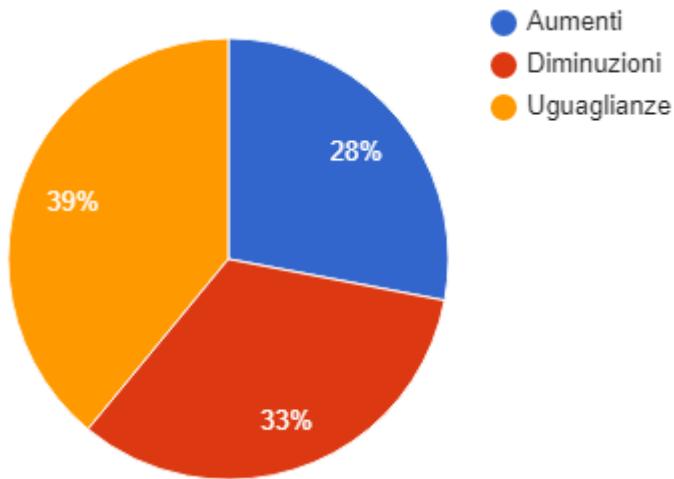


Figura 14: Variazione del Lab-SSIM

In questo caso, i giudizi rimangono invariati o si abbassano nel 72% dei casi. La tendenza sembra quindi essere confermata. Da questi risultati si può già dedurre che all'abbassarsi della risoluzione, le differenze percettive non solo rimangono percepibili ma tendono addirittura ad accentuarsi.

4.4 Addestramento della rete

Adesso che siamo in possesso degli input e degli output da fornire alla rete, possiamo passare ad effettuare un primo addestramento. Considerando x e t rispettivamente l'insieme delle feature estratte dalle immagini e l'insieme dei giudizi dati dall'osservatore, si addestra la rete utilizzando il seguente codice Matlab:

```
1 % Funzione di training
2 trainFcn = 'trainlm';
3
4 % Creazione della rete
5 hiddenLayerSize = 10;
6 net = patternnet(hiddenLayerSize, trainFcn);
7
8 % Suddivisione del dataset in training, test e validation set
9 net.divideFcn = 'dividerand';
10 net.divideMode = 'sample';
11 net.divideParam.trainRatio = 70/100;
12 net.divideParam.valRatio = 10/100;
13 net.divideParam.testRatio = 20/100;
14
15 % Funzione per il calcolo delle performance e lista di grafici da generare
16 net.performFcn = 'mse'; % mean squared error
17
18 net.plotFcns = {'plotperform','plottrainstate','ploterrhist',...
19 'plotconfusion', 'plotroc'};
20
21 % Addestramento della rete
22 [net,tr] = train(net,x,t);
23
24 % Test della rete e performance
25 y = net(x);
26 e = gsubtract(t,y);
27 performance = perform(net,t,y)
28 tind = vec2ind(t);
29 yind = vec2ind(y);
30 percentErrors = sum(tind ~= yind)/numel(tind);
```

Il dataset iniziale è stato diviso in 3 parti: Training Set, Validation Set e Test Set. Essi servono rispettivamente per addestrare, validare e testare la rete. In particolare, il Training Set serve per far capire alla rete quali risposte deve dare in base agli input che gli vengono forniti, il Validation Set serve per evitare il fenomeno dell'overfitting, mentre il Test Set serve per valutarne le performance alla fine dell'addestramento.

4.4 Addestramento della rete

4.4.1 Valutazione delle prestazioni

Una volta completato l'addestramento, per valutarne le prestazioni, analizziamo la matrice di confusione, che mostra, sulla diagonale, quanti sample sono stati valutati correttamente:

Training Confusion Matrix							Validation Confusion Matrix							
Output Class	1	340 27.0%	22 1.7%	7 0.6%	2 0.2%	2 0.2%	91.2% 8.8%	1	44 24.4%	10 5.6%	2 1.1%	2 1.1%	1 0.6%	74.6% 25.4%
	2	31 2.5%	263 20.9%	16 1.3%	5 0.4%	3 0.2%	82.7% 17.3%	2	6 3.3%	25 13.9%	3 1.7%	2 1.1%	0 0.0%	69.4% 30.6%
	3	0 0.0%	9 0.7%	84 6.7%	4 0.3%	0 0.0%	86.6% 13.4%	3	0 0.0%	4 2.2%	3 1.7%	6 3.3%	2 1.1%	20.0% 80.0%
	4	0 0.0%	2 0.2%	17 1.3%	223 17.7%	6 0.5%	89.9% 10.1%	4	1 0.6%	2 1.1%	5 2.8%	25 13.9%	7 3.9%	62.5% 37.5%
	5	0 0.0%	0 0.0%	2 0.2%	10 0.8%	212 16.8%	94.6% 5.4%	5	0 0.0%	0 0.0%	2 1.1%	5 2.8%	23 12.8%	76.7% 23.3%
		91.6% 8.4%	88.9% 11.1%	66.7% 33.3%	91.4% 8.6%	95.1% 4.9%	89.0% 11.0%		86.3% 13.7%	61.0% 39.0%	20.0% 80.0%	62.5% 37.5%	69.7% 30.3%	66.7% 33.3%
	1	2	3	4	5		1	2	3	4	5			
	Target Class	Target Class	Target Class	Target Class	Target Class		Target Class	Target Class	Target Class	Target Class	Target Class			
Test Confusion Matrix							All Confusion Matrix							
Output Class	1	83 23.1%	16 4.4%	4 1.1%	1 0.3%	0 0.0%	79.8% 20.2%	1	467 25.9%	48 2.7%	13 0.7%	5 0.3%	3 0.2%	87.1% 12.9%
	2	22 6.1%	50 13.9%	14 3.9%	4 1.1%	0 0.0%	55.6% 44.4%	2	59 3.3%	338 18.8%	33 1.8%	11 0.6%	3 0.2%	76.1% 23.9%
	3	1 0.3%	3 0.8%	15 4.2%	4 1.1%	2 0.6%	60.0% 40.0%	3	1 0.1%	16 0.9%	102 5.7%	14 0.8%	4 0.2%	74.5% 25.5%
	4	0 0.0%	9 2.5%	8 2.2%	42 11.7%	10 2.8%	60.9% 39.1%	4	1 0.1%	13 0.7%	30 1.7%	290 16.1%	23 1.3%	81.2% 18.8%
	5	0 0.0%	1 0.3%	1 0.3%	14 3.9%	56 15.6%	77.8% 22.2%	5	0 0.0%	1 0.1%	5 0.3%	29 1.6%	291 16.2%	89.3% 10.7%
		78.3% 21.7%	63.3% 36.7%	35.7% 64.3%	64.6% 35.4%	82.4% 17.6%	68.3% 31.7%		88.4% 11.6%	81.3% 18.8%	55.7% 44.3%	83.1% 16.9%	89.8% 10.2%	82.7% 17.3%
	1	2	3	4	5		1	2	3	4	5			
	Target Class	Target Class	Target Class	Target Class	Target Class		Target Class	Target Class	Target Class	Target Class	Target Class			

Figura 15: Matrice di confusione

Ciò che si evince è che l'accuratezza della rete è di poco inferiore al 70%, precisamente 68.3% sul Test set. La precisione della rete è bassa, e può sicuramente essere migliorata. Idealmente, si vorrebbe ottenere che il 100% dei giudizi fosse corretto e quindi la matrice dovrebbe avere valori soltanto sulla diagonale. In questo caso tuttavia, notiamo una buona precisione sulle

valutazioni estreme (H ed L), ma una più scarsa capacità di giudizio delle coppie di immagini che hanno ricevuto una valutazione meno netta. Ciò può essere dovuto al fatto che le reti neurali necessitano di un grande numero di dati nel training set affinché possano essere addestrate in maniera corretta. Euristicamente, dato x numero di feature in input alla rete, t numero di neuroni di uscita, e h numero di neuroni presenti nei livelli nascosti, il numero n adeguato di esempi che devono essere presenti nel dataset deve essere:

$$n \geq ((x * h) + (t * h)) * 5.$$

Nel nostro caso, le feature sono l'insieme di varianze e medie calcolate per tutte le coppie (108 feature), come descritto dettagliatamente nel paragrafo 4.2.2, i neuroni di output sono 5, mentre i neuroni nascosti sono fissati a 6 per ipotesi. Effettuando il calcolo di cui sopra, si ottiene:

$$n \geq ((108 * 6) + (5 * 6)) * 5 = 3390.$$

Quindi, per addestrare adeguatamente la nostra rete sarebbero necessari almeno 3390 esempi di confronti tra coppie di immagini. Non disponendo di questi dati, ciò che possiamo fare è effettuare la feature selection, cioè l'identificazione delle feature maggiormente discriminanti tra le 108 ottenute durante la feature extraction. Riducendo il numero di feature in ingresso alla rete, anche il numero di esempi necessari sarà ridotto, potendo addestrare adeguatamente la rete anche con soli 1800 confronti.

4.5 Feature selection

Durante la feature selection, si applica un algoritmo in grado di identificare, tra le varie feature estratte durante la feature extraction, quelle con il maggiore potere discriminante. Conoscendo questo dato, è poi facile diminuire il numero di feature in ingresso alla rete, andando a selezionare solo quelle realmente necessarie. Per effettuare la feature selection si comincia selezionando una feature dal dataset iniziale, essa sarà data in ingresso ad una rete che cercherà di calcolare il risultato di cui necessitiamo. Si osserva il risultato ottenuto e si sceglie la feature che, da sola, dà il più basso errore di classificazione. Si procede poi aggiungendo una feature e ritestando la rete con la coppia ottenuta. Si va avanti così aggiungendo di volta in volta una nuova feature ai test fino al soddisfacimento di un criterio. Questa tipologia di feature selection è detta forward sequential feature selection. Esiste anche quella backward in cui si parte dall'insieme totale delle feature e si procede eliminandone una ad ogni step. Alla fine di questo processo, si osservano i risultati della rete nei vari casi e si prendono le feature che hanno permesso di ottenere i risultati migliori.

4.5.1 Analisi dei risultati

Dopo oltre 300 iterazioni, la feature selection ha selezionato molteplici volte alcune particolari feature, considerandole quindi più importanti. Il seguente grafico mostra il numero di volte che una certa feature, appartenente al gruppo totale delle feature numerate da 1 a 108, è stata scelta durante la feature selection:

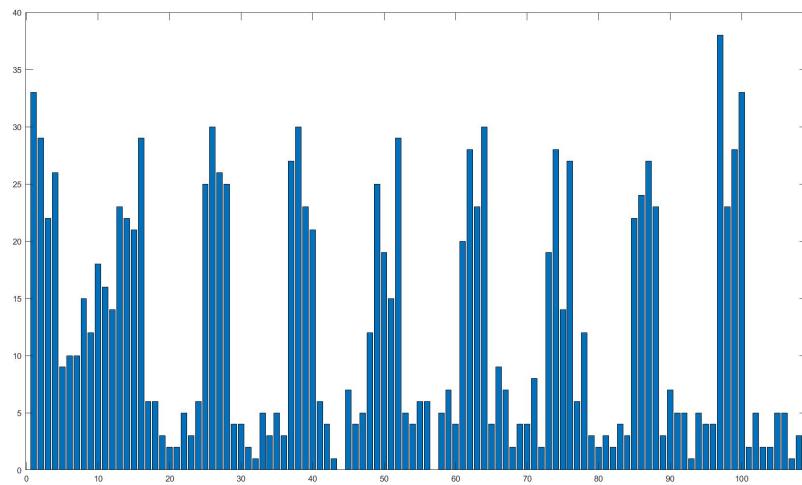


Figura 16: Risultati della Feature Selection

Analizzando il grafico si decide quindi di considerare per l'addestramento della rete soltanto le feature su cui sono presenti dei picchi evidenti di utilizzo. E' inoltre necessario assicurarsi di selezionare almeno una feature per ogni finestra, sia per l'immagine originale (x) che per quella a cui è stato applicato il filtro(y). Vengono quindi considerate le seguenti feature: $Mx1(L)$, $My1(L)$, $Sx2(L)$, $Sy2(L)$, $Sx3(L)$, $My3(L)$, $Mx4(L)$, $My4(L)$, $Mx5(L)$, $Sy5(L)$, $Sx6(L)$, $Sy6(L)$, $My7(L)$, $Sx7(L)$, $Sx8(L)$, $Sy8(L)$, $My9(L)$, $Sx9(L)$.

In questo modo siamo riusciti ad ottenere una diminuzione sostanziale delle feature in ingresso alla rete, passando da 108 a 18.

4.5.2 Undersampling e oversampling

Un’ulteriore causa della scarsa precisione della rete può essere lo sbilanciamento del dataset. E’ infatti molto probabile che, durante l’assegnamento dei giudizi da parte dell’osservatore umano, una certa classe di risultati sia stata usata più volte rispetto ad un’altra. Si procede quindi calcolando le occorrenze dei vari giudizi dati dall’osservatore:

- H: 528
- MH: 416
- M: 183
- ML: 349
- L: 324

I risultati evidenziano un palese sbilanciamento causato da un eccessivo numero di H e un non sufficiente numero di M. Per riequilibrare la situazione si effettua l’undersampling, cioè la diminuzione dei sample, di H e l’oversampling, cioè la duplicazione dei sample, di M.

- H: 428
- MH: 416
- M: 366
- ML: 349
- L: 324

A seguito di questa elaborazione, il dataset è bilanciato e contiene un totale di 1883 sample. In questo modo la rete potrà essere addestrata con un numero sufficiente di esempi per ogni possibile giudizio e potrà quindi essere più precisa.

4.6 Addestramento a seguito della feature selection

4.6 Addestramento a seguito della feature selection

Andando a rieffettuare lo stesso procedimento del paragrafo 4.4 ma usando 18 feature anziché 108, si addestra una nuova rete, ottenendo le seguenti matrici di confusione:



Figura 17: Matrice di confusione

Come si evince dalla matrice di test, il risultato è abbastanza migliorato, arrivando ad una precisione del 77%. A questo punto è evidente che, data la complessità del problema, potrebbero essere necessarie soluzioni più avanzate per ottenere una precisione superiore. Si ricorre quindi al classificatore ensemble.

4.6.1 Classificatore Ensemble

Un classificatore ensemble è un complesso di reti neurali che lavorano separatamente ma contribuiscono all'ottenimento di un risultato comune. Le feature di input vengono date in ingresso a tutte le reti e ognuna di queste restituisce un risponso sulla base della sua personale esperienza. Tutti i risultati delle reti vengono poi inviati ad una funzione che ha il compito di decidere il risultato finale. Per decidere, si usa semplicemente il principio di maggioranza, il risponso che è stato dato da più reti è probabilmente quello corretto. Nel nostro caso, sono state addestrate 100 reti neurali, dette classificatori, con 18 input e 25 neuroni nascosti.

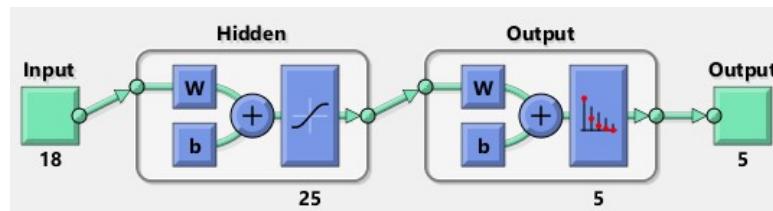


Figura 18: Struttura di un classificatore

Selezionando 5 reti che hanno mostrato buone prestazioni, il nostro classificatore ensemble avrà la seguente forma:

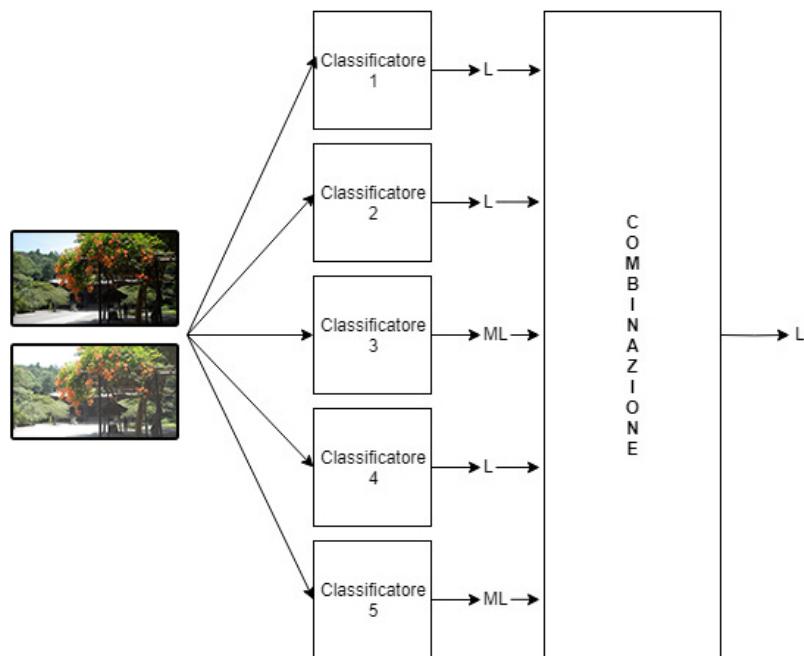


Figura 19: Classificatore Ensemble

Sorge un unico problema, cosa fare in una situazione di pareggio? Il criterio adottato è quello di effettuare la media tra i due giudizi vincenti. Se tuttavia la media non dovesse ricadere nettamente su un giudizio, ad esempio nel caso di due reti che restituiscono H e due reti che restituiscono M, il giudizio tenderà sempre verso il centro (e quindi verso M), poiché è considerata una risposta più plausibile in presenza di indecisione. Sia r la matrice contenente i responsi dei 5 classificatori ordinate per riga, la funzione Matlab per stabilire il giudizio finale è:

```

1 % Si calcola la somma per colonna
2 total = sum(r);
3
4 % L'indice della colonna associata alla somma maggiore
5 % corrisponde al giudizio maggiormente scelto
6 maxValue = max(total);
7 indexes = find(total == maxValue);
8
9 % Gestione dei pareggi
10 if length(indexes) > 1
11     average = mean(indexes)
12     if floor(average) ~= average
13
14         if average < 3
15             y = ceil(average)
16         else
17             y = floor(average)
18         end
19     else
20         y = average;
21     end
22 else
23     y = indexes;
24 end

```

4.6.2 Valutazione delle prestazioni del classificatore ensemble

Si valuta l'errore commesso dal classificatore ensemble su due dataset. Il primo, sarà l'intero dataset preso in considerazione durante la ricerca. Il secondo sarà il risultato dell'unione dei soli sample di test che sono stati usati durante l'addestramento delle singole reti. Di seguito sono riportate le matrici di confusione nei due casi:

Caso 1

374	46	5	3	0
104	197	91	19	5
34	40	252	34	6
16	16	50	236	31
2	5	29	66	222

Caso 2

201	23	2	2	0
56	113	54	11	2
17	17	152	18	2
9	4	32	121	22
0	1	20	41	121

Figura 20: Matrici di confusione

Analizzandole si ottiene una precisione di poco inferiore al 90%, per la precisione dell'88% in entrambi i casi. Abbiamo raggiunto una precisione adeguata, considerando che stiamo risolvendo un problema di classificazione a 5 classi e possiamo usare questa rete per effettuare il test.

5 Test

Si procede quindi sfruttando la rete neurale per capire se, al calare della risoluzione, le differenze percettive tra due immagini continuano ad essere visibili. Vengono quindi selezionati 8 campioni significativi dal dataset contenente i confronti tra le immagini e passati alla rete neurale. Dai risultati ottenuti dalla rete neurale, si tracciano dei grafici sull'andamento del giudizio considerandolo come un valore numerico da 1(L) a 5(H):

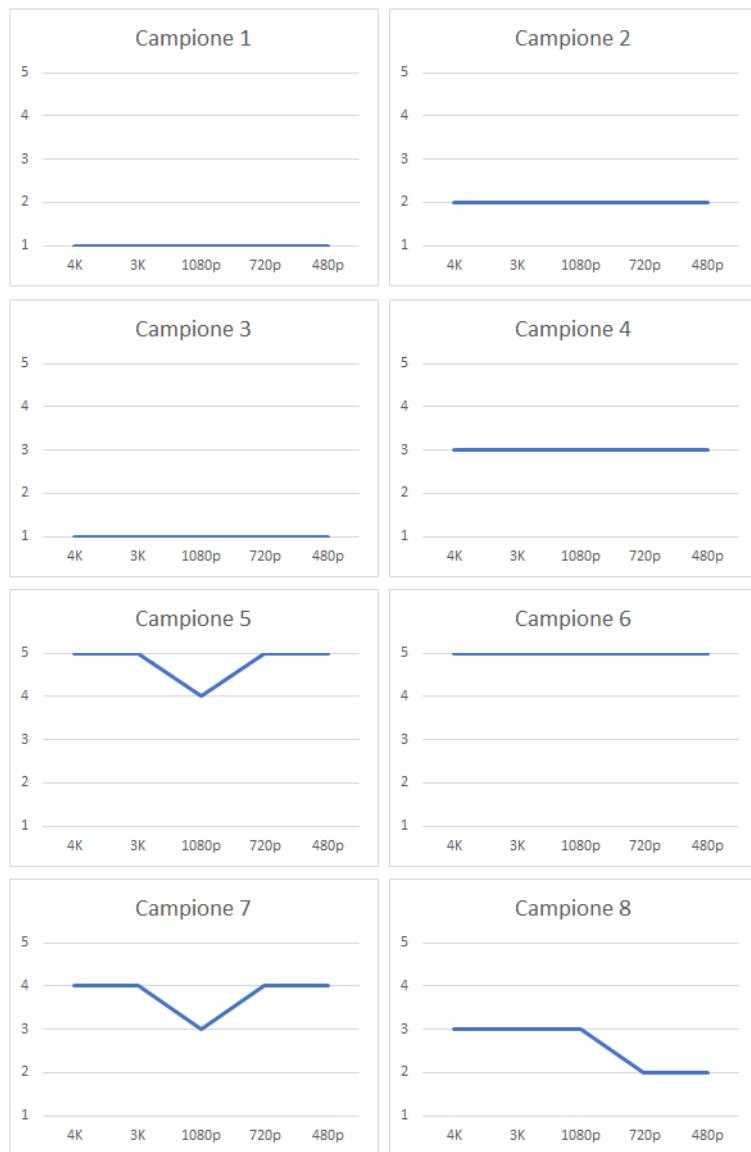


Figura 21: Test sui campioni

6 Conclusioni

Analizzando i risultati ottenuti durante la fase di test risulta evidente che, al calare della risoluzione, le differenze percettive tra due immagini continuano ad essere visibili e tendono leggermente ad accentuarsi. Si può quindi dire che queste elaborazioni possono essere fatte in sicurezza anche a 480p. Sfruttando l'esempio del paragrafo 2.3.1, un'elaborazione effettuata su 4K necessita di lavorare su 8.294.400 pixel, mentre su 480p i pixel si riducono a soli 261.120. Si ricava immediatamente che il risparmio di tempo derivante da questo abbassamento di risoluzione è di circa il 97%. Questo risultato permetterà di risparmiare tempo e denaro, utilizzando strumenti per l'acquisizione di immagini a risoluzioni più basse (e quindi più veloci), riducendo anche il tempo di elaborazione tra le immagini da confrontare.

Il risparmio di tempo consentirà inoltre di effettuare un maggior numero di test su più campioni diversi, aumentando notevolmente la possibilità di individuare falle nei processi produttivi e consentendo agli operatori di porvi prontamente rimedio.

7 Ringraziamenti

Questa avventura è giunta al termine e non posso non approfittarne per ringraziare tutte le persone che hanno reso speciali questi anni.

Il mio primo ringraziamento va a coloro che mi hanno permesso di intraprendere questo percorso, credendo in me e sostenendomi fin dal primo giorno, i miei genitori, Silvio e Aurora. Grazie per avermi appoggiato in tutte le mie scelte e per non aver mai smesso di essere orgogliosi di me.

A mia sorella Ambra e a mio cognato Giovanni che mi sono sempre stati vicini nonostante la distanza. Grazie per aver reso questi anni meno duri con la vostra presenza e i vostri preziosi consigli.

A mio nonno Vincenzo, alle mie nonne Lina e Gemma e a mia zia Anna, per essere stati un punto di riferimento durante tutti questi anni, accompagnandomi nella crescita e trasmettendomi tutti i vostri valori.

Grazie a mia zia Ornella, per avermi dato la spinta iniziale per affrontare tutte le sfide matematiche che questo percorso aveva in serbo per me.

Ringrazio poi il professor Pistolesi, per il grande aiuto e infinita disponibilità dimostratemi nel periodo passato a redigere la tesi insieme e per avermi trasmesso molte delle competenze che mi saranno fondamentali nella mia futura carriera.

Come non citare Simone, amico di una vita e fratello, onnipresente in tutti i momenti salienti della mia vita. Nel corso degli anni ci siamo fatti tanti film in testa, uno più assurdo dell'altro, ma il vero film è la nostra amicizia e io non avrei potuto scegliere persona migliore con cui girarlo.

Un grazie a Ciccio, per aver reso questi anni più spensierati con le sue idee fuori dagli schemi e la sua infinita simpatia.

Infine, voglio ringraziare tutti i miei colleghi e amici, per non avermi mai lasciato solo nelle difficoltà andando avanti uniti verso la meta e sostenendoci l'un l'altro.

Ne è passato di tempo da quando ho intrapreso questa strada tortuosa e piena di difficoltà, ma guardandomi indietro queste diventano irrilevanti e riaffiorano soltanto tutti i bei momenti che hanno costellato questi preziosi anni, resi indimenticabili da ognuno di voi.

Grazie a tutti!