**Università Degli Studi di Salerno**

**Dipartimento di informatica**

*Corso di Laurea Magistrale Cloud Computing*



Simulazione delle Reti Neurali Convoluzionali per la Compressione Dati

**Docente Gruppo: Negan**

Prof. Bruno Carpentieri Giuseppe Capaldo 0522500498

Raffaele Donadio 0522500423

**Anno Accademico 2017/2018**

Indice

[1. Introduzione 3](#_Toc509309155)

[2. Un primo tentativo 5](#_Toc509309156)

[3. Simulazione 27](#_Toc509309157)

[4. Testing 31](#_Toc509309158)

[5. Risultati Significativi 46](#_Toc509309159)

[6. Uso dell'applicazione 48](#_Toc509309160)

[7. Conclusioni 50](#_Toc509309161)

# 1. Introduzione

I segnali analogici (musica, voce, tv e film) sono più difficili da memorizzare e da trasmettere su lunghe distanze a differenza dei digitali. Lo svantaggio principale che riguarda le versioni digitali dei segnali è il fatto che richiedono più bit. Per ridurre i costi di memorizzazione e trasmissione, si adoperano tecniche di compressione, il cui scopo è minimizzare il numero di bit necessari per ottenere una versione accettabile del segnale sorgente originale. Però tutte le tecniche adoperate negli ultimi tempi non hanno tenuto conto di cosa bisognerebbe comprimere; questo ha portato all’immissione di troppi errori all’interno dell’immagine che si traduce in una pessima qualità visiva.

Il documento che ci è stato consegnato e che abbiamo esaminato è il paper Semantic

Perceptual Image Compression Using Deep Convolution Networks di Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo and James Storer della BrandeisUniversity. Nel suddetto si parla dell’utilizzo delle Reti Neurali Convoluzionali (CNN) e di come i recenti progressi nella potenza di calcolo, insieme alla disponibilità di ampi set di dati per addestramento, hanno aumentato l'interesse nell'applicazione di CNN di deep learning per affrontare il riconoscimento dell'immagine e le attività di elaborazione delle immagini. Questo paper riporta anche il link di una sua prima implementazione già presente su github; in cui vengono usati:

1. Tensorflow
2. Numpy
3. Pandas
4. Python PIL
5. PythonSKimage

Inoltre vengono suggeriti:

1. Imagemagick (per operazioni di immagine più veloci)
2. VQMT (per ottenere metriche di confronto delle immagini)

Questo progetto è fondato su Python e sulle CNN, ma purtroppo nessuno del gruppo ha avuto la possibilità di studiare l’ambito dell'intelligenza artificiale, quindi non avendo le giuste competenze per capirlo fino in fondo è stato preso in considerazione come una black-box.

Il progetto in questione si prefigge l’obiettivo di rendere jpeg content-aware, ovvero pur applicando sempre lo stesso algoritmo di compressione e decompressione, si cerca di cambiare dinamicamente la qualità della compressione appunto usando la CNN.

Infatti alla fine del paper in questione si spiega che l’unico punto su cui si può migliorare ciò che loro stessi hanno fatto è la le performance della CNN migliorandone.

Dovendoci fermare sul lavoro precedente per cause di forza maggiore, siamo passati a prendere in considerazione il seguente lavoro.

Il progetto è stato realizzato dai nostri colleghi Aiello Michele, Bisogno Giuseppe, Manzo

Gerardo, Paolella Vincenzo nell’anno Accademico 2007/2008 in cui si descrive una compressione dell’immagine mediante la separazione in più livelli della stessa, facendo sì che si possano usare livelli di compressione o, addirittura, tipi di compressione diversi su di essi facendo sì che al riassemblamento del tutto non si perda in qualità ma allo stesso tempo diminuendo il peso.

I livelli che loro generano sono 3:

Background, Foreground e Maschera.

Il Background è il livello dei pixel che indica oggetti di fondo a cui l’occhio umano presterà meno attenzione.

Il Foreground è il livello dei pixel che indica gli oggetti in primo piano, quelli a cui l’occhio umano presterà più attenzione.

La Maschera è il livello che ci permette di estrarre i livelli sopra-citati dall’immagine originale.

# 2. Un primo tentativo

Inizialmente siamo partiti con l’utilizzare il progetto dei colleghi, per separare l’immagine nei

3 livelli che la compongono e quindi usiamo una compressione molto forte sul livello di Background e una compressione lieve sul Foreground così da ottenere, alla ricomposizione dell’immagine, un ottimo valore di compressione senza perderne in qualità. Queste sono state le nostre aspettative iniziali ma v*edendo che i primi risultati non rispecchiavano le aspettative abbiamo capito che i colleghi INVERTIRONO il Foreground dell’immagine con il Background. Facendo numerosi test si è capito che Il problema fondamentale del progetto ereditato è che tutta la struttura software si basa su file jar per la compressione, il quale non lavora come dovrebbe. Infatti comprimendo la stessa immagine in due sessioni nelle quali venivano assegnati diversi parametri di qualità, il PSNR risultava sempre uguale.*

A questo punto confrontandoci anche con il professore sulla natura fallimentare di tutti questi tentativi, abbiamo deciso di cambiare algoritmo di segmentazione passando dalla suddivisione in Foreground e Background ad algoritmi di segmentazione del tipo superpixel.

Inoltre abbiamo creato un progetto basato su linguaggio Python.

Quindi abbiamo creato un progetto la cui idea di base consiste nel prendere in input un’immagine e applicarvi un algoritmo di segmentazione, ottenendo un’immagine segmentata secondo la logica del singolo algoritmo di segmentazione.

Una volta ottenuto i frammenti si procede con la selezione dei frammenti più rilevanti visivamente parlando (**simulando la rete neurale**).

**Data la seguente immagine:**



I frammenti più rilevanti potrebbero essere i frammenti che contengono il viso dell’astronauta e la tuta spaziale.

Una volta identificati i frammenti più importanti si procede con la compressione dei singoli, usando una qualità di compressione maggiore per i frammenti importanti e minore altrimenti. Finita la fase di compressione si esegue un merge di tutti i frammenti per riottenere l’immagine completa.

**Il processo può essere rappresentato tramite il seguente modello:**



## 1. Estrazione delle maschere

L’estrazione delle maschere avviene mediante l’algoritmo di segmentazione. Per la segmentazione abbiamo usato la tecnica del superpixel precisamente ci siamo concentrati sulle varianti Felzenswalb & Huttenlocher e SLIC.

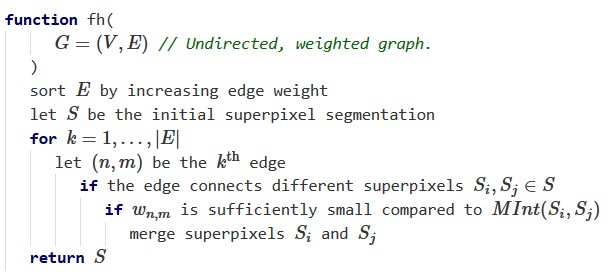
La variante Felzenswalb considera un'immagine 𝐼 ed un grafo 𝐺 = (𝑉, 𝐸) con nodi corrispondenti a pixel e i bordi tra pixel adiacenti come archi. Ad ogni bordo (𝑛, 𝑚) ⋲ 𝐸 viene assegnato un peso 𝑤𝑛,𝑚 = ‖𝐼(𝑛) − 𝐼(𝑚)‖2. Quindi, per i sottoinsiemi𝐴, 𝐵 ⊆ 𝑉definiamo

𝑀𝑆𝑇(𝐴)per essere il minimum spanning tree di ***A*** e

∫(𝐴) = 𝑚𝑎𝑥m,n∈𝑀𝑆𝑇(𝐴){𝑤𝑛,𝑚}

𝑀𝐼𝑛𝑡 (𝐴, 𝐵) = 𝑚𝑖𝑛.

Dove 𝑀𝐼𝑛𝑡 (𝐴, 𝐵) rappresenta la differenza interna minima tra i componenti *A* e *B*. Partendo da una prima segmentazione di superpixel in cui ogni pixel forma il proprio superpixel, l'algoritmo elabora tutti i bordi in ordine di aumento del peso del bordo. Ogni volta che un bordo collega due superpixel diversi, questi vengono uniti se il peso del bordo è piccolo rispetto alla differenza interna minima.

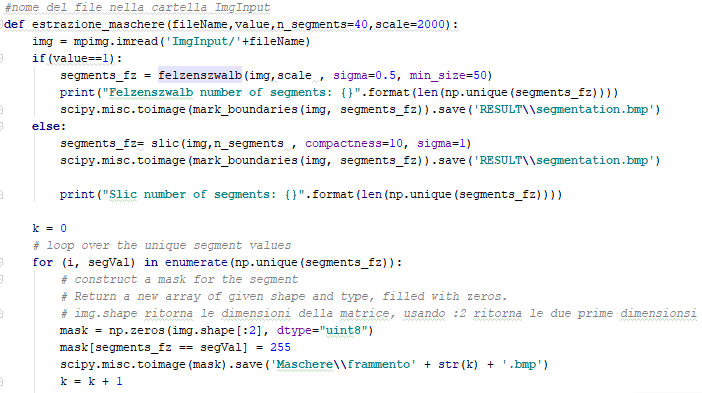


A differenza dell’approccio Felzenswalb, l'approccio SLIC quindi esegue un clustering locale di pixel nello spazio 5-D definito dai valori L, a, b dello spazio colore CIELAB e delle coordinate x, y dei pixel. Ha una diversa misurazione della distanza che consente compattezza e regolarità nelle forme dei superpixel e può essere utilizzata su immagini in scala di grigi e immagini a colori. SLIC genera superpixel raggruppandoli in base alla somiglianza e alla vicinanza dei colori nel piano dell'immagine. Per il clustering viene utilizzato uno spazio **[labxy]** a 5 dimensioni. Lo spazio colore CIELAB è considerato uniforme perpetuamente per piccole distanze di colore. Non è consigliabile utilizzare semplicemente la distanza euclidea nello spazio 5D e quindi gli autori hanno introdotto una nuova misura di distanza che considera la dimensione dei superpixel. Si inizia campionando K centri di cluster regolarmente distanziati e spostandoli in posizioni di seme corrispondenti alla posizione di gradiente più bassa in un quartiere 3 × 3. Questo è fatto per evitare di metterli a un bordo e ridurre le possibilità di scegliere un pixel rumoroso. I gradienti di immagine sono calcolati come

𝐺(𝑥, 𝑦) = ‖𝐼(𝑥 + 1, 𝑦) − 𝐼(𝑥 − 1, 𝑦)‖2 + ‖𝐼(𝑥, 𝑦 + 1) − 𝐼(𝑥, 𝑦 − 1)‖2

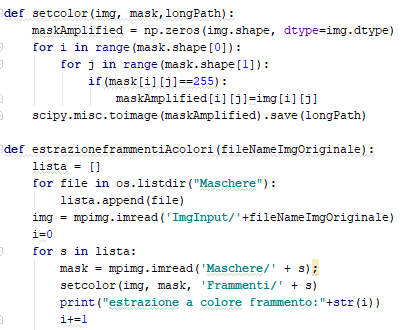
dove 𝐼(𝑥, 𝑦) è il vettore lab corrispondente al pixel in posizione (𝑥, 𝑦) e‖. ‖ è la norma L2. Questo tiene conto sia delle informazioni sul colore che sull'intensità. Ogni pixel dell'immagine è associato al centro del cluster più vicino la cui area di ricerca si sovrappone a questo pixel. Dopo che tutti i pixel sono associati al centro del cluster più vicino, un nuovo centro viene calcolato come vettore lab medio di tutti i pixel appartenenti al cluster. Al termine di questo processo, possono rimanere alcune etichette vaganti, ovvero pochi pixel in prossimità di un segmento più grande con la stessa etichetta ma non collegato ad esso. Mette in atto la connettività nell'ultimo passo dell'algoritmo rielaborando i segmenti disgiunti con le etichette del cluster più grande vicino.

**Ecco lo script python per l’estrazione delle maschere**

****

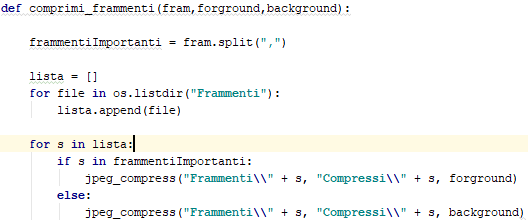
## 2. Estrazione dei frammenti

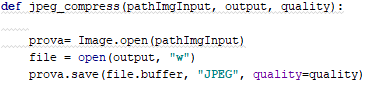
A partire dall’immagine di input e dalla maschera estraiamo i pezzi dell’immagine così da ottenere i frammenti di immagine a colore in formato bitmap in modo da poter scegliere quali comprimere di più e quali meno.



## 3. Compressione dei frammenti

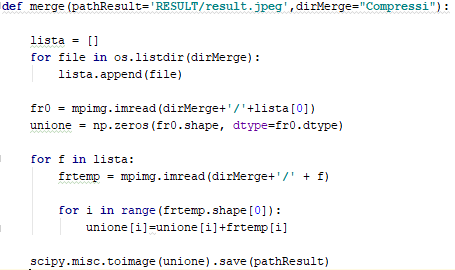
In questa fase si comprimono i frammenti utilizzando JPEG come algoritmo di compressione. Inoltre, scelti i frammenti importanti si usa un valore di qualità di compressione diverso dagli altri frammenti.





## 4. Unione dei frammenti

In questa fase ricombiniamo tutti i frammenti compressi in un’unica immagine

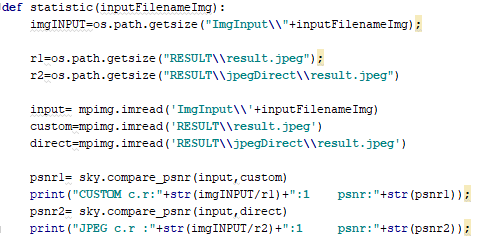


1. **Compressione diretta dell’immagine di input:**

Per ottenere un confronto con l’applicazione del semplice JPEG senza tutti i passaggi sopra citati chiamiamo la funzione jpeg\_compress con i dovuti parametri.

1. **Calcolo dei Compression Ratio e dei PSNR:**

Inoltre per ottenere dei dati di confronto abbiamo creato una funzione che estrae i parametri di confronto di qualità e compressione, ovvero CompressionRatio e PSNR.



**RISULTATI**

|  |
| --- |
|  |
| **Image Segmentation**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Most Important Segment Seleceted**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  |   **TEST1-2**   |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **SLIC** | |  | | --- | | **Quality Compression** | | 76 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 18:1 | 33.2 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 18:1 | 31.1 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Felzenswalb** | |  | | --- | | **Quality Compression** | | 76 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 18:1 | 33.2 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 18:1 | 27.1 | | |

|  |
| --- |
|  |
| **Image Segmentation**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Most Important Segment Seleceted**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  |     **TEST 3-4**     |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **SLIC** | |  | | --- | | **Quality Compression** | | 76 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 38.8 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 38.7 | | | **Originale** | **JPEG** | **Finale nostro** |  |  |  |  | | --- | --- | --- | |  |  |  | | **Felzenswalb** | |  | | --- | | **Quality Compression** | | 76 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 38.8 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 38.6 | | |

|  |
| --- |
|  |
| **Image Segmentation**     |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Most Important Segment Seleceted**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  |     **TEST 5-6**     |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **SLIC** | |  | | --- | | **Quality Compression** | | 94 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 1:1 | 60 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 1:1 | 19 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Felzenswalb** | |  | | --- | | **Quality Compression** | | 84 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 60 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 8:1 | 20 | | |

|  |
| --- |
|  |
| **Image Segmentation**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  |   **Most Important Images Seleceted**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | | **TEST 6-7** |  | | | **Originale** | **JPEG** | | | **Finale nostro** | |  |  | | |  | | **SLIC** | |  | | --- | | **Quality Compression** | | 82 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 23:1 | 33 | | | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 23:1 | 25 | | | **Originale** | **JPEG** | | | **Finale nostro** | |  |  | | |  | | **Felzenswalb** | |  | | --- | | **Quality Compression** | | 78 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 26:1 | 33 | | | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 26:1 | 26 | | |

|  |
| --- |
|  |
| **Image Segmentation**   |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Most Important Images Seleceted**     |  |  | | --- | --- | | **SLIC** | **Felzenswalb** | |  |  |   **TEST 8-9**   |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **SLIC** | |  | | --- | | **QualityCompression** | | 78 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 6:1 | 38 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 6:1 | 35 | | |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Felzenswalb** | |  | | --- | | **Quality Compression** | | 78 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 6:1 | 38 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 100 | 89 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 6:1 | 31 | | |

Dopo aver analizzato i risultati ottenuti è stata riscontrata la seguente anomalia.

*A parità di Compression Ratio JPEG puro ha un PSNR molto differente rispetto al PSNR misurato sull’immagine ottenuta dal processo di frammentazione e compressione delle ROI.*

Questi problemi sono dovuti alla cattiva applicazione di alcune funzioni dei pacchetti usati. Infatti ogni frammento estratto risulta essere un’immagine avente un background nero ed una piccola sezione di immagine originale ottenuta dal processo di segmentazione, quindi nel momento in cui si va a comprimere il singolo frammento in realtà viene processata oltre alla piccola sezione di immagine originale anche il restante dell’immagine inutilmente. Quindi il passo successivo sarebbe quello di prendere i frammenti ed eliminare completamente tutte le sezioni nere, cioè sostanzialmente eliminare il background. Ma questo processo risulta essere costoso in termini di computazione poiché questo task dovrebbe essere compiuto per ogni singolo frammento. Poiché l’obiettivo è simulare la rete neurale abbiamo optato per una strategia molto più semplice ossia eseguire una segmentazione matriciale costruita manualmente in modo da evitare frammenti con background nero.

Inoltre l’anomalia nei risultati non è dovuta solamente alle problematiche appena descritte ma anche al processo che effettua due compressioni per ottenere l’output del processo di frammentazione, ricombinazione e compressione.

Potrebbe sembrare una banalità o semplicemente un errore da parte nostra ma ciò ci ha illuminato su un problema serio e non evitabile.

**Cerchiamo di capire meglio il problema valutando tutto il processo:**

Data **A** un’immagine in input, sia **xi** l’i-esimo frammento ottenuto.

Il passo successivo è quello di comprimere tutti i frammenti e ottenere tante immagini in formato JPEG.

Dopo aver scompattato l’immagine e compresso i vari frammenti con jpeg ottenendo tanti file jpeg, non è possibile ottenere un unico file JPEG come output: l’unica soluzione sarebbe quella di modellare in memoria tutti i frammenti JPEG leggendo questi file mediante gli strumenti di Python i quali sostanzialmente consentono di modellare l’intera immagine come una matrice tridimensionale. In tal modo si ottengono delle matrici su cui è possibile effettuare il merge. A questo punto non sarebbe più possibile ottenere un salvataggio in jpeg altrimenti gli strumenti in uso comprimerebbero nuovamente l’immagine originale, generando quindi rumore dovuto alla doppia compressione applicata.

**Quindi la soluzione viene mostrata nel capitolo successivo**

# 3. Simulazione

Tutto il progetto è incentrato sulla compressione dinamica delle immagini e, come già è stato detto, non avendo le competenze di CNN, con l’approvazione del professore, abbiamo optato per un artefizio. Ovvero si segmenta l’immagine, successivamente vengono selezionati i frammenti più importanti e su quelli verrà effettuato una compressione minore.

Sui restanti frammenti dell’immagine si esegue una compressione molto più forte così da conservare il più possibile i dettagli importanti.

Questo secondo tentativo differisce dal primo perché:

* Viene effettuata una segmentazione matriciale, ovvero: data l’immagine si frammenta in tanti pezzi dove il pezzo generico x, y appartiene alla matrice di overlay creata sull’immagine.

Questo ci consente di ottenere esattamente il frammento senza il background nero, perché come già detto nel capitolo precedente, con gli strumenti utilizzati si ottengono dei frammenti con background nero che potrebbe influenzare, anche se non in maniera significativa, la compressione.

* Per evitare di comprimere l’immagine una seconda volta, dopo una prima compressione di tutti i frammenti in jpeg si procede nel seguente modo: Modellare in memoria tutti i frammenti ottenendo tante matrici quanti sono i frammenti, fare il merge di queste matrici e creare l’output in formato bmp da cui prendere il risultato visivo del processo di segmentazione e compressione e su questo calcolare il PSNR. Per quanto riguarda il calcolo del compression ratio, non avendo a disposizione un solo file jpeg, la dimensione del file di output si ottiene facendo la somma di tutti i frammenti in formato jpeg.

Dunque se si hanno a disposizione una serie di frammenti in formato jpeg, la dimensione complessiva è data dalla somma di tutte le dimensioni.

Però questo risultato non risulta essere veritiero poiché jpeg usa dei metadati per ogni file ed è chiaro che sommare le dimensioni di ogni frammento introduce errore, in quanto significa sommare tanti metadati di ogni frammento.

Per ottenere dei risultati migliori procediamo tenendo conto delle ridondanze, ovvero gli header di tutti i frammenti dei file jpeg, l’insieme dei frammenti jpeg ottenuti dalla segmentazione, vengono aggiunti ad un archivio ottenendo un unico file .tar.xz.

Il file .tar.xz deriva dall’applicazione tramite libreria prima del compressore tar e poi successivamente del compressore LZMA (Lempel-Ziv-Markov). Quest’ultimo è implementato anche come estensione .7z nel famoso programma LGPL (GNU Lesser General Public License) 7zip.

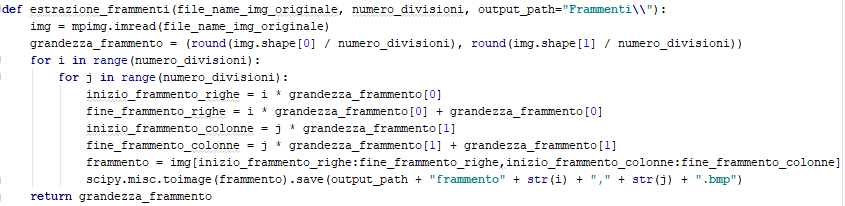
Così facendo abbiamo una dimensione quasi esatta dell’output del processo di segmentazione e compressione, poiché vengono eliminate tutte le ridondanze.

**Il processo può essere rappresentato tramite il seguente modello:**



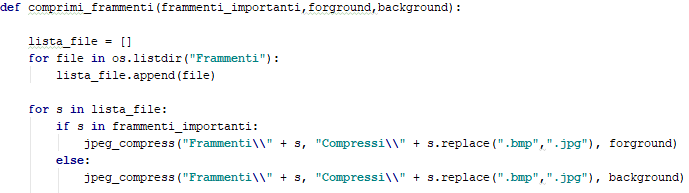
.**1. Estrazione dei frammenti**

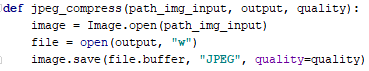
La prima operazione che viene effettuata a partire dall’immagine è la sua suddivisione in sottomatrici di pixel, tramite il seguente script python



## 2. Compressione dei frammenti

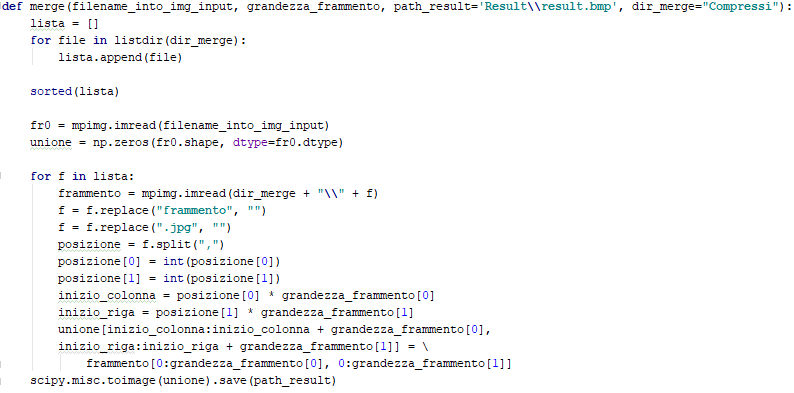
In questa fase si comprimono i frammenti utilizzando come algoritmo JPEG. Inoltre scelti i frammenti importanti, si mettono in evidenza con un valore di qualità diverso.





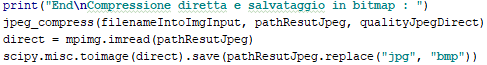
## 3. Unione dei frammenti

In questa fase si decomprimono e riuniscono i frammenti jpeg, successivamente si riuniscono in un unico file bitmap, rispettando l’ordine in cui erano state divise le sottomatrici.



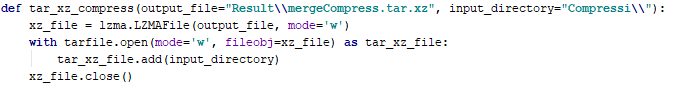
**4. Compressione diretta dell’immagine di input**

Per ottenere un confronto con l’applicazione del semplice JPEG senza tutti i passaggi sopracitati chiamiamo la funzione jpeg\_compress con i dovuti parametri.



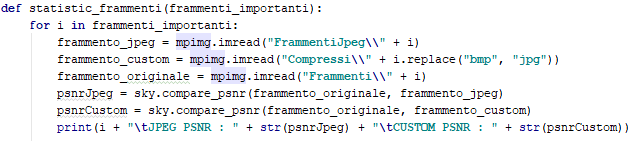
## 5. Compressione dei frammenti Jpeg in archivio tar.xz

Per migliorare il compression ratio della nostra suddivisione e togliere le ridondanze all’interno della cartella di risultato sui metadati di jpeg, utilizziamo due compressori lossless: tar e LMZA.

****

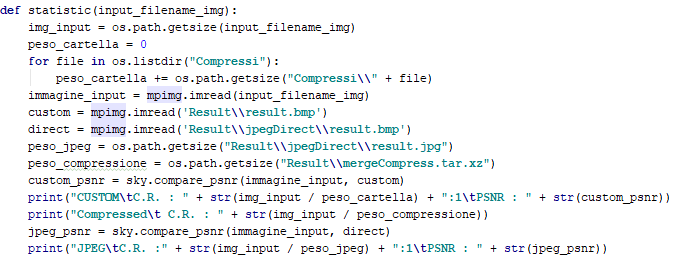
## 6. Statistiche dei frammenti

Per capire se il nostro procedimento migliora la qualità dei frammenti di interesse, li abbiamo confrontati con la controparte Jpeg.

****

## 7. Statistiche su tutto il file

Inoltre per ottenere dei dati di confronto è stata creata una funzione che ci estrae alcuni parametri di confronto qualità e compressione del risultato e della sua versione ricompressa.



# 4. Testing

**Per il test verrano usate due tabelle, la prima:**

|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | |  | |  | | --- | | **Quality Compression** | |  |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | |  |  | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | |  |  |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | |  |  | |  |  | | |

L’immagine in alto rappresenta l’input del testing, l’immagine verde rappresenta il risultato applicando direttamente Jpeg, l’immagine gialla è il risultato applicando l’algoritmo Custom.

Nella seconda parte della tabella sono presenti sia i paramentri della compressione che i risultati formali del testing.

Per l’algoritmo custom abbiamo due risultati per il Compression Ratio, come detto nei paragrafi precedenti.

**La seconda tabella:**

|  |  |  |
| --- | --- | --- |
| **frammento** | **Jpeg PSNR** | **Custom PSNR** |
| Frammento x | y | x |

In questa tabella vengono mostrati i risultati locali delle ROI, ovvero mostriamo che a parità di Compression Ratio e PSNR delle immagini, i frammenti più importanti hanno un PSNR maggiore applicando l’algoritmo Custom rispetto all’applicazione diretta di Jpeg.

|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 70 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 29.68:1 | 34.52 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 76 | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 22.35:1 | 34.53 | | 29.70:1 |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Examples Local Analysis**   |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 0,2 | 36.72 | 37.37 | | Frammento 1.0 | 33.65 | 34.18 | | Frammento 1.1 | 31.40 | 31.80 | | Frammento 1.2 | 31.83 | 32.28 |   Come possiamo vedere dai risultati il nostro algoritmo, scelte le zone di interesse, ottiene dei risultati molto vicini tra di loro.  Osservando i risultati delle due immagini complete ottenute dai due algoritmi, si può affermare che sono molto simili in termini di Compression Ratio e PSNR, ma facendo un’analisi locale ovvero sulle ROI si può osservare che il PSNR è migliore.  Inoltre, lo sforzo computazionale si attesta al massimo sull’ordine dei secondi. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | | | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 70 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 29.43:1 | 36.7 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 76 | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 30.57:1 | 36.30 | | 32.13:1 |  | | |
| **Examples Local Analysis**   |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 0,0 | 36.20 | 36.86 | | Frammento 0.2 | 43.50 | 44.50 | | Frammento 1.1 | 33.04 | 33.62 | | Frammento 1.3 | 37.84 | 38.54 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |

|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 70 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 31,68:1 | 32.6 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 75 | 65 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 26.53:1 | 32.4 | | 32.61:1 |  | |   **sdf** |
| **Examples Local Analysis**     |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 0,0 | 33.30 | 33.70 | | Frammento 0.1 | 37.27 | 37.60 | | Frammento 1,0 | 30.38 | 30.78 | | Frammento 1,2 | 30.95 | 31.25 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | | | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 69 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 33.40:1 | 37.30 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 76 | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 23.93:1 | 37.52 | | 33.0:1 |  | | |
| **Examples Local Analysis**     |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 0.2 | 36.0 | 36.77 | | Frammento 1.2 | 31.94 | 32.84 | | Frammento 2.1 | 35.0 | 35.7 | | Frammento 2.2 | 35.84 | 36.50 | | Frammento 3.1 | 35.25 | 36.1 | | Frammento 3.2 | 37.57 | 38.34 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |

|  |  |  |
| --- | --- | --- |
|  | | |
| **Originale** | **JPEG** | **Finale nostro** |
|  |  |  |
| **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 69 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 22.7:1 | 37.09 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 76 | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 14.73:1 | 37.18 | | 22.07:1 |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Examples Local Analysis**     |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 1.1 | 36.39 | 37.48 | | Frammento 1.2 | 34.22 | 35.29 | | Frammento 2.0 | 37.21 | 38.26 | | Frammento 2.1 | 33.43 | 34.44 | | Frammento 2.2 | 35.02 | 36.20 | | Frammento 2.3 | 36.40 | 37.57 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |

|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 39,68:1 | 36,4 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 90 | 45 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 26.53:1 | 36,3 | | 40.61:1 |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Examples Local Analysis** | | | |
| **Frammento** | **Jpeg PSNR** | **Custom PSNR** | |
| Frammento 0.1 | 41.33 | 46.65 | |
| Frammento 0.2 | 41.25 | 46.20 | |
| Frammento 1,1 | 40.56 | 44.94 | |
| Frammento 1,2 | 40.38 | 44.56 | |
| Frammento 2.1 | 37.25 | 42.04 | |
| Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. | | |  |
|  | | |

|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 5.5:1 | 40.1 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 90 | 45 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 2.1:1 | 40.0 | | 5.2:1 |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Examples Local Analysis**     |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 1.0 | 39.32 | 44.74 | | Frammento 1.1 | 39.70 | 45.73 | | Frammento 1,2 | 38.90 | 45.18 | | Frammento 1,3 | 38.17 | 44.0.6 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | | | | **Originale** | **JPEG** | **Finale nostro** | |  |  |  | | **Note: *A parità di Compression Ration il PSNR risulta molto simile*** | |  | | --- | | **Quality Compression** | | 65 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 28.8:1 | 34.0.1 | | |  |  | | --- | --- | | **Value Of**  **Fragment**  **Important** | **Value Of**  **Fragment Not Important** | | 75 | 60 |      |  |  | | --- | --- | | **Compression Ratio** | **PSNR** | | 19.0:1 | 30.0 | | 28.8:1 |  | | |
| **Examples Local Analysis**   |  |  |  | | --- | --- | --- | | **frammento** | **Jpeg PSNR** | **Custom PSNR** | | Frammento 0.2 | 33.49 | 34.69 | | Frammento 1.2 | 28.90 | 30.62 |   Anche in questo caso si riscontrano esattamente gli stessi risultati della foto precedente. |

# 

# 5. Risultati Significativi

I test condotti durante il progetto sono stati eseguiti usando la qualità del background e foreground come paramentro (**Foreground per i frammenti più importanti della foto mentre Backgroud per i frammenti meno importanti**), e nei test abbiamo assegnato questi due parametri per ottenere un Compression Ratio uguale, questo per ottenere un confronto equo tra i due algoritmi.

**Ma cosa succede se si comprime senza questa restrizione?**

I vantaggi sono notevoli, infatti a parità di PSNR il Compression Ratio è maggiore per l’algoritmo Custom, vediamo qualche esempio:

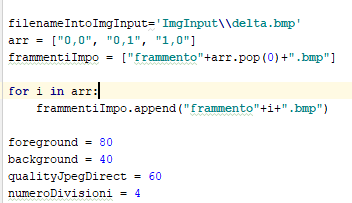
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Jpeg direct** | **Custom** | | C:\Users\Negan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\result.jpg | C:\Users\Negan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\result.bmp | | |  | | --- | | **Quality** | | **60** |  |  |  | | --- | --- | | **Compression-ratio** | **PSNR** | | **37.45:1** | 31 | | |  |  | | --- | --- | | **Background Quality** | **Foreground Quality** | | **40** | **80** |  |  |  | | --- | --- | | **Compression-ratio** | **PSNR** | | **41.92:1** | **31** | | |
| |  |  | | --- | --- | | **Jpeg direct** | **Custom** | | C:\Users\Negan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\result.jpg | C:\Users\Negan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\result.bmp | | |  | | --- | | **Quality** | | **70** |  |  |  | | --- | --- | | **Compression-ratio** | **PSNR** | | **30:1** | 34 | | |  |  | | --- | --- | | **Background Quality** | **Foreground Quality** | | **50** | **80** |  |  |  | | --- | --- | | **Compression-ratio** | **PSNR** | | **33:1** | **34** | | |

# 6. Uso dell'applicazione

Il progetto è basato su Python pertanto l’uso risulta molto semplice, per avviare l’applicazione basta semplicemente lanciare lo script main.py contenuto nella cartella principale del progetto.

Il file main.js è composto semanticamente da due parti, la prima rappresenta i parametri da dare in input, mentre la seconda contiene il motore dell’algoritmo, per usare l’algoritmo custom per la compressione delle immagini è necessario solamente assegnare i parametri.

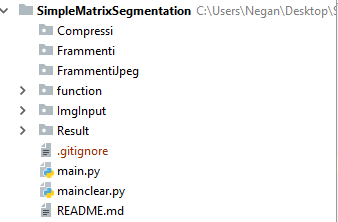
File main.js (**prima parte**)



Paramenti:

* **FilenameIntoImgInput** è la stringa che rappresenta l’immagine da testare, contenuta nella cartella ImgInput localizzata sotto la cartella principale.
* **arr** è un array di stringhe che rappresenta i frammenti più importanti dell’applicazione, una volta ottenuta la frammentazione dell’immagine in tanti frammenti, si possono selezionare i frammenti più importanti passando l’indice del frammento (nome sul discho del frammento).
* **Foregroung e Background** sono interi che rappresentano la qualità rispettivamente del Foreground (**Frammenti più rilevanti**) e Background (Frammenti meno rilevanti) da applicare all’algoritmo custom, sono interi tra 0 e 100.
* **QualityJpegDirect** è un intero che rappresenta la qualità da applicare a Jpeg quando si comprime la foto direttamente con Jpeg.
* **NumeroDivisioni** è un intero che rappresenta il numero di divisioni da fare sulla foto originale per ottenere i frammenti, se per esempio diamo il valore 4 otterremo una 4x4 frammenti

È importante anche conoscere la struttura delle cartelle per recuperare le informazioni dei test:



La cartella più importante è la cartella **Result** dove è possibile recuperare i risultati dell’applicazione, in particolare è presente il file ottenuto dalla compressione diretta con Jpeg e il file ottenuto dall’algoritmo custom.

Nella cartella **Frammenti** è possibile recuperare il file con estensione .bmp dei frammenti ottenuto dalla suddivisione matriciale.

E infine abbiamo la cartella **ImgInput** dove è possibile inserire l’immagine da testare.

Inoltre, è presente un file con nome **clear.py** che una volta avviato fa pulizia dei file ottenuti da una precedente esecuzione main.py, cancella i risultati, frammenti.

# 7. Conclusioni

Come possiamo osservare dai risultati ottenuti, si ricavano molteplici benefici nel procedere in questo modo. Il primo vantaggio è dato dalla possibilità di scegliere regioni di interesse a cui applicare una minore compressione così da preservarne la qualità. In secondo luogo usando una segmentazione matriciale possiamo vedere che risulta essere notevolmente più veloce rispetto a molti altri sistemi di segmentazione che possono essere addirittura NPhard. Il terzo vantaggio sta nel non allontanarci di molto da JPEG classico come

Compression Ratio. Quindi anche se computazionalmente questo metodo di compressione è molto oneroso, rispetto al solo JPEG come abbiamo potuto osservare beneficia di molteplici vantaggi che ne giustificano le performance. La qualità nelle zone di interesse è nettamente superiore e sacrificando la qualità del background si possono ottenere Compression Ratio molto alti. Inoltre, al nostro algoritmo può essere applicato un qualsiasi algoritmo lossy in circolazione basta cambiare le librerie associate alla compressione. Per quanto riguarda gli sviluppi futuri ci sono varie direzioni sulle quali indirizzarsi.

La prima è una migliore divisione matriciale per evitare il fenomeno del padding applicato da JPEG durante la compressione.

La seconda riguarda una miglior compressione dei frammenti JPEG evitando di passare per l’archivio tar ma applicare direttamente LMZA su tutti i file.