UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"Filtraggio e Pre-Processing di Dataset per lo Smistamento Automatico dei Rifiuti di Vetro"

Relatore: Prof. Emanuele Menegatti

Laureando: Leonardo Pontello

ANNO ACCADEMICO 2022 – 2023

Data di laurea 16/11/2023

Abstract

L'attività di smistamento dei rifiuti è un processo ripetitivo, logorante e spesso pericoloso, suscettibile a numerosi errori umani. L'intelligenza artificiale offre un potenziale significativo per automatizzare e migliorare questo lavoro.

Il sistema in questione deve saper individuare gli oggetti estranei dal flusso di rifiuti su un nastro trasportatore. Questo richiede un dataset di coppie di immagini che raffigurano la stessa porzione di nastro prima e dopo l'eliminazione degli oggetti intrusi.

Il dataset è stato raccolto da un sistema provvisorio e sperimentale trascurando alcune problematiche che hanno reso imperfetto il campionamento.

Questo studio si occupa di formulare un algoritmo che sia in grado di filtrare il dataset per scartare le foto inconsistenti.

Sono stati osservati diversi indici di similarità, alcuni ottenuti con le CNN, e attraverso l'analisi del PCA sono state individuate le misure più rilevanti.

Sono state scelte poi le 4 misure più importanti per rappresentare le coppie di foto in una classificazione attraverso un algoritmo di apprendimento supervisionato, il Support Vector Machine (SVM), e un algoritmo di apprendimento non supervisionato, il Clustering, in particolare sono stati osservati il k-means, l'Agglomerative Clustering, lo Spectral Clustering e il Birch.

Gli algoritmi sono stati testati e studiati con le loro molteplici configurazioni per ottenere il modello ottimo.

Infine, i modelli sono stati provati con ulteriori dati per analizzare le loro performance a larga scala e metterli a confronto

Indice

Capitolo 1

Introduzione

Il sistema di smistamento in questione consiste in un operatore umano che ha il compito di individuare ed eliminare corpi intrusi (oggetti di plastica, legno, metallo etc..) da un nastro trasportatore che trasporta i rifiuti del vetro. Alla lunga, questo compito risulta ripetitivo, impreciso e anche pericoloso per l'uomo, in quanto si ripetono le stesse operazioni per ore e ci si può tagliare con il vetro.

Un sistema automatizzato che sia in grado di compiere questa mansione deve innanzitutto saper individuare gli oggetti intrusi dal vetro.

Sono state collocate due telecamere rgb agli estremi dell'operatore (*Figura 1*) per poter inquadrare il nastro prima e dopo il suo operato. In questo modo, campionando le immagini in maniera sincronizzata, è possibile confrontare le due porzioni di nastro e notare gli oggetti che sono stati tolti identificando così gli intrusi.

L'approccio proposto consiste nel campionare un dataset di coppie di immagini e nell'utilizzarlo per allenare un algoritmo di riconoscimento self-supervised, che, grazie al confronto delle immagini, individua in modo autonomo gli intrusi, senza richiedere il labelling dei dati. In questo modo,
da una foto del nastro trasportatore, l'algoritmo sarà in grado di individuare i corpi intrusi.

Il dataset è stato campionato in 12 giorni ottenendo circa 16000 campioni.

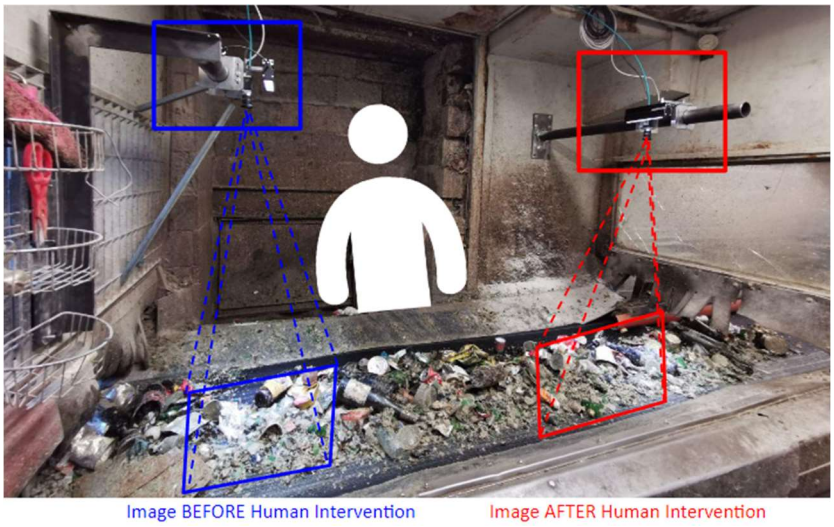


Figura 1: sistema di smistamento dei rifiuti con disposizione della strumentazione per il campionamento del dataset

1.1 Problemi di campionamento

Nel sistema in (*Figura 1)* le telecamere non sono state calibrate nella stessa maniera quindi, le immagini di ogni coppia hanno due prospettive diverse (*Figura 2*), il nastro trasportatore è sprovvisto

di encoder, quindi, non è stato possibile sincronizzare gli scatti con il nastro ottenendo delle immagini totalmente o parzialmente sfasate (*Figure 3 e 4*).

In 12 giorni di campionamento l'operatore ha commesso errori umani di valutazione che hanno compromesso il dataset per l'apprendimento del sistema (*Figure 5 e 6*).

Infine, quando il nastro era fermo, il sistema continuava a campionare, quindi, vi sono dei gruppi di foto identiche da eliminare.

Tutti questi problemi di sampling hanno portato il sistema di apprendimento a commettere bias e imperfezioni. Diventa quindi necessario filtrare tutto il dataset per eliminare gli accoppiamenti di foto sbagliate.

Figura 2: coppia di foto con prospettive diverse

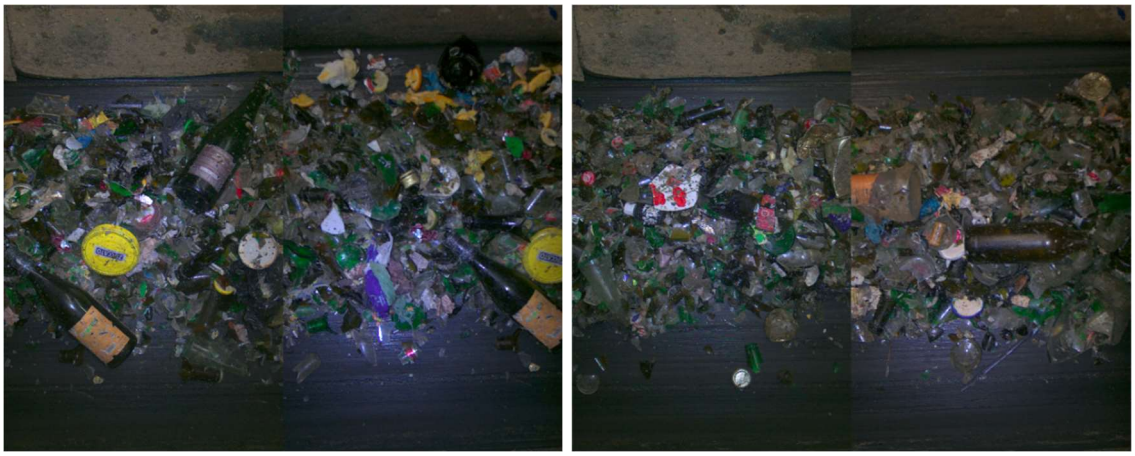


Figure 3 e 4: foto parzialmente e totalmente sfasate tra loro

Figure 5 e 6: bottiglia di plastica dimenticata dall'operatore e presenza umana nella foto

1.2 Scopo della tesi

L' obiettivo è quello di trovare un algoritmo che sia in grado di distinguere le coppie di foto del dataset da tenere da quelle da scartare.

Sono stati analizzati diversi parametri per misurare la similarità tra le immagini di ogni coppia e, attraverso il Principal Components Analysis (PCA), è stata fatta un'analisi del dataset per evidenziare le misure più rilevanti.

Dopo aver scelto le misure di similarità più adatte, per classificare le immagini in due classi ("corrette" e "da scartare"), è stata testata una tecnica di apprendimento supervisionato (SVM) e una di apprendimento non supervisionato (clustering). In conclusione, sono state analizzate e confrontate le performance dei due modelli, evidenziando i vantaggi e gli svantaggi dei due approcci.

1.3 Labelling dei dati

Attraverso il *codice 1* sono state etichettate "manualmente" le coppie di foto come idonee o non idonee per poi salvare gli esiti in un JsonArray. Il nome della directory della coppia di foto è la key e l'esito è il value.

La fase di labelling ha riportato 166 casi idonei e 36 non idonei, di conseguenza il training set risulta sbilanciato.

Capitolo 2

Misure di similarità

Esistono degli indicatori che esprimono quanto due immagini sono simili tra loro secondo diversi criteri. Questi possono basarsi su features globali o locali e in alcuni casi utilizzano delle reti neurali chiamate Convolutional Neural Network (CNN) che calcolano la convoluzione di un'immagine.

2.1 CNN

Una Convolutional Neural Network [1] (CNN) è una rete neurale composta da molti layers, addestrati per rilevare feature diverse di un'immagine. Ad ogni immagine in ingresso vengono applicati dei filtri a diverse risoluzioni e l'output viene utilizzato come input per il layer successivo. I filtri possono iniziare con feature molto semplici, ad esempio la luminosità o i bordi, e diventare sempre più complessi fino a includere feature che definiscono in modo univoco l'oggetto. Ogni layer va ad alterare i dati al fine di apprendere le feature specifiche dei dati stessi. I layers più diffusi sono convoluzionali, di attivazione (comunemente basato su funzioni ReLU) e di pooling (*Figura 8*).

- Un layer convoluzionale (*Figura 7*) sottopone il segnale in input a dei filtri convoluzionali chiamati kernel. Questi sono delle matrici con dei pesi che dipendono dalla feature cercata e che vengono applicate a tutte le regioni dell'immagine ottenendo la feature map.
- L'unità lineare rettificata (ReLU) va a mappare i valori negativi a zero rendendo l'addestramento più rapido ed efficiente in quanto solo le feature attivate vengono trasmesse ai layer successivi.
- Il pooling è un'operazione di downsampling non lineare, che riduce il numero di parametri che la rete dovrebbe apprendere e che va quindi a semplificare l'output

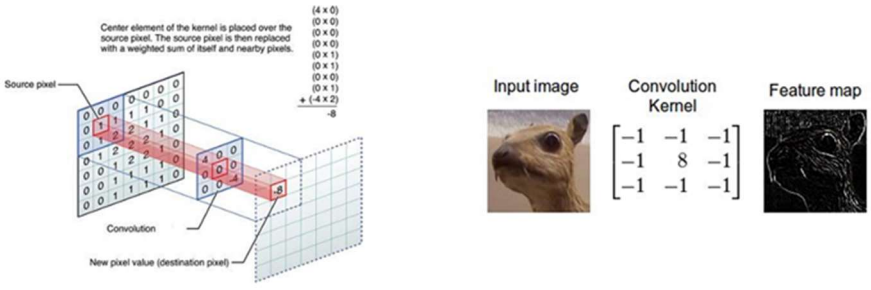


Figura 7: funzionamento filtro convoluzionale [16]

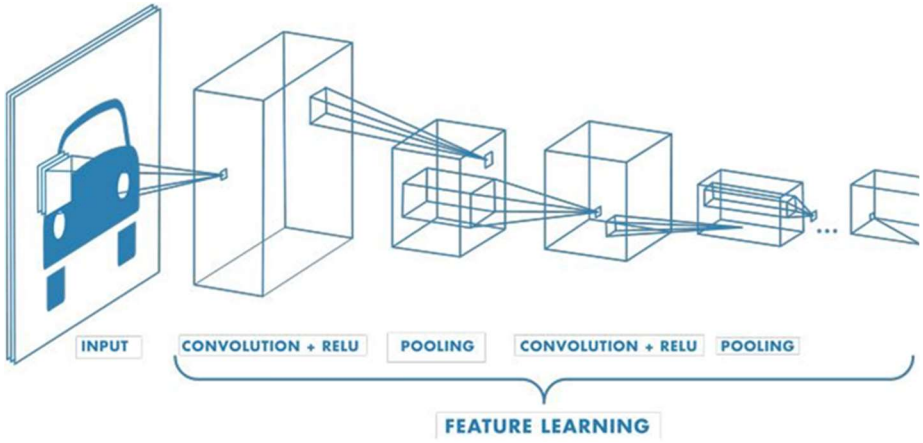


Figura 8: schema dei layer di una CNN [20]

2.2 Elenco delle misure

Per quantificare la similarità di due immagini possono essere esaminati diversi parametri e features. Le caratteristiche locali riguardano dettagli specifici in regioni diverse dell'immagine mentre quelle globali vanno a descrivere l'immagine per intero.

2.2.1 RMSE

Root Mean Square Error [2], ossia la media della differenza al quadrato di ogni pixel:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{h-1}\sum_{j=0}^{w-1}\left(I(i,j) - I'(i,j)\right)^2}{w \times h}}$$

- Matrice di riferimento I (o foto iniziale o foto finale)
- Matrice da confrontare I'
- Altezza h
- Larghezza w

2.2.2 PSNR

Il Peak Signal to Noise Ratio [2] corrisponde al rapporto tra la massima potenza di segnale dei pixel e la potenza del rumore in dB:

$$PSNR = 10\log_{10}\frac{MAX^2}{MSE}$$

- Quadrato del valor medio delle differenze tra pixel delle due immagini $MSE = RMSE^2$
- Valore massimo che può assumere un pixel MAX

2.2.3 SSIM

Lo Structural Similarity Index [3] è un indice che si misura grazie ad un modello che utilizza come percezione la degradazione dell'immagine, considerata come cambio di percezione delle informazioni strutturali, quindi, i pixel interdipendenti o spazialmente chiusi.

Il modello utilizza altri patterns come il mascheramento del contrasto e della luminanza per eliminare le distorsioni.

Nel modello in *Figura 9* i segnali x e y sono riguardano le stesse regioni delle due immagini a confronto.

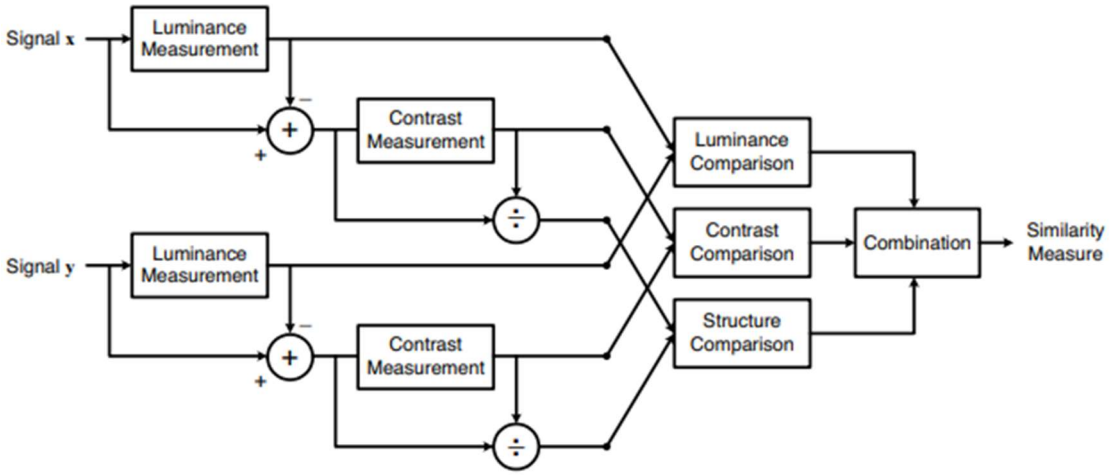


Figura 9: Diagramma del sistema per misurare SSIM [3]

Innanzi tutto, si confronta la luminanza, stimata come intensità media:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

(formula 1)

Per confrontare le due luminanze si utilizza la funzione

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

La costante C1 dipende dal range dinamico del valore dei pixel (L) e dalla costante K1<<1

$$C_1 = (LK_1)^2$$

Il contrasto invece viene stimato come deviazione standard

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_i)^2}$$

(formula 2)

Il confronto del contrasto avviene con la funzione

$$c(x,y) = \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Come per la costante C1, C2 dipende dal range dinamico del valore dei pixel (L) e dalla costante K2<<1

$$C_2 = (LK_2)^2$$

Infine

$$SSIM(x,y) = l(x,y)c(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

2.2.4 FSIM

Feature Based Similitary Index [4] è misura che si ricava in due fasi:

- Attraverso le CNN vengono cercate nelle immagini la congruenza di fase (PC) e il magnitude gradient (GM)
- Calcolo della somiglianza delle feature maps tra le due immagini, attraverso metodi più semplici come RMSE e PSNR oppure il pooling e il downsampling.

$$FSIM = \frac{\sum_{x\in\Omega} S_L(x) \cdot PC(x)}{\sum_{x\in\Omega} PC(x)}$$

S_L corrisponde al prodotto delle similarità per PC $S_{PC}(x)$ e per GM $S_G(x)$.

$$S_L = {S_{PC}}^{\alpha} {S_G}^{\beta}$$

α e β sono delle costanti per bilanciare i due parametri.

2.2.5 ISSM

L'Information Theoretic-based Statistic Similarity Measure [4] consiste nella ricerca delle similarità attraverso la teoria dell'entropia di Shannon e l'istogramma congiunto.
L'entropia corrisponde al valore atteso dell'informazione:

$$E_s(x) = -\sum_{i=1}^{n} p(x_i) \log_2[p(x_i)]$$

- $p(x_i)$ è la probabilità che il valore dell'informazione sia x_ie n è la cardinalità dei valori che può assumere

L'istogramma congiunto prende le caratteristiche locali dei pixel e costruisce un istogramma multidimensionale:

$$H(x,y) = H_{ij} = \frac{|\{x = i\} \cap \{y = j\}|}{M \times N}$$

- H_{ij} corrisponde alla probabilità che il pixel di intensità i dell'immagine x co-occorra con il pixel di intensità j della foto y, M e N sono le dimensioni (righe e colonne) delle due immagini.

Se si uniscono le due teorie si ottiene

$$EHS(x,y) = -\sum_{k=1}^{M\times N} T(k)\log_2 T(k)$$

- $T = H(:)$ rappresenta l'istogramma bidimensionale in un vettore unidimensionale.

A questi parametri si aggiunge anche il rilevamento dei bordi di Canny e l'indice SSIM:

$$C(x,y) = \left|\frac{\sum_i \sum_j (g_{ij} - g_0)(h_{ij} - h_0)}{\sqrt{\sum_i \sum_j (g_{ij} - g_0)^2 (h_{ij} - h_0)^2}}\right|$$

- h e g corrispondono alle immagini dopo l'applicazione del filtro per il rilevamento dei bordi su x e y.
- g_0 e h_0 corrispondono alle medie globali di g e di h.

Infine

$$ISSM(x,y) = \frac{0.7 \cdot C(x,y)EHS(x,y) + e}{0.3 \cdot C(x,y)EHS(x,y) + 0.5 \cdot EHS(x,y) + 0.7 \cdot SSIM(x,y) + e}$$

- Le costanti servono per il bilanciamento dei parametri.

2.2.6 SRE

Il Signal to Reconstruction error ratio [5] è un parametro che indica l'errore relativo alla potenza del segnale.

$$SRE = 10\log_{10}\frac{n{\mu_x}^2}{|x-y|^2}$$

- μ_x corrispondono alla media del segnale x.
- x e y sono i due segnali a confronto.

2.2.7 UIQ

L'Universal Image Quality Index [6] è forse la metrica più obsoleta in quanto ideata dall'IEEE nel 2002 poi sostituita dal SSIM (2.3).

Il calcolo dell'UIQ utilizza la luminanza μ_x (con la *formula 1*) e contrasto σ_x (con la *formula 2*):

$$UIQ = \frac{4\sigma_{xy}\mu_x\mu_y}{(\sigma_x^2 + \sigma_y^2)(\mu_x^2 + \mu_y^2)}$$

2.2.8 AlexNet

AlexNet [7][8] è una CNN ideata da Alex Krizhevsky nel 2012, nota per aver vinto il concorso ImageNet [7] e successivamente utilizzata in moltissime applicazioni di classificazione delle immagini.

Questa rete è composta da 5 strati convoluzionali, e 3 strati fully connected per fare la classificazione nei layers finali. Il sistema è oltretutto molto efficiente e performante in quanto, durante l'apprendimento, utilizza le risorse della GPU e la tecnica del dropout per evitare l'overfitting, in pratica vengono spenti casualmente dei neuroni della rete per generalizzare maggiormente l'apprendimento.

Questa rete è stata creata per riconosce e classificare diversi oggetti nelle foto ma, in questo contesto, è stata adattata per misurare la similarità tra due immagini.

In questa applicazione di AlexNet per ogni immagine vengono estratte delle features map da diversi layers della rete.

Per ogni feature map viene calcolata la distanza euclidea con la rispettiva dell'immagine a confronto. Le distanze trovate vengono messe in scala con il vettore w, e viene calcolata la media (*Figura 10*).

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot (y^l_{hw} - y^l_{0hw}) \right\|_2^2$$

- L è il numero di layers nella rete F.
- $H_l \; e \; W_l$ sono rispettivamente l'altezza e la larghezza delle feature map nello strato l.
- $y^l_{hw} - y^l_{0hw}$ sono le feature map normalizzate delle patch $x \; e \; x_0$ nello strato l.
- w_l è un vettore che scala le attivazioni canale per canale nello strato l.

infine, una piccola rete neurale G prende in input le distanze e determina l'indice di similarità.

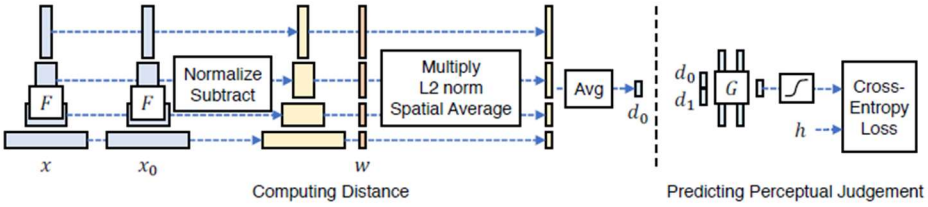


Figura 10: adattamento delle AlexNet per determinare la similarità [8]

2.2.9 VGG

Visual Geometry Group [9] è una CNN molto profonda che può avere 16 o 19 layers e che quindi, per l'apprendimento, va ad utilizzare più di 138 milioni di parametri.

La rete è composta da delle serie di layers convoluzionali susseguiti da un layer di pooling.

Nel modello a 16 layers la rete può applicare all'immagine, in dimensioni 224 x 224, 64 filtri convoluzionali fino ad arrivare a 4096 nel layer finale (*Figura 11*).

La particolarità di queste reti profonde è che, con tutti i filtri applicabili, il sistema raggiunge una vasta percezione delle immagini.

Le features vengono scelte automaticamente permettendo l'estrazione di pattern complessi e astrazioni da dati grezzi. Queste reti sono molto precise e accurate, tuttavia, essendo di taglie enormi, hanno bisogno di molto spazio, di molte risorse per funzionare e di molto tempo per l'apprendimento.

Anche in questo caso per calcolare la similarità è stato utilizzato lo stesso metodo usato per l'AlexNet

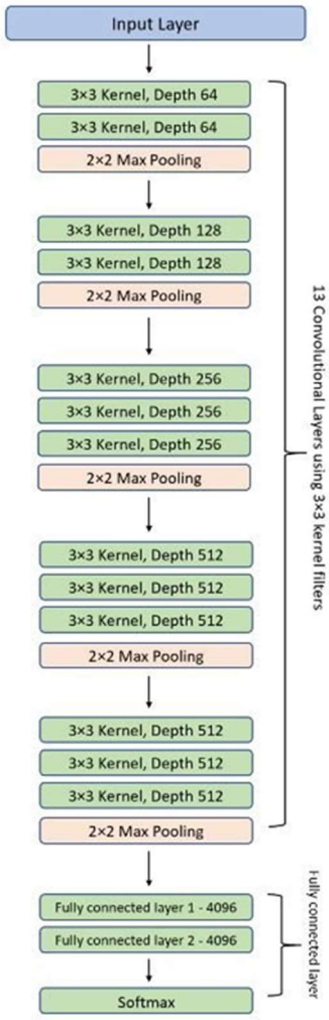


Figura 11: struttura di una VGG 16 [21]

2.3 Implementazione

Tutte le misure elencate nel capitolo 2.2 sono disponibili nelle librerie di Python image-similarity-measures [17] e lpips [18]. (Codice 2)

Capitolo 3

Analisi delle misure con il PCA

Dopo aver calcolato tutti gli indici di similarità per le coppie di immagini del dataset fornito, è emerso che i valori di ogni misura, presi singolarmente, non evidenziavano nessuna distinzione simile a quella trovata nella fase di labelling.

Successivamente sono state provate combinazioni lineari di massimo tre misure ma i grafici non hanno mostrato distinzioni soddisfacenti tra i dati (*Figure 12 e 13*).

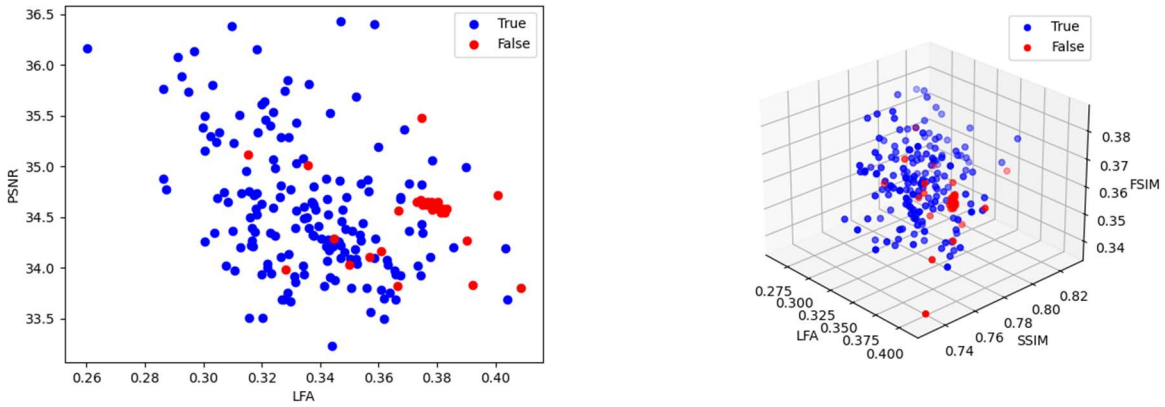


Figure 12 e 13: facendo combinazioni lineari tra le misure di similarità non si riesce a fare una distinzione netta tra dati "buoni" (true) e "da scartare" (false)

Creare un modello di classificazione a più di tre dimensioni è possibile ma è anche opportuno eliminare le misure superflue in quanto richiedono molto tempo e risorse per essere calcolate.

A tale scopo, è possibile adattare il Principal Components Analysis a questo contesto.

3.1 PCA

Il Principal Component Analysis [10] (PCA) è una tecnica che si utilizza per ridurre le dimensioni di un dataset riducendone la complessità.

Si suddivide in 4 passaggi:

- **Standardizzazione**, consiste nel mettere tutti i parametri nella stessa scala in modo che tutti contribuiscano alla stessa maniera. La caratteristica principale che serve al PCA è la variazione dei parametri all'interno del dataset.

- Trovare la **matrice di covarianza** serve per vedere se c'è qualche correlazione tra i parametri ed evidenziare informazioni che sono ridondanti.

$$\sigma_{xy}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \mu_x)(Y_i - \mu_y)$$

  - X e Y sono i due dati su cui calcolare la covarianza
  - μ_x e μ_y sono le medie dei dati X e Y

  Se le dimensioni fossero 3 (X, Y, Z) la matrice di covarianza sarebbe:

$$C = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{xz}^2 \\ \sigma_{yx}^2 & \sigma_y^2 & \sigma_{yz}^2 \\ \sigma_{zx}^2 & \sigma_{zy}^2 & \sigma_z^2 \end{bmatrix}$$

  Se la covarianza è positiva le due variabili sono direttamente correlate mentre, se negativa, sono inversamente correlate.

- Trovare le **componenti principali** (PC), ossia, vettori costituiti da combinazioni lineari dei parametri che rappresentano la direzione dei dati con la massima varianza cercando di ridurre il più possibile correlazioni tra dimensioni.

  Matematicamente le componenti principali sono gli autovettori della matrice di covarianza mentre gli autovalori servono per ricavare la varianza che ogni componente principale apporta ai dati.

$$Cv = \lambda v$$

  - v autovettore che rappresenta una PC
  - λ autovalore dell'autovettore v

  Infine, gli autovettori vengono ordinati secondo i loro autovalori in ordine decrescente.

- Ricavare il **feature vector**, quindi, scegliere se eliminare le componenti meno importati e comporre una matrice dove le colonne sono le componenti principali selezionate. Questo passaggio va sicuramente a perdere delle informazioni ma semplifica la lettura del dataset.
- Riprodurre i dati sullo spazio delle componenti principali dove il dataset standardizzato vene moltiplicato per il feature vector.

$$newDataSet = FeatureVector^T * StandardizedDataSet^T$$

3.2 Utilizzo del PCA

L'obiettivo del PCA in questa applicazione non è solo ridurre le dimensioni dei dati ma anche quello di andare ad eliminare le misure meno importati. L'idea è stata quella di creare un dataset composto

da vettori, che contengono tutte le 9 misure di similarità (un vettore per ogni coppia di immagini), applicare il PCA ai vettori dopo averli normalizzati, ricavare la varianza spiegata di ogni PC e moltiplicarla per tutti i 9 coefficienti della componente rispettiva.

Infine, sommando le componenti ponderate in base alla varianza si ottiene un vettore che riporta i pesi w di ogni dimensione.

In questo modo i coefficienti delle combinazioni lineari delle componenti con più varianza avranno un peso maggiore rispetto a quelli nelle componenti di varianza minore. Infine, sono state sommate tra loro le componenti principali ed è stata stilata una classifica dei pesi maggiori.

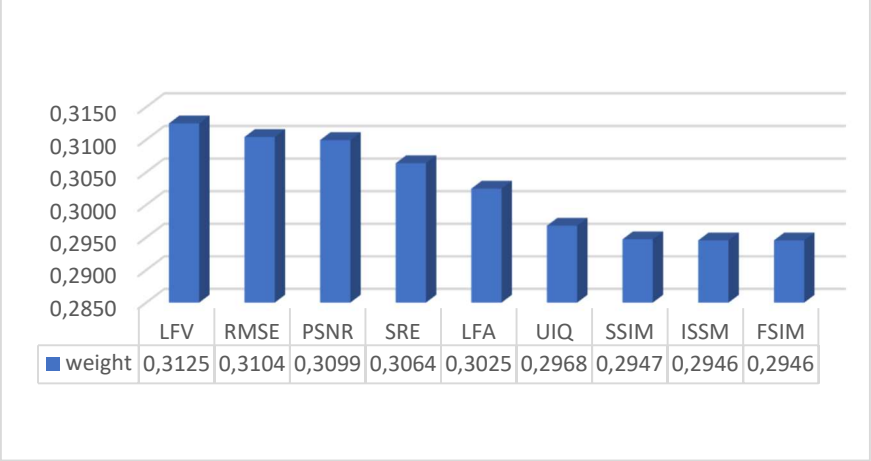
$$\begin{pmatrix} C_{1,1} & \cdots & C_{n,1} \\ \vdots & \ddots & \vdots \\ C_{1,d} & \cdots & C_{n,d} \end{pmatrix} \begin{pmatrix} {\sigma_1}^2 \\ \vdots \\ {\sigma_n}^2 \end{pmatrix} = \begin{pmatrix} w_1 \\ \vdots \\ w_d \end{pmatrix}$$

$$w_d = \sum_{c=1}^{n} {\sigma^2}_c \, C_{c,d}$$

- w_d peso della dimensione d
- n numero delle PC
- ${\sigma^2}_c$ varianza della componente c
- $C_{c,d}$ peso d della componente c

3.3 Risultati

Dopo aver mandato in esecuzione il codice le misure più rilevanti sono LFV, RMSE, PSNR e SRE:



Come si può vedere dal grafico, di tutte delle misure trovate con le CNN, solo LFV è considerata rilevante. Probabilmente ci saranno state delle correlazioni tra queste (LFV, LFA e FSIM). Stessa cosa può valere per UIQ, SSIM e ISSM che, basandosi tutte sulla luminanza e sul contrasto, sono state ritenute poco rilevanti.

Per le classificazioni sono stati creati dei vettori contenenti LFV, RMSE, PSNR e SRE per ogni coppia di immagini.

Capitolo 4

Classificazione con SVM

Per testare una classificazione a più di tre dimensioni attraverso un algoritmo di apprendimento supervisionato è stato usato il Support Vector Machine (SVM) in quanto efficace in spazi ad alta dimensione ed adatto per classificazioni non lineari.

4.1 SVM

Il Support Vector Machine [11] è un algoritmo di Machine Learning che serve per creare un modello di classificazione a una o più dimensioni di un dataset. Essendo un algoritmo di apprendimento supervisionato oltre all'insieme dei training data (X) ha bisogno anche dell'insieme delle etichette di uscita (Y) per l'addestramento.

Grazie all'apprendimento dei training data l'algoritmo va a cercare un iperpiano per separare il più possibile i dati nelle rispettive categorie.

Il caso più semplice è quando i dati sono linearmente separabili nello spazio originale del dataset.

Nel caso in cui le informazioni non sono linearmente separabili viene applicata ai dati una funzione Kernel che va a rappresentare il training set in uno spazio dove le classi possono essere separabili linearmente da un iperpiano (*Figura 14*).

Per l'SVM vengono comunemente utilizzate quattro kernel functions:

- Gaussiana (RBF): $K(x_1, x_2) = e^{-\frac{\| x_1 - x_2 \|^2}{\sigma}}$, σ equivale alla larghezza del Kernel
- Lineare: Lineare: $K(x_1, x_2) = x_1^T x_2$
- Polinomiale: $K(x_1, x_2) = (x_1^T x_2 + 1)^{\rho}$, ρ è l'ordine del polinomio
- Sigmoidale: $K(x_1, x_2) = \tanh(\beta_0 x_1^T x_2 + \beta_1)$

  x_1 e x_2 sono due dati del training set

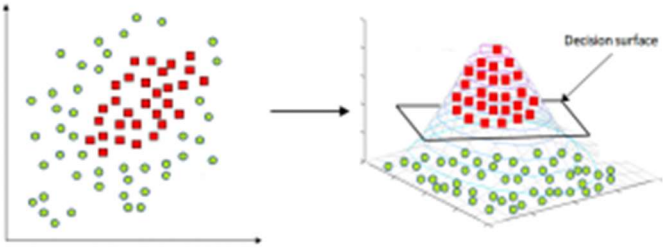


Figura 14: cambio dello spazio di rappresentazione grazie al kernel che aggiunge una dimensione [22]

Dopo aver trovato lo spazio adatto di rappresentazione dei dati vengono individuati i vettori di supporto (SV) (*Figura 15*), i punti dati nello spazio che si trovano più vicini all'iperpiano di decisione e rappresentano i casi più difficili da classificare.

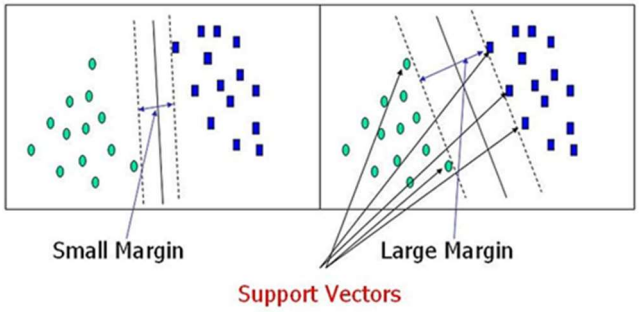


Figura 15: Margini e vettori di supporto [22]

Quando si tratta di trovare l'iperpiano ottimale che separa le classi si considerano tutti i possibili piani di separazione. Tuttavia, la scelta del piano di separazione ottimale viene fatta massimizzando la distanza dai vettori di supporto.

4.2 Implementazione

Scegliere il modello di SVM più opportuno non è semplice in quanto la configurazione ottimale dipende molto dalla struttura del dataset.

Il Set è stato suddiviso in training set (60%), test set (20%) e validation set (20%) che serve per identificare i parametri più appropriati.

Per la fase di validation è stato utilizzato l'algoritmo GridsearchCV() che richiede un dizionario di parametri da testare e che restituisce il modello ottimo per massimizzare diversi scores.

Quelli più rilevanti per una classificazione binaria con un dataset sbilanciato sono:

- Accuracy: indice di quante volte il modello ha azzeccato la decisione sul totale

$$accuracy = \frac{predictions\ corrette}{\#Validation\ set}$$

- Precision: considera i falsi positivi

$$precision = \frac{veri\ positivi}{Veri\ positivi + falsi\ positivi}$$

- Recall: considera i falsi negativi

$$recall = \frac{veri\ positivi}{veri\ positivi + falsi\ negativi}$$

- F1-score: considera sia Precision che Recall.

$$f1score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

I falsi positivi sono quei dati che sono stati labellati come non idonei ma vengono classificati dal modello come idonei, viceversa i falsi negativi sono i dati labellati come idonei e classificati dal modello come non idonei.

I veri positivi e i veri negativi sono i dati labellati rispettivamente come idonei e non idonei e la prediction del modello è stata corretta.

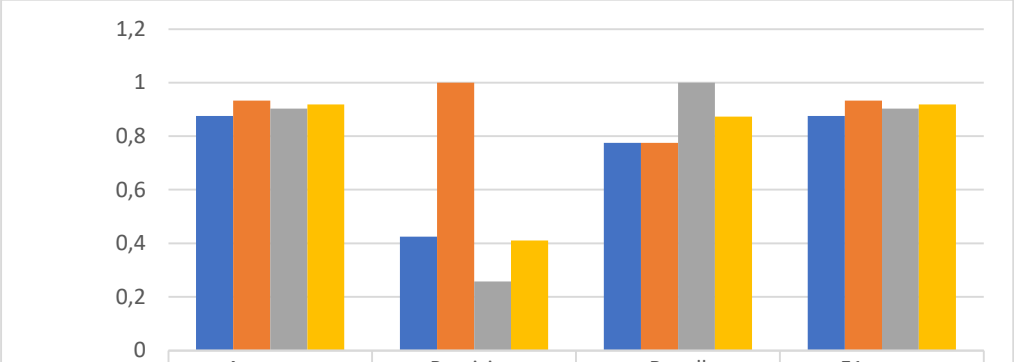
La cosa più importante in questa situazione è quella di andare a minimizzare i falsi positivi in quanto l'obbiettivo principale è quello di andare a togliere i dati che forniscono errori al sistema di riconoscimento self-supervised. Tuttavia, creare un filtro troppo selettivo causa la perdita di molti campioni buoni.

Il GridsearchCV() per fare la fase la validation utilizza una tecnica chiamata k-fold cross validation. Cioè, il validation set viene suddiviso in k subsets, su uno di questi viene eseguito il test mentre gli altri vengono usati per l'apprendimento.

Prima è stata fatta la validation per trovare il modello migliore sul validation set. Poi l'SVM è stato addestrato con il training set e, infine, testato con il test set. (*Codice 3*) [19]

4.3 Risultati

Per valutare le prestazioni del modello Accuracy, Precision, Recall e F1-score sono state calcolate anche per la fase di test.



	Accuracy	Precision	Recall	F1-score
Accuracy	0,875	0,425	0,775	0,875
Pecision	0,933	1	0,775	0,933
Recall	0,903	0,258	1	0,903
F1-score	0,918	0,41	0,873	0,918

```
ottimizzazione di accuracy:
{C: 1, class_weight: balanced, coef0: 0.0, degree: 1, gamma: 10, kernel: rbf}

ottimizzazione di precision:
{C: 0.001, class_weight: balanced, coef0: 0.0, degree: 4, gamma: 1, kernel: poly}

ottimizzazione di recall:
{C: 0.001, class_weight: None, coef0: 0.0, degree: 1, gamma: 0.001, kernel: linear}

ottimizzazione di f1:
{C: 1, class_weight: balanced, coef0: 0.0, degree: 1, gamma: 10, kernel: rbf}
```

Da questi risultati si può ben vedere che la configurazione migliore è quella che massimizza sia accuracy che f1-score. Negli altri casi, invece, si nota che il modello è affetto da overfitting in quanto solo lo score da massimizzare è buono, a discapito degli alti parametri.
Il modello definitivo ha una kernel function RBF con gamma 10.

Capitolo 5

Classificazione con Clustering

Per classificare i dati esistono anche degli algoritmi di apprendimento non supervisionato dove non è necessario fornire il set delle etichette in quanto l'algoritmo distingue autonomamente le classi in base a dei criteri di somiglianza dei dati. Un approccio del genere risulta molto comodo in quanto non richiede il labelling completo dei dati.

5.1 Clustering

Il clustering consiste nella ricerca delle strutture o dei gruppi di dati all'interno di un dataset e le classi che trova vengono definite clusters.
Per implementare questa pratica ci sono diversi algoritmi:

5.2.1 K-means

Il K-means [12] è uno degli algoritmi di clustering più diffusi e performanti. Per implementarlo è necessario conoscere il numero K di clusters che si vogliono trovare.
Questa tecnica si basa sulla ricerca dei centroidi, ossia, un punto di un cluster che media tra tutti i punti associati ad esso.
L'algoritmo esegue tali passi:

1. Scelta casuale dei centroidi, meglio se distanti tra loro
2. Calcolo della distanza euclidea tra dati e centroidi
3. Ogni dato viene abbinato al centroide più vicino
4. Ricalcolo dei centroidi facendo la media dei punti del cluster

   Si iterano i punti 2,3,4 finché sul punto 4 i centroidi trovati sono identici a quelli del ciclo precedente.

5.2.2 Agglomerative Clustering

L'Agglomerative Clustering [13] è un algoritmo di clustering gerarchico, cioè, i dati vengono raggruppati in una struttura ad albero (*Figura 16*), i clusters più piccoli sono figli dei clusters più grandi e la radice corrisponde a un mega cluster contenente tutto il dataset.

1. Inizialmente ogni punto viene considerato cluster separato.
2. Si calcolano le distanze tra i clusters.
3. Si trovano i due clusters più vicini e si uniscono.

Vengono ripetuti i punti 2 e 3 finché non si ottiene un cluster che contenga tutti i dati.

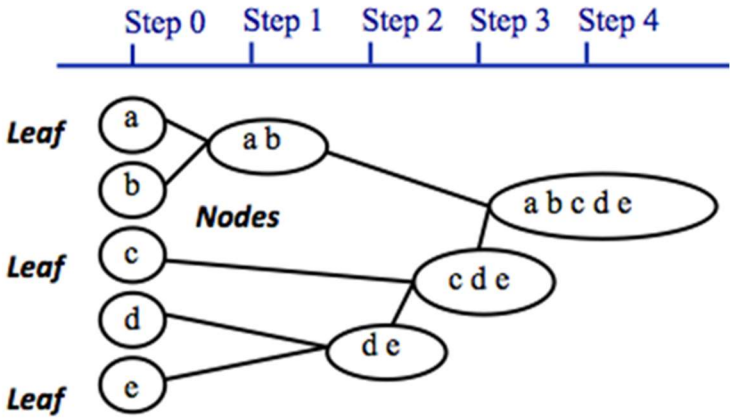


Figura 16: albero generato dall'Agglomerative Clustering [23]

5.2.3 Spectral Clustering

Lo Spectral Clustering [14] è una tecnica di clustering più complessa delle prime due. Si basa sulla trasformazione spettrale dei dati che poi vengono suddivisi in clusters.

Inizialmente il dataset viene rappresentato come un grafo pesato $G(V, E)$, dove i dati corrispondono ai vertici (V) e gli archi pesati (E) vengono creati in base alle misure di similarità calcolate con una kernel function.

$$W(A, B) = \sum_{i \in A, j \in B} w_{i,j}$$

- A e B sono due sottoinsiemi di vertici $A, B \subset V$
- $w_{i,j}$ corrisponde al peso dell'arco tra i e j

Si crea così la matrice di similarità W, che contiene tutte le misure di somiglianza (pesi degli archi) tra i dati.

Viene poi calcolata la matrice Laplaciana e normalizzata random walk

$$L = D - W$$

- D corrisponde alla matrice diagonale contenente la somma di tutti i pesi degli archi:

$$D_{ii} = \sum_{j} w_{i,j}$$

$$L^{rw} = D^{-1}L = I - D^{-1}W$$

- L^{rw} Laplaciana normalizzata random walk
- I matrice identità

Si trovano gli autovettori e autovalori di L^{rw}, si selezionano i k autovettori con autovalore più alto (k corrisponde al numero di clusters) e si rappresentano i dati sullo spazio costruito dai vettori selezionati.

Infine, si fa il clustering con le dimensioni spettrali attraverso algoritmi tradizionali come il k-means (5.2.1).

5.2.4 Birch

Come l'Agglomerative Clustering il Birch [15] è un Clustering gerarchico che cerca di raggruppare i clusters in un CF-Tree. Questo sistema sintetizza il più possibile le informazioni ed è progettato per fare classificazioni in grandi dataset.

I nodi del CF-Tree contengono delle statistiche relative al sottoinsieme di dati che rappresentano, come il numero di dati, la somma delle distanze etc.…

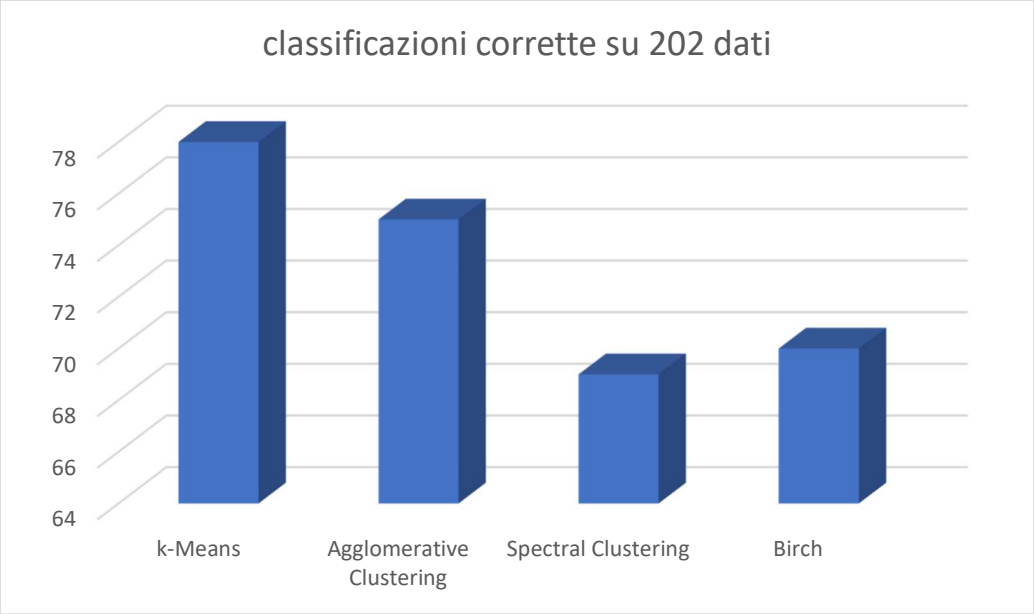
Se un nodo diventa troppo grande questo viene diviso in più sotto-clusters.

Quando l'albero risulta interamente costruito vengono raggruppati i clusters in base alle statistiche dei nodi e alle misure di similarità tra sotto-clusters.

Tutti gli algoritmi di clustering citati sono presenti nella libreria sklearn [19]. (*Codice 4*)

5.2 Risultati

Testando tutti gli algoritmi di clustering su tutto il dataset fornito i risultati sono stati questi:
per ogni modello il primo insieme rappresenta i dati idonei mentre il secondo quelli non idonei.
La cifra (1 o 0) rappresenta il cluster in cui sono stati collocati.

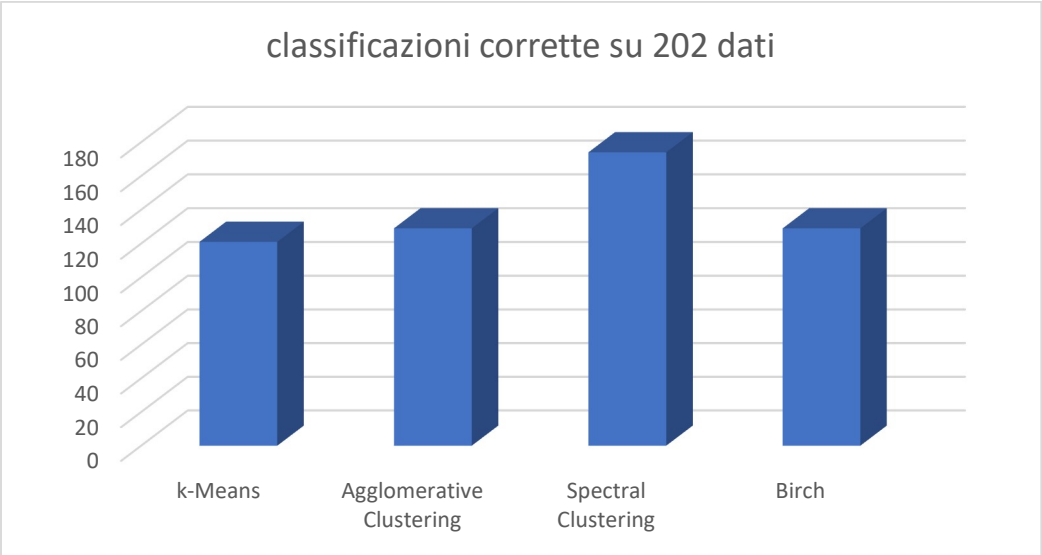


Tutti gli algoritmi di clustering non sono riusciti a trovare strutture tra i dati idonei e non idonei.

Probabilmente il dataset, avendo troppa varianza, dovuta alle misure di similarità evidenziate dall'utilizzo del PCA, fa sì che i dati siano troppo in disordine per poter fare un raggruppamento sensato.

Testando altre metriche sono stati riscontrati risultati più convincenti:

con vettori composti da LFV, LFA, UIQ e RMSE



In questo caso lo Spectral Clustering risulta essere in grado di distinguere le due classi con un accuracy dell'86,14%.

Probabilmente andando a rappresentare i dati nello spazio spettrale dei gruppi di dati true si sono allontanati dal gruppo di dati false.

Tra i falsi ci sono alcuni dati molto simili in quanto sono ottenuti dai campioni presi con il nastro fermo e quindi quasi tutti uguali. Questo gruppo è stato individuato da tutti gli algoritmi di clustering testati.

Capitolo 6

Test dei modelli con un nuovo dataset

Per verificare che i modelli di SVM e Clustering ottenuti siano performanti è necessario testare gli algoritmi su una porzione di dati più estesa di altre 200 copie di immagini.

In questa fase non è stato fatto il labelling dei dati in quanto risulta un'operazione particolarmente precisa e lunga.

I risultati sono stati analizzati in maniera più superficiale, guardando le proporzioni con cui è stato suddiviso il test set e scegliendo casualmente delle foto dalle due classi con lo scopo di verificare la presenza di falsi positivi e falsi negativi

6.1 Test del SVM

La prediction del modello di SVM ha etichettato come "true" tutti i dati, cioè ogni coppia di foto del nuovo dataset è stata ritenuta idonea. Naturalmente il modello classifica in maniera errata etichettando dati non idonei come idonei.

Questo problema può essere attribuito all'overfitting. Cioè, il modello si adatta troppo bene ai training data includendo rumore e variazioni casuali, in questo modo il modello ottiene ottime performance sui dati di addestramento ma esegue previsioni errate e troppo generalizzate sui dati nuovi. D'altronde, il dataset fornito conteneva 166 dati true e 36 false (17.8%), l'addestramento è stato fatto con il 60% dei dati (121), scelti casualmente, di cui 18 dati falsi previsti. Con un set così sbilanciato il modello non ha appreso bene i dati da etichettare come false.

Gli ottimi score ottenuti nel primo test sono dovuti appunto al fatto che ci sono pochi dati false quindi, l'accuratezza (0.875) e l'F1-score (0.918) sono buoni in quanto tutti i dati True sono stati classificati correttamente.

6.2 Test del Clustering

Il clustering ha etichettato 99/200 dati true e 101/200 dati false. Da queste informazioni si può dedurre che lo scarto di campioni, secondo questo modello, è circa del 50 %.

Andando poi a vedere le foto delle due classi ottenute si nota la presenza di parecchi errori di classificazione, sia di falsi positivi (*Figure 17 e18*) che di parecchi falsi negativi (*Figure 19 e 20*). Di conseguenza risulta essere poco performante anche questo algoritmo.

Questa situazione indica che non ci sono strutture di dati che rendono separabili i dati "idonei" da "non idonei" nel dataset.

L'unica cosa buona del clustering è la capacità di individuare i gruppi di dati quasi identici relativi alle immagini campionate quando il nastro restava fermo e le telecamere continuavano a fare scatti.

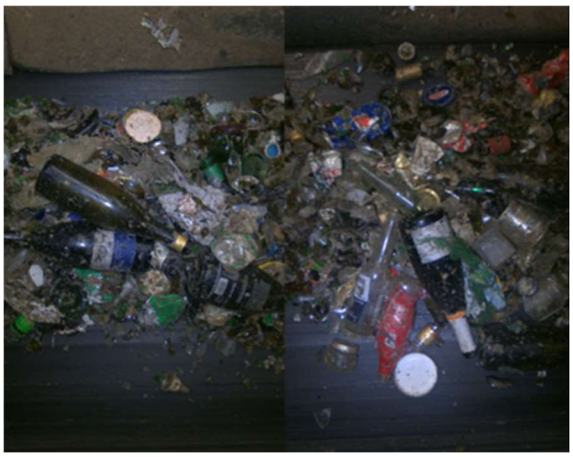



Figure 17 e 18: esempi di falsi positivi


Figure 19 e 20: esempi di falsi negativi

6.3 Conclusioni

Questo studio rispecchia un possibile approccio ad un problema di classificazione confrontando diversi modelli con diverse configurazioni.

Dato che le immagini in questione sono molto caotiche e presentano tutte particolari diversi non è stato possibile scegliere "manualmente" con quale criterio misurare la similarità ma è stato testato l'approccio con il PCA, fondamentale per eliminare la ridondanza delle misure in quanto correlate tra loro. Questo studio ha permesso quindi di capire tutte le difficoltà legate alla scelta dei parametri per la classificazione visto che questi dipendono molto dal training set fornito.

Sarebbe stato opportuno raccogliere un training set più bilanciato con più coppie di foto da etichettare come non idonee e includendo tutte le casistiche presentate nel punto 1.1

La scelta di affrontare tutti i problemi legati alle coppie di foto con una classificazione basata solo su alcune misure di similarità non si è dimostrata vincente, in quanto tutti i problemi legati ai dati da scartare si possono notare con metodi differenti e di complessità diversa.

Il problema legato allo sfasamento delle immagini potrebbe essere risolto con un algoritmo di computer vision che individua degli oggetti e che misura di quanto sono sfasati nelle due foto.

Le immagini che presentano i guanti dell'operatore possono essere eliminate attraverso una rete convoluzionale addestrata per individuarli mentre i gruppi di dati identici possono essere etichettati dal clustering.

Una soluzione del genere si potrebbe implementare applicando gradualmente questi algoritmi, partendo da quello di complessità minore fino a quello di complessità maggiore scartando così passo dopo passo le immagini sbagliate.

Codice

Codice 1: labelling

```
for i in folders:
    print(i)
    imm = cv2.imread(i + r"\DisplayImage.png")
    imm = cv2.resize(imm, (1000, 800))
    plt.imshow(cv2.cvtColor(imm, cv2.COLOR_BGR2RGB))
    plt.show(block = False)
    pmp = input("dire se la foto e' idonea[i] o non idonea[n]")
    plt.close()
    if pmp.lower()=="i":
        val = True
    else:
        val = False
    ratings[i] = val

with open(os.path.dirname(os.path.abspath(__file__))+r"\label.json", 'w') as
json_file:
    json.dump(ratings, json_file, indent=4)
```

Codice 2: calcolo delle misure di similarità

```
loss_fn_alex = lpips.LPIPS(net='alex')
loss_fn_vgg = lpips.LPIPS(net='vgg')
t_img_x = torch.zeros(224,224)
t_img_y = torch.zeros(224,224)
for k, v in data.items():
    for j in os.listdir(k):
        if j.endswith("first.png"):
            img_x = pp.preprocess(k + "\\" + j)
            t_img_x = transforms.functional.to_tensor(img_x)
        elif j.endswith("second.png"):
            img_y = pp.preprocess(k + "\\" + j)
            t_img_y = transforms.functional.to_tensor(img_y)
    evaluations = {"rmse": float(qm.rmse(img_x,img_y)),
                   "psnr": float(qm.psnr(img_x,img_y)),
                   "ssim": float(qm.ssim(img_x,img_y)),
                   "fsim": float(qm.fsim(img_x,img_y)),
                   "issm": float(qm.issm(img_x,img_y)),
                   "sre": float(qm.sre(img_x,img_y)),
                   "uiq": float(qm.uiq(img_x,img_y)),
                   "lfa": loss_fn_alex(t_img_x, t_img_y).item(),
                   "lfv": loss_fn_vgg(t_img_x, t_img_y).item()
                   }
```

Codice 3: SVM

```
X_train, X_temp, Y_train, Y_temp = train_test_split(X, Y, test_size=0.4,
random_state=77)
X_test, X_val, Y_test, Y_val = train_test_split(X_temp, Y_temp, test_size=0.5,
random_state=77)

param = {
    'C':[0.001, 0.01, 0.1, 1],
    'kernel':['linear', 'poly', 'rbf', 'sigmoid'],
    'degree': [1,2,3,4,5],
    'gamma': [0.001, 0.01, 0.1, 1, 10, 20,'scale'],
    'coef0': [0.0, 1.0, 5.0],
    'class_weight': [None,'balanced'],}

clf = svm.SVC(probability=True)
grid_search = GridSearchCV(clf, param, cv=5, scoring = 'precision',
verbose=1,n_jobs=-1)
grid_search.fit(X_val, Y_val)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

new_clf = best_model.fit(X_train,Y_train)
pred = new_clf.predict(X_test)

accuracy = accuracy_score(Y_test, pred)
precision = precision_score(Y_test, pred)
recall = recall_score(Y_test, pred)
f1 = f1_score(Y_test, pred)
print(best_params)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Codice 4: Clustering

```
models=[]
models.append(cl.KMeans(n_clusters=2))
models.append(cl.AgglomerativeClustering(n_clusters=2, affinity='euclidean'))
models.append(cl.SpectralClustering(n_clusters=2, random_state=77,gamma=2))
models.append(cl.Birch(n_clusters=2,threshold=0.05, branching_factor=25))

ins=[]

for model in models:
    cluster_labels = model.fit_predict(X)
    trues = cluster_labels[:t]
    falses = cluster_labels[t:]
    if np.count_nonzero(falses==0) > np.count_nonzero(falses==1):
       score =  np.count_nonzero(trues==1) + np.count_nonzero(falses==0)
    else:
        score =  np.count_nonzero(trues==0) + np.count_nonzero(falses==1)
    print(model)
    print(trues)
    print(falses)
    print("score:", score, r'/', len(X),"  ", score*100/len(X),"%")
```

Bibliografia

[1] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[2] Zhu, Yonggui & Yang, Xiaolan. (2011). "A Dyadic Wavelet Filtering Method for 2-D Image Denoising". Journal of Signal and Information Processing. 02. 10.4236/jsip.2011.24044.

[3] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, 2004, 13, 600 - 612, doi: 10.1109/TIP.2003.819861.

[4] Mohammed Abdulameer Aljanabi, Zahir M. Hussain, Noor Abd Alrazak Shnain & Song Feng Lu (2019) Design of a hybrid measure for image similarity: a statistical, algebraic, and information-theoretic approach, European Journal of Remote Sensing, 52:sup4, 2-15, DOI: 10.1080/22797254.2019.1628617

[5] Charis Lanaras, José Bioucas-Dias, Silvano Galliani, Emmanuel Baltsavias, Konrad Schindler, Super-resolution of Sentinel-2 images: Learning a globally applicable deep neural network, ISPRS Journal of Photogrammetry and Remote Sensing, Volume 146, 2018, Pages 305-319, ISSN 0924-2716.

[6] Zhou Wang and A. C. Bovik, "A universal image quality index," in *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81-84, March 2002, DOI: 10.1109/97.995823.

[7] Krizhevsky, A.; Sutskever, I. & Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, in F. Pereira; C. J. C. Burges; L. Bottou & K. Q. Weinberger, ed., 'Advances in Neural Information Processing Systems 25', Curran Associates, Inc., pp. 1097—1105.

[8] R. Zhang, P. Isola, A. Efros, E. Shechtman and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 586-595. DOI: 10.1109/CVPR.2018.00068

[9] Karen Simonyan and Andrew Zisserman(2015), Very Deep Convolutional Networks for Large-Scale Image Recognition, International Conference on Learning Representations

[10] Flandoli, F. (2013-14). "Vettori gaussiani e PCA". "Dispense di Statistica II" (Cap. 2). Università di Pisa.

[11] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector ma-chines," in *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18-28, July-Aug. 1998, doi: 10.1109/5254.708428.

[12] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, Jian, China, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.

[13] Daniel Müllner, "Modern hierarchical, agglomerative clustering algorithms", Stanford University, Department of Mathematics, 2011, https://doi.org/10.48550/arXiv.1109.2378.

[14] von Luxburg, U. A tutorial on spectral clustering. *Stat Comput* **17**, 395–416 (2007). https://doi.org/10.1007/s11222-007-9033-z

[15] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," ACM Sigmod Record, ACM, 1996, 25, 103–114.

[16] Bellotto N. (2023), "IA2223_Deep_learning_1", "Image convolution examples", Università degli Studi di Padova.

[17] https://pypi.org/project/image-similarity-measures/

[18] https://pypi.org/project/lpips/

[19] https://scikit-learn.org/stable/

[20] https://it.mathworks.com/discovery/convolutional-neural-network-matlab.html#:~:text=Una%20rete%20neurale%20convoluzionale%20(CNN,riconoscere%20oggetti%2C%20classi%20e%20categorie.

[21] https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

[22] https://medium.com/@apurvjain37/support-vector-machine-s-v-m-classifiers-and-kernels-9e13176c9396

[23] https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

Ringraziamenti

Ringrazio il professor Emanuele Menegatti per la disponibilità e per avermi accolto come tesista nel suo laboratorio. La sua proposta di argomento di tesi ha rappresentato un'opportunità unica per approfondire le mie conoscenze di intelligenza artificiale.

Ringrazio il dr. Alberto Bacchin per avermi seguito nella realizzazione del progetto, dandomi puntualmente consigli in maniera approfondita. Il suo supporto ha arricchito la mia esperienza e ha stimolato ulteriormente il mio interesse per la materia.

Ringrazio la mia famiglia che ha sempre creduto in me e che mi ha sempre sostenuto.

Ringrazio tutti gli amici che mi sono stati vicino in questi 3 anni. La vostra presenza e il vostro sostegno mi hanno fatto vivere un'esperienza accademica più serena e motivante.