Politecnico di Torino

Ingegneria Informatica

A.a. 2024/2025

Sessione di laurea Luglio 2025

Valutazione della qualità video tramite analisi riproducibili su dataset di grandi dimensioni in ambiente HPC

Relatore:

Enrico Masala

Candidato:

Francesco Santangelo

Indice

Introduzione

La fruizione di contenuti multimediali è oggi una parte fondamentale della nostra vita quotidiana, influenzando settori come l'intrattenimento, l'informazione e l'educazione. Grazie alla crescita di piattaforme come Netflix, YouTube e Twitch, è possibile accedere a una vasta gamma di contenuti, dai film e serie TV agli eventi sportivi live e ai video amatoriali. Oltre a questi servizi, i social media come Instagram e TikTok hanno introdotto nuovi formati di video brevi, progettati per catturare l'attenzione degli utenti in pochi secondi. Inoltre, lo sviluppo di tecnologie come la compressione video in tempo reale e il miglioramento delle infrastrutture di rete hanno reso possibile una fruizione più fluida anche in ambienti con connessioni limitate.

La codifica video è fondamentale in tutti questi contesti, consentendo di trasmettere contenuti di alta qualità con un uso efficiente della banda di rete, e la decodifica permette di riprodurli in modo accurato sui dispositivi finali. Codec avanzati, come H.264, H.265 e VP9, comprimono il video riducendo

la quantità di dati trasmessi, ma senza perdere dettagli essenziali, garantendo quindi un'esperienza visiva di alto livello. In questo panorama complesso, risulta essenziale l'analisi e il confronto dei dati relativi alla qualità video per garantire che l'utente finale percepisca il contenuto nel modo più fedele possibile. Strumenti di misurazione della qualità come VMAF (Video Multi-Method Assessment Fusion) sono diventati un punto di riferimento nel settore. VMAF combina diversi algoritmi per valutare la qualità video percepita dall'occhio umano, misurando come varia l'esperienza visiva tra un video originale e uno compresso. Questo tipo di valutazione è particolarmente utile per determinare quali parametri di compressione offrono il miglior compromesso tra qualità e risparmio di larghezza di banda.

L'analisi accurata della qualità video si basa anche sull'uso di dataset ampi e diversificati, contenenti video che coprono vari scenari, risoluzioni e condizioni di illuminazione. Questi dataset sono molto utili per la ricerca e lo sviluppo di nuovi algoritmi di compressione, consentendo di categorizzare e analizzare le caratteristiche specifiche di ciascun video. Lo studio di questi dataset permette ai ricercatori di creare soluzioni più accurate e generalizzabili, capaci di adattarsi a scenari reali.

Garantire la riproducibilità delle analisi è essenziale per validare i risultati e condurre test affidabili. L'uso di Singularity, in combinazione con sistemi HPC, permette di creare ambienti isolati e scalabili, ideali per eseguire

misurazioni coerenti su larga scala. Questa soluzione, ampiamente adottata in ambito accademico e industriale, riduce l'influenza di variabili esterne, favorendo confronti precisi e un monitoraggio costante dell'evoluzione delle tecnologie di compressione, con un impatto diretto sul miglioramento della qualità dell'esperienza utente.

In questo contesto, la tesi si propone di approfondire l'approccio descritto, concentrandosi sulla creazione di ambienti riproducibili per l'analisi della qualità video di dataset, nonché sulla raccolta e sintesi dei risultati ottenuti da tali analisi. Le valutazioni sono state eseguite utilizzando script in Python e Bash, sfruttando le risorse computazionali dell'HPC del Politecnico di Torino e assicurando la riproducibilità tramite l'ambiente Singularity. Per quanto riguarda la raccolta dati si sono sfruttati formati quali json e CSV, per una facile rappresentabilità e portabilità dei risultati. Il lavoro di tesi si struttura nei seguenti capitoli:

Il capitolo 1, *Caratteristiche video* tratta delle principali caratteristiche dei video digitali, come risoluzione, frame rate e bitrate, indispensabili per l'analisi della qualità.

Nel capitolo 2, *Codifica e Decodifica video*, vengono esplorati i metodi di compressione e decompressione video, con un focus sugli algoritmi H.264, H.265,VP9 e formati contenitore come MP4 e MKV.

Il capitolo 3, *Analisi qualità video*, si concentra sui metodi di valutazione soggettiva e oggettiva della qualità, con particolare attenzione all'algoritmo VMAF.

Nel capitolo 4, *Ambiente di riproducibilità*, viene analizzato l'uso di sistemi HPC e la containerizzazione tramite Singularity per migliorare l'efficienza, l'isolamento e la riproducibilità delle analisi.

Il capitolo 5, *Strategie di sviluppo*, descrive le scelte progettuali e lo sviluppo di script in Bash e Python per l'analisi dei dataset, insieme all'impostazione del sistema HPC con container Singularity.

Il capitolo 6, *Sintesi e valutazione dei risultati*, presenta i risultati ottenuti, rappresentandoli tramite grafici e tabelle per una rapida comprensione.

Infine, la *Conclusione* riassume i risultati ottenuti, suggerendo possibili sviluppi futuri.

Capitolo 1

Caratteristiche video

Questo capitolo fornisce una panoramica delle principali caratteristiche tecniche che definiscono un video digitale, le quali influiscono sulla sua qualità visiva e sulla gestione dei dati durante la compressione e la trasmissione. Verranno analizzati aspetti fondamentali come la risoluzione, il frame rate, il bit rate, la crominanza, la luminanza, la profondità del colore e la sua rappresentazione, con un focus su come questi elementi interagiscono tra loro e come influiscono sul risultato finale del video

1.1 Risoluzione

La risoluzione di un video indica il numero totale di pixel che compongono ogni fotogramma dell'immagine. Maggiore è il numero di pixel, più dettagliata e nitida sarà l'immagine. Le risoluzioni video più comuni sono:

- SD (Standard Definition) – 720x480 pixel: una risoluzione tipica dei DVD e della televisione analogica, con dettagli limitati rispetto agli standard moderni.

- HD (High Definition) – 1280x720 pixel: offre una qualità significativamente superiore rispetto alla SD, con immagini più definite e nitide.
- Full HD – 1920x1080 pixel: uno degli standard più diffusi per lo streaming, la TV digitale e il Blu-ray, con una qualità d'immagine elevata.
- Ultra HD – 3840x2160 pixel: quattro volte la risoluzione del Full HD, con un livello di dettaglio estremamente elevato, ideale per schermi di grandi dimensioni e contenuti di alta qualità.

Una risoluzione più alta non solo migliora la qualità visiva, ma aumenta anche la quantità di dati necessari per la trasmissione e l'archiviazione del video. Ad esempio, un video in 4K richiede più spazio su disco e maggiore larghezza di banda rispetto a un video in Full HD.

1.2 Frame e Bit rate

Il *frame rate* è il numero di fotogrammi visualizzati per secondo (fps). I valori comuni includono:

- 24 fps: standard per i film
- 30 fps: comune nei video online e nei programmi TV
- 60 fps ed oltre: utilizzato per video ad alta fluidità, come nei giochi o nelle trasmissioni sportive

Un frame rate più alto comporta una maggiore fluidità, ma aumenta anche la quantità di dati necessari per il video.

Il *bitrate* invece rappresenta la quantità di dati utilizzata per descrivere il video in un determinato periodo di tempo. È generalmente espresso in bit per secondo (bps). Il bitrate influisce sulla qualità del video e sulle dimensioni del file. Un bitrate più elevato ad esempio aumenta la qualità visiva del video, poiché ci sono più dati disponibili per descrivere ogni fotogramma, ma aumenta anche la dimensione del file. Viceversa uno più basso riduce la qualità, ma rende anche il file più piccolo e più facile da trasmettere o archiviare.

1.3 Profondità del Colore

La profondità del colore (o *bit-depth*) rappresenta il numero di bit utilizzati per definire il colore di ciascun pixel di un'immagine. Maggiore è la profondità del colore, più ampia è la gamma di colori che può essere rappresentata, permettendo una riproduzione più accurata e una transizione più graduale tra le sfumature di colore. I valori più comuni di profondità del colore includono:

- *8-bit*: Rappresenta 256 livelli per ciascuna delle tre componenti di colore (rosso, verde, blu), per un totale di circa 16,7 milioni di colori.

- *10-bit*: Rappresenta 1024 livelli per ciascun canale, consentendo una gamma di colori notevolmente più ampia e una transizione più fluida tra le sfumature. Questo livello di profondità del colore è ampiamente utilizzato nei video HDR (High Dynamic Range), dove è richiesta una maggiore precisione per rappresentare l'ampio spettro di luminosità.
- *12-bit e oltre*: Tipicamente utilizzato in ambito professionale, come nella produzione cinematografica o nei video HDR di altissima qualità, dove è necessario preservare la massima fedeltà dei dettagli visivi.

1.4 Rappresentazione e campionamento del colore

I colori nei video digitali possono essere rappresentati attraverso diversi modelli di colore, tra cui il modello RGB e il modello YUV. Il modello RGB (Red, Green, Blue) è basato sulla combinazione additiva di tre colori primari: rosso, verde e blu. Ogni colore visibile può essere ottenuto variando l'intensità di queste tre componenti, e la gamma cromatica riproducibile dal modello RGB è spesso rappresentata all'interno del diagramma di cromaticità CIE 1931. Questo diagramma mostra l'intero spettro di colori visibili all’occhio umano, evidenziando le limitazioni dello spazio colore RGB, che copre solo una porzione di tale spettro.

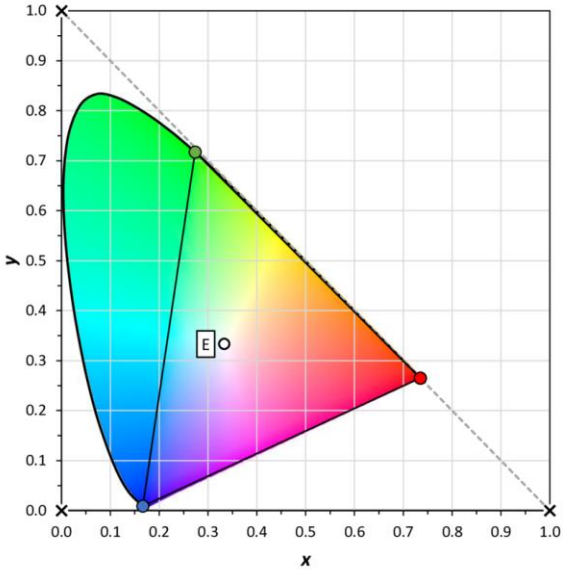

Figura 1.1: Diagramma CIE e spazio di colore coperto da RGB [1]

Nonostante la sua diffusione nei display digitali, il modello RGB non è ideale per la compressione e la trasmissione video, poiché non tiene conto del modo in cui l'occhio umano percepisce le immagini.

Per migliorare l'efficienza nella codifica video, l'RGB viene convertito nel modello YUV, che separa luminosità e colori in due componenti distinte:

- *Luminanza (Y)*: rappresenta la componente di luminosità di un'immagine. È la parte del video che controlla l'intensità di luce e ombra ed è la più rilevante per la percezione visiva generale.
- *Crominanza (U e V)*: rappresenta la componente di colore di un'immagine. Le due componenti U (Blue-difference chroma) e V (Red-difference chroma) indicano rispettivamente la differenza tra il valore del blu/rosso e la media della luminosità.

Spesso viene ridotta nella compressione video rispetto alla luminanza, poiché l'occhio umano è meno sensibile alle variazioni di colore rispetto alle variazioni di luminosità.

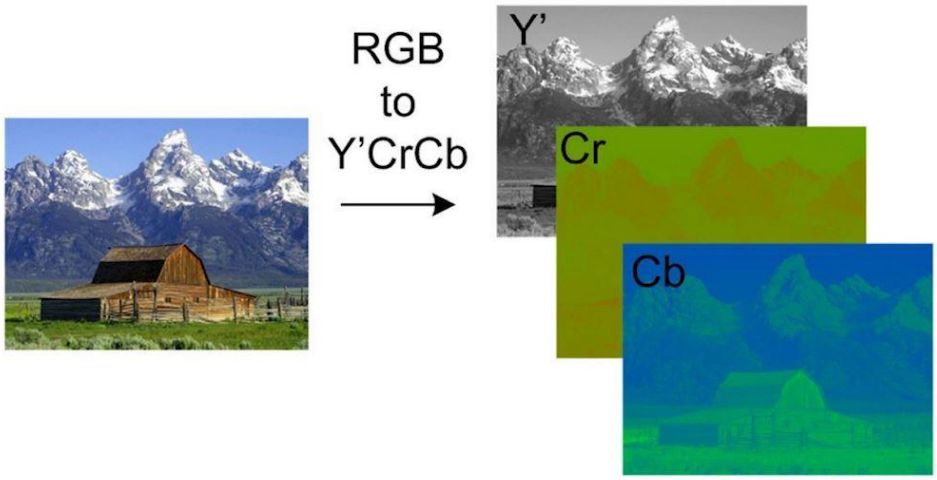

Figura 1.2: Esempio suddivisione luminanza e crominanza [2]

Il formato YUV è una rappresentazione analogica del colore e ne esistono diverse varianti, a seconda di come vengono campionate le componenti di crominanza in relazione alla luminanza, tra cui:

- *YUV 4:4:4*: tutte le componenti (Y, U, V) vengono campionate per ogni pixel, senza alcun subsampling. Questo formato è utilizzato nei video di alta qualità, come in produzione cinematografica.
- *YUV 4:2:2*: la crominanza orizzontale è campionata a metà della risoluzione orizzontale rispetto alla luminanza, ma la crominanza

verticale è campionata per ogni pixel. È comune nelle applicazioni professionali.

- *YUV 4:2:0*: la crominanza è campionata a metà della risoluzione sia orizzontale che verticale rispetto alla luminanza. Questo è il formato più comune nei video compressi, come quelli H.264 e H.265.

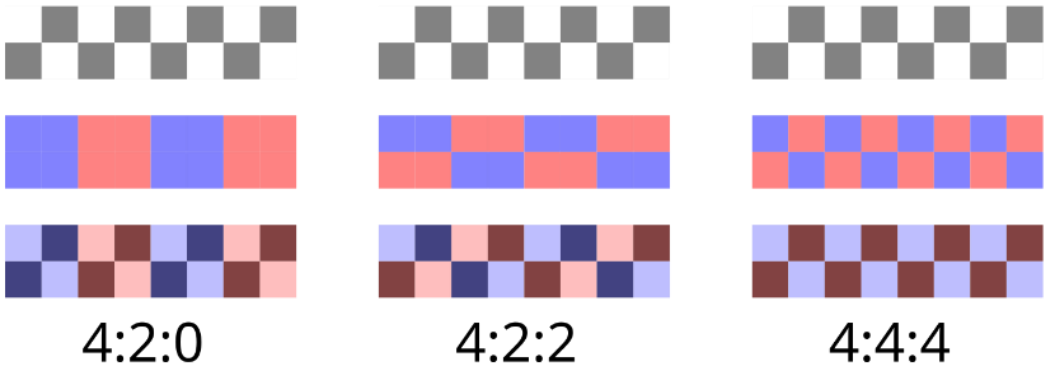

Figura 1.3: Sottocampionamento della crominanza [3]

Un file Y4M rappresenta una variante del formato YUV, strutturato per semplificare l'elaborazione e la gestione di video non compressi. Il formato Y4M è frequentemente utilizzato in software di elaborazione video, come FFmpeg, per agevolare il trattamento di video raw, permettendo l'accesso e la manipolazione dei dati senza necessità di compressione. Oltre ai dati video in formato YUV, il formato Y4M include anche una serie di metadati che forniscono informazioni cruciali riguardo le caratteristiche del video. All'inizio di ogni file Y4M è presente una riga di intestazione che descrive parametri quali la risoluzione, il frame rate, il formato di campionamento e

la profondità di colore. Il corpo del file Y4M contiene una sequenza di fotogrammi video codificati in formato YUV, con ogni fotogramma che rappresenta un'istanza completa del video. I dati sono organizzati in modo tale che ogni fotogramma possa essere facilmente estratto e processato da software di analisi video, particolarmente in contesti che richiedono una manipolazione dettagliata dei dati pixel-per-pixel. Il formato Y4M è compatibile con diverse applicazioni di elaborazione video, come software di transcodifica, analisi della qualità video e visualizzazione. La sua struttura semplice lo rende particolarmente adatto per operazioni che richiedono l'elaborazione di video non compressi, come la misurazione della qualità tramite VMAF.[4]

Capitolo 2

Codifica e decodifica

La rappresentazione digitale dei contenuti video richiede una grande quantità di dati. Sebbene la capacità di archiviazione e trasmissione dei dati continui a crescere, senza tecniche di compressione efficaci, il volume dei dati prodotto dalla digitalizzazione sarebbe eccessivo e difficile da gestire con le infrastrutture odierne. Ad esempio, un video non compresso in 1080p (1920x1080) a 30 fotogrammi al secondo può arrivare a occupare circa 3 Gbps.

La *codifica* (o *compressione*) *video* è il processo che permette di ridurre il peso di un video digitale, trasformandolo in un flusso numerico con dimensioni più appropriate per l'archiviazione e la trasmissione. La *decodifica* (o *decompressione*) *video* è il processo inverso: a partire dal flusso numerico compresso, il decoder ricostruisce il segnale video originale, applicando trasformazioni inverse per ripristinare l'immagine. L'integrazione dei due approcci è possibile distribuire video in streaming, trasmettere

segnali televisivi digitali e archiviare grandi quantità di contenuti con un'efficienza elevata. Senza la compressione, la trasmissione di video in alta definizione o in 4K sarebbe impraticabile a causa delle enormi quantità di dati coinvolte. Allo stesso modo, la decompressione garantisce che i contenuti compressi possano essere ricostruiti e visualizzati con una qualità elevata, bilanciando efficienza e fedeltà del segnale originale.

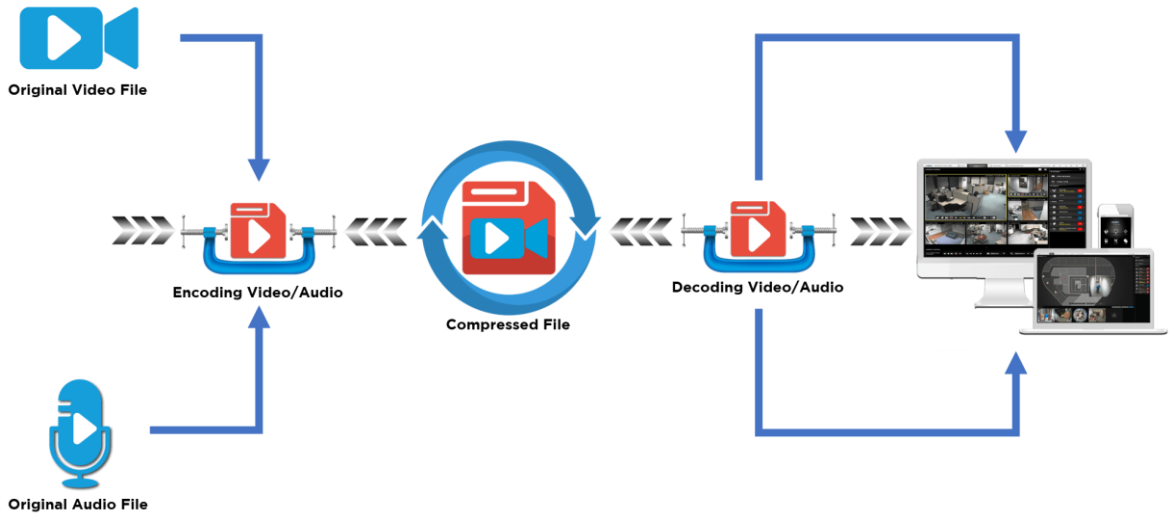

Figura 2.1: Processo di codifica e decodifica [5]

2.1 Codifica Video

La codifica video ha due obiettivi principali: ridurre la dimensione dei file e ottimizzare la qualità percepita. I video non compressi richiedono quantità enormi di spazio, ma comprimendo i dati si possono ridurre

significativamente le dimensioni mantenendo una buona qualità visiva. La codifica video rende così possibile lo streaming e l'archiviazione efficienti.

Vi sono due principali tecniche di compressione, temporale e spaziale.

2.1.1 Compressione Temporale (o Interframe)

La compressione temporale rappresenta una strategia essenziale nell'ambito della codifica video, in quanto sfrutta le ridondanze presenti tra fotogrammi consecutivi. In particolare, quando i cambiamenti tra un fotogramma e l'altro sono minimi – come in presenza di scene statiche o con movimenti lenti – diventa superfluo memorizzare integralmente ogni fotogramma. Al contrario, è possibile codificare solamente le differenze rilevate, ottenendo così una significativa riduzione del volume di dati da elaborare e trasmettere.

Un ruolo centrale in questo processo è svolto dalla predizione del moto, la quale si articola in tre fasi fondamentali: *motion estimation*, *motion prediction* e *motion compensation*. Queste tecniche, strettamente correlate, consentono di sfruttare al meglio la ridondanza temporale, migliorando l'efficienza della compressione video e riducendo il bitrate necessario per la trasmissione o l'archiviazione dei dati.

La fase di *motion estimation* costituisce il primo step del processo. In questa fase, il fotogramma corrente viene suddiviso in blocchi di dimensioni

standard, generalmente 4x4, 8x8 o 16x16 pixel. Per ciascun blocco, l'encoder esamina uno o più fotogrammi di riferimento (I-frame, P-frame o B-frame) al fine di individuare la porzione d'immagine che risulti maggiormente affine. Il confronto, effettuato mediante algoritmi sofisticati, permette di determinare i *motion vector*, ovvero i vettori che indicano lo spostamento del blocco dalla sua posizione originale a quella presente nel fotogramma di riferimento. Questa fase è cruciale in quanto fornisce la base informativa necessaria per prevedere i successivi cambiamenti nei fotogrammi.

Una volta ottenuti i motion vector, si passa alla fase di *motion prediction*. In questa fase, l'informazione derivata dalla motion estimation viene utilizzata per anticipare il contenuto del fotogramma successivo, basandosi su uno o più fotogrammi precedenti. L'ipotesi alla base di questa tecnica è che, nella maggior parte dei video, il movimento degli oggetti avvenga in maniera regolare e prevedibile. Conoscendo la direzione e l'entità dello spostamento, è possibile predire in maniera accurata il nuovo fotogramma, codificando quindi soltanto le differenze residue rispetto all'immagine attesa. Tale approccio consente di ridurre ulteriormente la quantità di dati da trasmettere, concentrando l'attenzione solo sulle variazioni rilevanti.

La fase finale è quella della *motion compensation*, che applica concretamente i motion vector calcolati nella fase di motion estimation per ricostruire il

nuovo fotogramma. In pratica, il processo di motion compensation utilizza il fotogramma di riferimento e trasla i blocchi d'immagine secondo i vettori individuati, permettendo così di ricostruire il fotogramma corrente. In questo modo, anziché codificare ogni singolo pixel, il sistema trasmette esclusivamente i motion vector e le differenze residue rispetto alla previsione effettuata. Questo metodo non solo riduce il bitrate, ma permette anche di mantenere elevata la qualità visiva, bilanciando l'efficienza della compressione con la fedeltà dell'immagine.

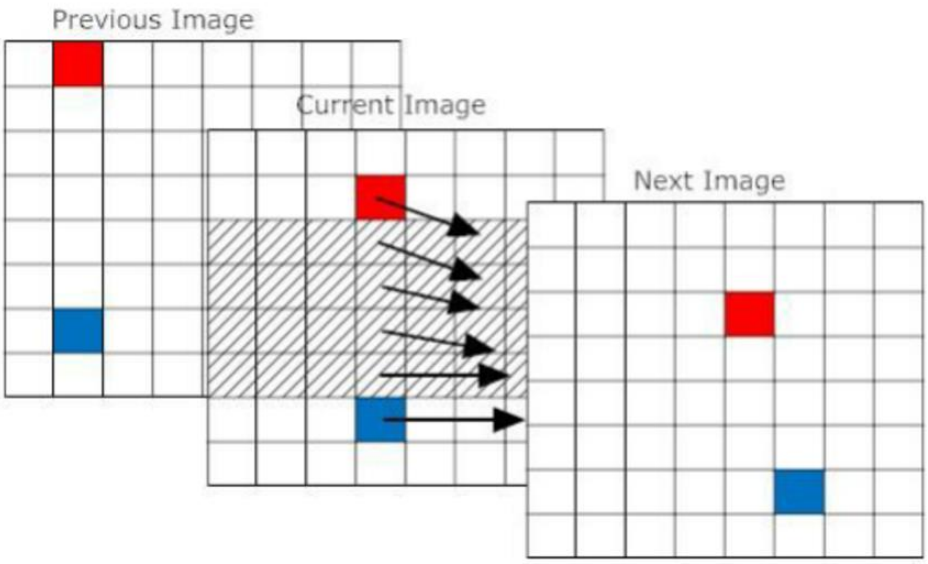

Figura 2.2: Motion prediction tramite motion vector [6]

La predizione del moto e la compensazione del moto vengono implementate attraverso tre tipi principali di frame:

- I-Frame (Intracoded Frame): è un fotogramma completo, senza riferimenti ad altri frame, ed è solitamente più grande. Viene usato come punto di partenza per i successivi frame compressi.
- P-Frame (Predicted Frame): è un fotogramma codificato in base a un I-Frame o a un altro P-Frame precedente, contenendo solo le differenze necessarie per ricostruire l'immagine.
- B-Frame (Bidirectional Frame): utilizza informazioni sia dai fotogrammi precedenti che successivi, garantendo una compressione più efficiente e una maggiore riduzione del bitrate.

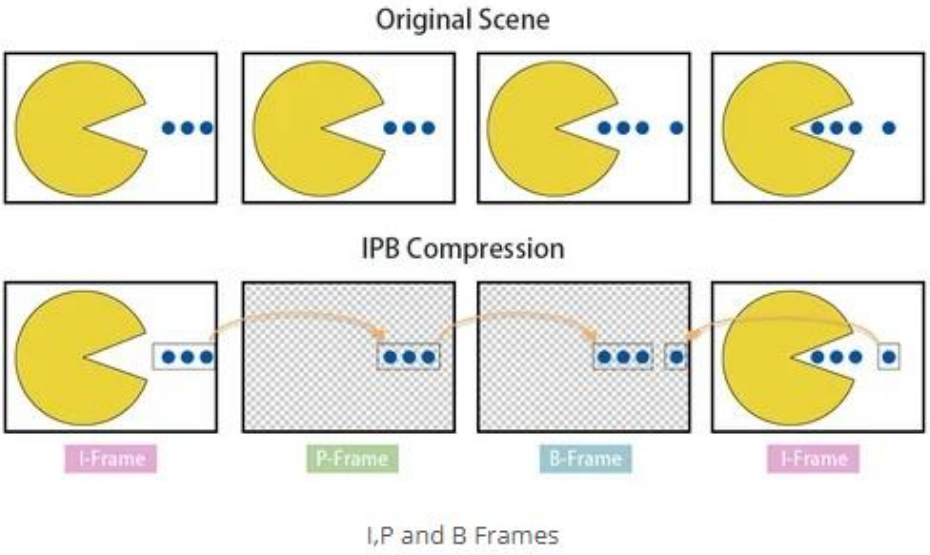

Figura 2.3: Esempio di frame di tipo I, P e B [7]

I fotogrammi all'interno di un video non vengono gestiti in modo lineare durante la codifica. In effetti, il processo di codifica prevede un trattamento dei fotogrammi che può sembrare contrario all'ordine temporale in cui

dovrebbero apparire nella riproduzione. Questo approccio si riflette nei concetti di codec e display order, che determinano rispettivamente l'ordine in cui i fotogrammi vengono codificati e decodificati, e quello in cui devono essere visualizzati durante la riproduzione del video.

Il *codec order* si riferisce all'ordine in cui i fotogrammi vengono effettivamente elaborati dai codec durante il processo di codifica e decodifica. In questo ordine, i fotogrammi vengono codificati e decodificati in base a come dipendono l'uno dall'altro. Il *display order*, d'altro canto, è l'ordine in cui i fotogrammi devono essere visualizzati durante la riproduzione del video. Questo ordine segue la sequenza temporale naturale dei fotogrammi, ossia l'ordine in cui dovrebbero apparire sullo schermo durante la visione del video. Esso non coincide sempre con quello di codifica, e il motivo è che alcuni fotogrammi, come i B-Frame, vengono effettivamente codificati prima di altri fotogrammi ma devono essere visualizzati solo dopo i fotogrammi di riferimento. In pratica, quindi, anche se un fotogramma viene codificato prima, non verrà mostrato fino a quando non saranno decodificati i fotogrammi sui quali dipende.

2.1.2 Compressione Spaziale (o Intraframe)

La compressione intraframe si concentra sull'analisi e sulla compressione dei dati all'interno di ciascun singolo fotogramma. Questo processo mira a rimuovere le ridondanze interne presenti nel fotogramma, come ad esempio

pattern e colori ripetuti, che non aggiungono valore all'immagine finale e occupano spazio inutile. Tra le varie tecniche utilizzate una delle più importanti è la Trasformata Discreta del Coseno (DCT). La DCT è una trasformazione matematica che converte un segnale spaziale (i pixel di un'immagine) in una rappresentazione di frequenze.

Il processo consiste inizialmente nel dividere l'immagine in blocchi 8x8 (o altre dimensioni). Successivamente, applicando la DCT su ogni blocco, si convertono i valori di intensità dei pixel in una combinazione di frequenze spaziali, restituendo come risultato una matrice di coefficienti che, partendo dall'elemento in alto a sinistra, rappresenta le frequenze più basse (informazioni di base sull'immagine), scendendo verso destra quelle più alte (informazioni più dettagliate come ad esempio i bordi). Poiché l'occhio umano è meno sensibile ai dettagli fini, i coefficienti ad alta frequenza vengono quantizzati, in modo tale da poter essere scartati successivamente.

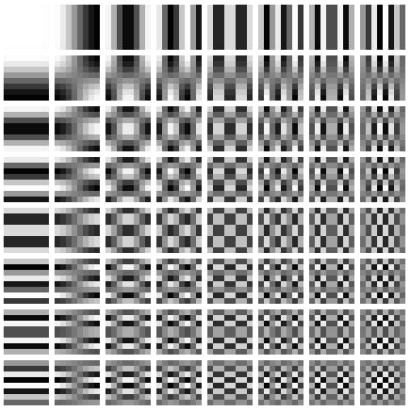

Figura 2.4: Matrice dei coefficienti risultante dalla DCT [8]

Poi tale matrice viene condensata in una sequenza di coefficienti tramite il riordinamento Zig-zag, che permette così di avere in coda alla sequenza valori nulli consecuitivi. Tale sequenza infine passa attraverso tecniche di codifica entropica, quali Run-Length (che elimina sequenze di valori ripetuti) e Huffman (assegna codici binari più corti ai simboli più frequenti), riducendo così la dimensione complessiva dei dati.

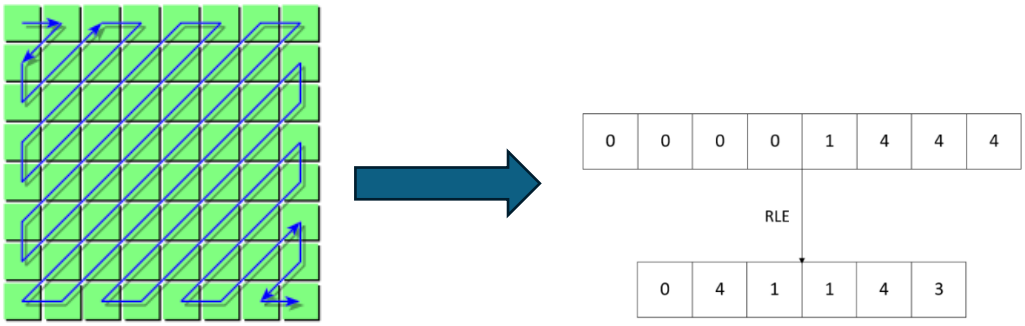

Figura 2.5: Esempio di riordinamento zig-zag e successiva codifica run-length

[9] [10]

2.2 Codec e formati contenitore

I codec (coder/decoder) sono algoritmi fondamentali per comprimere e decomprimere file audio e video, consentendo la trasmissione e la memorizzazione efficiente dei contenuti multimediali.

Lo sviluppo dei codec video è iniziato negli anni '80 e '90 con MPEG-1, utilizzato per i primi formati video digitali, e MPEG-2, che ha facilitato la diffusione della televisione digitale e dei DVD. Con l'aumento della

domanda di contenuti in alta definizione, è stato introdotto H.264 (AVC), divenuto lo standard principale per lo streaming e i Blu-ray. Successivamente, H.265 (HEVC) ha ridotto il bitrate per video in alta definizione e 4K, mentre VP9 ha offerto un'alternativa open-source. Oggi, standard come AV1 e VVC migliorano ulteriormente l'efficienza, supportando contenuti in 8K, HDR e realtà virtuale.

Nel contesto della compressione video, i codec sono progettati per bilanciare l'efficienza di compressione e la qualità dell'immagine. Alcuni, come H.264, H.265 e VP9, adottano una compressione *lossy*, eliminando informazioni ridondanti per ridurre la dimensione del file, mentre altri, come Apple ProRes, sono progettati per una compressione *lossless*, preservando integralmente i dati originali per applicazioni professionali.

Nel contesto di questa tesi, sono stati analizzati dataset di sequenze video a compressione lossy, tra i quali emergono quelli di seguito descritti.

2.2.1 H.264

H.264, noto anche come Advanced Video Coding (AVC), è un codec video che utilizza la compressione interframe per ridurre la ridondanza temporale tra fotogrammi consecutivi. Questo è possibile grazie a tecniche avanzate come la previsione del moto e la compensazione dei blocchi, che consentono di codificare solo le differenze rispetto ai fotogrammi precedenti piuttosto

che l'intero contenuto. La codifica avviene suddividendo i fotogrammi in macroblocchi, solitamente di dimensione 16×16 pixel, che vengono poi compressi attraverso la DCT. Questa operazione consente una rappresentazione più efficiente dei dati, migliorando la qualità a parità di bitrate rispetto agli standard precedenti.

Un aspetto chiave di H.264 è l'uso di due metodi di codifica entropica: *Context-Adaptive Variable Length Coding* (CAVLC) e *Context-Adaptive Binary Arithmetic Coding* (CABAC). CABAC, sebbene più complesso dal punto di vista computazionale, offre una maggiore efficienza di compressione rispetto a CAVLC, consentendo di ottenere una migliore qualità visiva a parità di bitrate. Inoltre, H.264 supporta diverse modalità di predizione intra e interframe, adattandosi a diverse esigenze applicative, dalla trasmissione in rete alla compressione per l'archiviazione locale.[11]

2.2.2 H.265

H.265, noto come High Efficiency Video Coding (HEVC), è il successore di H.264 e introduce un'architettura di codifica più avanzata, progettata per gestire in modo più efficiente i contenuti ad alta risoluzione, fino a 8K. Una delle principali innovazioni di H.265 è l'introduzione delle Coding Tree Units (CTU), che sostituiscono i macroblocchi di H.264 e possono variare

dinamicamente in dimensione fino a 64×64 pixel. Questo consente una maggiore flessibilità nella rappresentazione delle aree dell'immagine e migliora la compressione in scene complesse o con dettagli fini. H.265 ottimizza ulteriormente la compressione interframe permettendo l'uso di più fotogrammi di riferimento e una segmentazione più avanzata dei blocchi di predizione. Integra anche miglioramenti nella DCT, utilizzando blocchi di dimensione 4×4, 8×8, 16×16 e 32×32, riducendo in modo più efficace la ridondanza spaziale. La codifica entropica si basa esclusivamente su CABAC, garantendo un'efficienza superiore nella compressione rispetto a H.264. Grazie a queste ottimizzazioni, H.265 può ridurre il bitrate di circa il 50% rispetto a H.264, mantenendo la stessa qualità visiva. Questo lo rende ideale per applicazioni che richiedono alta efficienza di banda, come lo streaming video in 4K, la trasmissione su reti mobili e la distribuzione di contenuti HDR. Tuttavia, la maggiore complessità computazionale del codec richiede hardware più potente per la decodifica, il che ha influenzato la sua adozione su dispositivi meno recenti.[12]

2.2.3 VP9

VP9, sviluppato da Google, è un codec open source royalty-free progettato per migliorare l'efficienza di compressione rispetto ai suoi predecessori (come VP8) e per competere con H.265. VP9 utilizza tecniche avanzate come la compressione basata su blocchi di dimensioni variabili, che consente

una gestione più precisa delle informazioni video. Inoltre, VP9 include una codifica predittiva bidirezionale, che migliora la qualità visiva senza richiedere un incremento significativo del bitrate.

VP9 è particolarmente efficiente nella compressione dei contenuti HDR e 4K, riducendo la larghezza di banda necessaria per lo streaming ad alta definizione, pur mantenendo una qualità visiva elevata.[13]

2.2.4 MP4 e MKV

Oltre ai codec, i *formati contenitore* (come MP4 e MKV) svolgono un ruolo fondamentale nella gestione dei file multimediali. Un contenitore è un file che può includere video, audio, sottotitoli e metadati, combinando questi elementi in un'unica struttura. La differenza chiave tra *contenitore* e *codec* è che il primo definisce il metodo di compressione e decompressione del video o dell'audio, mentre il secondo determina come questi flussi vengono organizzati all'interno del file.

Il formato MP4, noto anche come MPEG-4 Part 14, è uno dei più diffusi per la memorizzazione e la distribuzione di contenuti multimediali. La sua popolarità deriva dalla straordinaria compatibilità con una vasta gamma di dispositivi, sistemi operativi e piattaforme di streaming. Se hai mai scaricato un video o guardato un film su uno smartphone, una console o un computer, è molto probabile che fosse in formato MP4. Questo perché è stato progettato

per essere leggero ed efficiente, riuscendo a bilanciare una buona qualità video con dimensioni contenute.

Uno dei punti di forza di MP4 è la sua stretta integrazione con H.264 (AVC) e H.265 (HEVC), che permettono di ottenere una qualità elevata riducendo lo spazio occupato dal file. Proprio per questa efficienza, MP4 è il formato di riferimento per il video streaming.

Dal punto di vista della gestione dei contenuti, MP4 supporta tracce audio e sottotitoli, ma con alcune limitazioni. Se è vero che consente di incorporare più tracce audio (utile per i film multilingua) e sottotitoli, non è il formato ideale per chi ha bisogno di un'elevata personalizzazione nella gestione di questi elementi. Ad esempio, le opzioni per i sottotitoli sono piuttosto basilari rispetto a formati più versatili come MKV.

Passando proprio a MKV, o Matroska Video, è un formato open-source creato per offrire la massima flessibilità. Uno dei principali vantaggi è il supporto avanzato per tracce multiple. Un singolo file può contenere non solo diverse tracce video e audio, ma anche una vasta gamma di sottotitoli in vari formati, inclusi quelli con effetti grafici e animazioni. Inoltre, MKV permette di integrare capitoli interattivi, come quelli di un DVD o Blu-ray, e persino metadati avanzati.

Un altro aspetto che distingue MKV da MP4 è la capacità di gestire video non compressi o compressi senza perdita di qualità. Questa caratteristica lo rende particolarmente utile per chi lavora con il montaggio video e ha bisogno di preservare ogni dettaglio dell'immagine.

Tuttavia, la compatibilità con i dispositivi è inferiore rispetto a MP4. Non tutti i lettori video supportano MKV nativamente, e spesso è necessario installare software dedicati, come VLC Media Player o altri codec specifici. [14] [15]

2.3 Decodifica Video

La decodifica video è il processo che consente di ripristinare i dati video compressi in un formato che possa essere visualizzato e riprodotto correttamente dai dispositivi, come lettori multimediali o piattaforme di streaming. In sostanza, la decodifica applica le operazioni inverse rispetto alla codifica, recuperando i fotogrammi e le informazioni mancanti per ricreare il video originale.

2.3.1 Decodifica Intraframe

In questa fase, i fotogrammi completi, noti come I-frame, vengono ricostruiti utilizzando solo le informazioni interne a ciascun fotogramma. Le

trasformate inverse, come la IDCT (Inverse Discrete Cosine Transform), vengono applicate per recuperare i dati visivi compressi, restituendo così l'immagine che corrisponde al fotogramma originale.

2.3.2 Decodifica Interframe

In questa fase, i P-frame e B-frame vengono ricostruiti sulla base dei dati memorizzati e delle differenze rispetto agli altri fotogrammi. Per fare ciò, vengono riassemblati i blocchi di movimento e i dati predetti vengono riagganciati ai fotogrammi di riferimento. Questo processo permette di ricostruire ogni fotogramma in modo fluido e continuo. Gli algoritmi di compensazione del moto e predizione temporale, che erano stati applicati durante la codifica, vengono invertiti per restituire il contenuto visivo in modo che possa essere visualizzato correttamente sul dispositivo di riproduzione. Nonostante la decodifica richieda comunque risorse significative, specialmente per codec ad alta compressione come l'H.265, essa è generalmente meno onerosa rispetto alla codifica. Questo perché, mentre la codifica deve eseguire operazioni computazionalmente costose come la motion estimation, la decodifica dispone già dei motion vector calcolati e si limita ad applicare la motion compensation, ricostruendo il

fotogramma senza dover effettuare ulteriori stime. Di conseguenza, la decodifica risulta un processo più leggero in termini computazionali.

2.4 FFmpeg

FFmpeg è un software potente e versatile, utilizzato per l'elaborazione e la gestione di contenuti multimediali, inclusi video e audio. Fin dagli inizi, grazie al suo carattere open source e al contributo di una vasta comunità internazionale di sviluppatori, FFmpeg ha saputo evolversi rapidamente, adattandosi alle nuove tecnologie e alle esigenze del settore. L'adozione di una licenza open source e la portabilità su piattaforme diverse (Windows, Linux, macOS e dispositivi embedded) hanno favorito la sua diffusione in ambiti sia accademici che industriali, rendendolo uno strumento imprescindibile per la conversione, l'editing e lo streaming dei media.

Una delle sue principali funzionalità è la conversione tra formati, che consente di trasformare file da un formato all'altro in modo efficiente. Ad esempio, è possibile convertire un video da MP4 o MKV a YUV o Y4M, mantenendo la qualità visiva originale, grazie alla possibilità di utilizzare una vasta libreria di codec. Questa capacità di conversione è fondamentale per ottimizzare i file in base alle specifiche esigenze, ad esempio per l'analisi

della qualità video, poiché i formati YUV e Y4M sono ampiamente utilizzati per test e valutazioni senza perdita di qualità dovuta alla compressione.

Oltre alla conversione, FFmpeg offre un ampio supporto per l'elaborazione avanzata dei video tramite filtri, che consentono di eseguire operazioni come il ridimensionamento, il ritaglio, l'aggiunta di effetti speciali o la modifica del frame rate, ottimizzando così la fluidità e la qualità del contenuto. Inoltre, è possibile combinare vari filtri per applicare modifiche complesse, come l'integrazione di sottotitoli o la regolazione del contrasto e della saturazione, rendendo FFmpeg uno strumento estremamente versatile per la post-produzione video.

Un'altra caratteristica distintiva di FFmpeg è la sua capacità di supportare lo streaming. Grazie a numerosi protocolli come RTMP, HLS e RTSP, il software permette la trasmissione in tempo reale di video su Internet o reti private, ottimizzando la latenza e il buffering per garantire una trasmissione fluida e di alta qualità. La sua architettura flessibile consente anche la transcodifica in tempo reale, adattando il flusso video alle condizioni di rete o alle specifiche dei dispositivi di destinazione.

Un elemento fondamentale alla base del successo di FFmpeg è la sua architettura modulare. Il software è composto da diverse librerie specializzate, come *libavcodec*, *libavformat* e *libavfilter*, ciascuna dedicata a specifiche operazioni di elaborazione multimediale. Questa struttura

modulare non solo permette aggiornamenti e sostituzioni indipendenti dei componenti, ma facilita anche l'integrazione di nuove funzionalità e il supporto a codec emergenti. Di conseguenza, FFmpeg può essere facilmente incorporato in pipeline di elaborazione personalizzate e in applicazioni più ampie, adattandosi in modo flessibile alle specifiche esigenze dei vari progetti.[16]

Capitolo 3

Analisi qualità video

Le analisi della qualità video si suddividono principalmente in due categorie: *analisi soggettive* e *analisi oggettive*. Questi approcci vengono impiegati per valutare la qualità percepita o misurata dei contenuti video, in ambiti che spaziano dal broadcasting allo streaming, fino alla compressione video. Mentre le analisi soggettive si basano sul giudizio umano, quelle oggettive fanno uso di algoritmi per fornire una valutazione più automatizzata e replicabile. Entrambi gli approcci presentano vantaggi e limiti, e spesso sono usati in modo complementare per ottenere una valutazione più completa.

3.1 Analisi Soggettive

Le analisi soggettive si basano su osservatori umani che valutano la qualità percepita dei contenuti video in condizioni specifiche e controllate. Queste

valutazioni risultano fondamentali per comprendere come gli utenti percepiscono la qualità del video, indipendentemente dai risultati numerici ottenuti attraverso metriche oggettive. Le analisi soggettive sono considerate il metodo più accurato per misurare la qualità percepita dai consumatori finali. Tuttavia, presentano diversi limiti: richiedono tempo, sono costose e necessitano di condizioni di visione rigorosamente controllate per ridurre al minimo le variabili che potrebbero influenzare il giudizio umano.

3.1.1 MOS

Il MOS (Mean Opinion Score) è uno dei metodi più utilizzati e standardizzati nell'analisi soggettiva. Agli osservatori viene chiesto di assegnare un punteggio a un video su una scala tipicamente da 1 a 5:

- 1 – Qualità molto bassa (Bad): L'esperienza visiva è gravemente compromessa. Il video presenta artefatti evidenti come forte blocchettizzazione, perdita di dettagli, distorsioni di colore o fluidità molto ridotta. La qualità è inaccettabile per la maggior parte degli spettatori.
- 2 – Qualità scarsa (Poor): Il video è fruibile ma con evidenti difetti che ne compromettono l'esperienza. Possono esserci sfocature,

compressione e distorsioni che rendono difficile distinguere i dettagli. Gli artefatti sono frequenti e fastidiosi.

- 3 – Qualità discreta (Fair): Il video è accettabile, ma la qualità non è ottimale. Sono presenti alcune imperfezioni, come una leggera perdita di dettagli o piccoli artefatti di compressione, ma non compromettono significativamente la visione. La qualità è sufficiente per un utilizzo generale.
- 4 – Qualità buona (Good): Il video è chiaro e piacevole da guardare, con una qualità visiva generalmente elevata. Possono esserci minimi difetti, ma la maggior parte degli utenti non li nota o li trova trascurabili.
- 5 – Qualità eccellente (Excellent): Il video è privo di difetti visibili, con una qualità pari o molto vicina a quella originale. I dettagli sono nitidi, i colori fedeli e la fluidità perfetta.

La media dei punteggi raccolti viene calcolata per ottenere una misura globale della qualità percepita. Questo metodo è semplice da implementare, ma può essere influenzato da fattori soggettivi come l'esperienza, la familiarità con il contenuto o le preferenze personali degli osservatori.

3.1.2 DSCQS e ACR

Il metodo DSCQS (Double Stimulus Continuous Quality Scale) è un approccio di confronto diretto. Gli osservatori visualizzano due versioni dello stesso contenuto: una di riferimento (generalmente priva di distorsioni) e una distorta. A ciascuna versione viene assegnato un punteggio su una scala continua. La differenza tra i punteggi delle due versioni indica il livello di degradazione percepita. Questo metodo è utile per analisi comparative, in cui l'obiettivo è valutare la qualità relativa rispetto a un contenuto ideale

L'ACR (Absolute Category Rating) differisce dal DSCQS perché i partecipanti vedono un solo video alla volta, che viene valutato indipendentemente da una versione di riferimento. Anche in questo caso, il punteggio si basa su una scala predefinita. Questo metodo è più rapido rispetto al DSCQS, ma potrebbe risultare meno preciso, soprattutto quando le differenze di qualità tra i video sono sottili.

3.1.3 Pairwise Comparison

Questo metodo prevede la presentazione simultanea di due video agli osservatori, che devono scegliere quale dei due abbia una qualità superiore. Il Pairwise Comparison è particolarmente utile per valutare differenze

minime nella qualità video, ad esempio quando si testano algoritmi di compressione con parametri simili. Tuttavia, a differenza del DSCQS, il Pairwise Comparison non fornisce una misura assoluta della qualità, ma solo un confronto relativo tra due video. Inoltre, mentre il DSCQS riduce il bias introducendo una valutazione indipendente per ciascun video su una scala continua, il Pairwise Comparison si basa esclusivamente sulla preferenza tra due opzioni, rendendolo meno adatto per studi che richiedono una quantificazione dettagliata della qualità percepita. [17] [18]

3.2 Analisi Oggettive

L'analisi oggettiva della qualità video è un metodo per valutare la qualità di un contenuto senza dover ricorrere a valutazioni soggettive da parte di osservatori umani. Questo tipo di analisi si basa su metriche calcolate automaticamente da algoritmi, permettendo di ottenere risultati quantitativi e riproducibili. È particolarmente utile in ambiti come la compressione video, lo streaming e la trasmissione televisiva, dove è necessario monitorare la qualità in modo costante e su larga scala. Uno dei principali vantaggi dell'analisi oggettiva è la sua coerenza: a differenza delle valutazioni soggettive, che possono variare da persona a persona, le metriche forniscono risultati standardizzati. Questo è essenziale, ad esempio, per confrontare

diversi metodi di codifica o per ottimizzare la qualità di un video in base alla larghezza di banda disponibile. Inoltre, la velocità con cui queste metriche possono essere calcolate consente di applicarle in tempo reale, rendendole ideali per servizi di streaming o trasmissione live. Queste metriche possono essere suddivise in tre categorie principali. Le metriche *Full-Reference* richiedono l'accesso al video originale non degradato per calcolare le differenze rispetto alla versione alterata. Sono le più affidabili ma richiedono l'originale, il che può essere un limite in alcuni scenari. Le metriche *Reduced-Reference*, invece, necessitano solo di informazioni parziali sul video originale, come parametri statistici o feature specifiche, e rappresentano un compromesso tra precisione e disponibilità di dati. Infine, le metriche *No-Reference* analizzano esclusivamente il video degradato, senza alcuna informazione sull'originale. Questo tipo di approccio è utile nei casi in cui il video originale non è disponibile, come nella trasmissione televisiva o nel monitoraggio di qualità dei servizi di streaming in tempo reale. Tuttavia, l'analisi oggettiva presenta anche dei limiti. Le metriche più semplici, come il PSNR, non sempre riflettono accuratamente la qualità percepita dagli utenti. Per questo motivo, sono state sviluppate metriche più avanzate, come il VMAF di Netflix, che combinano diversi fattori per ottenere una stima più realistica della qualità visiva. Un altro aspetto da considerare è che l'efficacia di queste metriche può dipendere molto dal tipo

di contenuto analizzato: un cartone animato, per esempio, può restituire valori molto diversi rispetto a un film d'azione, pur avendo la stessa qualità percepita. Inoltre, alcune metriche richiedono un'elevata potenza computazionale, il che può rappresentare una sfida quando si tratta di implementarle in dispositivi con risorse limitate. Di seguito vengono presentate le metriche che sono state utilizzate nelle analisi dei dataset, insieme ai dettagli tecnici che le caratterizzano

3.2.1 PSNR

Il PSNR (Peak Signal-to-Noise Ratio) è una delle metriche più tradizionali e ampiamente utilizzate. Esso misura il rapporto tra il segnale massimo possibile (intensità del pixel) e il rumore introdotto dalla compressione o da altre forme di degradazione. Si calcola come:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right)$$

Dove:

- MAX è il valore massimo del pixel (es. 255 per immagini a 8 bit).
- MSE (Mean Squared Error) è l'errore quadratico medio tra il video originale e quello distorto.

Il PSNR è semplice e veloce da calcolare, ma ha limiti significativi: non considera la percezione visiva umana e, quindi, può fornire valutazioni inaccurate in termini di qualità percepita. [19]

3.2.2 SSIM

L'SSIM (Structural Similarity Index) migliora rispetto al PSNR includendo aspetti legati alla percezione visiva, come la struttura, la luminanza e il contrasto dell'immagine. L'SSIM è definito come:

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

Dove:

- μ_x, μ_y sono le medie delle intensità dei pixel per le due immagini x e y.
- σ_x, σ_y sono le deviazioni standard per le immagini x e y.
- σ_{xy} è la covarianza tra x e y.
- $C1$ e $C2$ sono costanti di stabilità per evitare la divisione per zero. [20]

3.2.3 VIF

Nella valutazione della qualità di immagini e video, il Visual Information Fidelity (VIF) è una metrica avanzata che quantifica la fedeltà visiva di un

contenuto degradato rispetto all'originale. A differenza di metriche più semplici come PSNR o SSIM, il VIF si concentra sulla quantità di informazione visiva preservata durante il processo di degradazione, adottando un approccio più vicino alla percezione umana. Per calcolarlo, l'immagine viene scomposta in più livelli di dettaglio tramite tecniche di decomposizione multiscala, come le trasformate wavelet. Questo passaggio è cruciale perché la percezione della qualità dipende dalla distribuzione delle informazioni a diverse scale spaziali. Successivamente, si stima quanta parte dell'informazione originale sia ancora presente nella versione degradata. I valori del VIF variano tra 0 e 1, dove 1 indica una perfetta fedeltà e valori più bassi segnalano una perdita crescente di qualità. Questa metrica è particolarmente utile per valutare algoritmi di compressione video (come H.264 o HEVC), monitorare la qualità delle trasmissioni su reti con perdita di pacchetti e ottimizzare sistemi di acquisizione e riproduzione visiva. [21]

3.2.4 CIEDE2000

Il CIEDE2000 è una metrica avanzata per misurare la differenza tra due colori percepiti, progettata per riflettere meglio la sensibilità dell'occhio umano rispetto a modelli più semplici. A differenza di metodi basati sulla

distanza euclidea nello spazio CIE Lab, il CIEDE2000 introduce correzioni per migliorare la precisione della valutazione cromatica.

La formula tiene conto di tre fattori chiave:

- Chiarezza (Lightness): Compensa l'influenza della luminosità sulla percezione del colore.
- Croma (Chroma): Regola la saturazione per evitare distorsioni nella differenza cromatica.
- Rotazione della tinta (Hue Rotation): Corregge variazioni percettive non lineari, specialmente per colori come blu e viola.

Il valore 0 indica colori identici, mentre valori più alti segnalano differenze crescenti. Generalmente:

- $\Delta E < 1 \rightarrow$ differenza impercettibile.
- ΔE tra 1 e 2 $\rightarrow$ differenza appena visibile.
- ΔE tra 2 e 5 $\rightarrow$ differenza chiara.
- $\Delta E > 5 \rightarrow$ colori significativamente diversi.

Nel contesto della qualità video, il CIEDE2000 viene impiegato per analizzare la perdita di fedeltà cromatica nei video compressi, fornendo una misura più accurata rispetto a metriche puramente matematiche come il PSNR. [22]

3.2.5 CAMBI

La metrica CAMBI (Contrast-Aware Multiscale Banding Index) è progettata per valutare il banding, un fenomeno visivo che si verifica quando un gradiente di colore, che dovrebbe essere uniforme, appare suddiviso in bande ben distinte e visibili. Questo problema è particolarmente evidente nei video compressi, dove la riduzione della qualità del colore e la quantizzazione possono creare transizioni nette tra i colori, dando luogo a un aspetto "a bande" anziché a sfumature fluide.

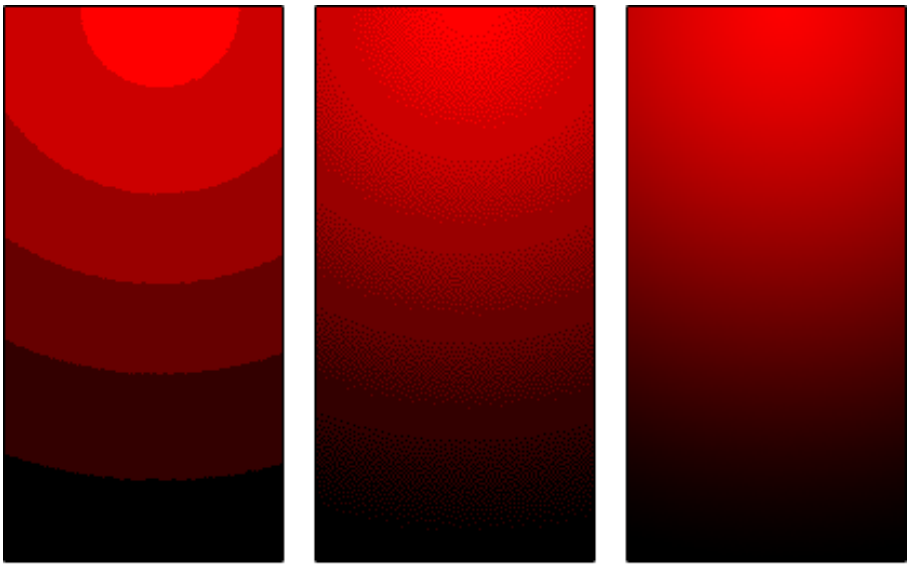

Figura 3.1: Effetto banding [23]

CAMBI analizza la presenza e la gravità del banding nei video attraverso una serie di tecniche che esaminano le variazioni di colore su più livelli di

dettaglio, identificando aree in cui la qualità visiva è compromessa da artefatti legati a una codifica insufficiente dei gradienti di colore.

La metrica è particolarmente utile per la valutazione della qualità in video ad alta compressione, in cui i fenomeni di banding sono più frequenti e visibili. [24]

3.3 VMAF

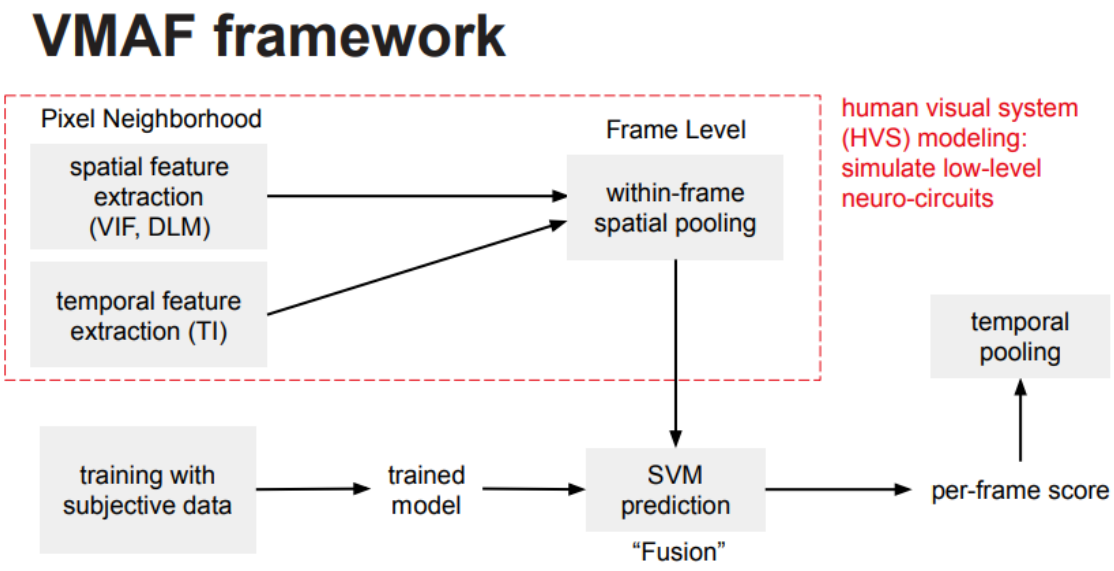

Figura 3.2: Framework VMAF [25]

VMAF (Video Multimethod Assessment Fusion) è un avanzato modello di valutazione della qualità video sviluppato da Netflix, progettato per stimare la qualità visiva percepita confrontando un video originale con una sua versione compressa o distorta. A differenza di metriche tradizionali come PSNR e SSIM che valutano aspetti specifici e isolati della qualità, VMAF adotta un approccio integrato combinando diverse metriche, tra cui VIF, per calcolare un punteggio che si avvicina alla percezione umana. Questo approccio multidimensionale consente una stima più precisa e affidabile della qualità visiva, essenziale per garantire un'esperienza utente ottimale nei sistemi di distribuzione e streaming video.

L'architettura di VMAF si basa su un flusso ben definito che combina analisi spaziali e temporali. Il processo inizia con l'estrazione delle caratteristiche, dove vengono analizzati sia elementi spaziali che temporali. Le caratteristiche spaziali, come dettagli strutturali e contrasto, vengono valutate tramite metriche avanzate quali il DLM (Detail Loss Metric), mentre gli aspetti temporali, fondamentali per rappresentare la fluidità e la coerenza del movimento tra i frame, sono misurati utilizzando indicatori come il TI (Temporal Information). Questo approccio permette di simulare i meccanismi alla base del sistema visivo umano (HVS), riproducendo il modo in cui i dettagli e le distorsioni vengono percepiti a livello neuro cognitivo.

Una volta raccolte queste informazioni, i dati vengono aggregati a livello di singolo frame tramite un'operazione di "within-frame spatial pooling", che sintetizza le caratteristiche più rilevanti per ogni frame. Questi dati vengono poi elaborati mediante un modello di machine learning basato su Support Vector Machine (SVM), progettato per combinare le caratteristiche estratte in modo da generare un punteggio che rispecchia fedelmente la percezione visiva di un osservatore umano. Il modello SVM è addestrato su un vasto dataset soggettivo, composto da giudizi di qualità forniti da osservatori reali, garantendo che la valutazione finale tenga conto delle preferenze umane. Infine, i punteggi per frame vengono integrati tramite un pooling temporale,

che calcola un punteggio complessivo per il video considerando sia la qualità globale che la coerenza temporale.

VMAF si distingue per la sua capacità di ottimizzare l'intero processo di codifica video. Questo lo rende uno strumento indispensabile per il monitoraggio e la valutazione delle prestazioni di codec e tecniche di compressione. La possibilità di confrontare l'efficacia di diverse configurazioni garantisce che i sistemi di streaming possano offrire video di alta qualità anche in condizioni di rete variabili. Inoltre, la sua architettura flessibile e modulare lo rende particolarmente adatto per la ricerca e lo sviluppo, favorendo l'innovazione nel campo della compressione video e delle tecnologie multimediali.

Durante il lavoro di tesi sono state analizzate nove varianti di VMAF, tutte basate sullo stesso principio fondamentale di valutazione della qualità video attraverso un approccio integrato che combina diverse metriche per simulare la percezione umana. Ciò che accomuna queste versioni è l'intento di misurare in modo affidabile le qualità visive di un video, sfruttando modelli addestrati su dati soggettivi. Le differenze tra le varianti riguardano principalmente tre aspetti: il tipo di aritmetica utilizzata (punto fisso contro virgola mobile), la gestione dei valori negativi e l'ottimizzazione per contenuti ad altissima risoluzione (4K), nonché gli aggiornamenti apportati in alcune versioni (come nel ramo "b").

La versione di riferimento, *vmaf_v0.6.1*, utilizza l'aritmetica a punto fisso e rappresenta il modello standard. Da questa derivano varianti come la *vmaf_v0.6.1neg*, che integra una logica per gestire i valori negativi emersi in particolari condizioni di degradazione video. In questo modo, pur mantenendo la stessa struttura di base, il modello "neg" riesce a riflettere in modo più accurato la percezione umana nelle situazioni limite. Un ulteriore gruppo di varianti introduce l'uso dell'aritmetica in virgola mobile per incrementare la precisione dei calcoli. Così, la *vmaf_float_v0.6.1* rispecchia la versione base, ma con una maggiore accuratezza numerica. Unendo anche il trattamento dei valori negativi, si ottiene la *vmaf_float_v0.6.1neg*, che beneficia sia della precisione della virgola mobile sia della gestione più fine delle anomalie, migliorando ulteriormente la correlazione con le percezioni soggettive. Un ulteriore sviluppo è rappresentato dalle versioni *vmaf_b_v0.6.3* e *vmaf_float_b_v0.6.3*. Queste varianti, rispettivamente a punto fisso e in virgola mobile, derivano da un ramo aggiornato del modello (indicabile con il suffisso "b"), che incorpora affinamenti basati su dataset più recenti e miglioramenti nell'algoritmo. Il risultato è una maggiore robustezza e versatilità nella valutazione, particolarmente utile in contesti operativi variegati. Infine, dato il crescente utilizzo dei contenuti in risoluzione 4K, sono state sviluppate versioni specifiche per questo ambito. La *vmaf_4k_v0.6.1* è ottimizzata per il 4K

mantenendo l'aritmetica a punto fisso, mentre la *vmaf_float_4k_v0.6.1* sfrutta la precisione del calcolo in virgola mobile per gestire in maniera più accurata la maggiore densità di pixel. Inoltre, la *vmaf_4k_v0.6.1neg* unisce l'approccio della versione 4K a punto fisso con la logica di gestione dei valori negativi, garantendo una stima della qualità che risponde efficacemente anche alle situazioni di forte degradazione.

Questa pluralità di versioni ha permesso di valutare in modo approfondito durante le analisi come ciascun approccio possa rispondere alle diverse esigenze, migliorando progressivamente l'accuratezza e l'affidabilità della misurazione della qualità video.[26]

Capitolo 4

Ambiente di riproducibilità

In questo capitolo si affronta il concetto di ambiente di riproducibilità, analizzando le infrastrutture e gli strumenti fondamentali per garantire l'esecuzione consistente e verificabile di esperimenti computazionali. In particolare, verranno esaminati in modo approfondito gli ambienti HPC (High Performance Computing) e l'utilizzo di Singularity, una piattaforma di containerizzazione sviluppata per il settore scientifico.

4.1 High Performance Computing e job

L'HPC è un insieme di tecnologie e metodologie che consente di eseguire calcoli complessi e intensivi utilizzando sistemi di elaborazione ad alte prestazioni, come supercomputer e cluster di computer interconnessi. Questo approccio si basa sulla distribuzione parallela dei carichi di lavoro, utilizzando CPU e GPU in configurazioni scalabili, che possono variare da

poche unità di calcolo a migliaia. L'HPC permette di processare dati e simulare fenomeni su una scala altrimenti irraggiungibile con i tradizionali sistemi di calcolo. È largamente utilizzato per simulazioni fisiche, modellazioni chimiche, analisi di dati climatici e molte altre applicazioni critiche. Ciò è reso possibile grazie a un'infrastruttura che combina architetture computazionali avanzate, reti di interconnessione ad alta velocità e sistemi di memoria e storage distribuiti. Nel contesto dell'HPC, un *job* è un'unità di lavoro che viene inviata al sistema per essere processata. I job rappresentano attività specifiche che includono il codice da eseguire, i parametri di configurazione e le risorse necessarie, come il numero di CPU, GPU, memoria e tempo di esecuzione richiesto.

I job sono gestiti tramite un *job scheduler*, come SLURM (Simple Linux Utility for Resource Management), che si occupa di allocare le risorse necessarie e monitorare lo stato dei job in esecuzione. Gli utenti interagiscono con il sistema di calcolo inviando job tramite file di script, solitamente con estensione .sbatch. Tali script contengono le istruzioni per il job, come il programma da eseguire e i requisiti di risorse.

```
#!/bin/bash
#SBATCH --job-name=simple_python_job      # Nome del job
#SBATCH --output=output_%j.log            # File di output, %j rappresenta l'ID del job
#SBATCH --time=00:10:00                   # Tempo massimo di esecuzione (hh:mm:ss)
#SBATCH --ntasks=1                        # Numero di task (processi) da eseguire
#SBATCH --mem=1G                          # Memoria richiesta (esempio: 1 GB)

# Carica i moduli necessari (se richiesto)
module load intel/python/3/2019.4.088     # Esempio di caricamento di un modulo Python

# Comando per eseguire lo script Python
python3 my_script.py
```

Figura 4.1: Esempio di file .sbatch per eseguire uno script Python

Nel contesto del presente lavoro di tesi, è stato utilizzato il sistema di calcolo ad alte prestazioni (HPC) *Legion*, messo a disposizione dal HPC@POLITO.[27]

4.2 Singularity

In questo contesto, *Singularity* si distingue come una piattaforma di containerizzazione appositamente progettata per gli ambienti HPC. Singularity offre un metodo sicuro ed efficiente per creare, distribuire ed eseguire applicazioni in ambienti isolati e altamente riproducibili, garantendo al contempo la compatibilità con i sistemi multiutente e con le risorse computazionali avanzate che caratterizzano i cluster HPC.

Una delle caratteristiche principali di Singularity è la sua capacità di confezionare applicazioni e ambienti software in un unico file immagine. Questo processo inizia con la creazione di un file di definizione, che rappresenta il cuore della costruzione del container. Il file di definizione, scritto in formato testo, specifica dettagliatamente le configurazioni

necessarie per creare l'immagine container. Tra le informazioni incluse nel file di definizione vi sono l'immagine base del sistema operativo (ad esempio, una distribuzione Linux), le dipendenze richieste (come librerie e pacchetti software), e i comandi di configurazione necessari per predisporre l'ambiente di esecuzione.

```
Bootstrap: library
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${SINGULARITY_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $SINGULARITY_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author d@sylabs.io
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

Figura 4.2: Esempio di costruzione file .def [28]

Una volta creato il file di definizione, si procede alla costruzione del container utilizzando il comando singularity build. Questo comando genera un file immagine in formato *.sif* (Singularity Image File), che contiene l'intero ambiente di esecuzione. Il formato .sif è altamente compatto, sicuro e portatile, poiché racchiude tutto il necessario per l'esecuzione del software, comprese librerie, configurazioni e binari. Grazie a questa struttura, l'immagine può essere trasferita e utilizzata su diverse piattaforme senza richiedere ulteriori configurazioni, garantendo un ambiente identico a prescindere dal sistema sottostante.

Un aspetto distintivo del processo di costruzione di immagini in Singularity è la possibilità di integrare le configurazioni di sistema e le risorse necessarie in modo modulare. Durante la costruzione, è possibile definire script di installazione che vengono eseguiti automaticamente all'interno del container per installare pacchetti o configurare ambienti specifici. Questo approccio consente di ottimizzare l'immagine per applicazioni specifiche, riducendo al minimo le dimensioni del file e mantenendo l'efficienza.

Una volta costruito il container, esso può essere eseguito attraverso i comandi *singularity run* o *singularity exec*. Il comando run avvia il processo principale specificato durante la costruzione del container, mentre il comando exec consente di eseguire comandi arbitrari all'interno

dell’ambiente del container. Questo offre grande flessibilità, permettendo di accedere alle funzionalità del container in modo selettivo.

Singularity garantisce un'integrazione nativa con i sistemi HPC, consentendo un accesso trasparente alle risorse di calcolo avanzate, come GPU e reti ad alte prestazioni. Inoltre, poiché esegue i container come processi utente senza richiedere privilegi di root, Singularity si adatta perfettamente agli ambienti multiutente, rispettando i requisiti di sicurezza richiesti dai cluster di supercomputer. Questa caratteristica elimina gran parte dei rischi associati alla containerizzazione tradizionale, permettendo agli utenti di eseguire applicazioni in modo sicuro e senza interferire con altre attività in esecuzione sul sistema.

L'adozione di Singularity negli ambienti HPC ha rivoluzionato il modo in cui le applicazioni e gli ambienti software vengono confezionati ed eseguiti, offrendo agli utenti un potente strumento per garantire la riproducibilità, la portabilità e l'efficienza nell’utilizzo delle risorse computazionali. Grazie alla semplicità del processo di costruzione delle immagini e alla sua compatibilità con le infrastrutture HPC, Singularity si è affermato come uno standard de facto per la containerizzazione in ambiti scientifici e industriali avanzati. [29]

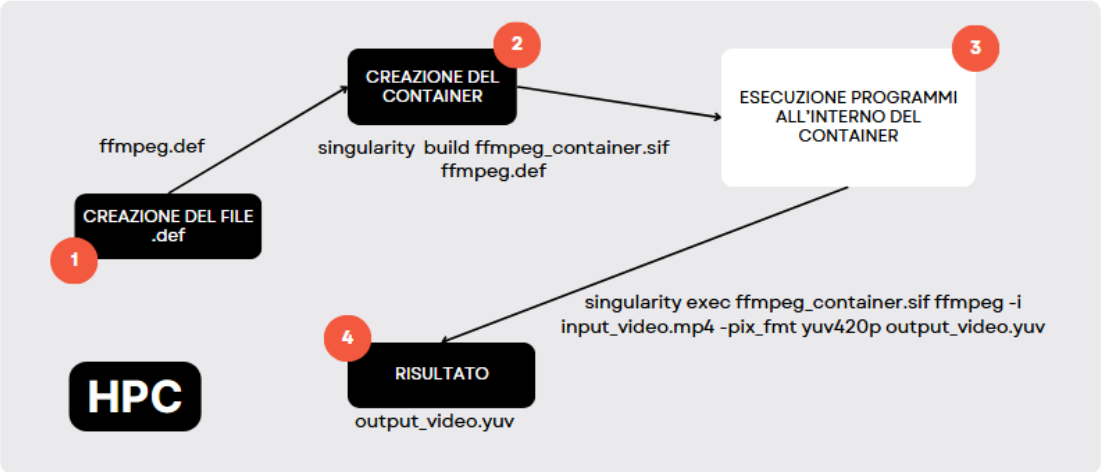

Figura 4.3: Esempio di FFmpeg con Singularity per la conversione di un file MP4 in YUV

singularity exec ffmpeg_container.sif ffmpeg -i input_video.mp4 -pix_fmt yuv420p output_video.yuv

con:

- *singularity exec /ffmpeg_container.sif* esegue FFmpeg all'interno di un conteiner Singularity
- *ffmpeg* è il programma da eseguire all'interno del contenitore.
- *-i input_video.mp4* indica il file di input (in questo caso un video MP4).
- *-pix_fmt yuv420p* imposta il formato di pixel su YUV 4:2:0
- *output_video.yuv* è il file di output in formato YUV

Capitolo 5

Strategie di Sviluppo

Le analisi condotte sono state organizzate secondo il seguente ordine operativo:

1. Creazione e costruzione delle immagini per i container Singularity, configurati rispettivamente per l'utilizzo degli strumenti *ffmpeg* e VMAF;
2. Preparazione del dataset, con suddivisione dei file originali e compressi in directory dedicate;
3. Esecuzione di uno script finalizzato alla generazione e al deployment dei file .sbatch, in modo tale che ogni job inviato all'HPC effettui l'analisi su una singola coppia di file originale/compresso;
4. All'interno di ciascun job, esecuzione di uno script .sh che, in base alla natura del dataset, può includere una conversione preliminare del

video distorto in un formato non compresso compatibile con VMAF, utilizzando il container singularity *ffmpeg*;

5. Esecuzione dell'analisi VMAF all'interno dello stesso job;
6. Al termine dei job, raccolta dei risultati dai file .json generati da VMAF e dai file .log. Da tali file sono stati estratti i valori medi delle metriche principali, insieme ai tempi di esecuzione e alla memoria utilizzata, organizzandoli in un file .csv per garantire un formato standard di consultazione

5.1 Datasets e riproducibilità

L'ambiente utilizzato per lo sviluppo e la scrittura degli script in locale è stato configurato su una macchina virtuale, eseguita tramite VirtualBox. Il sistema operativo installato è Ubuntu 20.04.6 LTS, con architettura a 64 bit. La macchina virtuale è stata configurata con una memoria RAM di 4096 MB, al fine di garantire prestazioni adeguate all'esecuzione delle operazioni richieste durante le fasi di testing preliminari all'uso sull' HPC.

La selezione dei dataset è stata effettuata in accordo con il relatore di tesi, al fine di garantire una scelta mirata e coerente con gli obiettivi della ricerca.

Il *GamingVideoSET* e il *Kingston University Gaming Video Dataset (KUGVD)* comprendono video non compressi della durata di 30 secondi, con risoluzione 1080p e 30 fps, rappresentativi di diversi giochi popolari. Questi video sono stati codificati utilizzando il codec H.264 in diverse combinazioni di risoluzione e bitrate, generando numerose sequenze video distorte. Il GamingVideoSET include 24 video rappresentativi di 12 giochi popolari, mentre il KUGVD comprende 6 video di giochi ad alta qualità. [30] [31] Tra i dataset selezionati figura anche una coppia di raccolte sviluppate in collaborazione con l'Institute for Telecommunication Sciences (ITS), parte della National Telecommunications and Information Administration (NTIA) degli Stati Uniti. Si tratta dei dataset *ITS4S* e *AGH-NTIA-Dolby*, entrambi accomunati dall'obiettivo di valutare sistemi di streaming e metriche di qualità video attraverso sequenze con scene non ripetute, una caratteristica pensata per simulare meglio le condizioni reali di fruizione e supportare nuovi disegni sperimentali. L'ITS4S è stato progettato per studiare sistemi di streaming adattivo su dispositivi mobili ad alta definizione. I video originali, della durata di 4 secondi ciascuno, sono stati convertiti a 720p e 24 fps, uno standard frequentemente adottato per favorire compatibilità e coerenza nelle valutazioni metriche. Il dataset AGH/NTIA/Dolby, frutto della collaborazione tra AGH University of Science and Technology, NTIA/ITS e Dolby, include sequenze codificate con MPEG-2, H.264 e

H.265 (HEVC) della durata di 8 secondi. [32] [33] Infine, l'*AVT-VQDB-UHD-1*, sviluppato dal gruppo Audiovisual Technology (AVT) dell'Università Tecnica di Ilmenau (TU Ilmenau), è un ampio database di qualità video per contenuti 4K. Comprende video sorgente codificati con H.264, HEVC e VP9. Le risoluzioni variano da 360p a 2160p, con framerate da 15 fps a 60 fps. [34]

Viene qui presentata una tabella riassuntiva che illustra le caratteristiche principali di ciascuna sequenza.

Nome	N. sequenze originali	Risoluzione originale	Framerate	Durata	Codec	N. sequenze compresse	Risoluzioni/ Formati
KUGVD	6	480p,720p, 1080p	30 fps	30 sec	H264	144	480p,720p,1080p/ MP4
ITS4S	514	720p	24 fps	4 sec	H264	514	Fino al 720p/Y4M
GamingVideoSET	24	1080p	30 fps	30 sec	H264	576	480p,720p,1080p/ MP4
AGH-NTIA-Dolby	84	720p	60 fps	8 sec	MPEG-2,H264, H265	201	720p/Y4M
AVT-VQDB-UHD-1	12	2160p (1 a 2250p)	60 fps	8-10 sec	H264, H265, VP9	432	Dal 360p fino al 2160p/MP4,MKV

Per garantire la riproducibilità e la parallelizzazione dell'esecuzione sull'HPC, è stato sviluppato uno script che automatizza l'analisi della qualità. Questo script sfrutta il sistema di gestione delle risorse SLURM per sottomettere job in modo efficiente, consentendo l'analisi simultanea di più coppie di video originali e distorti.

L'utente fornisce in input le directory contenenti i file video originali (.yuv/.y4m) e distorti (.mp4), la cartella di output per i risultati e il modello VMAF da utilizzare. È inoltre possibile specificare funzionalità VMAF aggiuntive tramite un parametro opzionale.

Il flusso di esecuzione prevede:

1. L'individuazione automatica delle corrispondenze tra i file originali e distorti.
2. La generazione e la sottomissione di job SLURM per eseguire l'analisi di ciascuna coppia tramite uno script dedicato.
3. Il salvataggio di una lista associativa tra job ID e file di log in un file di testo, permettendo di recuperare in seguito il consumo di CPU e memoria utilizzata da ciascun job.

Questo approccio permette di scalare facilmente le analisi su un gran numero di file, ottimizzando le risorse disponibili e garantendo risultati riproducibili. La scelta relativa alla durata e alla memoria utilizzata dal job è stata effettuata

in base alla natura del dataset, in particolare per quanto riguarda la risoluzione e la durata della singola sequenza.

```
#!/bin/bash
#SBATCH --job-name=cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1
#SBATCH --output=results/AVT-VQDB-UHD-1/test_1/vmaf_v0.6.1/cutting_orange_tuil_4
0000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log
#SBATCH --ntasks=1
#SBATCH --mem=1536M
#SBATCH --time=02:30:00

module load intel/python/3/2019.4.088
module load singularity/3.2.1

# Record start time
start_time=$(date +'%s')

echo "Job started at: $(date)" >> results/AVT-VQDB-UHD-1/test_1/vmaf_v0.6.1/cutt
ing_orange_tuil_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log
echo "Job ID: $SLURM_JOB_ID" >> results/AVT-VQDB-UHD-1/test_1/vmaf_v0.6.1/cuttin
g_orange_tuil_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log

# Call the generic script to process the single file
bash ~/scripts/AVT-VQDB-UHD-1_scripts/analysis.sh "datasets/AVT-VQDB-UHD-1/src_v
ideos/cutting_orange_tuil.yuv" "datasets/AVT-VQDB-UHD-1/distorted/test_1/segment
s//cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc.mp4" "results/AVT-VQDB-UHD-
1/test_1/vmaf_v0.6.1" "vmaf_v0.6.1" --feature cambi float_ssim psnr float_ms_ssi
m ciede psnr_hvs >> results/AVT-VQDB-UHD-1/test_1/vmaf_v0.6.1/cutting_orange_tui
l_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log 2>&1

# Record end time
end_time=$(date +'%s')

# Calculate and print execution duration
duration=$((end_time - start_time))

echo "Job finished at: $(date)" >> results/AVT-VQDB-UHD-1/test_1/vmaf_v0.6.1/cut
ting_orange_tuil_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log
echo "Total execution time: $duration seconds" >> results/AVT-VQDB-UHD-1/test_1/
vmaf_v0.6.1/cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc_vmaf_v0.6.1.log
```

Figura 5.1: Job creato per l'analisi della sequenza

cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc.mp4

Per tutte le analisi è stata scelta una versione specifica di VMAF e FFmpeg per la creazione dei container,rispettivamente la 3.0.0 e la 5.1.6.

Questa scelta è stata effettuata con l'obiettivo di garantire coerenza e confrontabilità tra i risultati ottenuti. L'utilizzo di versioni standardizzate degli strumenti riduce i potenziali effetti di variabilità derivanti da aggiornamenti software o cambiamenti nelle librerie dipendenti, rendendo i risultati delle analisi replicabili nel tempo.

Vengono di seguito mostrati i due file .def dai quali sono stati creati i file .sif.

```
Bootstrap: docker
From: debian:stable

%runscript
    #echo "This is what happens when you run the container..."
    #pwd
    #echo $@
    #cd /vmaf/libvmaf/build/tools
    #./vmaf $@
    /vmaf/libvmaf/build/tools/vmaf $@

%environment
    export LC_ALL=C

#%files
#    . /vmaf

%post
    echo "Hello from inside the container"
    apt-get update && \
    apt-get install -y --no-install-recommends \
    build-essential \
    git \
    meson \
    ninja-build \
    nasm \
    xxd \
    ca-certificates \
    && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists
    git clone https://github.com/Netflix/vmaf.git
    cd vmaf
    git checkout tags/v3.0.0
    meson setup libvmaf/build libvmaf --buildtype release -Denable_float=true && \
    ninja -vC libvmaf/build
    pwd
    ls -l libvmaf/build/tools/vmaf
```

Figura 5.2: vmaf300.def

```
Bootstrap: docker
From: debian:stable

%runscript
    ffmpeg $@

%environment
    export LC_ALL=C

%post
    echo "Installing dependencies..."
    apt-get update && \
    apt-get install -y --no-install-recommends \
    autoconf \
    automake \
    build-essential \
    cmake \
    git \
    libass-dev \
    libfreetype6-dev \
    libtool \
    libvorbis-dev \
    libvpx-dev \
    pkg-config \
    texinfo \
    wget \
    yasm \
    zlib1g-dev \
    libx264-dev \
    libx265-dev \
    libnuma-dev \
    coreutils \
    && apt-get clean && rm -rf /var/lib/apt/lists/*

    echo "Downloading and building FFmpeg 5.1.6..."
    mkdir -p /opt/ffmpeg && cd /opt/ffmpeg
    wget --no-check-certificate -O ffmpeg-5.1.6.tar.gz https://ffmpeg.org/releases/ffmpeg-5.1.6.tar.gz
    tar xzf ffmpeg-5.1.6.tar.gz && cd ffmpeg-5.1.6

    ./configure --prefix=/usr/local \
                --enable-gpl \
                --enable-libass \
                --enable-libfreetype \
                --enable-libvorbis \
                --enable-libvpx \
                --enable-libx264 \
                --enable-libx265 \
                --enable-nonfree

    make -j$(nproc) && make install

    echo "FFmpeg 5.1.6 installation complete!"

    cd / && rm -rf /opt/ffmpeg

%test
    ffmpeg -version
```

Figura 5.3: ffmpeg.def

5.2 Preparazione, conversione e analisi dei file video

Per ottenere una valutazione affidabile della qualità video, è fondamentale partire da dati non alterati che preservino ogni dettaglio visivo senza compromissioni. L'utilizzo di file raw è essenziale per l'analisi condotta da

VMAF poiché garantisce l'integrità dei dati video. In particolare, i file raw, come quelli in formato YUV/Y4M, mantengono intatte tutte le informazioni pixel-per-pixel, consentendo così a VMAF di confrontare in maniera precisa e accurata ogni dettaglio tra il video di riferimento e quello distorto. Utilizzare formati compressi, ad esempio MP4, comporta il rischio che il processo di compressione introduca artefatti e alterazioni che possono confondere l'analisi, rendendo difficile distinguere tra le imperfezioni causate dalla compressione e quelle realmente presenti nel materiale video. Inoltre, la natura non elaborata dei file raw assicura che la valutazione si basi su dati omogenei, privi di ulteriori modifiche derivanti da codec o da processi di decodifica, migliorando così la coerenza e l'affidabilità della misurazione VMAF.

È altresì importante sottolineare che, qualora vi siano differenze tra il video originale e quello compresso in termini di risoluzione, bitrate, framerate o bit-depth, è necessario apportare le opportune operazioni di normalizzazione. Tali operazioni possono includere il ridimensionamento del video per uniformare la risoluzione, l'adeguamento del bitrate per mantenere livelli di compressione comparabili, l'allineamento del framerate e la corrispondenza della profondità di bit. Solo in questo modo si garantisce che l'analisi di VMAF si basi su dati comparabili e non influenzati da eventuali discrepanze tecniche, consentendo così una valutazione accurata e

valida della qualità video. In alcuni casi, ad esempio quando nella directory dei file sorgenti originali del dataset erano presenti file .yuv con una nomenclatura descrittiva delle caratteristiche principali della sequenza, è stato sufficiente estrarle direttamente da essa. In altri dataset, come AVT-VQDB-UHD-1, i nomi dei file non riportavano le informazioni fondamentali, ma è stato possibile ricavarle dal corrispondente file .mp4, utilizzando FFmpeg per l'estrazione dei dati.

Dopo aver creato il file distorto riadattato si procede con l'analisi VMAF vera e propria. Anche in questo caso a seconda del formato dei file originali l'analisi cambia leggermente in termini di argomenti da passare allo script. Infatti, nel caso in cui i file originali presentino formato YUV, non contenendo intestazione, è necessario, affichè l'analisi venga effettuata correttamente, passare al comando VMAF informazioni aggiuntive, quali risoluzione, bit-depth e formato del sottocampionamento cromatico.

A complemento dei parametri tecnici relativi a risoluzione, bit-depth e sottocampionamento cromatico, il comando integra una serie di feature aggiuntive per affinare l'analisi VMAF. In particolare, la feature *cambi* evidenzia variazioni locali e differenze cromatiche che il semplice confronto segnale/rumore potrebbe non rilevare. Le metriche *float_ssim* e *float_ms_ssim* offrono una valutazione dettagliata della similarità strutturale, operando rispettivamente a scala singola e multipla per cogliere degradazioni

anche su differenti livelli di dettaglio. Ad essi si affianca *ciede*, che misura con precisione le variazioni di colore secondo lo standard CIEDE2000, mentre *psnr* e *psnr_hvs* forniscono, rispettivamente, un'analisi classica del rapporto segnale/rumore e una versione raffinata che tiene conto delle caratteristiche del sistema visivo umano. Queste feature, impiegate in sinergia, permettono di ottenere una valutazione complessiva della qualità video che rispecchia in modo più accurato la percezione soggettiva.

Un esempio del comando è il seguente:

singularity run vmaf300.sif -w 3840 -h 2160 -p 422 -b 10

-r cutting_orange_tuil.yuv

-d cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc.yuv

- o cutting_orange_tuil_40000kbps_2160p_59.94fps_hevc__10sec__vmaf_v0.6.1.json

--json -m version=vmaf_v0.6.1

--feature cambi --feature float_ssim --feature psnr

--feature float_ms_ssim --feature ciede --feature psnr_hvs

- *singularity run vmaf300.sif*: Esegue l'immagine Singularity vmaf300.sif, che contiene l'applicazione VMAF.
- *-w -h -p -b* : Imposta rispettivamente la larghezza e l'altezza in pixel del video, il formato dei pixel con subsampling cromatico ed il bit-depth.

- *-r*: Indica il file video originale (di riferimento).
- *-d*: Indica il file video distorto da analizzare.
- *-o*: Specifica il file di output per i risultati dell'analisi.
- *--json*: Genera l'output in formato JSON.
- *-m version*: Specifica la versione del modello VMAF da utilizzare.
- *--feature*: opzionale le feature aggiuntive che si vogliono misurare

Nel caso in cui invece il formato sia Y4M, VMAF è in grado di estrarre direttamente le informazioni principali (come risoluzione, bit-depth e formato del subsampling cromatico) direttamente dall'header del file. In questo modo, non sarà necessario specificare manualmente questi parametri, semplificando notevolmente il comando.

Una volta eseguite tutte le analisi per un dato dataset, sono stati raccolti i valori di consumo di CPU e memoria utilizzata nei rispettivi file di log, atttraverso il comando SLURM *seff*. Infine, sono stati scaricati i risultati in ambiente locale e sono stati raccolti i valori più significativi, attraverso lo script python aggregate_csv.

Questo script è progettato per analizzare una cartella contenente file JSON e log relativi alle metriche VMAF, estraendo le informazioni rilevanti da ciascun file e generando un file CSV riepilogativo. Inizialmente, lo script legge i file JSON per recuperare le metriche "pooled_metrics" e calcolare il valore medio di ciascuna metrica. Successivamente, analizza i file log

utilizzando espressioni regolari per estrarre informazioni sul tempo di utilizzo della CPU (in secondi) e sulla memoria utilizzata, la quale viene convertita in MB. Per quanto riguarda i nomi dei file, lo script sfrutta una regex per ottenere dettagli utili come il nome della sequenza, il bitrate, la risoluzione, il frame rate, il codec, la durata e la versione del modello VMAF. Una volta estratti questi dati, lo script associa il file JSON al corrispondente file log, estraendo, se disponibile, anche le informazioni sulla CPU e la memoria. Tutte le informazioni raccolte vengono quindi aggregate in un dizionario, che viene successivamente scritto in un file CSV. Il CSV include le intestazioni principali in un ordine preciso, partendo dai dettagli del video (nome della sequenza, bitrate, risoluzione, frame rate, codec e durata), seguito dal modello VMAF utilizzato e dai dati sulle risorse computazionali impiegate, come il tempo di CPU e la memoria. Infine, vengono aggiunte in ordine alfabetico le metriche VMAF estratte dai file JSON, con i relativi valori medi calcolati.

Capitolo 6

Sintesi e valutazione dei risultati

In questo paragrafo si procederà a un riepilogo e a una valutazione delle principali analisi effettuate sui dataset di riferimento. Verrà fornita una panoramica dei risultati ottenuti, mettendo in evidenza le caratteristiche più rilevanti emerse durante le varie fasi di studio. A supporto di questa sintesi, saranno presentati grafici esplicativi costruiti a partire dai dati memorizzati nei file .CSV generati al termine di ogni singola analisi, che permetteranno di visualizzare le tendenze e i comportamenti osservati.

6.1 VMAF e PSNR a confronto

Questo paragrafo si propone di analizzare e confrontare la relazione tra VMAF e Psnr_y sui dataset ITS4S (figure 6.1 e 6.2), KUGVD (figure 6.3 e 6.4), AVT-VQDB-UHD-1 (figure 6.5 e 6.6) e GamingVideoSET (figure 6.7

e 6.8), impiegando due specifiche versioni del modello VMAF, la v0.6.1 e la 4k_v0.6.1.

Gli scatterplot presentano sull'asse x il valore medio del vmaf , sull'asse y il valor medio del Psnr_y. Ogni punto su questi grafici corrisponde a una singola sequenza video. Per arricchire l'analisi, i punti sono visualizzati con colori che indicano diverse categorie di bitrate e con simboli che rappresentano le risoluzioni dei video distorti. In tutti i grafici analizzati emerge una chiara correlazione positiva tra VMAF e Psnr_y: all'aumentare di Psnr_y, anche i punteggi VMAF crescono, indicando una qualità percepita migliore. Questo è atteso, poiché VMAF integra anche metriche di fedeltà al segnale. Inoltre, video con bitrate e risoluzioni più alti si collocano in alto a destra nei grafici (qualità migliore), mentre quelli con valori inferiori si trovano in basso a sinistra, confermando che più dati e pixel portano in genere a una qualità superiore. Le principali differenze tra i grafici derivano dalle caratteristiche intrinseche dei dataset, unite alla differente valutazione dei due modelli, soprattutto a bassi bitrate e risoluzioni. Infatti, se nella zona in alto a destra (VMAF e Psnr_y alti) generalmente i due modelli sono quasi sovrapponibili, con qualche valore poco più alto per il modello 4k, scendendo sempre più su risoluzioni e bitrate inferiori, i risultati dati dal modello v0.6.1 risultano sensibilmente più bassi. Suggerisce che il modello 4k_v0.6.1, pur essendo tarato per risoluzioni superiori e distanze di visione

più ravvicinate, potrebbe avere una calibrazione che lo porta ad assegnare punteggi VMAF leggermente più generosi anche a risoluzioni inferiori ed una diversa sensibilità a specifici artefatti che si traduce in un punteggio complessivamente più ottimistico.

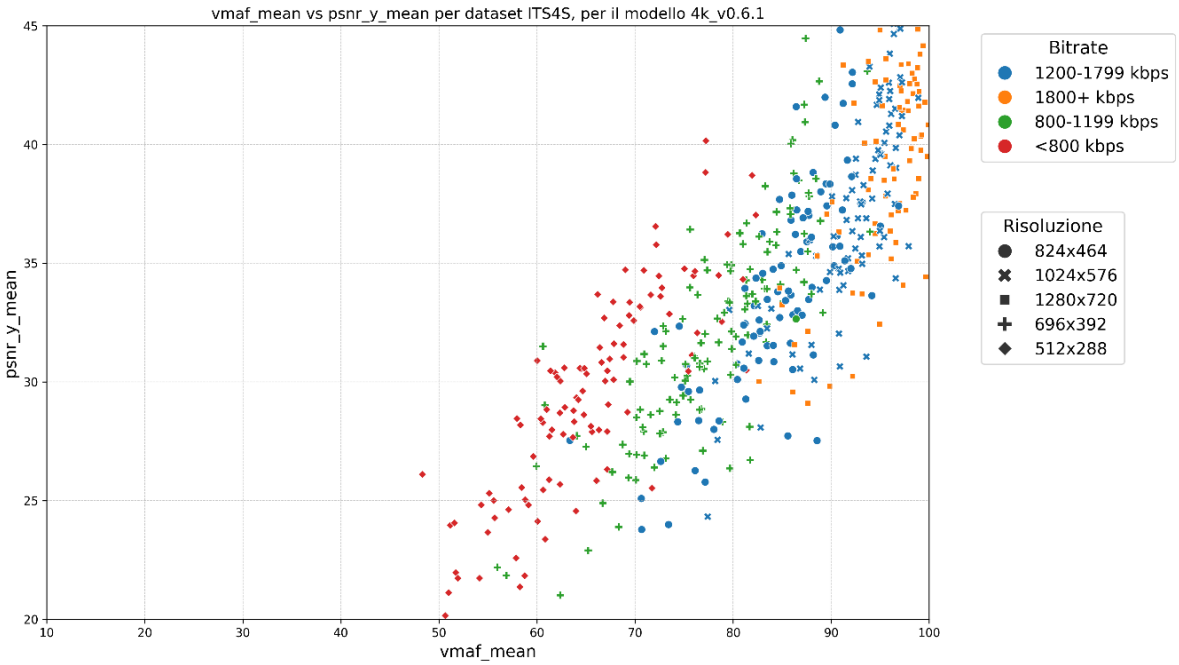

Figura 6.1: VMAF vs Psnr_y per dataset ITS4S, modello 4k_v0.6.1

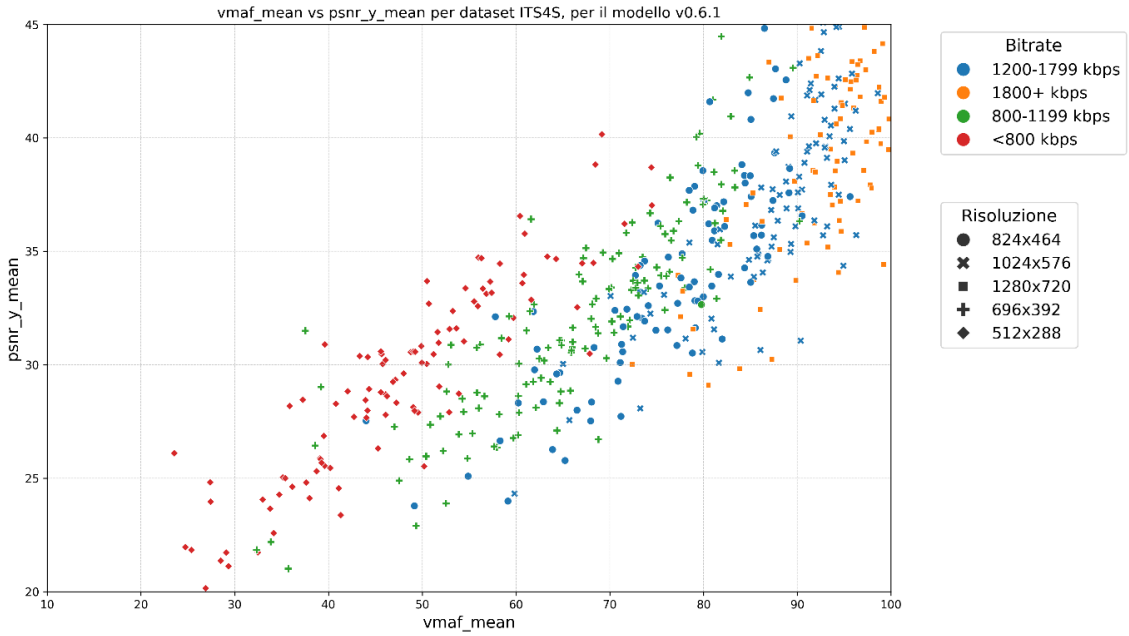

Figura 6.2: VMAF vs Psnr_y per dataset ITS4S, modello v0.6.1

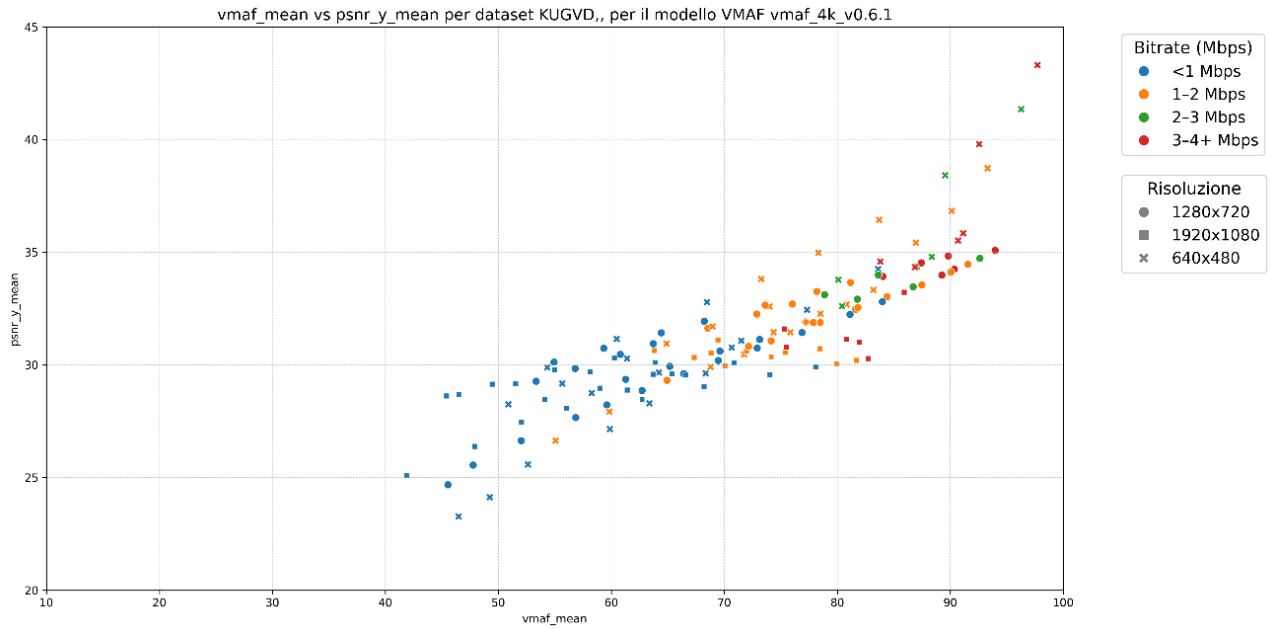

Figura 6.3: VMAF vs Psnr_y per dataset KUGVD, modello 4k_v0.6.1

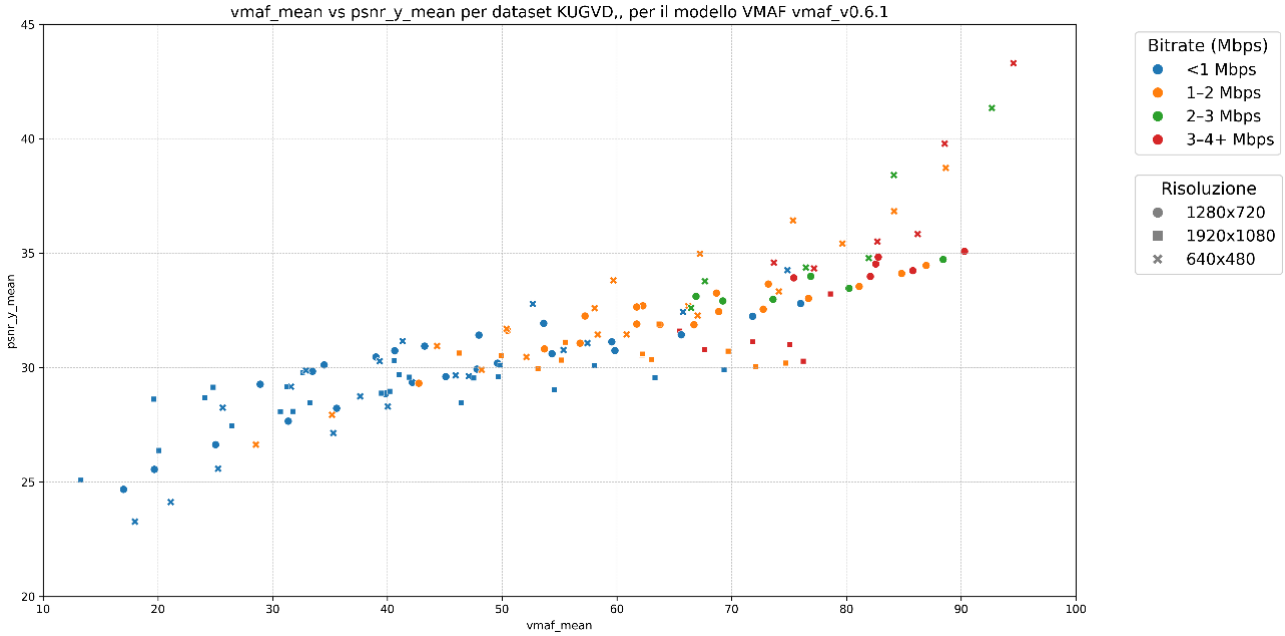

Figura 6.4: VMAF vs Psnr_y per dataset KUGVD, modello v0.6.1

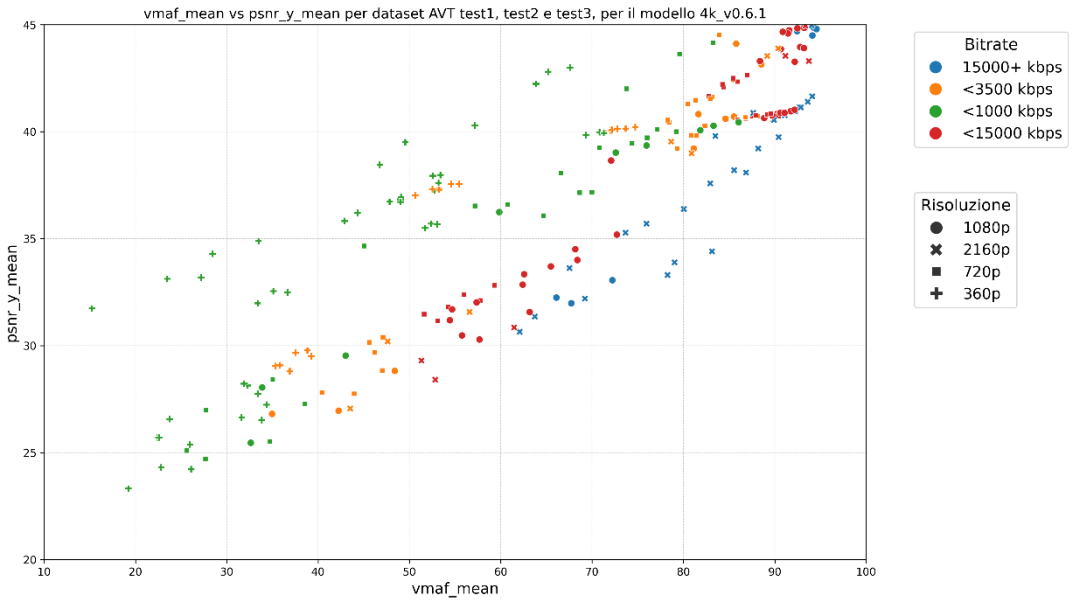

Figura 6.5: VMAF vs Psnr_y per dataset AVT, modello 4k_v0.6.1

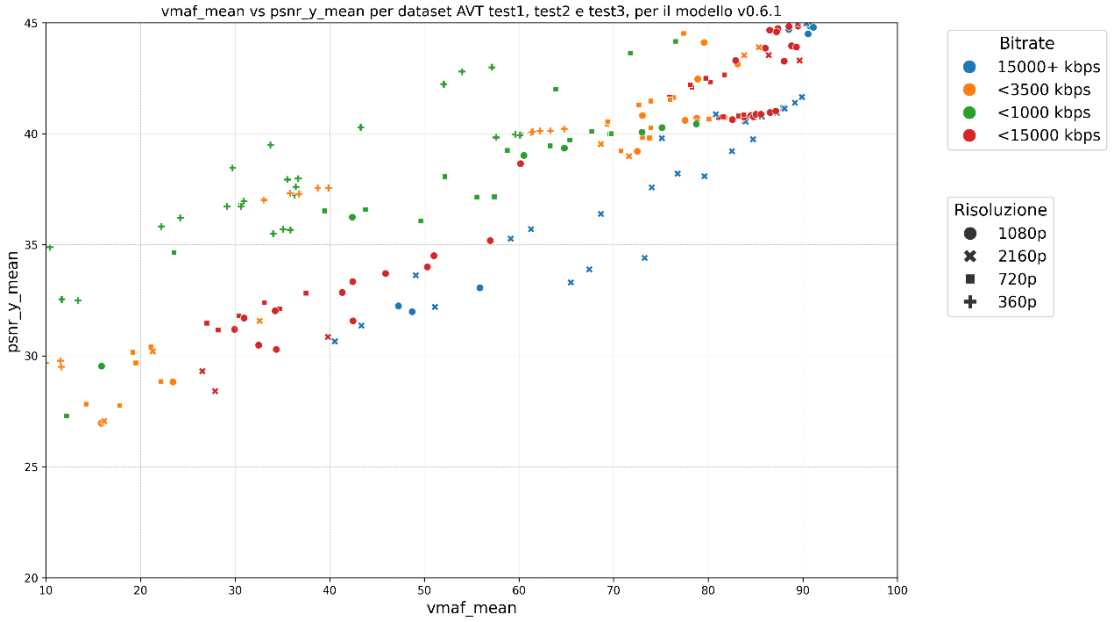

Figura 6.6: VMAF vs Psnr_y per dataset AVT, modello v0.6.1

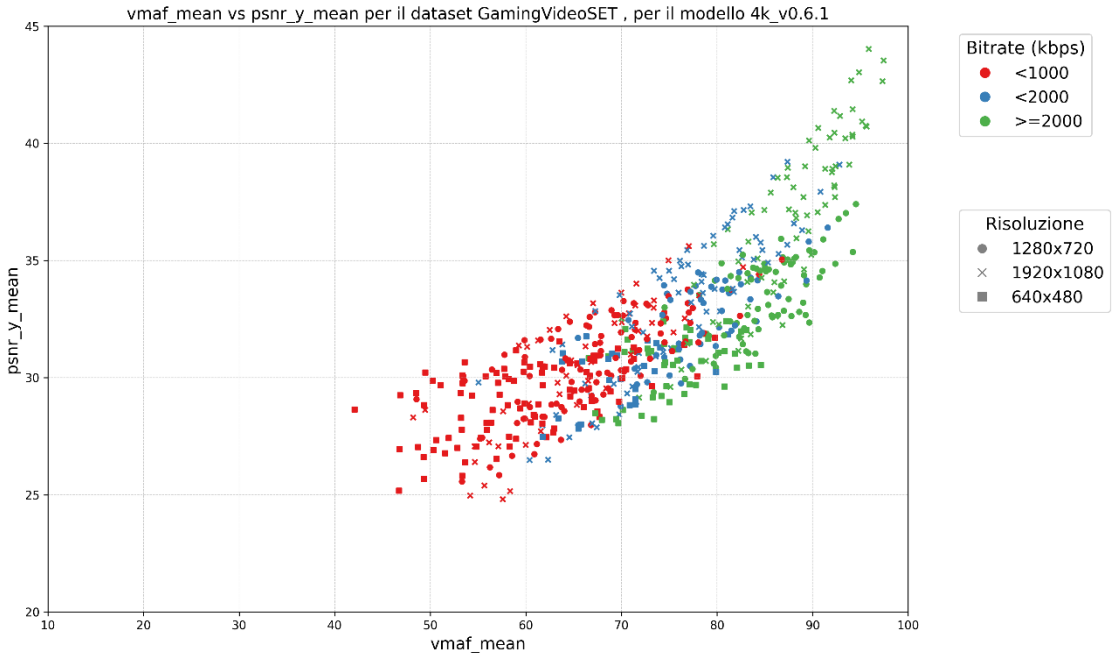

Figura 6.7: VMAF vs Psnr_y per dataset GamingVideoSET, modello 4k_v0.6.1

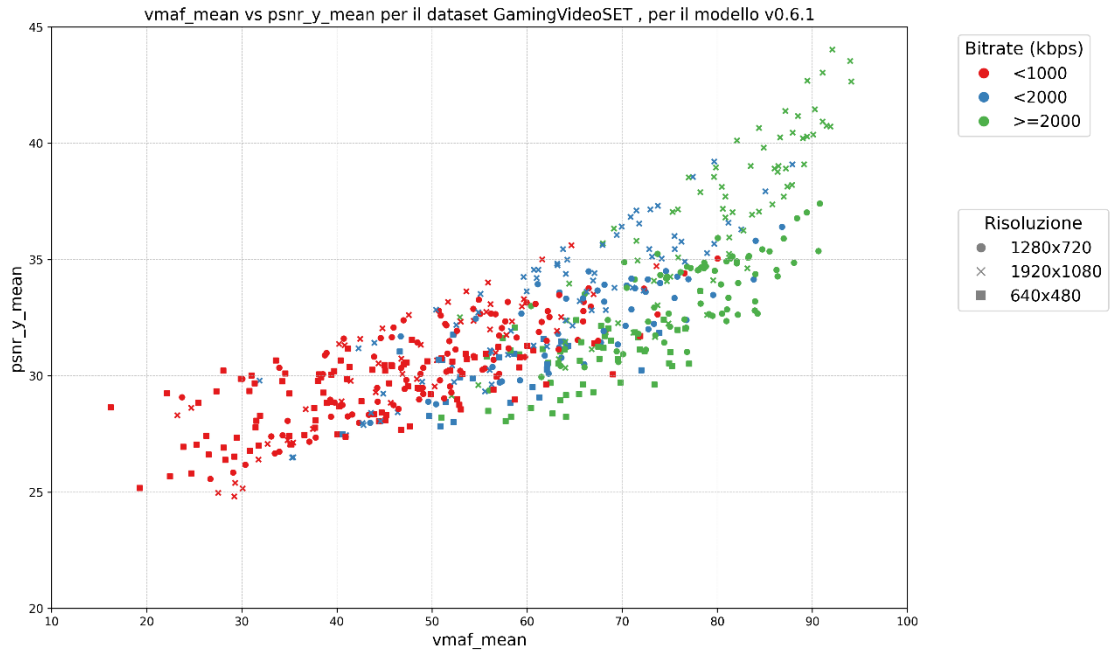

Figura 6.8: VMAF vs Psnr_y per dataset GamingVideoSET, modello v0.6.1

6.2 Feature a confronto

Oltre all'analisi standard, sul dataset ITS4S è stata condotta un'analisi comparativa focalizzata esclusivamente sul modello VMAF v0.6.1. Inizialmente, l'analisi è stata eseguita senza l'integrazione di feature aggiuntive, per poi essere ripetuta aggiungendo, una per volta, le diverse feature opzionali. L'obiettivo era valutare l'impatto di ciascuna aggiunta in termini di utilizzo della CPU e consumo di memoria. Si precisa che, in questa fase, non è stato considerato il tempo impiegato per la conversione preliminare dei file da MP4 a Y4M, poiché tale processo è stato eseguito separatamente per disporre dei file raw già pronti all'analisi. I risultati evidenziano come l'aggiunta di feature quali *CIEDE2000* e *float_ms_ssim* comporti un carico computazionale significativamente maggiore rispetto alle altre feature opzionali considerate. L'utilizzo della CPU in assenza di feature aggiuntive rimane mediamente intorno ai 3-5 secondi. Nel caso del *float_ms_ssim* (figura 6.9), l'utilizzo sale, con picchi che si attestano intorno ai 25-30 secondi, con in media 20 secondi in più.

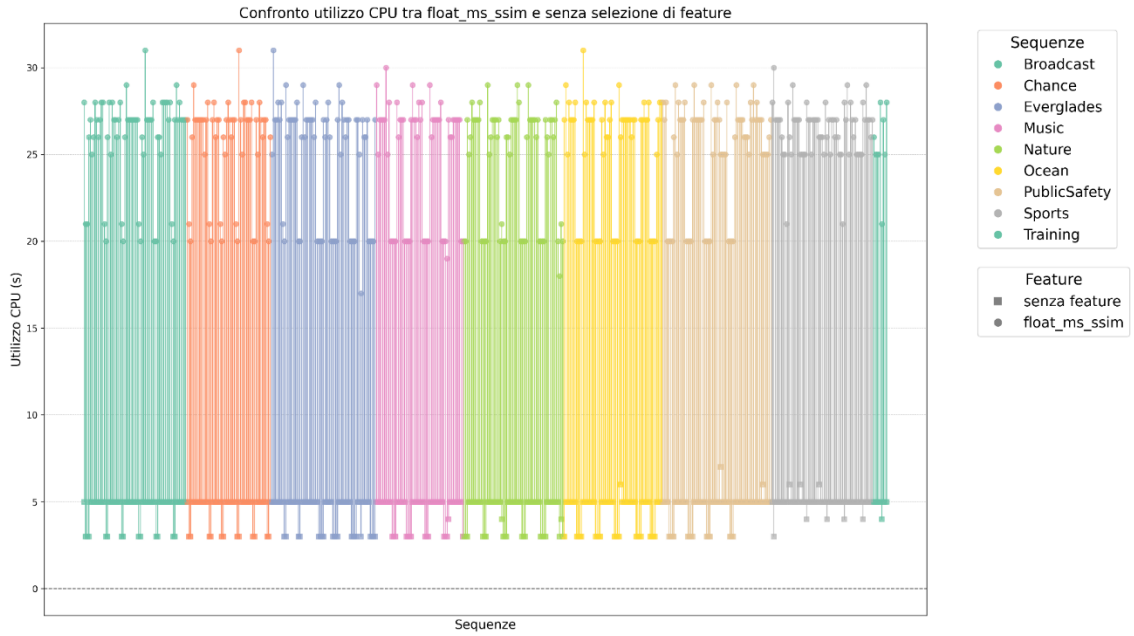

Figura 6.9: Utilizzo CPU per il dataset ITS4S con feature aggiuntiva float_ms_ssim

L'aumento è dovuto alla sua natura multi-scala e all'uso della virgola mobile. La metrica esegue analisi su più livelli di risoluzione per ogni frame, con operazioni matematiche complesse e ad alta precisione, aumentando significativamente il carico computazionale rispetto ad altre features.

Con l'aggiunta invece del *CIEDE2000* (figura 6.10) il dato è ancora più incisivo. Infatti l'aumento medio è di circa 54 secondi, con picchi che vanno anche oltre i 70 secondi.

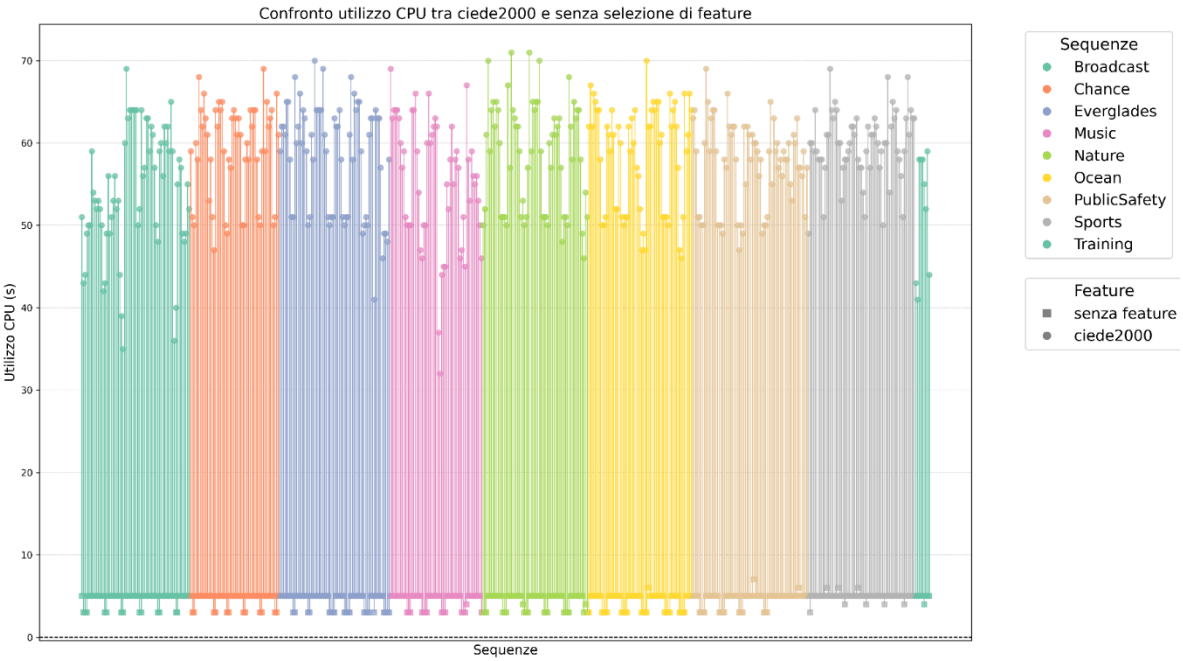

Figura 6.10: Utilizzo CPU per il dataset ITS4S con feature aggiuntiva CIEDE2000

In termini di memoria utilizzata invece, se nel caso semplice e con l'aggiunta del *float_ms_ssim* non si nota alcuna differenza, è molto evidente il grande uso di memoria aggiuntiva da parte di *CIEDE2000* (figura 6.11). Infatti se nel resto delle analisi l'uso della memoria risulta attorno al centinaio di KB, nel caso del ciede si arriva ad una media di 80 MB. Questo marcato incremento sia nell'utilizzo della CPU che della memoria da parte della metrica CIEDE2000 è attribuibile alla natura del suo calcolo.

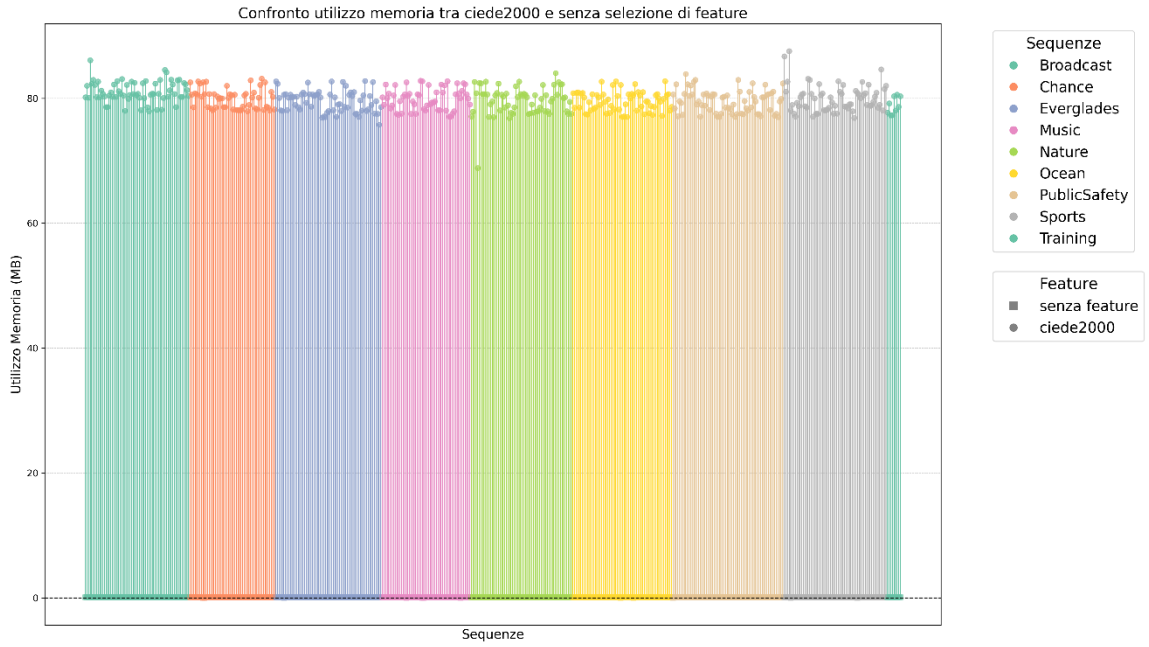

Figura 6.11: Utilizzo memoria per il dataset ITS4S con feature aggiuntiva CIEDE2000

A differenza delle altre features, *CIEDE2000* effettua una valutazione dettagliata delle differenze cromatiche pixel per pixel, operando su frame interi convertiti nello spazio colorimetrico CIE Lab, molto più complesso da gestire computazionalmente. Questo comporta un doppio impatto: da un lato un'elaborazione molto più intensa per ogni singolo frame, dall'altro la necessità di mantenere in memoria grandi quantità di dati intermedi. Inoltre, l'assenza di ottimizzazioni specifiche nel framework VMAF per questa metrica accentua ulteriormente il carico.

6.3 Codec e modelli a confronto

Vengono qui presentati i valori VMAF ottenuti dal dataset AVT-VQDB-UHD-1 (figure 6.12, 6.13 e 6.14), evidenziando le differenze tra i modelli VMAF impiegati e le prestazioni dei diversi codec. In questi plot, il punto centrale di ciascun set di dati rappresenta il valore VMAF ottenuto utilizzando il modello b_v0.6.3, mentre la barra superiore indica il modello 4k_v0.6.1, l'inferiore mostra invece il v0.6.1. L'analisi dei grafici rivela una chiara variabilità nei valori VMAF a seconda del modello utilizzato. Per la maggior parte delle sequenze e risoluzioni, il modello 4k_v0.6.1 tende a produrre i valori VMAF più elevati, indicando una maggiore ottimizzazione per contenuti ad alta risoluzione o una diversa sensibilità alla qualità percepita che lo porta a assegnare punteggi più alti. Al contrario, il modello v0.6.1 produce costantemente i valori VMAF più bassi, suggerendo una valutazione più stringente o una calibrazione diversa che risulta in punteggi inferiori per la stessa qualità video. Il modello b_v0.6.3 si posiziona tipicamente tra i due, fornendo un valore VMAF intermedio. Questa gerarchia di punteggio tra i modelli (v0.6.1 < b_v0.6.3 < 4k_v0.6.1) è evidente in quasi tutti i punti dati e sottolinea l'importanza di specificare il modello VMAF utilizzato per qualsiasi valutazione di qualità, in quanto modelli diversi possono portare a conclusioni significativamente diverse sulla performance percepita. Le differenze tra i modelli diventano

particolarmente pronunciate a risoluzioni più elevate (es. 1080p e 2160p), dove la complessità dei dettagli e la percezione della qualità possono essere interpretate diversamente dagli algoritmi di ciascun modello.

Analizzando le prestazioni dei codec, si osserva un quadro chiaro e interessante. Il codec h264 (cerchi), pur essendo ancora ampiamente diffuso, tende a mostrare i valori VMAF più bassi tra i tre, un divario che si accentua notevolmente con l'aumentare delle risoluzioni. Mentre a 360p e 720p le sue prestazioni sono ancora accettabili, oltre queste soglie la sua efficienza nel mantenere un'elevata qualità percepita diminuisce progressivamente, soprattutto in presenza di bitrate elevati o contenuti video complessi. È inoltre interessante notare come l'ampiezza dei valori VMAF tra i diversi modelli (indicata dalla lunghezza delle linee verticali) sia spesso più marcata per h264, suggerendo che i vari algoritmi VMAF possano avere interpretazioni particolarmente divergenti della qualità di video codificati con questo codec. Passando al codec hevc (quadrati) si riscontra una superiorità costante e marcata rispetto a h264 in quasi tutti gli scenari. Questo codec raggiunge valori VMAF significativamente più elevati, in particolare alle risoluzioni di 1080p e 2160p. Tale performance riflette la sua maggiore efficienza di compressione, che gli consente di preservare meglio i dettagli e la qualità percepita a parità di bitrate, o di ottenere la stessa qualità con un bitrate inferiore. I margini di variazione nella valutazione per hevc sono

generalmente più contenuti rispetto a quelli di h264, indicando una maggiore consistenza tra i modelli VMAF nella valutazione della qualità dei video codificati con hevc.

Infine, il codec vp9 (croci), presente nei grafici AVT_test1 e AVT_test3, dimostra prestazioni estremamente competitive. Spesso si allinea, o talvolta supera, hevc alle risoluzioni più elevate, in particolare per sequenze come "water_netflix" e "bigbuck_bunny_8bit". Similmente a hevc, anche vp9 mantiene una stabilità di valutazione relativamente elevata tra i diversi modelli VMAF, a testimonianza di una buona coerenza nei suoi risultati.

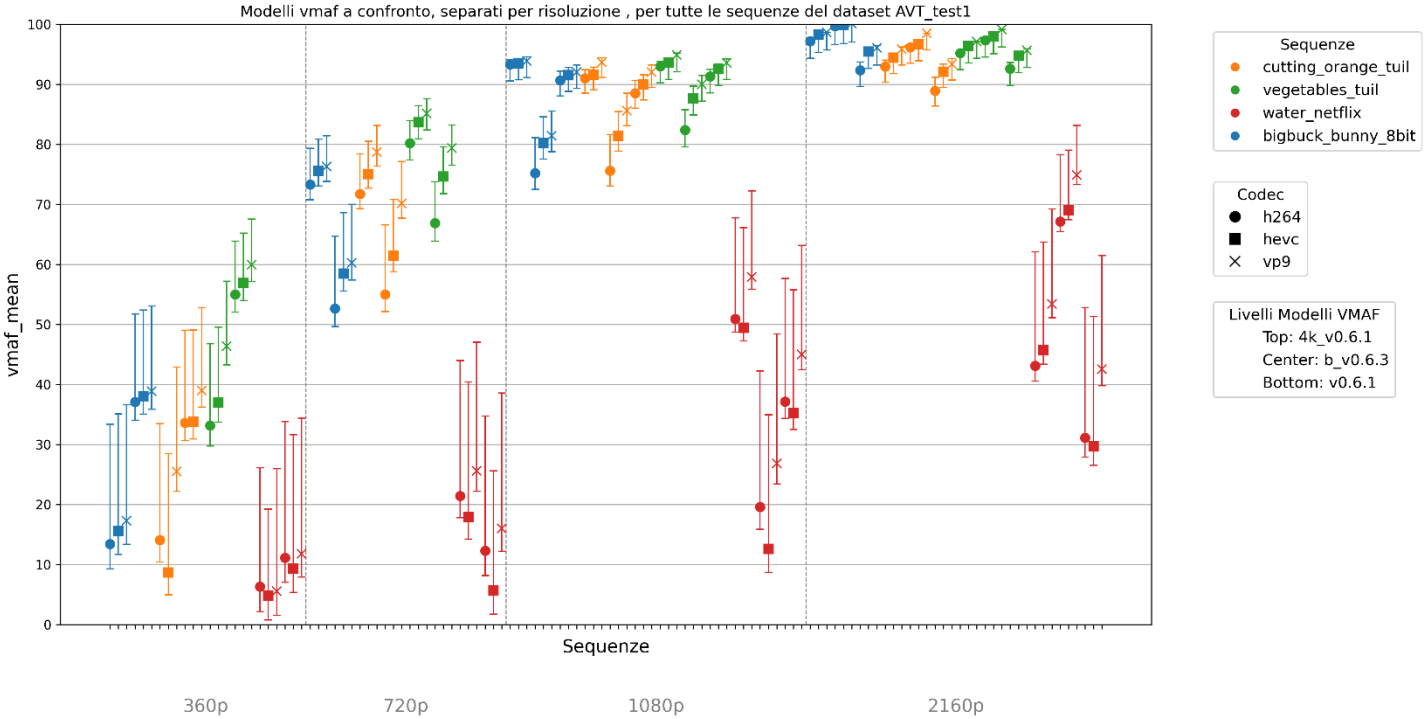

Figura 6.12: Modelli VMAF e codec a confronto, separati per risoluzione, per il dataset AVT_test1

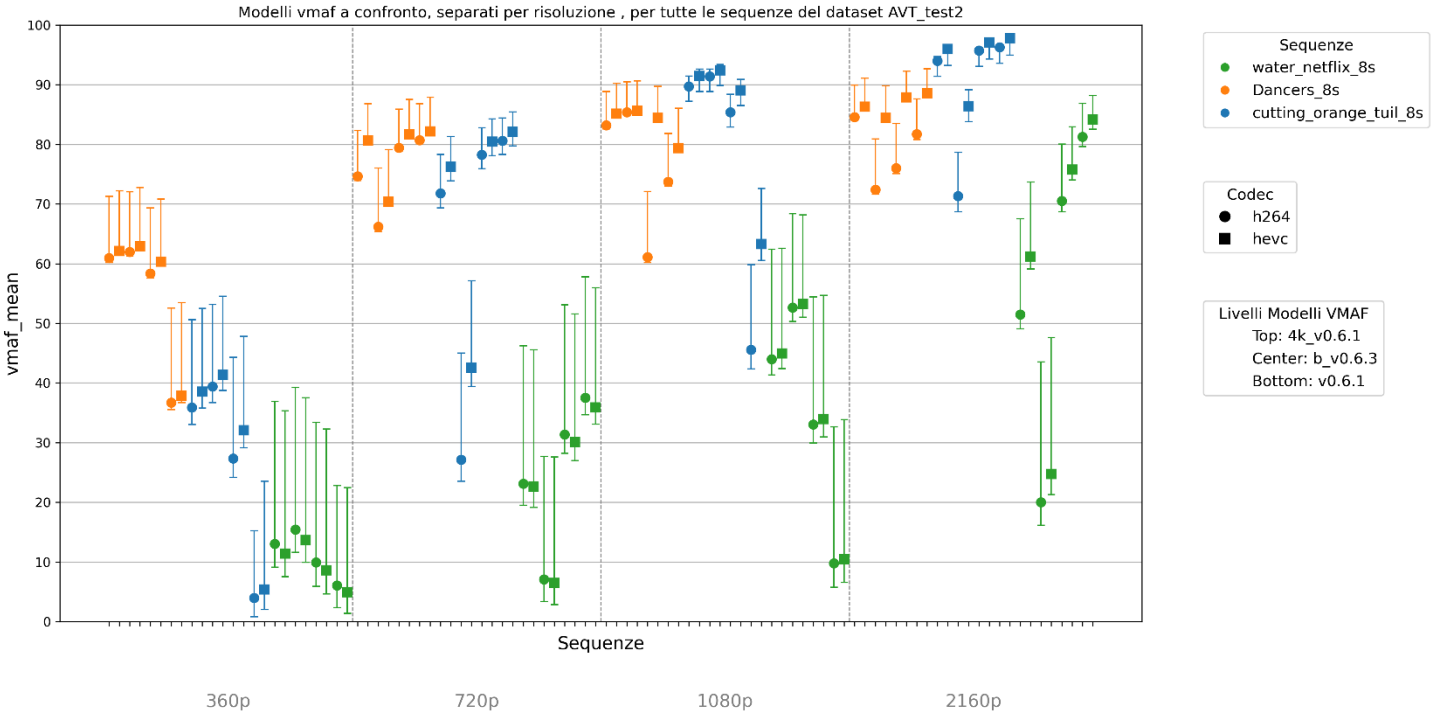

Figura 6.13: Modelli VMAF e codec a confronto, separati per risoluzione, per il dataset AVT_test2

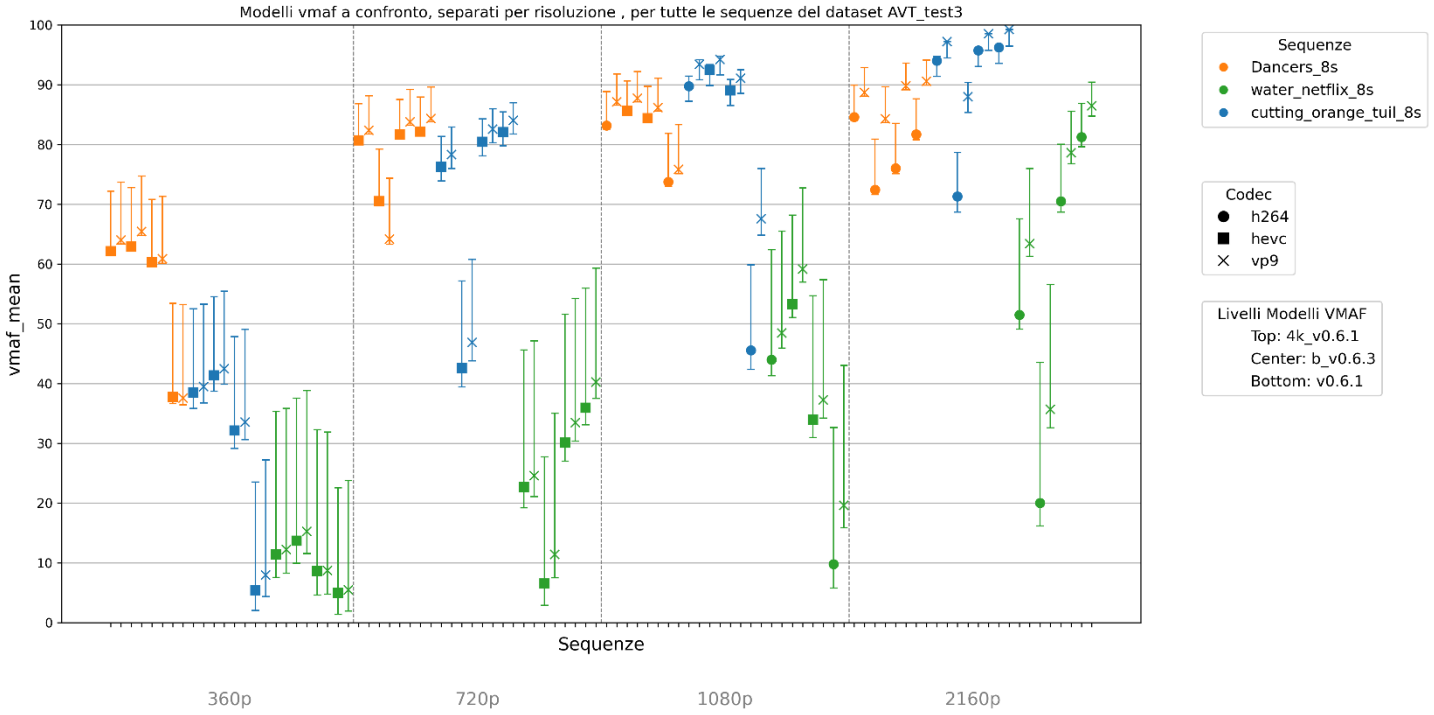

Figura 6.14: Modelli VMAF e codec a confronto, separati per risoluzione, per il dataset AVT_test3

6.4 Consumo CPU e memoria

In quest'ultimo paragrafo vengono mostrati i valori di consumo medio di CPU e memoria risultanti dalle analisi di tutti i dataset (figure 6.15 e 6.16). Dai grafici, risulta evidente che il dataset AVT-VQDB-UHD-1 è di gran lunga il più esigente. Le sue sequenze richiedono un utilizzo medio della CPU che eccede i 4600 secondi (circa 77 minuti) e un consumo di memoria superiore ai 1000 MB (circa 1 GB) per ciascuna elaborazione. Questa elevata richiesta è direttamente attribuibile alle caratteristiche di questo dataset, il quale comprende principalmente contenuti 4K (fino a 2160p) e framerate elevati (fino a 60 fps). La valutazione di un volume di pixel per frame e di un numero di frame per secondo così elevato comporta un carico di lavoro computazionale massiccio, che si traduce in tempi di calcolo prolungati e un fabbisogno di memoria considerevole.

In una fascia intermedia di consumo si posizionano i dataset KUGVD e GamingVideoSET. Entrambi in media richiedono un utilizzo della CPU di circa 2000 secondi (circa 33 minuti) e un consumo di memoria attorno ai 240 MB. Questi valori sono coerenti con le specifiche dei due dataset, caratterizzati da video non compressi a 1080p e 30 fps, con una durata di 30 secondi. Sebbene non raggiungano le risoluzioni estreme del dataset AVT, la loro risoluzione Full HD, combinata con la durata e la complessità dei

contenuti di gioco, li rende significativamente più onerosi da elaborare rispetto ad altri set di dati.

I dataset AGH-NTIA-Dolby e ITS4S manifestano l'utilizzo di risorse più contenuto. AGH-NTIA-Dolby richiede circa 500 secondi di CPU e 120 MB di memoria, mentre ITS4S è il meno esigente, con meno di 100 secondi di CPU e circa 110 MB di memoria. Questa maggiore efficienza computazionale è spiegata dalle loro caratteristiche: i video di ITS4S hanno una durata di soli 4 secondi e sono a 720p e 24 fps, mentre le sequenze di AGH-NTIA-Dolby durano 8 secondi con risoluzioni che non raggiungono quelle dei dataset gaming o AVT. La durata più breve e le risoluzioni inferiori riducono drasticamente il volume di dati da processare per ciascuna sequenza.

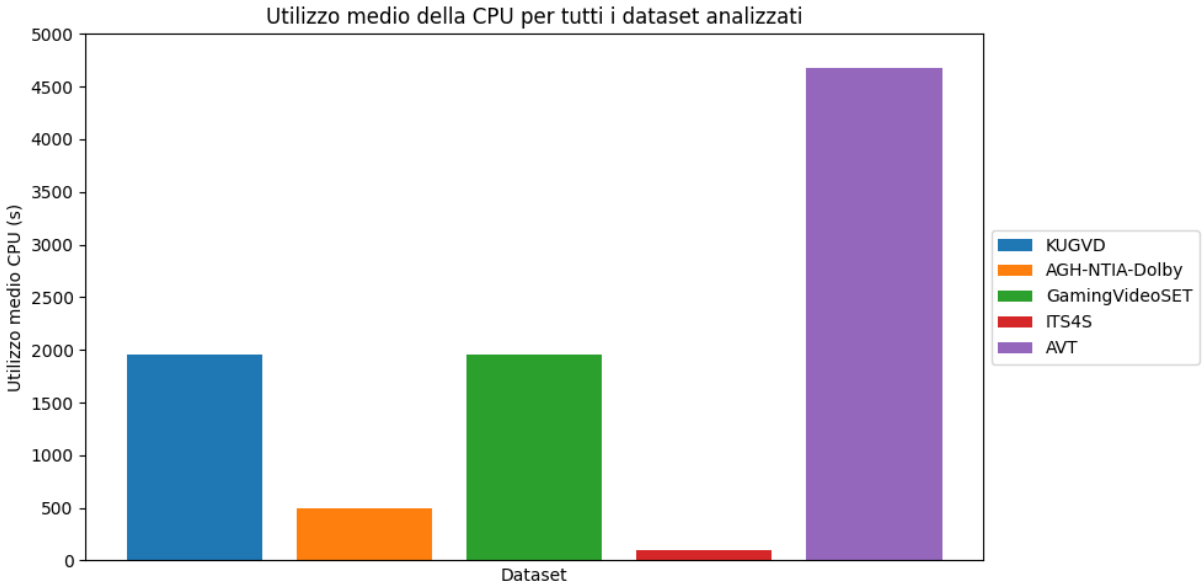

Figura 6.15: Consumo medio CPU per tutti i dataset

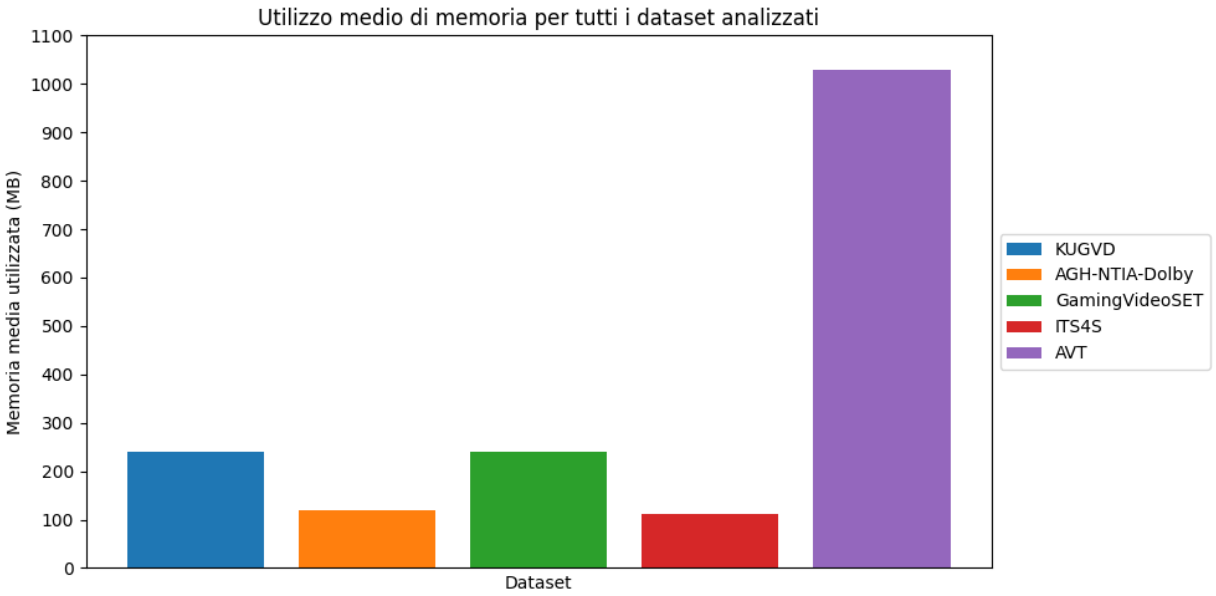

Figura 6.16: Consumo medio memoria per tutti i dataset

Capitolo 7

Conclusione

Il lavoro di tesi svolto ha avuto come obiettivo principale la valutazione oggettiva della qualità video mediante l'utilizzo di VMAF (Video Multimethod Assessment Fusion), applicato su dataset di grandi dimensioni codificati con i principali codec moderni (H.264, H.265 e VP9). L'implementazione di un framework riproducibile, basato su script automatizzati in Python e Bash, ha consentito di gestire l'intero processo di pre-processing, analisi e raccolta dati su infrastrutture HPC del Politecnico di Torino, garantendo elevata efficienza e scalabilità grazie all'utilizzo di container Singularity.

I risultati ottenuti hanno permesso di confrontare in modo approfondito le prestazioni dei diversi modelli VMAF, evidenziando come le variazioni percettive siano strettamente legate al tipo di codifica, alla risoluzione e al bitrate applicato. Inoltre, sono state analizzate anche le implicazioni computazionali in termini di consumo di CPU e memoria, offrendo così una

visione completa dei compromessi esistenti tra qualità visiva e risorse necessarie.

Il framework sviluppato si è dimostrato efficace e versatile, facilmente adattabile per ulteriori studi comparativi su nuovi modelli di valutazione della qualità video e su contenuti compressi con codec emergenti. Le potenzialità offerte da questa infrastruttura possono rappresentare un valido supporto per attività future sia in ambito accademico, sia per applicazioni industriali legate allo streaming video, alla trasmissione in rete e all'ottimizzazione dei sistemi di compressione.

Infine, il lavoro svolto costituisce una base solida per eventuali sviluppi futuri, quali l'estensione a contenuti UHD/8K, l'integrazione con metriche no-reference, e l'ottimizzazione del consumo computazionale nell'ambito di analisi massive in contesti di intelligenza artificiale e machine learning.

Elenco figure

Bibliografia

[1]C. Yamahata, Human vision and the CIE chromaticity diagram, Marzo 2024,https://www.researchgate.net/publication/378936775_Human_vision_and_the_CIE_chromaticity_diagram

[2] G.Capri,Color Correction: i prerequisiti base, 4 Gennaio 2020, https://montalo.net/color-correction-nozioni-base/

[3] Sottocampionamento della crominanza, From Wikipedia, the free encyclopedia, https://it.wikipedia.org/wiki/Sottocampionamento_della_crominanza#/media/File:Chroma_subsampling_ratios.svg

[4] Poynton, C. (2012). "Digital Video and HD: Algorithms and Interfaces" https://www.google.it/books/edition/Digital_Video_and_HD/Sa-cEY_ZeEQC?hl=it&gbpv=0

[5] The LENSEC Perspective Newsletter, https://lensec.com/about/news/newsletter/lpn-03-20/

[6] A. Suissa, A Novel Video Packet Loss Concealment Algorithm & Real Time Implementation, Novembre 2008, https://www.researchgate.net/publication/29601288_A_Novel_Video_Packet_Loss_Concealment_Algorithm_Real_Time_Implementation

[7] Difference between I-Frames, P-Frames and B-Frames, https://mymusing.co/difference-between-i-frames-p-frames-and-b-frames/

[8] Discrete cosine transform, https://en.wikipedia.org/wiki/Discrete_cosine_transform#/media/File:DCT-8x8.png

[9] JPEG, From Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/JPEG#/media/File:JPEG_ZigZag.svg

[10] B. Pervan, MIDOM—A DICOM-Based Medical Image Communication System, Maggio 2023, https://www.researchgate.net/figure/An-example-of-run-length-encoding_fig2_370778264

[11] H.264 : Advanced video coding for generic audiovisual services, https://www.itu.int/rec/T-REC-H.264

[12] H.265 : High efficiency video coding, https://www.itu.int/rec/T-REC-H.265

[13] VP9, From Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/VP9

[14] ISO/IEC 14496-14:2020Information technology — Coding of audio-visual objects,Part 14: MP4 file format, https://www.iso.org/standard/79110.html

[15] Matroska Media Container, https://www.matroska.org/

[16] FFmpeg, https://www.ffmpeg.org/

[17] ITU-R BT.500 – Methodology for the subjective assessment of the quality of television pictures, https://www.itu.int/rec/R-REC-BT.500/en

[18] ITU-T P.910 – Subjective video quality assessment methods for multimedia applications, https://www.itu.int/rec/T-REC-P.910/en

[19] Peak signal-to-noise ratio, From Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[20] P.Simoncelli,Z. Wang, H.R. Sheikh, A.C. Bovik, “Image Quality Assessment: From Error Visibility to Structural Similarity”, 2004,

https://www.researchgate.net/publication/3327793_Image_Quality_Assessment_From_Error_Visibility_to_Structural_Similarity

[21] H.R. Sheikh, A.C. Bovik , “Image Information and Visual Quality” , https://live.ece.utexas.edu/publications/2004/hrs_ieeetip_2004_imginfo.pdf

[22] M. Luo, D. R. C. Hill, e R. K. M. Jayant, "Color Difference Formulae and Their Application to Image Quality Assessment" , 2001, https://www.researchgate.net/publication/258382945_Color_image_quality_assessment_based_on_CIEDE2000

[23] Colour banding, From Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Colour_banding#/media/File:Colour_banding_example01.png

[24] Pulkit Tandon, Mariana Afonso, Joel Sole, Lukáš Krasula,” CAMBI: Contrast-aware Multiscale Banding Index”,2021, https://www.researchgate.net/publication/348959018_CAMBI_Contrast-aware_Multiscale_Banding_Index

[25] C. Bampis,Measuring Video Quality with VMAF: Why You Should Care, 15 Ottobre 2019, http://downloads.aomedia.org/assets/pdf/symposium-2019/slides/ChristosBampis_Netflix.pdf

[26] VMAF - Video Multi-Method Assessment Fusion

https://github.com/Netflix/vmaf/tree/master

[27] Cluster Legion,HPC@POLITO, https://www.hpc.polito.it/

[28] Definition Files, https://docs.sylabs.io/guides/3.7/user-guide/definition_files.html

[29] Singularity User Guide, https://docs.sylabs.io/guides/3.7/user-guide/index.html

[30] GamingVideoSET: a dataset for gaming video streaming applications,

https://eprints.kingston.ac.uk/id/eprint/41300/

[31] Development of a Bitstream-based Video Quality Prediction Model Using Deep Learning Techniques for Gaming Streaming Services,

https://github.com/nasimjamshidi/Deep-BVQM

[32] ITS4S: A Video Quality Dataset with Four-Second Unrepeated Scenes, https://its.ntia.gov/publications/details?pub=3194

[33] Evaluating Experiment Design with Unrepeated Scenes for Video Quality Subjective Assessment,

https://its.ntia.gov/publications/details?pub=3240

[34] AVT-VQDB-UHD-1 video quality database,

https://telecommunication-telemedia-assessment.github.io/AVT-VQDB-UHD-1/

Ringraziamenti

Voglio innanzitutto ringraziare la mia famiglia, che mi ha accompagnato in questi anni di studio con pazienza e grande supporto, consigliandomi nei momenti di difficoltà e gioendo con me per ogni traguardo raggiunto. Ve ne sarò sempre riconoscente.

A Chiara, che ha sempre trovato il tempo per parlare e confrontarsi, sia nei momenti felici che in quelli più difficili.

Ad Alessio, con cui ho condiviso tutta la mia vita: un sostegno costante, che mi ha aiutato a crescere tanto. Grazie.

Agli amici di sempre – Davide, Alessio, Gimmy e Mauro – con cui ho vissuto momenti indimenticabili e con cui spero di condividerne molti altri ancora.

A Debora, amica ritrovata, che avrà sempre il mio rispetto e la mia stima. Il tuo amico silenzioso ti augurerà sempre il meglio.

Un ringraziamento a Eugenio, Enrico e Simone, coinquilini con cui ho condiviso gli anni torinesi, scambiandoci tradizioni, consigli e opinioni.

Vorrei infine ringraziare con sincera riconoscenza il professor Enrico Masala per la sua costante disponibilità, il prezioso supporto e la guida competente che hanno accompagnato ogni fase di questo lavoro.