

Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

A.A. 2025/2026

Sessione di laurea settembre 2026

Sviluppo di un framework per la valutazione di codec video neurali in scenari real-time con perdita di dati

Relatore:
Enrico Masala

Candidato:
Emanuele Frisi

Every little boy should have a hat

Giustino Bagge

Sommario

L'esplosione del traffico dati globale ha reso indispensabile lo sviluppo di tecnologie di compressione video sempre più efficienti. In questo contesto, i codec video basati su reti neurali rappresentano un'alternativa promettente ai tradizionali codec ibridi. Questo elaborato analizza la resilienza e le prestazioni del modello neurale DCVC-RT in scenari di rete non ideali, caratterizzati da perdita di dati e latenza.

A tal fine, è stato sviluppato un framework di simulazione in grado di riprodurre differenti condizioni di rete, controllando il tasso di perdita dei dati e il ritardo del canale di feedback. Il framework automatizza la generazione degli scenari di perdita, la codifica e la decodifica delle sequenze, consentendo di valutare il comportamento del codec al variare delle condizioni di rete e su molteplici tracce di perdita. Lo studio analizza inoltre l'integrazione del *Frame Freeze* in fase di decodifica come strategia di *error concealment*.

I risultati evidenziano una marcata dicotomia tra valutazione oggettiva ed esperienza visiva soggettiva. Sebbene le metriche penalizzino il disallineamento temporale introdotto dal congelamento dell'immagine, l'analisi visiva mostra che il *Frame Freeze* può migliorare significativamente la qualità percepita a basse latenze, limitando temporaneamente la propagazione degli artefatti senza introdurre overhead di banda. L'efficacia della tecnica dipende tuttavia dalla dinamica della scena e dal ritardo del canale di feedback: mentre i contenuti prevalentemente statici risultano maggiormente tolleranti, nelle sequenze dinamiche il beneficio percettivo si riduce all'aumentare della latenza, fino a risultare insufficiente a fronte della prolungata propagazione dell'errore.

Indice

Elenco delle figure VI

Introduzione 1

1 Background tecnologico 3
1.1 Compressione video: dai codec tradizionali alle reti neurali 3
1.2 Concetti fondamentali 4
1.2.1 Tipi di fotogrammi 4
1.2.2 *Group of pictures* (GOP) 5
1.2.3 Bitrate 5
1.3 Evoluzione della famiglia di codec DCVC 5
1.4 Metriche di valutazione della qualità video 6

2 Analisi preliminare e modularizzazione del codec 8
2.1 Analisi dell'architettura originale 8
2.2 Modularizzazione dell'ambiente di test 10
2.3 Analisi dei costi: I-frame vs P-frame 11
2.3.1 Impatto sul bitrate 11
2.3.2 Impatto della riduzione del GOP su resilienza e bitrate . . . 12
2.3.3 Impatto sulle metriche di qualità 13

3 Studio della resilienza ed error concealment 15
3.1 Tecniche di corruzione del bitstream 15
3.2 Impatto sul codec DCVC-RT e propagazione dell'errore 16
3.3 Tentativi di error concealment sul DCVC-RT 17
3.4 Test comparativi sul codec DCVC-FM 18

4 Sviluppo del framework di simulazione per scenari real-time 21
4.1 Architettura del framework 21
4.1.1 Motivazione 21
4.1.2 Orchestratore del framework 21

4.1.3 Gestione del flusso dati 22
4.2 Simulazione del canale di feedback mediante il ritardo M 22
4.3 Integrazione del *Frame Freeze* 23
4.4 Approccio Monte Carlo e gestione delle tracce multiple 24
4.5 Analisi delle metriche ed esportazione video 24

5 Risultati sperimentali e analisi 26
5.1 Setup sperimentale 26
5.1.1 Sequenze video utilizzate 26
5.1.2 Parametri di simulazione 27
5.1.3 Hardware e metriche di riferimento 27
5.2 Resilienza naturale e sistema di feedback 27
5.2.1 Andamento delle metriche 28
5.2.2 Varianza statistica 30
5.2.3 Analisi visiva del degrado 34
5.3 Mitigazione tramite *Frame Freeze* 36
5.3.1 Andamento delle metriche e riduzione della varianza 37
5.3.2 Analisi visiva del degrado: efficacia a basse latenze 40
5.3.3 Analisi visiva del degrado: limiti ad alta latenza 42
5.4 Valutazione complessiva: curve RD 44

Conclusioni e sviluppi futuri 47

Bibliografia 50

Elenco delle figure

3.1 Propagazione dell'errore per il codec DCVC-RT 17
3.2 Evoluzione del degrado con error concealment su DCVC-FM 19

5.1 Andamento metriche per RaceHorses (QP=47) 28
5.2 Andamento metriche per BQSquare (QP=47) 29
5.3 Dispersione statistica per RaceHorses (QP=47) 31
5.4 Dispersione statistica per RaceHorses (QP=47) con M=2 31
5.5 Dispersione statistica per RaceHorses (QP=47) con M=6 32
5.6 Dispersione statistica per RaceHorses (QP=47) con M=10 32
5.7 Dispersione statistica per BQSquare (QP=47) 33
5.8 Confronto fotogramma 51 per RaceHorses (M=2) 34
5.9 Confronto fotogramma 55 per RaceHorses (M=6) 35
5.10 Confronto fotogramma 59 per RaceHorses (M=10) 36
5.11 Andamento metriche per RaceHorses (QP=47) con *Frame Freeze* . . 37
5.12 Andamento metriche per BQSquare (QP=47) con *Frame Freeze* . . 38
5.13 Dispersione statistica per BQSquare (QP=47) con *Frame Freeze* . . 39
5.14 Dispersione statistica per RaceHorses (QP=47) con *Frame Freeze* . 40
5.15 Confronto fotogramma 51 per RaceHorses (M=2) con *Frame Freeze* 41
5.16 Confronto fotogramma 55 per RaceHorses (M=6) con *Frame Freeze* 42
5.17 Confronto fotogramma 56 per RaceHorses (M=10) con *Frame Freeze* 43
5.18 Confronto fotogramma 59 per RaceHorses (M=10) con *Frame Freeze* 43
5.19 Curve Rate-Distortion per RaceHorses (M=10) 44
5.20 Curve Rate-Distortion per BQSquare (M=10) 45

Introduzione

L'avvento del *Deep Learning* ha introdotto nuove possibilità nel campo della compressione video, dando origine a codec neurali in grado di raggiungere risultati competitivi rispetto ai tradizionali standard ibridi. Tuttavia, la valutazione di queste architetture avviene frequentemente in condizioni di test ideali, nelle quali il flusso di dati raggiunge il decoder senza perdite o corruzioni. Questo approccio non considera una delle principali criticità di una trasmissione reale: le reti di comunicazione sono soggette a condizioni variabili, caratterizzate da latenza, congestione e possibili perdite di dati.

L'obiettivo di questo elaborato è analizzare la resilienza di uno dei codec neurali studiati per applicazioni real-time, il DCVC-RT, in presenza di condizioni di trasmissione non ideali. Abbandonando l'ipotesi di un canale trasmissivo perfetto, il lavoro analizza gli effetti prodotti dalla corruzione e dalla perdita di dati, osservando come un singolo difetto nel bitstream possa innescare una rapida propagazione degli artefatti, compromettendo progressivamente la ricostruzione dei fotogrammi successivi e, nei casi più severi, portando al collasso del processo di decodifica.

Per poter studiare, quantificare e tentare di mitigare questi fenomeni, si è resa necessaria la progettazione e lo sviluppo di un framework di simulazione automatizzato. L'ambiente software realizzato supera le limitazioni del codice originale, permettendo di manipolare il flusso dati, simulare un canale di feedback per la richiesta dinamica di nuovi fotogrammi chiave (I-frame) e integrare strategie di error concealment in fase di decodifica.

La presente tesi è organizzata in cinque capitoli, strutturati come di seguito specificato:

- Nel **Capitolo 1** verrà introdotto il background tecnologico, delineando il passaggio dai codec tradizionali al paradigma neurale e presentando alcuni dei principali concetti alla base della codifica video, necessari per comprendere i capitoli successivi, insieme alle metriche utilizzate per la valutazione della qualità visiva.

- Nel **Capitolo 2** verrà esaminata l'architettura del codice sorgente del DCVC-RT, illustrando il processo di disaccoppiamento e modularizzazione necessario

per poter manipolare il bitstream intermedio.

- Nel **Capitolo 3** verrà studiata la propagazione dell'errore a fronte di corruzioni mirate, analizzando il comportamento del codec e i limiti delle tecniche di error concealment applicate direttamente in fase di decodifica.
- Nel **Capitolo 4** verrà descritto lo sviluppo del framework di simulazione, illustrando la modellazione della latenza del canale di feedback, la generazione dei pattern di errore e l'integrazione iterativa della tecnica di error concealment denominata *Frame Freeze*.
- Nel **Capitolo 5** verranno presentati e analizzati i risultati sperimentali, valutando il comportamento del codec al variare delle condizioni di perdita e di latenza, e confrontando le metriche oggettive con l'analisi visiva delle sequenze ricostruite.
- Infine, nella sezione **Conclusioni e sviluppi futuri** verranno sintetizzati i principali risultati emersi dal lavoro, discutendo i limiti dell'approccio proposto e le possibili direzioni di ricerca.

Capitolo 1

Background tecnologico

1.1 Compressione video: dai codec tradizionali alle reti neurali

L'esplosione del traffico dati globale, legata alla continua distribuzione di contenuti multimediali, ha reso indispensabile lo sviluppo di tecnologie di trasmissione video sempre più efficienti. Standard di compressione come H.264/AVC, H.265/HEVC e il più recente H.266/VVC si sono evoluti proprio per far fronte a questa necessità, con l'obiettivo di abbattere la larghezza di banda necessaria per la trasmissione dei video, senza penalizzare la qualità visiva.

Questi standard condividono un paradigma di codifica ibrida basato su moduli progettati analiticamente: stima del movimento, predizione spaziale e temporale, trasformata e quantizzazione del residuo, e infine codifica entropica. Il vero tallone d'Achille di tale approccio è la crescente complessità computazionale; per garantire incrementi anche solo marginali nelle prestazioni, i nuovi standard introducono un numero sempre maggiore di modalità e strumenti di codifica, portando a un aumento quasi insostenibile dei tempi di elaborazione.

L'applicazione del *Deep Learning* ha permesso di introdurre un nuovo approccio alla compressione video. Le reti neurali consentono di ridurre la dipendenza dalla progettazione manuale dei singoli moduli, apprendendo direttamente dai dati le trasformazioni più efficaci per la rappresentazione e la compressione dell'informazione. Questo rende possibile la realizzazione di codec video *end-to-end*, nei quali le diverse fasi della pipeline sono integrate in un'unica architettura neurale e ottimizzate congiuntamente, al fine di trovare un compromesso efficace tra il tasso di compressione e la fedeltà visiva della ricostruzione.

La principale differenza tra l'approccio tradizionale e quello neurale risiede nel dominio di elaborazione. I codec classici lavorano sui pixel grezzi, affidandosi a trasformate lineari (come la DCT) per compattare l'informazione visiva. I

codec neurali derivano invece dalle architetture ad autoencoder, originariamente impiegate per comprimere le singole immagini. Anziché manipolare direttamente il fotogramma, una prima rete neurale (l'encoder) estrae dai pixel un insieme di feature ad alta densità. L'immagine viene così proiettata in una rappresentazione compatta, nota come **spazio latente**, al cui interno le dimensioni fisiche del fotogramma si riducono drasticamente, a favore di una maggiore concentrazione di informazioni. La compressione effettiva avviene quantizzando questa rappresentazione latente e sottoponendola a codifica entropica. A valle della trasmissione, interviene una seconda rete (il decoder) con il compito di invertire il processo, decodificando e destrutturando la rappresentazione latente per ricostruire i pixel dell'immagine di partenza [1].

1.2 Concetti fondamentali

Prima di analizzare nello specifico le architetture neurali, è necessario fissare alcuni concetti alla base dell'elaborazione video, indispensabili per seguire senza difficoltà il resto dell'elaborato.

1.2.1 Tipi di fotogrammi

All'interno di un flusso video compresso, le singole immagini non sono codificate in maniera uniforme. A seconda della tecnica di predizione impiegata per ridurre la mole di dati, i fotogrammi si dividono storicamente in tre categorie:

- **I-frame (Intra-coded picture)**: è un fotogramma codificato in maniera del tutto autonoma, sfruttando esclusivamente la ridondanza spaziale all'interno dell'immagine. Il processo è concettualmente identico alla compressione di una singola fotografia; essendo codificato in modo del tutto indipendente dagli altri fotogrammi, l'I-frame agisce come punto di riferimento essenziale per avviare la decodifica o ripristinare la corretta riproduzione del flusso.

- **P-frame (Predicted picture)**: viene elaborato analizzando la ridondanza temporale rispetto ai fotogrammi passati. Il processo di compressione codifica esclusivamente le informazioni di movimento e l'errore residuale rispetto a un'immagine di riferimento precedente, permettendo un abbattimento drastico del peso in bit rispetto a un I-frame.

- **B-frame (Bi-predicted picture)**: raggiunge il livello massimo di compressione sfruttando simultaneamente fotogrammi passati e futuri per calcolare la predizione temporale. Tuttavia, è fondamentale precisare che all'interno di scenari operativi a bassa latenza, come le comunicazioni real-time, i B-frame

sono generalmente evitati. La necessità di attendere la ricezione di fotogrammi futuri, per poter ricostruire l'immagine attuale, introdurrebbe infatti un ritardo di decodifica che violerebbe i requisiti di bassa latenza.

1.2.2 *Group of pictures* (GOP)

La struttura periodica con cui le diverse tipologie di fotogrammi si susseguono all'interno del flusso prende il nome di *Group of pictures* (GOP). Un GOP si apre regolarmente con un I-frame, a cui segue una catena di fotogrammi predittivi. La dimensione del GOP indica il numero di fotogrammi che compongono l'intervallo tra due I-frame consecutivi, includendo il fotogramma di riferimento iniziale. Si tratta di un importante parametro di codifica, poiché la frequenza di inserimento degli I-frame influenza direttamente la resilienza del flusso agli errori temporali: un nuovo I-frame ristabilisce infatti un punto di riferimento indipendente dai fotogrammi precedenti, interrompendo la propagazione degli artefatti derivanti dalla perdita o dalla corruzione di frame antecedenti.

1.2.3 Bitrate

Il bitrate quantifica il volume di dati al secondo, necessario per trasmettere o archiviare un flusso video. Rappresentando una misura diretta del carico imposto alla rete, il bitrate costituisce una metrica di riferimento fondamentale. Tuttavia, nell'ambito specifico dei codec neurali, per valutare le prestazioni dei modelli in modo indipendente dalla risoluzione e dal framerate delle sequenze video, il costo trasmissivo viene comunemente normalizzato e misurato in bit per pixel (bpp). Poiché le due metriche sono direttamente correlate e descrivono entrambe le dimensioni del flusso compresso, d'ora in avanti il termine **bitrate** verrà utilizzato convenzionalmente per fare riferimento al valore in bpp.

1.3 Evoluzione della famiglia di codec DCVC

Introdotto nel 2021, il progetto DCVC (*Deep Contextual Video Compression*) [2] si distingue dalle architetture neurali precedentemente proposte, poiché sostituisce la tradizionale codifica del residuo con il paradigma della codifica condizionale. Anziché limitarsi a calcolare una predizione nel ristretto dominio dei pixel, il modello apprende dinamicamente a estrarre informazioni spaziali e temporali proiettandole in una rappresentazione latente; quest'ultima viene utilizzata come base per condizionare e ottimizzare l'intera fase di compressione di un fotogramma.

L'evoluzione di questo approccio ha portato allo sviluppo del codec **DCVC-RT** [3], ovvero la variante specifica su cui si concentra il presente elaborato. Il codec nasce con l'obiettivo di rendere la codifica neurale applicabile in scenari

real-time, intervenendo in particolare sulla complessità operazionale, legata a fattori quali gli accessi alla memoria e l'overhead delle chiamate di funzione, che possono rappresentare un collo di bottiglia più rilevante dei costi puramente computazionali. A differenza delle versioni precedenti, che facevano ancora affidamento su moduli espliciti per la stima e la compensazione del movimento, il DCVC-RT introduce una strategia di modellazione temporale di tipo implicito, riducendo così il numero di operazioni e accessi alla memoria, necessari durante i processi di codifica e decodifica. A ciò si aggiunge l'utilizzo di una singola rappresentazione latente a bassa risoluzione, in sostituzione delle strutture a risoluzione progressivamente decrescente. Le modifiche descritte consentono di ridurre significativamente la complessità operazionale e la latenza del sistema, mantenendo al contempo un'elevata efficienza di compressione.

Si precisa che, al netto di un'ulteriore iterazione del codec, pubblicata in concomitanza con le fasi conclusive di stesura, il DCVC-RT viene mantenuto come stato dell'arte di riferimento per tutte le simulazioni condotte nel presente elaborato.

1.4 Metriche di valutazione della qualità video

Per quantificare in modo preciso e riproducibile le prestazioni di un algoritmo di compressione video, è necessario affiancare al bitrate una valutazione oggettiva della distorsione subita dall'immagine ricostruita. A tal fine, verranno utilizzate tre delle metriche oggettive di *Video Quality Assessment* (VQA) più diffuse per l'analisi dei codec:

- **PSNR** (*Peak Signal-to-Noise Ratio*): è una delle metriche di riferimento nella valutazione della qualità delle immagini e dei video. La differenza tra l'immagine originale e quella ricostruita viene quantificata attraverso l'errore quadratico medio (*Mean Squared Error*, MSE), dal quale viene calcolato il PSNR. Il risultato è espresso in decibel (dB) su scala logaritmica, dove un valore più alto indica un minore errore di ricostruzione e quindi una maggiore fedeltà rispetto al fotogramma di riferimento. Sebbene sia una metrica ampiamente adottata per la sua semplicità e rapidità di calcolo, la pura differenza matematica tra i pixel non sempre è allineata con il modo in cui il sistema visivo umano percepisce effettivamente il degrado, gli artefatti o la perdita di nitidezza.

- **MS-SSIM** (*Multi-Scale Structural Similarity Index Measure*) [4]: rappresenta un'estensione del classico indice SSIM, introdotto per superare i limiti del PSNR. Mentre SSIM valuta la similarità in termini di luminanza, contrasto e struttura dell'immagine anziché basarsi esclusivamente sulla differenza tra i singoli pixel, la variante **Multi-Scale** (MS) compie un passo ulteriore. Essa

espande l'analisi calcolando la somiglianza a diverse scale spaziali, tenendo conto di come l'occhio umano percepisce la qualità delle immagini a differenti distanze di osservazione e risoluzioni. L'algoritmo restituisce un valore compreso tra 0 e 1, dove 1 indica una perfetta corrispondenza strutturale con l'immagine di riferimento.

- **VMAF** (*Video Multimethod Assessment Fusion*): sviluppata da Netflix in collaborazione con il mondo accademico, questa metrica è stata progettata per fornire una stima della qualità video maggiormente correlata alla percezione umana. Oltre a includere caratteristiche puramente spaziali, il VMAF si distingue per l'integrazione di specifiche metriche temporali nell'analisi. Sfruttando un modello di *Machine Learning* addestrato su valutazioni soggettive umane, l'algoritmo combina diverse caratteristiche relative agli artefatti spaziali, alla conservazione dei dettagli e alle variazioni temporali tra i frame. Il punteggio finale, espresso su una scala da 0 a 100 dove valori più elevati indicano una qualità percepita maggiore, viene calcolato aggregando i valori dei singoli fotogrammi per restituire una valutazione complessiva dell'intera sequenza video [5].

Capitolo 2

Analisi preliminare e modularizzazione del codec

2.1 Analisi dell'architettura originale

Il punto di partenza per lo sviluppo dell'ambiente di simulazione è stato lo studio del codice sorgente fornito dagli autori del framework DCVC-RT. Pur appoggiandosi a componenti logiche e librerie strutturate in directory dedicate, lo script di esecuzione che ne coordinava l'utilizzo si presentava, all'atto pratico, come un blocco monolitico. Questa caratteristica ostacolava la flessibilità del sistema, rendendo difficile la sua modularizzazione, un passaggio tuttavia essenziale per poter isolare il bitstream intermedio e consentire la manipolazione e l'analisi dei dati in modo automatizzato.

L'analisi dettagliata dello script ha evidenziato come il codec sia basato su due autoencoder distinti, uno dedicato agli I-frame e l'altro per i P-frame. Nonostante l'obiettivo finale sia la compressione di un flusso video continuo, i modelli ragionano effettivamente per singolo fotogramma. Una caratteristica architetturale fondamentale della versione RT, che la distingue nettamente da tutti i modelli precedenti della famiglia DCVC, è l'assenza di un modulo esplicito per la stima e la compensazione del movimento (*Motion Estimation and Motion Compensation*, MEMC), sostituita interamente da una modellazione temporale di tipo implicito.

L'output generato dalla codifica neurale è un flusso di dati binari con estensione `.bin`. Analizzando le routine di lettura e scrittura a basso livello, si evince che il file è strutturato in una sequenza logica di pacchetti, richiamando l'approccio delle unità NAL (*Network Abstraction Layer*) tipiche dei codec tradizionali. Nello specifico, il file non presenta una singola intestazione globale e rigida, ma contiene uno o più pacchetti di configurazione (*Sequence Parameter Set*, SPS) dinamicamente, qualora i parametri di codifica subiscano variazioni. A questi fanno seguito i pacchetti dei fotogrammi veri e propri. Ciascun frame è introdotto da un header identificativo,

che ne specifica la natura intra o predittiva, l'indice dell'SPS di riferimento e il parametro di quantizzazione. A tale intestazione fa seguito il bitstream effettivo, il quale costituisce la codifica in formato binario della rappresentazione latente del fotogramma.

La gestione dell'equilibrio tra compressione e fedeltà visiva è demandata al parametro di quantizzazione, comunemente indicato in questo elaborato come *Quality Parameter* (QP), il quale accetta valori interi compresi in un range da 0 a 63. A differenza della logica dei codec tradizionali, l'implementazione del DCVC inverte la metrica: il valore 0 rappresenta il livello di qualità minima (massima compressione), mentre il valore 63 corrisponde alla qualità massima. L'impostazione del QP in fase di test viene gestita tramite gli argomenti dello script originale: è possibile fornire esplicitamente allo script una sequenza mirata di QP per gli I-frame e per i P-frame, utilizzando rispettivamente i parametri `qp_i` e `qp_p`. Qualora non vengano fornite queste liste, il campionamento viene delegato all'argomento `rate_num`, tramite il quale lo script calcola autonomamente i valori da testare, spaziandoli in modo lineare sull'intero intervallo disponibile. Ad esempio, impostando `rate_num` a 2, il sistema predispone i test solo per i due valori estremi 0 e 63; un valore pari a 4 genera invece automaticamente quattro bitstream a qualità equamente distanziate (0, 21, 42 e 63).

La definizione del GOP è regolata dal parametro `force_intra_period`: impostandolo al valore predefinito di -1, lo script inserisce un singolo I-frame all'inizio della sequenza, codificando tutti i fotogrammi successivi come P-frame. Tuttavia, il parametro permette di definire intervalli a piacimento (come 32 o 64), abilitando il test di diverse configurazioni.

La scelta delle sequenze video in input avviene attraverso un file di configurazione JSON, il cui percorso viene specificato a riga di comando tramite il parametro `test_config`. Per quanto riguarda i dati in ingresso, i modelli supportano nativamente sequenze video in formato YUV420 e PNG, sebbene tutte le simulazioni condotte in questo elaborato si siano concentrate in via esclusiva sul formato YUV. Il file fornito originariamente dagli autori struttura il dataset in classi di riferimento standard, tra cui UVG, MCL-JCV e le varie categorie HEVC. Nulla vieta, comunque, di definire classi personalizzate: l'ambiente di test può essere facilmente esteso aggiungendo le proprie sequenze YUV, a patto di specificare i metadati di base necessari al codec, come la risoluzione spaziale e il numero totale di fotogrammi. Infine, per ciascuna classe configurata, un apposito flag booleano consente di attivare o escludere dinamicamente l'intero gruppo di sequenze dall'elaborazione.

Per valutare la distorsione tra la sequenza originale e quella ricostruita, lo script calcola nativamente il PSNR e, attraverso l'opzione `calc_ssim`, permette di abilitare anche il calcolo dell'MS-SSIM. L'attivazione di questa metrica addizionale rende tuttavia il test più oneroso, incrementando sensibilmente il tempo di esecuzione.

2.2 Modularizzazione dell'ambiente di test

Come evidenziato nell'analisi preliminare, l'approccio d'esecuzione dello script originale costituiva un ostacolo per gli obiettivi di questo elaborato. Il software eseguiva infatti le fasi di codifica, calcolo delle metriche e decodifica in un unico flusso continuo. Tale struttura, pur essendo funzionale per test isolati, gestiva l'intera catena all'interno di un singolo processo. Sebbene fosse teoricamente possibile integrare logiche di manipolazione dei dati direttamente all'interno dello script monolitico, questo avrebbe compromesso la flessibilità dell'ambiente di test. Si è resa invece necessaria un'architettura che permettesse di interrompere il flusso logico per estrarre e isolare fisicamente il bitstream intermedio, accedendovi al di fuori dell'esecuzione originaria del programma, prima di avviare in modo indipendente la fase di decodifica e ricostruzione visiva.

Per abilitare questa modularità, si è proceduto a un profondo refactoring del codice, disaccoppiando l'architettura in script indipendenti, eseguibili manualmente in successione. La nuova organizzazione del flusso di lavoro si articola in tre componenti software principali:

- **Modulo di compressione (`compress.py`)**: costituisce il primo stadio della catena. Riceve in input le sequenze video non compresse (limitate per le simulazioni al formato nativo YUV420) sulla base del file di configurazione JSON e applica gli autoencoder neurali per la riduzione spaziale e temporale. L'output principale generato da questo modulo è un file binario (`.bin`). Rispetto alla versione originale, lo script è stato esteso per salvare in automatico lo stato della simulazione: al termine della codifica, genera un file di configurazione generale (`global_args.json`) e un file manifesto (identificato dal suffisso `_manifest.json`) per ogni combinazione di sequenza e parametri testata. Questi file di supporto sono essenziali per trasferire i metadati e le esatte impostazioni ai moduli successivi, garantendo così la coerenza dell'intera pipeline.

- **Modulo di corruzione (`corrupt.py`)**: è un modulo opzionale introdotto per la manipolazione del bitstream intermedio. Lo script intercetta il file binario generato dallo stadio precedente e vi applica un pattern di errore, alterando i dati prima della ricostruzione. Il modulo genera un nuovo file binario corrotto, preservando l'originale, e aggiorna la configurazione generale con i metadati sull'errore iniettato. Tale strumento consente di definire con precisione il fotogramma bersaglio, la modalità di corruzione e l'intensità in percentuale dell'alterazione da applicare.

- **Modulo di decompressione (`decompress.py`)**: chiude il flusso di elaborazione. Riceve in ingresso i file binari ed esegue la decodifica per ricostruire la

sequenza video. Sfruttando i metadati generati durante la fase di codifica, lo script è in grado di utilizzare dinamicamente il bitstream originario o quello manipolato, attivando il parametro dedicato `use_corrupted_bin`. In questa fase il modulo si occupa anche del calcolo delle metriche oggettive di qualità, salvando le statistiche finali per la valutazione delle prestazioni e del degrado visivo.

A supporto della catena di test, sono stati implementati ulteriori script accessori per l'automazione del flusso di lavoro: uno script dedicato alla pulizia delle directory e al reset dell'ambiente tra diverse sessioni di test (`clean_outputs.sh`), oltre a un modulo di conversione (`export_video.py`), progettato per transcodificare automaticamente le sequenze YUV ricostruite nel più accessibile formato compresso AVC/MP4, al fine di agevolare una rapida revisione qualitativa dei risultati.

Questa riorganizzazione architetturale ha permesso di intervenire sui singoli fotogrammi nel bitstream e, parallelamente, di ottimizzare i tempi complessivi di esecuzione. Isolando la fase di compressione, che richiede tempi di elaborazione non trascurabili, è diventato infatti possibile eseguire test di corruzione iterativi sullo stesso file `.bin` senza dover ricalcolare l'intera codifica neurale a ogni esecuzione.

2.3 Analisi dei costi: I-frame vs P-frame

Per valutare in modo accurato la resilienza dell'architettura neurale in scenari di rete ostili, è essenziale quantificare preventivamente il costo, in termini di banda, generato dall'introduzione di I-frame aggiuntivi. Come in qualsiasi standard di compressione video, anche nel DCVC-RT l'I-frame interrompe la propagazione dell'errore, azzerando la dipendenza temporale, ma ciò richiede una quantità di bit significativamente maggiore rispetto a un P-frame.

2.3.1 Impatto sul bitrate

L'analisi preliminare si è concentrata su due sequenze appartenenti alla Classe D del dataset ufficiale HEVC Common Test Conditions (CTC) [6], caratterizzate da dinamiche visive differenti: BQSquare e RaceHorses. Assumendo come caso ideale un GOP di 64 fotogrammi, i dati evidenziano come il peso relativo di un I-frame vari sensibilmente in funzione della complessità della scena e del parametro QP.

Nella sequenza BQSquare, a un livello di compressione elevato (QP 16), un I-frame risulta essere 22.2 volte più oneroso di un P-frame (0.184 bpp contro 0.008 bpp). Il rapporto decresce all'aumentare della qualità desiderata: impostando il codec alla minima compressione (QP 63), il divario si riduce a un fattore di 8.5×. Una dinamica simile si riscontra in RaceHorses, dove il rapporto passa da 7.1× (QP 16) a 4.2× (QP 63). Ciò dimostra che il costo relativo per l'inserimento di

un I-frame è inversamente proporzionale alla qualità complessiva del flusso video elaborato. I dati confermano empiricamente anche un altro aspetto: la dinamicità della scena influisce sul costo dei P-frame. Il maggiore movimento in RaceHorses, infatti, richiede più bit per codificare i continui cambiamenti visivi, riducendo sensibilmente il divario di peso con gli I-frame rispetto a una sequenza più statica come BQSquare.

2.3.2 Impatto della riduzione del GOP su resilienza e bitrate

Sebbene la forte compressione dei P-frame garantisca un elevato risparmio di banda, essa espone il flusso video a una grave vulnerabilità. In presenza di errori o perdite durante la trasmissione di rete, la mancata decodifica di un fotogramma genera artefatti visivi che possono propagarsi a cascata sui frame successivi dipendenti da esso. In un'architettura priva di meccanismi di feedback o ritrasmissione, l'unico modo per interrompere tale propagazione consiste nell'attendere la ricezione del successivo I-frame, che consente di ristabilire un punto di riferimento indipendente dai fotogrammi precedenti.

Poiché la distanza temporale tra due I-frame è dettata esclusivamente dalla dimensione del GOP, esiste una stretta correlazione tra quest'ultima e la tolleranza teorica agli errori di rete: per limitare la durata degli artefatti visivi, la frequenza di aggiornamento deve aumentare al peggiorare delle condizioni della rete. Seguendo questa logica, per gestire condizioni di rete via via più degradate, si passa da un caso ideale con GOP composto da 64 fotogrammi a un GOP di 32 e 16 frame, fino ad arrivare a un GOP di soli 8 frame, riducendo progressivamente l'intervallo temporale entro il quale l'errore può propagarsi. I valori percentuali derivati da queste riduzioni ($1/64 = 1.56\%$, $1/32 = 3.12\%$, $1/16 = 6.25\%$ e $1/8 = 12.50\%$) devono pertanto essere interpretati rigorosamente come proporzioni teoriche della frequenza di aggiornamento del frame di riferimento.

La riduzione del GOP introduce tuttavia un inevitabile compromesso: se da un lato un intervallo più breve tra gli I-frame migliora la resilienza del flusso agli errori, dall'altro penalizza l'efficienza di compressione. Come si evince dai dati riportati nella Tabella 2.1, la progressiva riduzione del GOP comporta un aumento del costo di trasmissione, dovuto proprio alla maggiore frequenza di inserimento degli I-frame. Nel caso estremo di BQSquare ad alta compressione (QP 16), il passaggio a un GOP composto da 8 frame determina un incremento del bitrate pari al 298.2% rispetto al caso ideale.

I dati mostrano tuttavia che l'aumento del bitrate varia a seconda della sequenza e del livello di compressione. Gli incrementi maggiori si registrano nelle sequenze più statiche e nei casi di maggiore compressione. In questi scenari, infatti, i P-frame possono essere rappresentati con un numero estremamente ridotto di bit; di

conseguenza, l'inserimento frequente di I-frame incide in misura molto più rilevante sul bitrate complessivo. Al contrario, nelle sequenze caratterizzate da un elevato movimento e da una compressione meno aggressiva, come RaceHorses a QP 63, il costo relativo degli I-frame si assottiglia e l'incremento percentuale del bitrate è pertanto più contenuto.

Sequenza	**QP**	**Metrica**	**GOP 64**	**GOP 32**	**GOP 16**	**GOP 8**
BQSquare	16	bpp (Δ%)	0.0112	+58.3%	+148.7%	+298.2%
		PSNR	30.46	30.70 (*+0.24*)	30.59 (*+0.13*)	30.35 (*-0.11*)
		VMAF	79.3	81.3 (*+2.0*)	80.9 (*+1.6*)	79.7 (*+0.4*)
	63	bpp (Δ%)	0.1885	+15.1%	+41.0%	+89.8%
		PSNR	36.37	36.43 (*+0.06*)	36.50 (*+0.13*)	36.64 (*+0.27*)
		VMAF	97.4	97.4 (*+0.0*)	97.5 (*+0.1*)	97.6 (*+0.2*)
RaceHorses	16	bpp (Δ%)	0.0144	+16.0%	+39.2%	+84.0%
		PSNR	29.53	29.71 (*+0.18*)	29.84 (*+0.31*)	30.05 (*+0.52*)
		VMAF	57.3	58.9 (*+1.6*)	60.1 (*+2.8*)	61.6 (*+4.3*)
	63	bpp (Δ%)	0.2547	+6.4%	+17.6%	+39.9%
		PSNR	38.14	38.22 (*+0.08*)	38.30 (*+0.16*)	38.48 (*+0.34*)
		VMAF	99.5	99.6 (*+0.1*)	99.6 (*+0.1*)	99.6 (*+0.1*)

Tabella 2.1: Impatto sul bitrate (bpp) e sulla qualità visiva (PSNR e VMAF) riducendo il GOP rispetto al caso ideale (GOP 64), valutato a livelli di qualità distanti (QP 16 e QP 63).

Nota: i valori tra parentesi indicano la differenza (in dB per il PSNR e in punti per il VMAF) rispetto al caso ideale.

2.3.3 Impatto sulle metriche di qualità

L'enorme spesa in termini di risorse di rete non trova giustificazione in un corrispettivo guadagno qualitativo. A qualità massima (QP 63), il passaggio da un GOP pari a 64 fotogrammi a uno di 8 in RaceHorses fa lievitare la banda del +39.9%, a fronte di un incremento delle metriche del tutto trascurabile: +0.1 punti di VMAF e +0.34 dB di PSNR.

Più indicativo è il comportamento di BQSquare ad alta compressione (QP 16): nonostante un incremento di banda di quasi il 300%, dovuto all'inserimento di un I-frame ogni 8 fotogrammi, la qualità globale subisce un effettivo peggioramento rispetto alle configurazioni intermedie. Il PSNR scende al di sotto del valore del caso ideale, con una variazione di -0.11 dB, mentre il VMAF, dopo aver raggiunto un picco impostando il GOP a 32 fotogrammi, subisce a sua volta un calo.

I risultati esposti mostrano come l'inserimento troppo frequente di I-frame non rappresenti una soluzione efficace negli scenari real-time soggetti a perdite di dati. Si rende pertanto necessario esplorare strategie di mitigazione alternative che non gravino in modo significativo sul costo trasmissivo. Come verrà discusso nel prossimo capitolo, un primo tentativo in questa direzione consisterà nell'applicazione di tecniche di error concealment in fase di decodifica.

Capitolo 3

Studio della resilienza ed error concealment

3.1 Tecniche di corruzione del bitstream

Come introdotto nel capitolo precedente, l'indagine preliminare sulla resilienza del codec si è avvalsa del modulo `corrupt.py`. In questa fase lo scopo è isolare e studiare la reazione del decoder a fronte di una singola anomalia. Per questo motivo, il modulo interviene una sola volta all'interno del file binario compresso (`.bin`), applicando la corruzione esclusivamente su un singolo fotogramma bersaglio, prima della fase di decodifica.

Per valutare il comportamento del codec in questi scenari isolati, lo script richiede l'attivazione di una specifica modalità di esecuzione. Tramite gli argomenti a riga di comando, oltre a definire l'indice esatto del fotogramma bersaglio da corrompere, è possibile selezionare una delle seguenti opzioni di alterazione:

- **Azzeramento totale** (opzione `corrupt_entire_frame`): tecnica che simula la perdita totale del flusso di bit associato a un fotogramma. A livello implementativo, il modulo individua l'intestazione del frame bersaglio all'interno del bitstream e sovrascrive l'intera porzione di dati successiva con bit impostati a zero, mantenendo tuttavia inalterata la lunghezza complessiva del segmento originario, così da non invalidare la lettura sequenziale del file.
- **Corruzione percentuale** (opzione `corrupt_frame_percentage`): strategia che modella la ricezione di un fotogramma parzialmente danneggiato. Fornendo in input la percentuale di corruzione desiderata, lo script calcola il numero esatto di bit da alterare sul totale del fotogramma. Successivamente, selezionando in modo pseudo-casuale la loro posizione all'interno del segmento binario del frame, forza l'inversione logica dei singoli bit.

3.2 Impatto sul codec DCVC-RT e propagazione dell'errore

L'architettura di test sviluppata è stata inizialmente impiegata per condurre una serie di simulazioni mirate sul DCVC-RT. Le prove hanno coinvolto sequenze a diverse risoluzioni, attingendo alle classi HEVC_D (416×240) e HEVC_B (1920×1080), al fine di verificare se la densità di informazione e la complessità spaziale della scena potessero influenzare la resilienza del codec.

Per garantire la coerenza dell'analisi, i flussi video sono stati generati utilizzando parametri di codifica uniformi, con QP impostato a 63, corrispondente alla minima compressione, e definendo un GOP della durata di 64 fotogrammi. Il modulo di corruzione è stato impostato per intervenire sul fotogramma con indice 10 (l'undicesimo della sequenza, considerando l'indicizzazione a partire da zero), iniettando una corruzione pari allo 0.1% dei bit che lo compongono.

L'effettiva entità della percentuale di corruzione introdotta è riportata nei log di esecuzione del modulo: operando a qualità massima, sulla sequenza BQTerrace (HEVC_B) lo 0.1% si traduce nell'inversione logica di appena 179 bit su un totale di 179.344 presenti all'interno del fotogramma bersaglio, mentre su BasketballPass (HEVC_D) la corruzione si attesta a 6 bit su 6.040.

I risultati emersi dall'analisi visiva hanno mostrato un'evidente fragilità del codec neurale nei confronti di questo tipo di alterazione. Il decodificatore, trovandosi a processare un input contenente bit alterati, perde rapidamente la propria coerenza interna. Dal punto di vista percettivo, l'impatto iniziale sul fotogramma bersaglio risulta estremamente localizzato: nella sequenza BasketballPass, ad esempio, l'alterazione si manifesta unicamente come un lieve artefatto sulla scarpa del giocatore di sinistra, lasciando intatto il resto della scena.

Tuttavia, la principale criticità del codec risiede nella propagazione temporale del difetto. I fotogrammi predittivi successivi ereditano il disallineamento e, in assenza di meccanismi di recupero, il degrado tende ad amplificarsi. Esso si accumula progressivamente per tutta l'estensione del GOP, raggiungendo l'apice sul fotogramma 63, l'ultimo istante predittivo prima dell'inizio di un nuovo GOP.

Come si nota in Figura 3.1, l'evoluzione del degrado è strettamente dipendente dalla dinamica della scena. In sequenze ad alto movimento, come BasketballPass, l'errore genera un effetto di trascinamento inarrestabile che disgrega completamente le geometrie. Al contrario, in scene caratterizzate da uno sfondo fisso e minore dinamicità, come BQTerrace, la struttura globale del fotogramma riesce a preservarsi: l'errore non distrugge l'intera scena, ma rimane confinato sotto forma di evidenti chiazze nere, che persistono e si evolvono unicamente in corrispondenza degli elementi in movimento, come il flusso dell'acqua o le persone. In entrambi i casi, un difetto originatosi in pochi bit compromette irrimediabilmente la fedeltà

visiva dell'intero intervallo predittivo.



Figura 3.1: Propagazione dell'errore per il codec DCVC-RT (GOP 64, QP 63, corruzione del frame 10 dello 0.1%). In alto la sequenza BasketballPass, in basso BQTerrace. A sinistra il fotogramma bersaglio (frame 10), a destra lo stesso flusso al fotogramma 63.

3.3 Tentativi di error concealment sul DCVC-RT

Preso atto dell'entità del degrado, la fase successiva dell'indagine si è concentrata sull'applicazione di tecniche di error concealment nel tentativo di ripristinare la coerenza visiva della sequenza. In primo luogo, si è cercato di intervenire direttamente sui metodi di decompressione degli autoencoder. Poiché la ricostruzione del fotogramma avviene a partire da un formato fortemente compresso, si è tentato di manipolare direttamente le rappresentazioni interne e lo spazio latente, con l'obiettivo di attenuare le anomalie prima che si propagassero ai fotogrammi successivi.

Tuttavia, nonostante le modifiche apportate alla logica del decodificatore neurale, nessuna delle tecniche attive considerate ha prodotto miglioramenti visivi significativi: la propagazione dei difetti spaziali e di trascinamento è rimasta pressoché identica a quella osservata in assenza di error concealment. L'unica tecnica in grado di offrire una mitigazione quantomeno stabile si è rivelata di natura puramente passiva. Essa consiste nello scartare integralmente il fotogramma identificato come

corrotto e nel riproporre l'ultimo fotogramma decodificato con successo, creando un effetto percettivo di "congelamento" della scena video, da cui il nome *Frame Freeze*.

Sebbene questa tecnica eviti la visualizzazione degli artefatti, la sua applicazione su un flusso video configurato con un GOP di 64 fotogrammi si è dimostrata del tutto impraticabile. Qualora la corruzione interessi un fotogramma posto nelle fasi iniziali dell'intervallo predittivo, l'approccio *Frame Freeze* costringe il decoder a mantenere l'immagine statica fino alla ricezione del successivo I-frame. L'effetto è quello di un video bloccato per quasi l'intera durata del GOP, uno scenario incompatibile con i requisiti minimi di fluidità di una riproduzione multimediale.

L'inefficacia di questi tentativi ha portato a ipotizzare che una delle possibili cause risiedesse nell'assenza, in DCVC-RT, di un modulo esplicito di stima e compensazione del movimento (MEMC) direttamente accessibile. Lavorando con una rappresentazione implicita del moto, senza vettori di movimento espliciti su cui intervenire, risulta infatti impossibile isolare la porzione di immagine alterata. Per verificare se la disponibilità di informazioni di moto potesse rendere più efficaci le strategie di error concealment, l'analisi è stata quindi estesa al codec DCVC-FM [7], la versione immediatamente precedente al DCVC-RT, caratterizzata dalla presenza di un modulo MEMC esplicito.

3.4 Test comparativi sul codec DCVC-FM

L'obiettivo iniziale era quello di sfruttare l'accesso diretto ai vettori di moto per implementare strategie attive di error concealment, tentando di estrapolare e correggere la dinamica della scena per mascherare la corruzione. Tuttavia, i numerosi tentativi di manipolare, in modo mirato, il movimento all'interno del decodificatore si sono rivelati inefficaci. Per un'indagine preliminare sulla resilienza del codec, è stato condotto un test base iniettando una corruzione minima, pari all'alterazione di un singolo bit in un fotogramma posto nelle fasi iniziali dell'intervallo predittivo. Lasciando propagare liberamente l'errore senza alcuna mitigazione, gravi distorsioni a macchia si sono manifestate quasi istantaneamente, seguite in breve tempo da un collasso totale del processo di decodifica, il quale ha iniziato a restituire interi fotogrammi di colore verde.

A fronte dell'impossibilità di correggere attivamente il difetto, l'indagine si è spostata su una tecnica di mitigazione in fase di decodifica, basata sull'azzeramento forzato degli stati interni. Questo approccio mira a emulare i principi del *Frame Freeze* visti nel paragrafo precedente, operando, tuttavia, a un livello più profondo. In corrispondenza del fotogramma corrotto, il sistema interviene a livello visivo scartando i dati difettosi e forzando a schermo l'ultimo fotogramma valido recuperato dal DPB. Parallelamente, per impedire la propagazione dell'errore negli

stati interni, le informazioni ereditate dalla MEMC vengono forzatamente azzerate. Alla base di questo intervento vi era l'ipotesi che, ricevendo i flussi di bit dei successivi P-frame sani, il codec fosse in grado di riprendere la normale decodifica, ancorandosi all'immagine congelata e rigenerando progressivamente le *feature* dello spazio latente, ripristinando così il movimento e la coerenza della scena.

L'applicazione pratica di questa tecnica si è tuttavia limitata a ritardare l'inevitabile. Poiché il bitstream alterato viene scartato integralmente a prescindere dall'entità del danno, l'intervento riesce inizialmente a mascherare l'errore, mantenendo l'immagine statica. Come si evince analizzando la sequenza BasketballPass in Figura 3.2, nell'esperimento, analogo a quello illustrato nel Paragrafo 3.2, il fotogramma 10 costituisce il punto di applicazione della tecnica; al fotogramma 31 la scena non è ancora collassata e mantiene una coerenza globale, seppur con la comparsa delle prime macchie cromatiche e distorsioni geometriche.

Tuttavia, questa stabilità apparente si rivela puramente transitoria. A partire dal fotogramma 48, si innesca un degrado che origina dalla base dell'immagine e si espande progressivamente a macchia. Al fotogramma 52 (Figura 3.2, al centro) è ben visibile come la macchia si sia ulteriormente allargata, compromettendo gran parte del contenuto visivo. Il degrado raggiunge il livello massimo poco dopo: dal fotogramma 53 fino alla chiusura del GOP (fotogramma 63, Figura 3.2, a destra), il decodificatore non è più in grado di produrre una ricostruzione coerente e restituisce un'immagine verde uniforme. Questo comportamento, analogo a quello riscontrato nel test senza error concealment, evidenzia come l'intervento sugli stati interni non sia sufficiente a impedire la propagazione dell'errore e il successivo degrado della ricostruzione.



Figura 3.2: Evoluzione del degrado con error concealment su DCVC-FM (sequenza BasketballPass): da sinistra a destra, fotogrammi 31, 52 e 63.

Infine, l'estrazione delle metriche dai log ha rilevato un'ulteriore criticità per lo scenario di riferimento. Pur considerando che le tempistiche assolute dipendono intrinsecamente dall'hardware utilizzato, i test condotti sulla sequenza in esame mostrano come il DCVC-FM risulti circa dieci volte più lento rispetto alla versione RT. Tale divario prestazionale, unito all'estrema instabilità in fase di decodifica a fronte di errori, ne conferma la totale impraticabilità per servizi di trasmissione real-time.

I risultati ottenuti hanno evidenziato come le strategie di error concealment applicate direttamente al processo di decodifica non siano sufficienti a impedire la propagazione dell'errore. Il semplice mascheramento del fotogramma corrotto, infatti, non interrompe le dipendenze temporali che coinvolgono i successivi P-frame. Si è quindi reso necessario adottare un approccio differente, nel quale la rilevazione della perdita inneschi un meccanismo di recupero a livello di codifica, attraverso la generazione di un nuovo I-frame. Da questa esigenza nasce lo sviluppo di un framework in grado di simulare scenari di trasmissione caratterizzati da perdita di dati e ritardo del canale di feedback, descritto nel capitolo successivo.

Capitolo 4

Sviluppo del framework di simulazione per scenari real-time

4.1 Architettura del framework

4.1.1 Motivazione

Sebbene la modularizzazione dell'ambiente di test, descritta nel Capitolo 2, abbia permesso di superare la natura monolitica del codice originale, isolando le diverse fasi di elaborazione, l'architettura risultante rimaneva intrinsecamente statica. La pipeline sviluppata consentiva infatti di manipolare il bitstream e iniettare errori per analizzarne gli effetti sulla fase di decodifica, ma non forniva alcun meccanismo per variare il tipo di fotogramma generato in risposta a eventi esterni.

Come evidenziato dai risultati del capitolo precedente, le strategie puramente passive non sono sufficienti. Per ripristinare la coerenza visiva è quindi necessario che la rilevazione di un difetto possa generare una risposta alla sorgente. Il framework sviluppato introduce a questo scopo un canale di feedback tra ricevitore e trasmettitore, permettendo di gestire dinamicamente le condizioni di perdita e di mantenere il controllo e la riproducibilità degli esperimenti.

4.1.2 Orchestratore del framework

Il cuore di questa nuova architettura è rappresentato dallo script `run_simulator.py`, che ha il compito di iterare in totale autonomia le diverse configurazioni, incrociando i parametri qualitativi, i tassi di perdita e le latenze di rete simulate, senza richiedere alcun intervento manuale tra un'esecuzione e l'altra.

Alcuni dei parametri di configurazione derivano direttamente dallo script originale del codec, opportunamente adattato al nuovo framework. Altri sono stati aggiunti, come il *Packet Loss Rate* (PLR), il quale rappresenta in percentuale la gravità delle perdite e il ritardo M, che quantifica in frame la latenza del canale di feedback.

Una volta acquisiti gli input, `run_simulator` gestisce l'esecuzione sequenziale dei vari sottomoduli, automatizzando l'intero processo per ogni singola iterazione.

4.1.3 Gestione del flusso dati

La comunicazione tra le diverse parti del framework segue un flusso logico ben definito:

- Inizialmente, l'orchestratore avvia lo script `generate_patterns.py`, il quale genera i file testuali contenenti le sequenze di perdita, in cui `0` rappresenta i frame ricevuti e `1` quelli persi. Sulla base di questi pattern e del ritardo di rete impostato, lo script calcola la successione esatta degli I-frame e dei P-frame.

- Successivamente, viene richiamato `compress.py`, derivato dalla scomposizione del codice originale e incaricato di generare il bitstream delle sequenze. Durante questa fase, per ogni sequenza elaborata e per ciascuna combinazione di parametri ricevuta dall'orchestratore, il sistema genera un file JSON dedicato, che archivia tutti i metadati e i parametri utilizzati per quella specifica codifica.

- Infine, il modulo di decompressione (`decompress.py`) legge il file JSON per garantire la perfetta coerenza dei parametri tra la fase di codifica, la ricostruzione visiva e il calcolo delle metriche.

4.2 Simulazione del canale di feedback mediante il ritardo M

La valutazione del codec in scenari di rete realistici richiede necessariamente la modellazione di dinamiche di recupero degli errori. A tal fine, l'architettura di test integra un meccanismo di feedback che simula la comunicazione bidirezionale tra il decodificatore (ricevitore) e il codificatore (mittente). L'obiettivo è interrompere la propagazione del degrado visivo, forzando il codificatore nel trasmettere un nuovo I-frame non appena viene notificata l'avvenuta corruzione del flusso video.

La variabile che modella questo meccanismo è il ritardo M, parametro che quantifica, in numero di frame, il tempo necessario affinché il mittente riceva la notifica della perdita, generi un nuovo I-frame e lo trasmetta al destinatario.

Dal punto di vista implementativo, l'architettura simula offline una comunicazione bidirezionale, riproducendo l'intera logica di feedback. Il compito di tradurre questa latenza è affidato allo script `generate_patterns.py`. Oltre a generare autonomamente le tracce di perdita, il modulo le analizza e, al verificarsi della perdita di un fotogramma, forza la codifica di un I-frame esattamente M frame successivi. L'output di questa elaborazione produce a sua volta delle tracce che fungono da input per il modulo di compressione. Quest'ultimo riceve in ingresso una timeline già perfettamente temporizzata e genera il bitstream, posizionando gli I-frame nelle esatte posizioni stabilite a monte da `generate_patterns`.

Questo approccio permette di osservare la risposta del codec sotto molteplici profili di latenza. Un valore di M ridotto modella scenari in cui l'architettura di rete gode di un canale di feedback estremamente reattivo. All'aumentare di M, la simulazione ricrea condizioni di congestione più severe, in cui il decoder è costretto ad affidarsi alla naturale capacità resiliente del codec per un numero prolungato di istanti, prima di poter beneficiare dell'aggiornamento mediante I-frame.

4.3 Integrazione del *Frame Freeze*

Il framework sviluppato supera i limiti evidenziati nel capitolo precedente integrando la richiesta di feedback con una logica di error concealment iterativa, configurabile tra i parametri di avvio del simulatore. Come emerso dalle analisi preliminari, intervenire direttamente sugli stati interni del modello neurale risulta inefficace; pertanto, tale soluzione opera a livello di gestione esterna dei fotogrammi in fase di decodifica, in modo del tutto trasparente rispetto ai meccanismi del codec. Questa strategia permette di mascherare il degrado durante il tempo di latenza necessario alla ricezione del nuovo I-frame.

In assenza di tale funzionalità, la gestione degli errori è passiva: durante l'esecuzione, il modulo di decodifica legge il pattern di errore e individua gli istanti di perdita. Se un fotogramma risulta mancante, indipendentemente dal fatto che si tratti di un I-frame o di un P-frame, lo script passa in ingresso al decoder un flusso di bit artificialmente azzerato. Di conseguenza, il codec è costretto a tentare la predizione basandosi su dati nulli, generando inevitabilmente artefatti visivi che si accumulano e si propagano sui fotogrammi successivi.

Attivando l'opzione di *Frame Freeze*, al momento della perdita, il decoder non riceve il flusso di dati azzerato, ma al contrario, lo script accede al *Decoded Picture Buffer* (DPB), ovvero la memoria in cui il codec conserva i fotogrammi decodificati in precedenza, a partire dall'ultimo I-frame, da impiegare per la decodifica dei successivi P-frame. Da qui il sistema estrae l'ultima immagine valida disponibile, la clona e la reinserisce nel DPB.

Avviene, quindi, una pura e semplice copia dei dati dell'immagine, producendo sull'osservatore un effetto percettivo di "congelamento" della sequenza video. L'operazione di copia viene reiterata ogni qualvolta si verifichi una perdita di un fotogramma.

4.4 Approccio Monte Carlo e gestione delle tracce multiple

Valutare l'impatto delle perdite, basandosi su un'unica simulazione per una combinazione specifica di parametri, produrrebbe un risultato fortemente aleatorio e poco affidabile, in quanto il degrado visivo è strettamente legato al contesto temporale in cui si verifica la perdita. Ad esempio, l'assenza di un fotogramma durante una scena caratterizzata da elevato movimento produce, in genere, una degradazione visiva più evidente rispetto a una scena prevalentemente statica.

Per abbattere tale incertezza, viene adottato un approccio basato sul metodo Monte Carlo. Tramite il parametro di input `num_traces` fornito allo script `run_simulator`, il framework genera molteplici scenari di perdita indipendenti per ciascuna configurazione di parametri di input. Così facendo, è possibile valutare le prestazioni medie del codec in modo affidabile, riducendo l'impatto dei casi isolati.

Infine, per garantire la totale riproducibilità degli esperimenti, la generazione pseudo-casuale dei pattern di perdita è vincolata a un seed deterministico, il cui valore può essere definito in fase di avvio dell'orchestratore.

4.5 Analisi delle metriche ed esportazione video

Al termine del processo di simulazione, il framework si avvale di due moduli dedicati all'analisi statistica (`analyze_simulations.py`) e visiva dei risultati (`export_video.py`).

Il primo modulo, oltre a collezionare le prestazioni ideali del codec (il caso base a PLR 0%) da usare come termine di paragone, raccoglie i risultati delle varie tracce di errore, ne calcola la media e ne analizza la varianza. Infine, genera in autonomia i grafici necessari per l'analisi:

- andamenti delle metriche al variare del PLR, per osservare il degrado qualitativo all'aumentare delle perdite;
- curve Rate-Distortion (RD), per valutare l'efficienza di compressione;
- diagrammi a scatola e baffi (boxplot), per visualizzare la dispersione dei dati e confermare l'affidabilità delle misurazioni.

Il secondo modulo, concepito come un'evoluzione dell'utility di base introdotta nel Capitolo 2, è stato esteso per gestire le molteplici tracce generate dalla simulazione e per integrare diverse opzioni da riga di comando. L'ispezione visiva degli artefatti e l'estrazione dei singoli fotogrammi avvengono direttamente sui file grezzi decompressi (in formato YUV), avvalendosi del software YUView[1], garantendo così un'analisi priva di ulteriori degradazioni. Tuttavia, per facilitare una rapida revisione qualitativa delle sequenze, lo script `export_video` automatizza la conversione dei file YUV nel più accessibile formato MP4, codificato in AVC utilizzando l'encoder libx264 tramite *FFmpeg*[2].

I dati statistici elaborati dal primo modulo, uniti all'ispezione visiva diretta sui file YUV, costituiscono la base per la valutazione delle prestazioni, esposta nel capitolo successivo.

[1] `https://github.com/IENT/YUView`

[2] `https://ffmpeg.org/`

Capitolo 5

Risultati sperimentali e analisi

5.1 Setup sperimentale

Prima di procedere con l'analisi dei risultati, è fondamentale definire i confini operativi entro cui è stata valutata l'efficacia del DCVC-RT. In questo paragrafo verrà pertanto descritta la configurazione dell'ambiente di test, definendo le caratteristiche delle sequenze video in esame e i parametri scelti per la valutazione.

5.1.1 Sequenze video utilizzate

I test sono stati eseguiti su un insieme di sequenze video non compresse (formato YUV 4:2:0), appartenenti alla Classe D del dataset HEVC CTC introdotto in precedenza. Tali sequenze, caratterizzate da una risoluzione WQVGA (416×240), sono state scelte per bilanciare l'elevata complessità computazionale richiesta dal codec neurale, con la necessità di condurre simulazioni statistiche su tracce multiple:

- **BasketballPass** (500 frame, 50 fps) e **RaceHorses** (300 frame, 30 fps), sequenze caratterizzate da elevato movimento e da continue occlusioni, utili per osservare la naturale capacità resiliente del codec su scene molto dinamiche.
- **BlowingBubbles** (500 frame, 50 fps), caratterizzata da un movimento irregolare e disomogeneo su gran parte dei fotogrammi, ma con uno sfondo prevalentemente statico.
- **BQSquare** (600 frame, 60 fps), caratterizzata da un movimento di camera molto lento e da un'elevata densità di particolari sullo sfondo.

5.1.2 Parametri di simulazione

Le condizioni di trasmissione non ideali sono state modellate usando i seguenti parametri:

- **Dimensione del GOP**: fissata a 32 fotogrammi.
- **Punti di qualità (QP)**: la codifica è stata eseguita su quattro livelli di compressione distinti; nello specifico sono stati scelti QP pari a 16, 32, 47 e 63. Si è scelto di escludere il parametro QP pari a 0 poiché, imponendo il massimo grado di compressione, restituisce un livello qualitativo inaccettabile per le finalità di questo studio.
- **Packet Loss Rate (PLR)**: sono stati considerati cinque diversi livelli di perdita dei dati, ovvero 1%, 5%, 10%, 15% e 20%.
- **Latenza di rete (parametro M)**: sono stati analizzati tre diversi livelli di ritardo di trasmissione, pari a 2, 6 e 10 fotogrammi.
- **Numero di tracce di perdita**: per ciascuna configurazione di parametri sono state generate dieci tracce di perdita pseudo-casuali, applicando un approccio di tipo Monte Carlo per l'estrazione delle medie statistiche, come discusso nel capitolo precedente.

5.1.3 Hardware e metriche di riferimento

Trattandosi di un framework basato su un codec neurale, le operazioni di codifica e decodifica sfruttano l'accelerazione hardware per velocizzare il carico computazionale richiesto. Tutte le simulazioni sono state eseguite su un sistema dotato di un processore Intel Core i7-7700 e una GPU NVIDIA GeForce GTX 1070 Ti.

Per valutare le prestazioni complessive e la qualità delle sequenze ricostruite, sono state utilizzate le metriche precedentemente introdotte: il bitrate, per quantificare l'efficienza di compressione, e PSNR, MS-SSIM e VMAF per misurare la qualità visiva rispetto al riferimento originale.

5.2 Resilienza naturale e sistema di feedback

In questo paragrafo si analizza il comportamento del DCVC-RT, valutando sia la sua capacità intrinseca di tollerare la perdita di dati, sia l'efficacia del sistema di feedback nel gestire l'invio di I-frame al variare del parametro M. I risultati presentati di seguito si concentrano su un valore di qualità intermedio (QP 47), assunto come caso di studio rappresentativo. Andamenti analoghi si riscontrano in modo coerente anche per gli altri livelli di compressione.

5.2.1 Andamento delle metriche

I grafici seguenti illustrano l'andamento delle tre metriche di qualità e del costo in bitrate all'aumentare del PLR. La linea orizzontale tratteggiata definisce il caso ideale, ovvero le prestazioni del sistema in assenza di perdite (PLR = 0%). Le tre curve mostrano invece la reazione del sistema al variare del ritardo di rete M, il quale regola la latenza di iniezione forzata di un I-frame. I punti lungo ciascuna curva corrispondono ai livelli di PLR simulati.

La Figura 5.1 mostra i risultati per la sequenza RaceHorses.

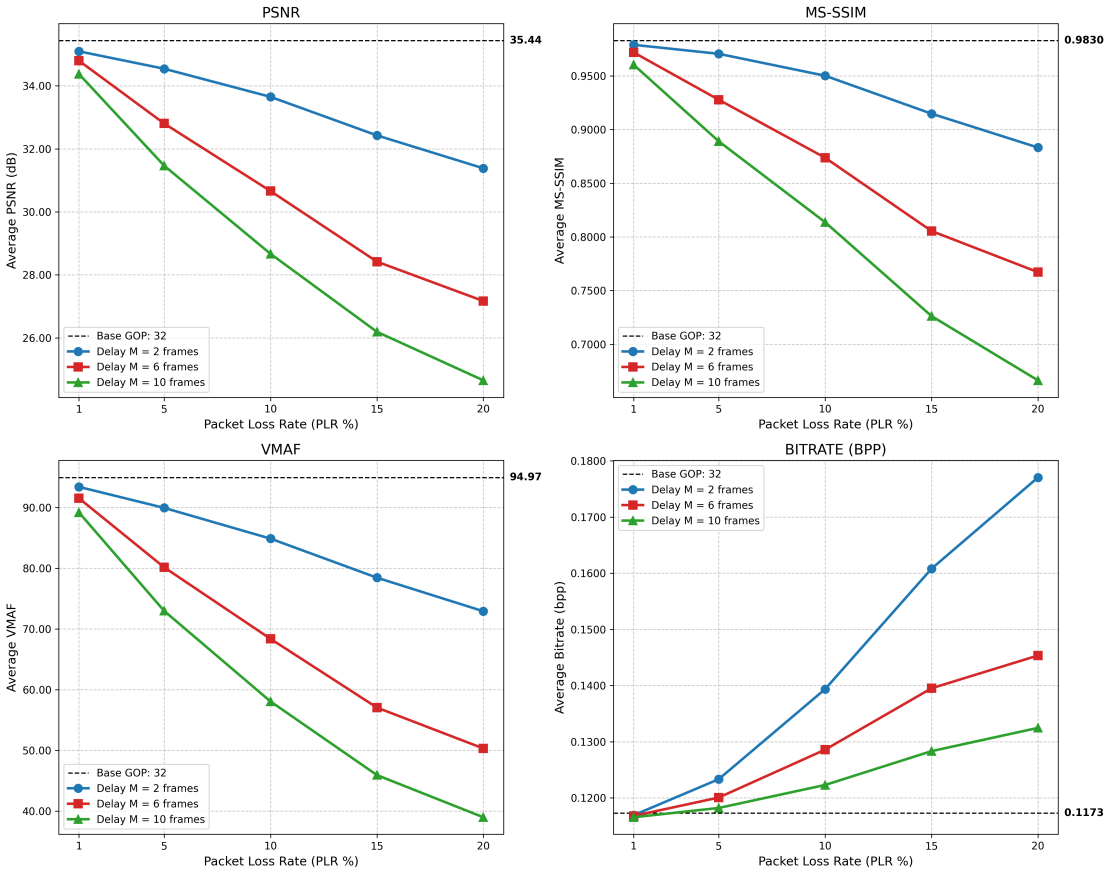


Figura 5.1: Andamento delle metriche di qualità e del bitrate al variare del PLR e del ritardo di rete M per la sequenza RaceHorses (QP=47).

Analizzando le curve, si nota come ad alta reattività ($M = 2$) il sistema intervenga tempestivamente, limitando la propagazione dell'errore a una finestra di soli due fotogrammi. Questo si traduce in metriche qualitative che resistono bene anche a PLR elevati. Al contrario, imponendo un ritardo di rete maggiore, si osserva un crollo qualitativo, poiché l'errore ha il tempo di propagarsi spazialmente e

temporalmente prima dell'arrivo di un nuovo I-frame. Questo degrado è catturato in modo particolare dalla metrica VMAF, che penalizza gli artefatti percepibili dall'occhio umano in modo molto più severo rispetto alle altre metriche.

Si pone quindi a confronto questo comportamento con una sequenza a minore dinamicità, come BQSquare, illustrata in Figura 5.2.

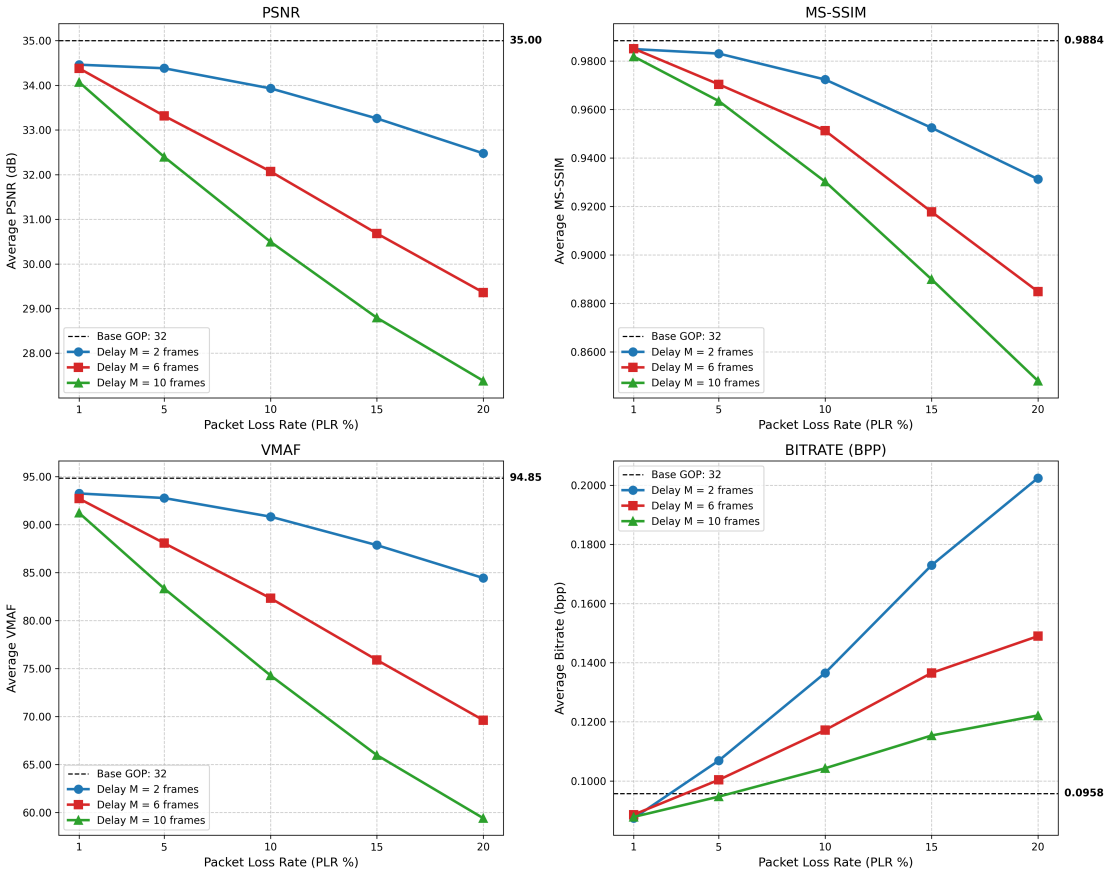


Figura 5.2: Andamento delle metriche di qualità e del bitrate al variare del PLR e del ritardo di rete M per la sequenza BQSquare (QP=47).

Il confronto diretto tra le due sequenze dimostra l'impatto del movimento temporale. In assenza di perdite, entrambe le sequenze partono da metriche qualitative pressoché identiche; l'unica differenza risiede nel bitrate intrinsecamente maggiore per la prima, a causa dell'elevata dinamicità della scena. Non appena si introduce una percentuale di perdita, tuttavia, le metriche di BQSquare si mantengono sistematicamente superiori per qualsiasi livello di PLR. Questo andamento valida la teoria secondo la quale un'elevata staticità dello sfondo frena la naturale propagazione dell'errore nella decodifica del DCVC-RT.

Passando all'analisi del costo di rete (bitrate, nei grafici in basso a destra), si nota un comportamento controintuitivo, particolarmente evidente per la sequenza BQSquare: a tassi di perdita molto bassi (PLR = 1%), il sistema registra un bitrate medio inferiore rispetto al caso ideale. Questo accade perché l'iniezione imprevista di un I-frame, per coprire un errore, interrompe il normale ciclo di codifica preimpostato (GOP). Di conseguenza, il codec compensa questa anomalia applicando automaticamente una compressione più aggressiva ai P-frame immediatamente successivi, generando un risparmio di dati tale da compensare il peso dell'I-frame aggiunto. Al contrario, ai livelli di PLR più alti (15-20%), la continua e ravvicinata iniezione di fotogrammi di tipo I satura la banda, provocando l'esplosione del bitrate complessivo.

Si nota, infine, come una latenza maggiore riesca a contenere parzialmente tale esplosione rispetto a una bassa latenza, proprio perché il sistema attende più a lungo prima di forzare l'invio di nuovi e pesanti I-frame.

5.2.2 Varianza statistica

I grafici presentati nel paragrafo precedente mostrano l'andamento medio delle metriche, calcolato aggregando i risultati delle dieci simulazioni per ciascun livello di PLR. Tuttavia, un'analisi basata esclusivamente su questo singolo dato non è sufficiente a descrivere appieno il comportamento del sistema in reti non ideali. Poiché la perdita di dati in una rete avviene in modo del tutto casuale, l'impatto di un medesimo PLR può variare radicalmente a seconda degli specifici fotogrammi colpiti. Per questo motivo, l'approccio statistico basato sul metodo Monte Carlo è stato approfondito analizzando i risultati attraverso l'uso di diagrammi a scatola e baffi, anche detti boxplot.

Questi grafici permettono di visualizzare la distribuzione dei risultati sulle dieci tracce, evitando così di limitarsi al solo dato medio. La "scatola" centrale racchiude il 50% dei dati (l'intervallo interquartile), indicando al suo interno la mediana, mentre i "baffi" si estendono fino ai valori minimi e massimi, escludendo eventuali valori anomali estremi chiamati *outlier*, rappresentati come punti isolati.

Al fine di offrire una visione sia comparativa che di dettaglio, i risultati vengono presentati in due formati complementari. La Figura 5.3 mostra una vista raggruppata per confrontare direttamente l'impatto delle diverse latenze, mentre le Figure 5.4, 5.5 e 5.6 offrono una visualizzazione separata per ciascun valore di M.

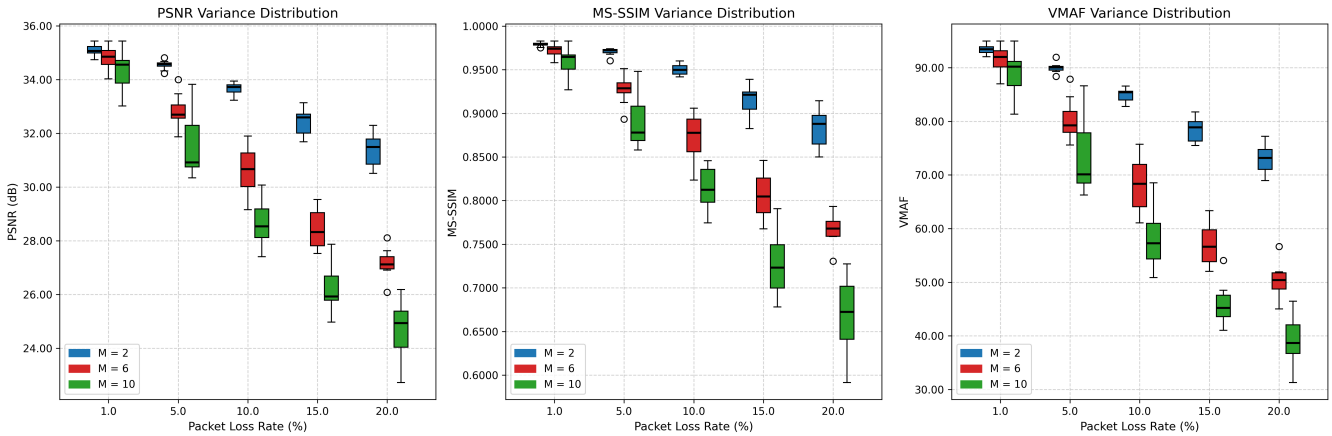


Figura 5.3: Dispersione statistica raggruppata sulle dieci tracce simulate per la sequenza RaceHorses (QP=47), per un confronto diretto tra le latenze M=2, M=6 e M=10.

Nota: per evitare la sovrapposizione visiva, i diagrammi relativi ai diversi valori di M sono stati affiancati e leggermente traslati sull'asse orizzontale; tutte le scatole di ciascun raggruppamento si riferiscono rigorosamente all'esatto valore di PLR indicato al centro.

Come si può notare, all'aumentare del PLR non solo si assiste a un calo delle metriche, come già descritto nel paragrafo precedente, ma anche a un incremento della dispersione dei risultati, il cui impatto è strettamente legato al ritardo di rete. Per analizzare più nel dettaglio queste variazioni e isolare il comportamento di ciascuna latenza, è utile esaminare le singole dispersioni.

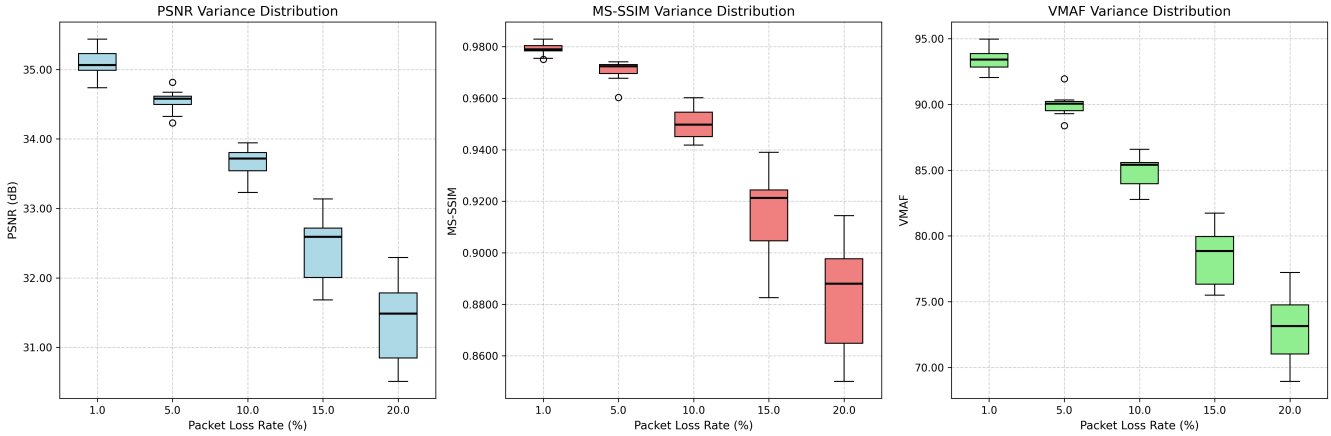


Figura 5.4: Dispersione statistica delle metriche per la sequenza RaceHorses (QP=47) per M=2.

Nella Figura 5.4 le distribuzioni rimangono compatte per i livelli di perdita

fino al 10%. A queste soglie, l'intervento rapido del sistema annulla quasi del tutto l'effetto casuale delle perdite. Tuttavia, ai tassi di PLR del 15% e 20%, i boxplot si allargano e mostrano una maggiore dispersione dei dati. In ogni caso, pur aumentando, questa variabilità risulta molto più contenuta rispetto agli scenari con ritardi di rete maggiori.

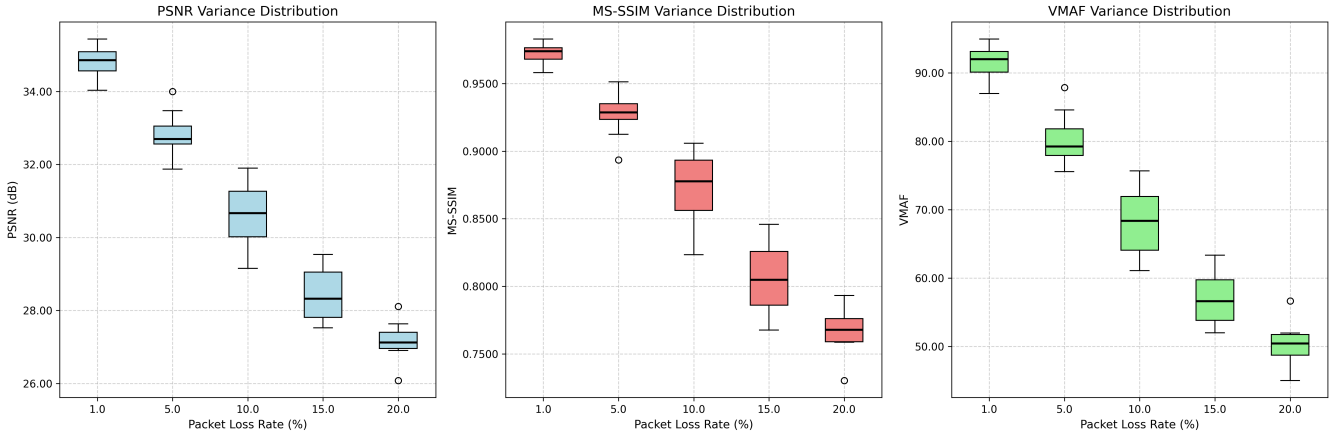


Figura 5.5: Dispersione statistica delle metriche per la sequenza RaceHorses (QP=47) per M=6.

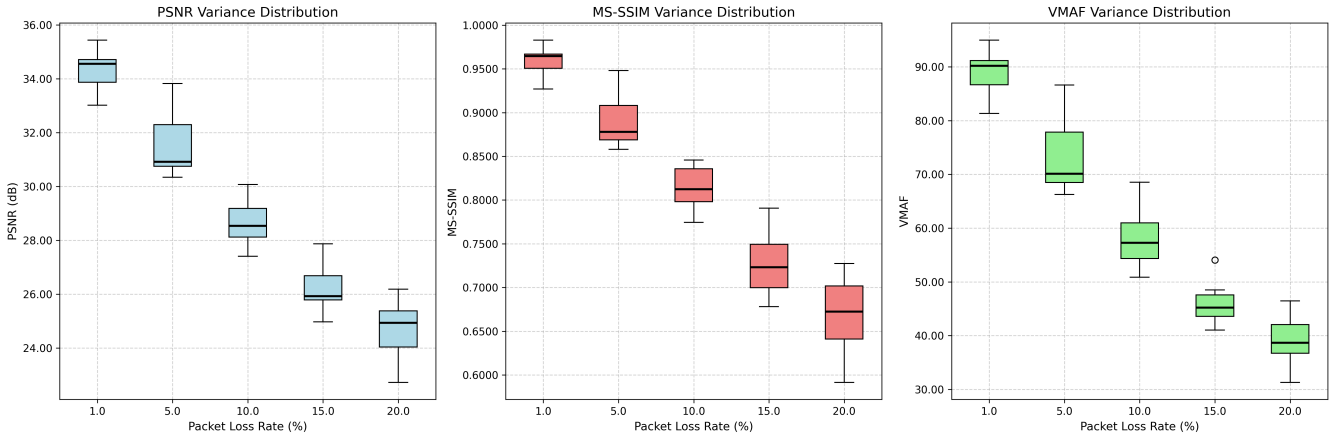


Figura 5.6: Dispersione statistica delle metriche per la sequenza RaceHorses (QP=47) per M=10.

Passando all'analisi degli scenari con latenza maggiore (Figure 5.5 e 5.6), la distribuzione statistica dei dati cambia radicalmente. All'aumentare di M, l'escursione delle metriche si allarga già a partire dai livelli di perdita intermedi (5% e 10%). Guardando, ad esempio, la metrica VMAF nello scenario ad alta latenza

($M = 10$), la differenza qualitativa tra le tracce simulate arriva a sfiorare i 20 punti. Questo accade perché subentra una forte componente di casualità, per cui, se la perdita colpisce fotogrammi caratterizzati da un'alta complessità spaziale o temporale, il degrado si accumula rapidamente e la qualità crolla; se, al contrario, l'errore incide segmenti della sequenza più statici o uniformi, il degrado risulta più contenuto.

Tuttavia, in corrispondenza delle soglie di perdita del 15% e 20% di PLR, le metriche si dividono. La varianza di PSNR e VMAF torna a ridursi, schiacciandosi verso il basso. I frame persi sono talmente numerosi, che la qualità visiva del video risulta compromessa e inaccettabile in quasi tutte le tracce simulate.

Al contrario, l'MS-SSIM mantiene una forte varianza anche al PLR 20%, con una scatola visibilmente più estesa rispetto alle altre metriche.

Per verificare quanto l'impatto della latenza dipenda dal movimento nella scena, la Figura 5.7 mostra la dispersione statistica delle metriche per BQSquare.

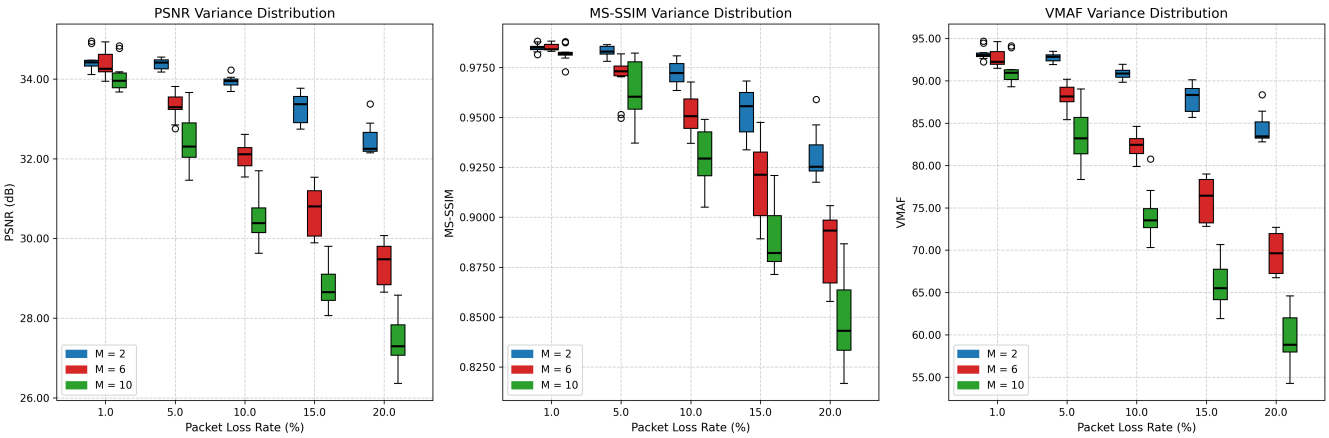


Figura 5.7: Dispersione statistica raggruppata sulle dieci tracce simulate per la sequenza BQSquare (QP=47).

Dal confronto con RaceHorses emergono subito due differenze importanti. La prima riguarda l'andamento ad alti tassi di perdita, in cui la forte varianza non sparisce, ma persiste per le configurazioni con latenza superiore a M=2. La seconda differenza riguarda l'ampiezza di questa varianza: sebbene la dispersione rimanga evidente, l'escursione totale dei dati è praticamente dimezzata; osservando il VMAF, l'intervallo massimo copre circa 10 punti, contro i quasi 20 registrati in RaceHorses.

In sintesi, un'alta latenza continua a rendere i risultati imprevedibili da una traccia all'altra, ma la staticità della sequenza mitiga i danni, impedendo un tracollo totale delle metriche.

5.2.3 Analisi visiva del degrado

Dopo aver quantificato l'impatto della latenza attraverso le metriche, è fondamentale esaminare visivamente come la perdita di dati comprometta la qualità dell'immagine. Il codec analizzato manifesta un evidente fenomeno di propagazione dell'errore: un danno iniziale non si limita a un fotogramma isolato, ma si accumula lungo i frame successivi fino all'arrivo di un nuovo I-frame, che ripristina la qualità dell'immagine.

Per illustrare questo comportamento, è stata selezionata la sequenza RaceHorses sottoposta a un tasso di perdita del 20%, isolando una specifica traccia di simulazione (seed 81300). Tra le numerose perdite distribuite lungo i 300 fotogrammi, si è scelto di focalizzare l'analisi su un particolare istante: la perdita consecutiva dei frame 50 e 51. Poiché il posizionamento degli I-frame varia in base alla latenza M, i tre scenari raggiungono il picco di degrado in istanti diversi. L'analisi visiva che segue si concentra quindi per ciascun valore di M sul momento di massimo accumulo dell'errore, ovvero l'ultimo fotogramma degradato prima dell'iniezione forzata di un I-frame.

Nelle descrizioni dei singoli scenari verrà sempre specificata la posizione dell'ultimo I-frame ricevuto prima dell'errore. Fornire questo contesto è essenziale, poiché l'impatto di una perdita, oltre a dipendere esclusivamente dal ritardo di feedback, è fortemente influenzato anche dallo stato della catena predittiva pregressa. Non tutte le perdite, infatti, colpiscono il flusso nelle medesime condizioni: maggiore è il numero di P-frame elaborati tra l'ultimo I-frame e l'istante della perdita, più la predizione dei successivi fotogrammi risulta vulnerabile.

Nelle figure presentate di seguito, a sinistra viene mostrato il fotogramma originale, mentre a destra la rispettiva versione degradata.

Analisi del fotogramma 51, $M = 2$



Figura 5.8: Confronto visivo del fotogramma 51 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=2: fotogramma originale (a sinistra) e versione degradata (a destra).

Per M=2, l'ultimo I-frame ricevuto risale al fotogramma 40. La perdita del frame 50 innesca la richiesta di un nuovo I-frame, che viene recapitato tempestivamente al frame 52. L'analisi si concentra quindi sul fotogramma 51, come mostrato in Figura 5.8. Come si può osservare, la finestra temporale molto ristretta impedisce all'errore di degenerare e propagarsi all'intera scena. Il degrado è visibile sotto forma di artefatti localizzati e perdita di nitidezza: si nota, ad esempio, una evidente distorsione sul muso del cavallo di sinistra. Tuttavia, al di fuori di questo difetto, la struttura complessiva della scena rimane solida e perfettamente riconoscibile.

Analisi del fotogramma 55, $M = 6$



Figura 5.9: Confronto visivo del fotogramma 55 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=6: fotogramma originale (a sinistra) e versione degradata (a destra).

Per M=6, l'I-frame precedente si posiziona al fotogramma 44. A causa della maggiore latenza, a seguito della perdita dei fotogrammi 50 e 51, l'I-frame di ripristino viene recapitato solo al frame 56. Osservando il fotogramma 55, mostrato in Figura 5.9, è evidente come il maggior numero di fotogrammi compromessi aggravi notevolmente la propagazione dell'errore. Si nota, infatti, una deformazione decisamente marcata dei soggetti: le zone ad alto movimento, come la testa e gli arti del cavallo o la divisa del fantino di destra, iniziano a sfaldarsi e a fondersi, perdendo i contorni originali.

Analisi del fotogramma 59, $M = 10$

Come mostrato in Figura 5.10, osservando il fotogramma 59 si assiste al collasso totale della scena. Il codec non è più in grado di compensare il movimento, generando un vistoso trascinamento dei pixel, per cui i soggetti perdono i propri bordi e si fondono tra loro e con lo sfondo, rendendo l'immagine confusa, illeggibile e qualitativamente inaccettabile. Questa devastazione visiva è il risultato diretto dell'elevato numero di fotogrammi su cui l'errore ha avuto modo di propagarsi (dal

frame 50 al 59). In questo scenario, infatti, l'I-frame precedente risale al fotogramma 48; questo dimostra chiaramente come la breve distanza tra il fotogramma chiave e la perdita non sia sufficiente a proteggere la sequenza. A causa dell'elevata latenza, la perdita consecutiva dei frame 50 e 51 innesca un errore che non viene frenato a tempo debito, poiché l'I-frame di ripristino viene recapitato solamente al frame 60.



Figura 5.10: Confronto visivo del fotogramma 59 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=10: fotogramma originale (a sinistra) e versione degradata (a destra).

L'analisi visiva fin qui condotta conferma quanto emerso dai dati statistici: all'aumentare di M, specialmente in sequenze dinamiche, la propagazione incontrollata dell'errore compromette severamente la struttura dei fotogrammi, giustificando il crollo verticale delle metriche oggettive di qualità, esposto precedentemente.

5.3 Mitigazione tramite *Frame Freeze*

Come anticipato nel Capitolo 3, l'applicazione delle tecniche di error concealment, tra cui il *Frame Freeze*, si è scontrata con un limite intrinseco ai modelli neurali. Concepire la mitigazione come un intervento "una tantum" per tentare di ripristinare un intero GOP si è rivelato inefficace.

Tuttavia, l'analisi condotta nei paragrafi precedenti ha dimostrato come l'impiego di un canale di feedback per la richiesta di I-frame confini la propagazione dell'errore all'interno di una finestra temporale ben definita. È proprio in questo scenario che il *Frame Freeze* trova la sua reale applicazione. Come descritto nel Paragrafo 4.3, questa tecnica interviene in fase di decodifica, clonando l'ultimo fotogramma valido presente nel *Decoded Picture Buffer* (DPB). Il frame duplicato, oltre a sostituire in uscita il fotogramma mancante, viene reinserito all'interno del DPB per fungere da base di riferimento per la predizione dei frame futuri. In questo modo si impedisce al codec di operare su flussi di bit azzerati, in attesa dell'imminente I-frame di ripristino.

L'analisi porterà alla luce un netto contrasto tra la valutazione algoritmica e quella umana. Da un lato, l'impiego del *Frame Freeze* verrà penalizzato dalle metriche oggettive, a causa del disallineamento temporale indotto dall'algoritmo; dall'altro, il vero successo della tecnica si misurerà nell'esperienza soggettiva, garantendo all'osservatore finale una sequenza più pulita, fluida e comprensibile.

5.3.1 Andamento delle metriche e riduzione della varianza

Si pone a confronto diretto quanto analizzato nel Paragrafo 5.2 (Figura 5.1 e 5.2) con i nuovi risultati ottenuti a seguito dell'attivazione del Frame Freeze, mostrati rispettivamente in Figura 5.11 e Figura 5.12.

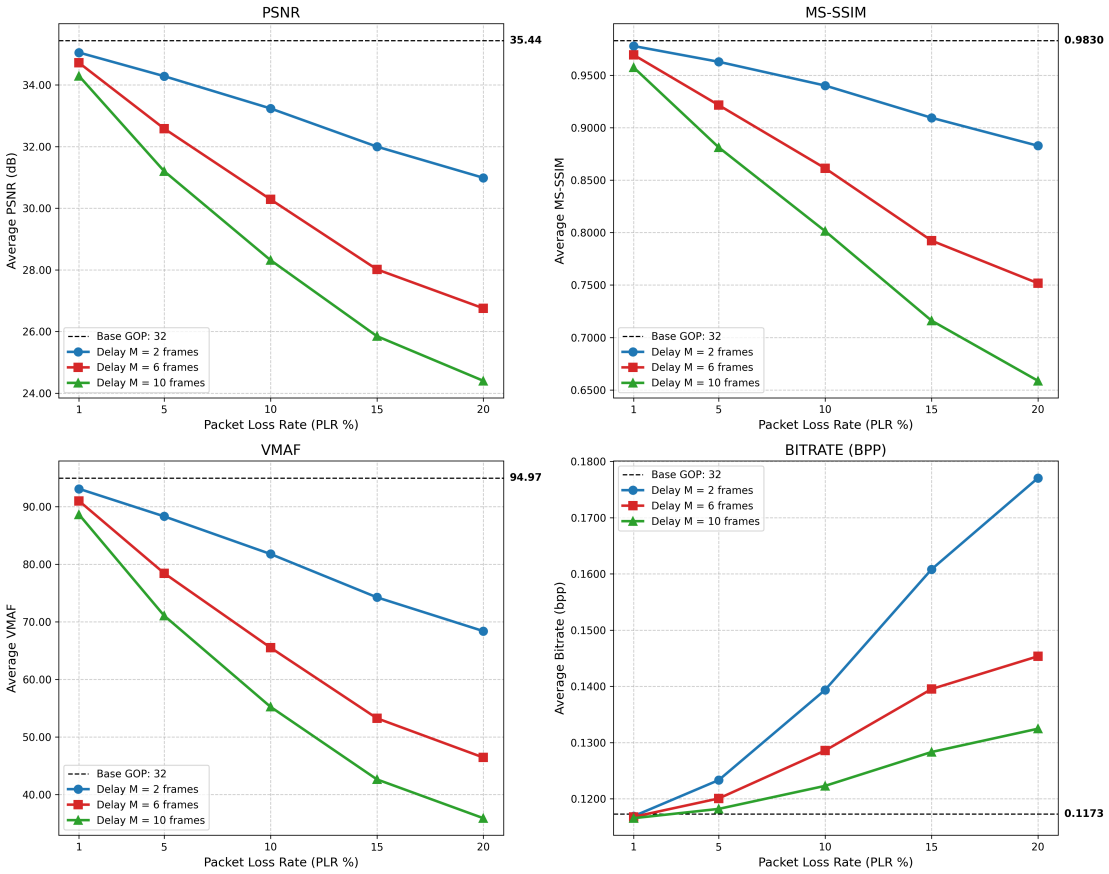


Figura 5.11: Andamento delle metriche di qualità e del bitrate al variare del PLR e del ritardo di rete M per la sequenza RaceHorses (QP=47) con *Frame Freeze* attivo.

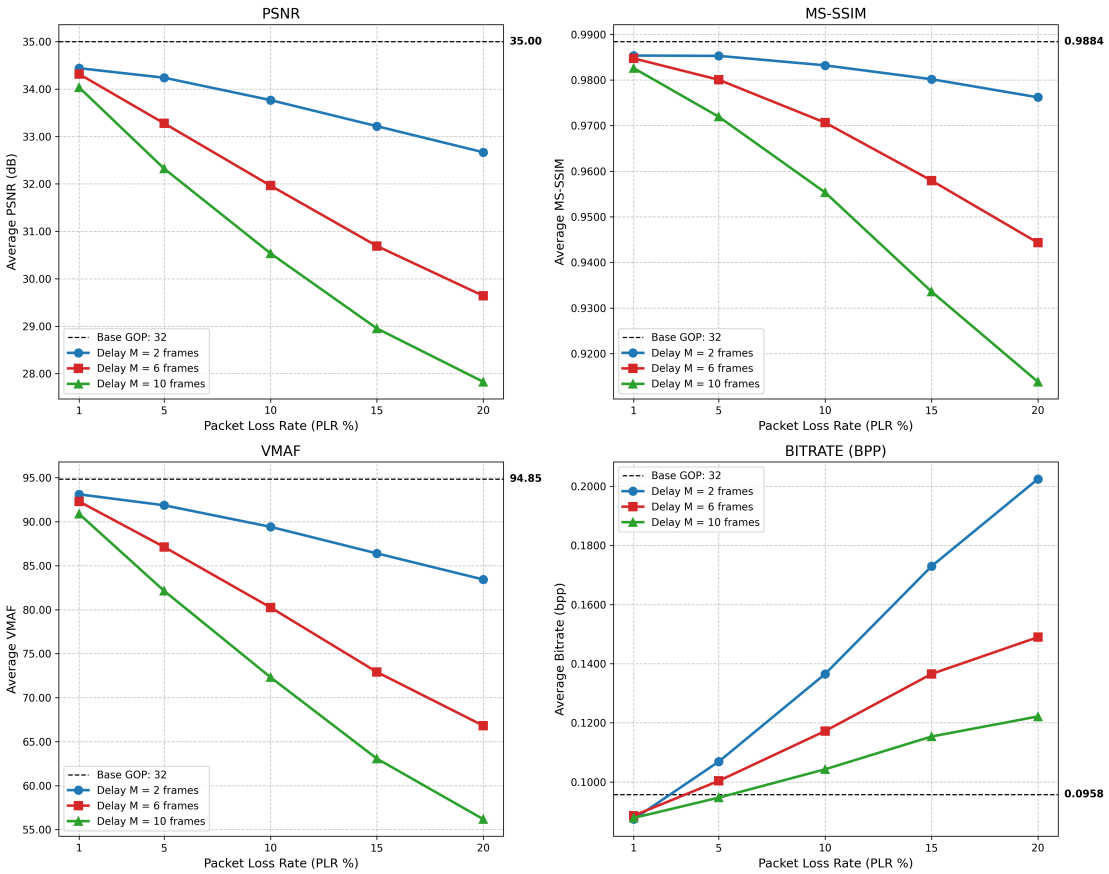


Figura 5.12: Andamento delle metriche di qualità e del bitrate al variare del PLR e del ritardo di rete M per la sequenza BQSquare (QP=47) con *Frame Freeze* attivo.

Un primo risultato riguarda il costo di rete. Le curve del bitrate restano assolutamente identiche a quelle dello scenario base. Infatti, operando esclusivamente sul DPB in fase di decodifica, il *Frame Freeze* non richiede l'invio di dati aggiuntivi.

Focalizzando l'attenzione sulle metriche di qualità, emerge in modo evidente la dinamica controintuitiva anticipata nell'introduzione: in RaceHorses il *Frame Freeze* non apporta benefici tangibili alle medie generali, inducendo anzi una sistematica penalizzazione.

Difatti, osservando la metrica VMAF, si nota un generale abbassamento delle curve. Considerando, ad esempio, un tasso di perdita del 20% e M=2, il valore medio crolla dai circa 73 punti dello scenario base a circa 69 punti con il *Frame Freeze* attivo. Un degrado analogo si registra anche per gli scenari con latenza maggiore. Anche il PSNR subisce un lieve e costante schiacciamento verso il basso su tutte le configurazioni, ma in misura più contenuta.

La causa di questa penalizzazione risiede interamente nella desincronizzazione temporale. Le metriche oggettive di qualità, per loro natura, valutano il degrado confrontando matematicamente il flusso decodificato con la sequenza originale di riferimento, operando un riscontro pixel per pixel e fotogramma per fotogramma. Congelando l'immagine, il *Frame Freeze* blocca il movimento dei soggetti. Poiché la scena originale continua il suo naturale avanzamento, il confronto diretto con il frame clonato genera un inevitabile scostamento spaziale. L'algoritmo viene, così, penalizzato nel punteggio finale, nonostante l'immagine congelata risulti effettivamente nitida e priva di artefatti visivi.

L'impatto del disallineamento temporale trova conferma esaminando i risultati della sequenza BQSquare, ma con andamenti molto più stabili rispetto a RaceHorses: per M=2 il VMAF si abbassa di circa un punto (contro il calo di 5 punti osservato in RaceHorses) e, pur registrando flessioni maggiori ad alta latenza (fino a circa 3 punti per M=10), il degrado si mantiene più contenuto. Metriche strettamente legate alla fedeltà strutturale dell'immagine, come l'MS-SSIM e il PSNR, mostrano addirittura lievi e costanti incrementi. Poiché il contenuto visivo è prettamente statico, la duplicazione del fotogramma prelevato dal DPB non genera scostamenti spaziali significativi rispetto al frame originale atteso.

Questa maggiore stabilità risulta evidente anche analizzando i boxplot relativi alla dispersione statistica di BQSquare, riportati in Figura 5.13. Coerentemente con l'andamento dei valori medi, MS-SSIM e PSNR confermano il lieve incremento complessivo e, in aggiunta, mostrano una varianza ridotta. La migliore stabilità si riflette in intervalli interquartili e range dei baffi molto più schiacciati rispetto allo scenario base. Fa eccezione il VMAF, in cui vi è una semplice traslazione verso il basso dei valori, mantenendo l'ampiezza della dispersione pressoché invariata.

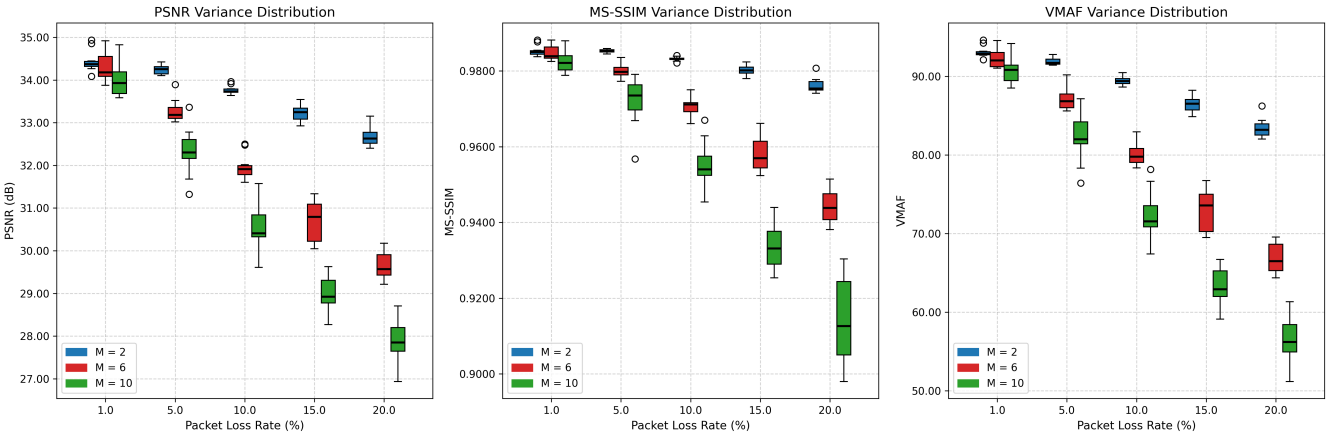


Figura 5.13: Dispersione statistica raggruppata sulle dieci tracce simulate per la sequenza BQSquare (QP=47) con *Frame Freeze* attivo.

Ritornando sulla sequenza RaceHorses, i grafici di Figura 5.14 mostrano come il *Frame Freeze* non apporti alcun beneficio in termini di varianza. Poiché la scala dei grafici rimane esattamente identica a quella dello scenario base, il confronto visivo risulta immediato: in tutte le metriche analizzate, l'ampiezza della varianza si mantiene pressoché invariata. L'intera distribuzione statistica subisce esclusivamente una netta traslazione verso il basso, confermando come la penalizzazione dovuta al disallineamento temporale si applichi su tutte le tracce simulate.

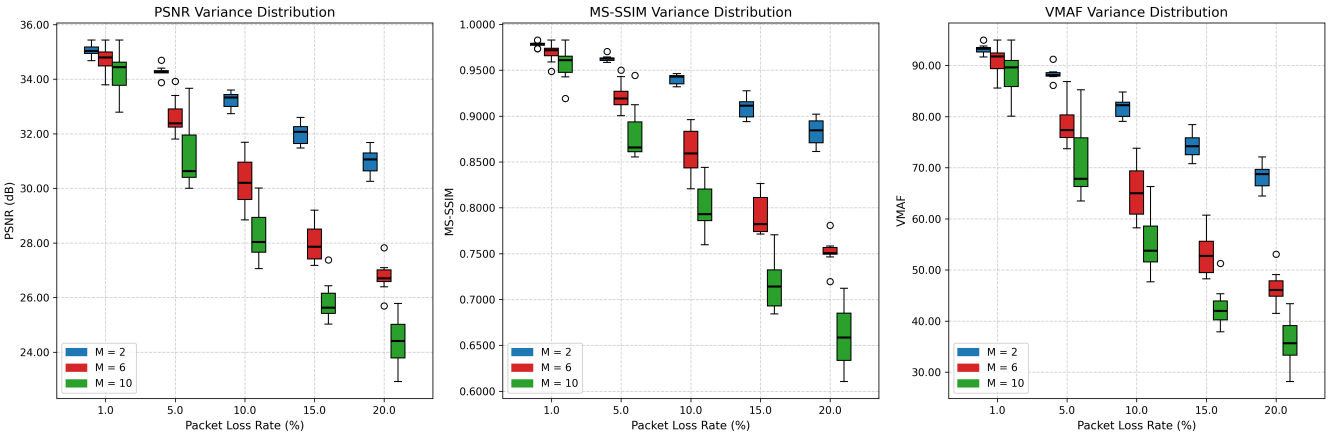


Figura 5.14: Dispersione statistica raggruppata sulle dieci tracce simulate per la sequenza RaceHorses (QP=47) con *Frame Freeze* attivo.

In sintesi, l'analisi puramente numerica delle metriche suggerirebbe che il *Frame Freeze* sia una tecnica inefficace, se non addirittura dannosa per i contenuti dinamici. Ciò rende necessaria la successiva analisi visiva, l'unica in grado di dimostrare come questa tecnica rappresenti, in realtà, un netto miglioramento percettivo.

5.3.2 Analisi visiva del degrado: efficacia a basse latenze

L'analisi si concentra sulla medesima traccia di simulazione esaminata nel Paragrafo 5.2.3, ovvero la sequenza RaceHorses sottoposta a un PLR del 20% con seed 81300. L'obiettivo è osservare come la tecnica riesca a mascherare i danni causati dalla perdita consecutiva dei fotogrammi 50 e 51, prima dell'arrivo dell'I-frame di ripristino.

Nelle figure seguenti, al fine di garantire una maggiore leggibilità dei dettagli, si omette il fotogramma originale, per proporre un confronto diretto tra la decodifica standard con frame perso (a sinistra) e la decodifica con *Frame Freeze* attivato (a destra).

Analisi del fotogramma 51, $M = 2$



Figura 5.15: Confronto visivo del fotogramma 51 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=2: decodifica standard (a sinistra) e con *Frame Freeze* attivo (a destra).

Nel caso di latenza minima, l'osservazione del fotogramma 51 dimostra l'eccellente resa visiva del *Frame Freeze*. Come si evince dal confronto in Figura 5.15, in assenza di mitigazione la scena presenta una perdita di nitidezza e la comparsa di artefatti localizzati sui soggetti in movimento, specialmente sul muso del cavallo di sinistra. Attivando il *Frame Freeze*, il difetto viene annullato completamente.

In questo scenario di doppia perdita consecutiva, il fotogramma 51 elaborato dal sistema è di fatto una copia del fotogramma 50, il quale a sua volta è una copia clonata del fotogramma 49, ovvero l'ultimo frame ricevuto regolarmente. Grazie a questa operazione, i difetti visivi scompaiono e i contorni rimangono ben delineati.

Sebbene questa strategia introduca, come precedentemente discusso, un inevitabile disallineamento temporale rispetto al reale movimento della scena, la ridotta finestra (due frame) fa sì che tale compromesso sia ampiamente giustificato. Durante la riproduzione della sequenza al suo framerate nominale, questo lieve scostamento temporale risulta pressoché impercettibile all'occhio umano.

Analisi del fotogramma 55, $M = 6$

Analizzando il caso M=6, mostrato in Figura 5.16, l'impatto positivo del *Frame Freeze* risulta evidente, pur presentando alcune limitazioni intrinseche. Rispetto alla grave deformazione e allo sfaldamento dei soggetti osservati nella decodifica standard, l'attivazione dell'algoritmo argina con successo questa dinamica: i contorni dei cavalli e dei fantini risultano maggiormente definiti e la separazione tra i diversi elementi della scena viene preservata.



Figura 5.16: Confronto visivo del fotogramma 55 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=6: decodifica standard (a sinistra) e con *Frame Freeze* attivo (a destra).

In sostanza, in entrambi gli scenari analizzati, pur con l'emergere di corruzioni localizzate all'aumentare della latenza, il *Frame Freeze* previene il collasso qualitativo dell'immagine. La tecnica si dimostra quindi essenziale per garantire un'integrità e una leggibilità della scena ampiamente superiori rispetto allo scenario base.

5.3.3 Analisi visiva del degrado: limiti ad alta latenza

Mantenendo la medesima traccia di simulazione, la prossima analisi evidenzia i limiti del *Frame Freeze* ad alta latenza (M=10). Per illustrare la progressione del degrado, vengono esaminati il fotogramma intermedio 56 e il fotogramma 59; quest'ultimo rappresenta il momento di massimo accumulo dell'errore prima dell'arrivo dell'I-frame di ripristino. Nelle Figure 5.17 e 5.18 viene proposto il confronto diretto tra lo scenario base e quello con *Frame Freeze* attivato, omettendo anche in questo caso i fotogrammi originali per garantire la massima leggibilità dei dettagli.

Analisi del fotogramma 56, $M = 10$

Nello scenario base, la prolungata assenza di I-frame di ripristino innesca un degrado incontrollabile. Già osservando il fotogramma intermedio 56, mostrato in Figura 5.17, la propagazione dell'errore inizia a compromettere severamente la scena. Attivando il *Frame Freeze* si osserva invece un lieve miglioramento: il sistema previene la fusione dei soggetti, restituendo contorni più nitidi. In particolare, i dettagli anatomici, come la testa del cavallo centrale, si mantengono più definiti e riconoscibili rispetto alla decodifica standard.

Tuttavia, il *Frame Freeze* non garantisce un ripristino perfetto: mentre la porzione destra risulta molto meno distorta rispetto allo scenario base, l'area sinistra rimane ugualmente compromessa. Ciononostante, riuscire a contenere il

collasso visivo, a ben sette frame dall'ultimo riferimento valido, dimostra la notevole solidità dell'approccio.



Figura 5.17: Confronto visivo del fotogramma 56 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=10: decodifica standard (a sinistra) e con *Frame Freeze* attivo (a destra).

Analisi del fotogramma 59, $M = 10$



Figura 5.18: Confronto visivo del fotogramma 59 per la sequenza RaceHorses (PLR = 20%, seed 81300) con latenza M=10: decodifica standard (a sinistra) e con *Frame Freeze* attivo (a destra).

Come documentato nel Paragrafo 5.2.3, in questo specifico istante lo scenario base è caratterizzato da un collasso visivo totale.

Attivando il *Frame Freeze*, tuttavia, si nota un leggero miglioramento. La tecnica riesce infatti a preservare la struttura di diversi elementi, ad esempio mantenendo una parziale leggibilità della testa del cavallo di destra, della macchia sulla destra e di altre porzioni localizzate. Tuttavia, nonostante questi salvataggi, la situazione visiva complessiva rimane gravemente compromessa. Il *Frame Freeze* ha di fatto esaurito la sua reale capacità di mascheramento.

In sintesi, a latenze elevate l'ispezione visiva e le misurazioni oggettive tornano ad allinearsi. Il progressivo aumento della latenza satura l'efficacia del *Frame Freeze*, rendendolo incapace di contenere le distorsioni della scena. La perdita di qualità effettiva che ne consegue risulta quindi del tutto coerente con il calo delle prestazioni registrato dai dati statistici.

5.4 Valutazione complessiva: curve RD

Al fine di validare i risultati su tutti i livelli di compressione testati, si è condotta un'analisi aggregata tramite le curve Rate-Distortion. Per evidenziare il comportamento del sistema nelle condizioni di massimo stress, l'analisi si concentra sullo scenario ad alta latenza. In corrispondenza del valore massimo di M testato, le Figure 5.19 e 5.20 pongono a confronto, rispettivamente per RaceHorses e BQSquare, lo scenario base con quello in cui è attivo il *Frame Freeze*.

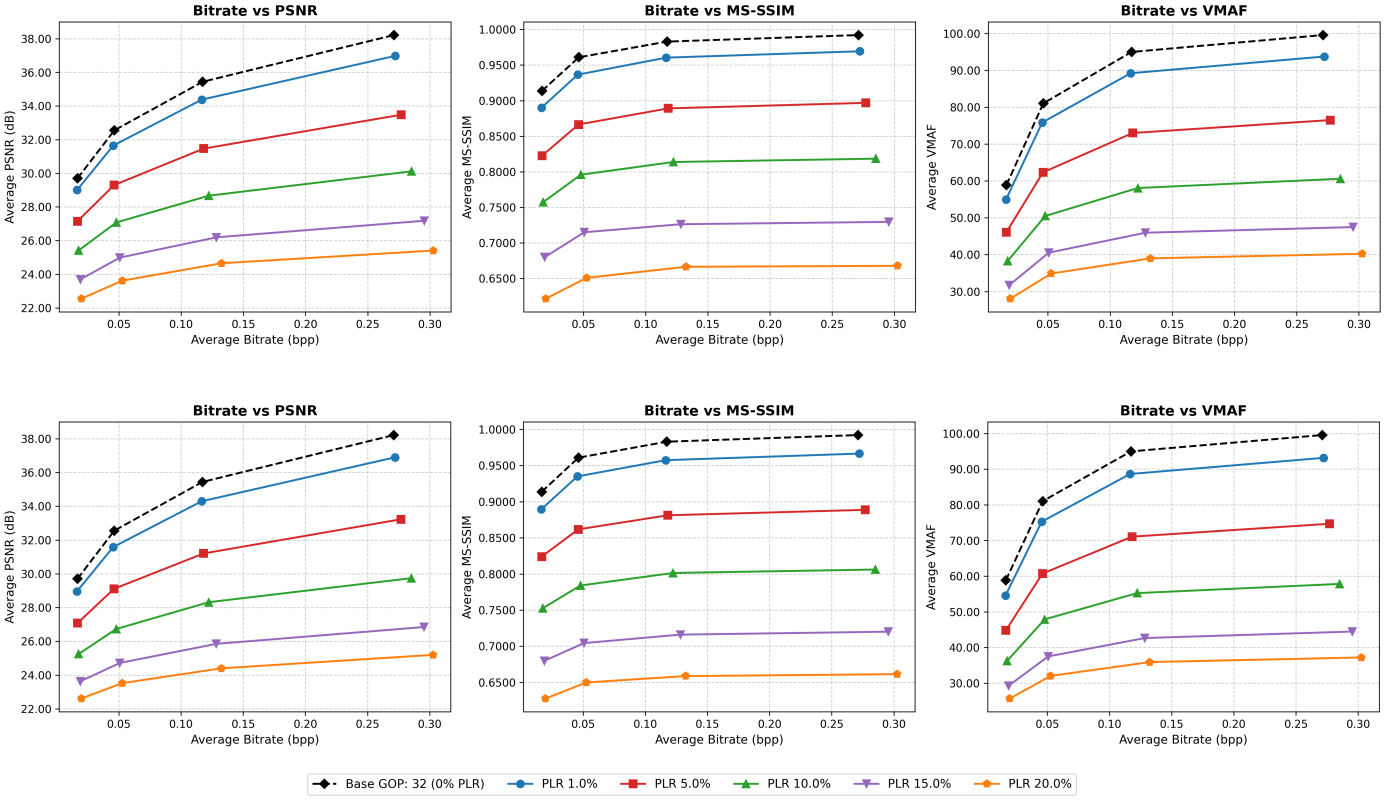


Figura 5.19: Curve Rate-Distortion per la sequenza RaceHorses con M=10: confronto tra lo scenario base (in alto) e la configurazione con *Frame Freeze* attivo (in basso).

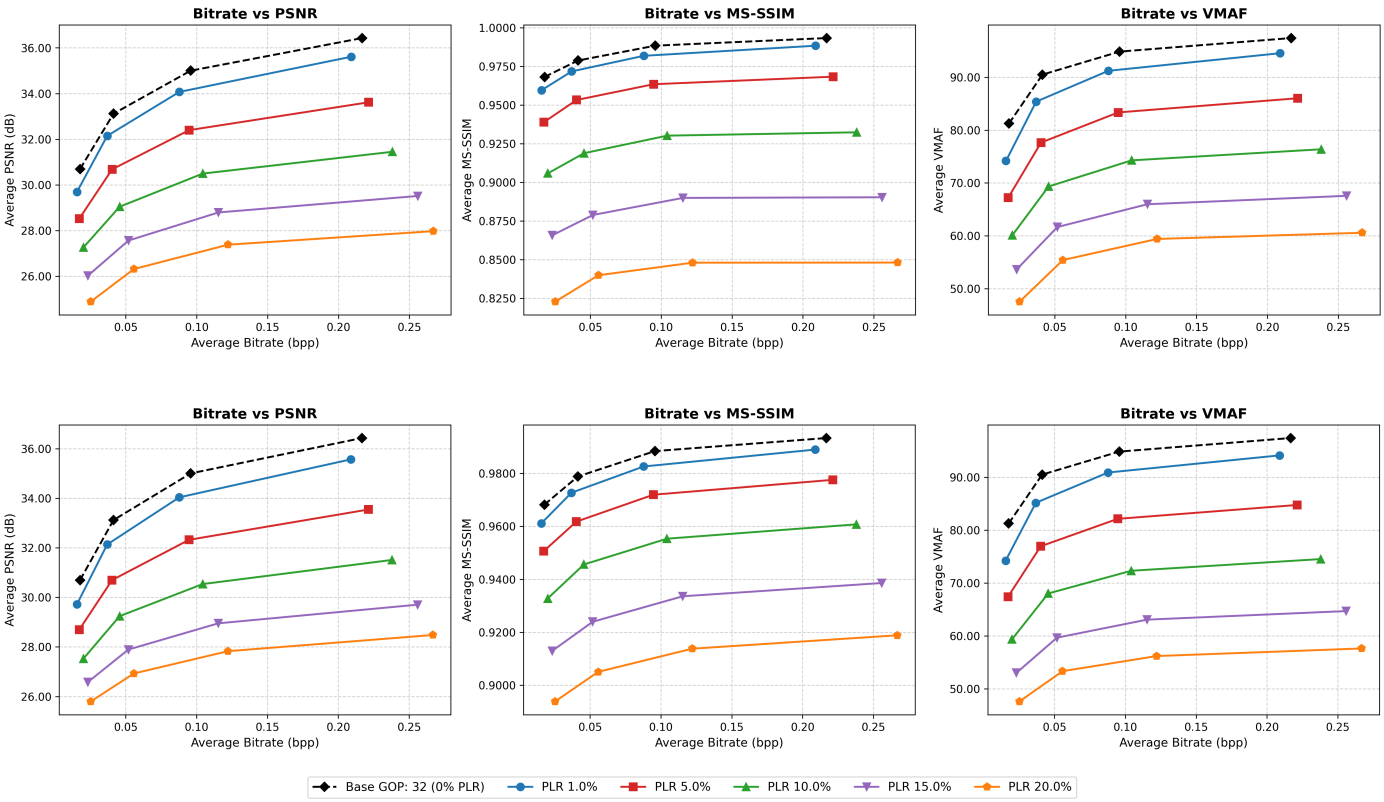


Figura 5.20: Curve Rate-Distortion per la sequenza BQSquare con M=10: confronto tra lo scenario base (in alto) e la configurazione con *Frame Freeze* attivo (in basso).

Come già evidenziato nell'analisi precedente, l'attivazione del *Frame Freeze* non introduce alcun overhead di trasmissione, condizione che nei grafici si traduce nella condivisione della medesima ascissa per i punti corrispondenti allo stesso livello di compressione.

Spostando invece l'attenzione sull'asse delle ordinate, l'analisi delle curve di qualità sull'intera classe di test mostra come esista una dualità legata alla natura del contenuto visivo:

- **Sequenze dinamiche (RaceHorses e BasketballPass)**: per qualsiasi valore di latenza, l'attivazione del *Frame Freeze* si traduce in una traslazione delle curve verso il basso per tutte le metriche valutate: ciò conferma che il disallineamento temporale penalizza matematicamente il sistema a prescindere dal livello di compressione (QP) applicato.
- **Sequenze statiche (BQSquare e BlowingBubbles)**: lo scenario cambia. Mentre le curve di PSNR e VMAF subiscono variazioni minori e bilanciate,

la metrica MS-SSIM mostra un lieve miglioramento, come confermato in precedenza. Questo dimostra come il *Frame Freeze* sia efficace in assenza di movimenti accentuati.

In conclusione, i risultati sperimentali definiscono con precisione il perimetro di efficacia del *Frame Freeze*. Sebbene la tecnica offra il vantaggio di operare a costo di rete nullo, i dati dimostrano che non può essere considerata una cura universale, bensì un "cerotto". Mentre sui contenuti statici offre una mitigazione solida, negli scenari dinamici si riduce a un espediente puramente temporaneo, i cui benefici visivi iniziali vengono rapidamente annullati dal crescere della latenza. L'analisi conferma, in definitiva, che una simile tecnica di error concealment, operando in sola decodifica, non ha la forza per sopperire a un sistema di feedback di rete non sufficientemente tempestivo.

Conclusioni e sviluppi futuri

L'obiettivo di questo lavoro è stato analizzare il comportamento della famiglia di codec video neurali DCVC in condizioni operative diverse da quelle ideali, normalmente adottate per la loro valutazione. L'analisi si è concentrata in particolare sull'architettura DCVC-RT, affiancata dalla valutazione comparativa della precedente versione DCVC-FM, considerando scenari di rete caratterizzati da perdita e corruzione dei dati. In particolare, sono state studiate la propagazione temporale degli errori e la possibilità di limitarne gli effetti attraverso meccanismi attivi di recupero e strategie passive di error concealment.

Per rendere possibile l'analisi, è stato necessario superare i limiti dell'implementazione originale del codec, concepita per esecuzioni monolitiche. È stato quindi sviluppato un framework di simulazione automatizzato, in grado di orchestrare l'intera catena di test. Il framework simula le anomalie di rete attraverso diverse tracce di errore pseudo-casuali, che determinano a monte gli istanti esatti in cui si verifica la perdita. In corrispondenza di tali eventi, il sistema interviene in fase di decodifica, gestendo l'assenza del fotogramma mediante l'ingresso forzato di flussi di bit azzerati, oppure attivando il congelamento della scena. Questo approccio ha permesso di ripetere gli esperimenti in modo sistematico su molteplici tracce indipendenti, valutando le prestazioni del codec al variare del tasso di perdita e del ritardo del meccanismo di feedback.

I risultati hanno mostrato che la perdita o l'alterazione dei dati di un singolo fotogramma non produce effetti limitati all'istante dell'errore, ma viene ereditata dai fotogrammi predittivi successivi, determinando un progressivo accumulo di artefatti spaziali. In assenza di un intervento di ripristino, tale propagazione può compromettere significativamente la qualità della ricostruzione.

L'analisi ha inoltre evidenziato il ruolo del meccanismo di recupero. Il ripristino della decodifica viene ottenuto forzando il codificatore a trasmettere un nuovo fotogramma intra-codificato (I-frame), che interrompe le dipendenze temporali compromesse e consente di ristabilire la coerenza della ricostruzione. L'efficacia di questa soluzione è tuttavia fortemente condizionata dalla latenza di rete: all'aumentare del ritardo con cui la perdita viene rilevata e il nuovo I-frame raggiunge il decodificatore, aumenta anche l'intervallo durante il quale il degrado può propagarsi.

Per mitigare gli effetti di tale intervallo, sono state valutate strategie di recupero dell'errore in fase di decodifica, tra cui il *Frame Freeze*. Il mantenimento dell'ultimo fotogramma valido presente nel DPB si è dimostrato una strategia efficace nel contenere la propagazione degli artefatti senza introdurre overhead di rete. L'efficacia della tecnica dipende tuttavia dalla durata dell'intervallo di recupero: a bassa latenza il congelamento consente di preservare una ricostruzione priva delle principali distorsioni, mentre all'aumentare del ritardo la sua capacità di mascheramento si riduce progressivamente. Inoltre, il congelamento dell'immagine introduce un disallineamento temporale rispetto all'evoluzione reale della scena, penalizzando le metriche di qualità oggettive, soprattutto nei contenuti ad alta dinamicità, pur garantendo in determinate condizioni una maggiore stabilità percettiva rispetto a un flusso affetto da artefatti.

Nel complesso, i risultati mostrano che la valutazione di un codec neurale non può limitarsi alle sole prestazioni di compressione misurate in condizioni ideali. Nelle applicazioni multimediali real-time, la capacità di gestire la perdita dei dati e di ripristinare rapidamente la coerenza della ricostruzione costituisce un requisito fondamentale per determinarne la reale applicabilità.

Il framework sviluppato può essere utilizzato come base per ulteriori attività sperimentali. Come anticipato nel Capitolo 1, nelle fasi finali della stesura del presente elaborato è stata pubblicata una nuova iterazione della famiglia di codec DCVC. Una prima naturale direzione di ricerca consiste, pertanto, nell'estendere la valutazione a questa nuova architettura, mantenendo il processo di generazione e analisi degli errori già adottato.

Il nuovo modello, denominato DCVC-UF [8], rappresenta un'evoluzione diretta del DCVC-RT e introduce un innovativo paradigma di *chunk coding*: a differenza dell'elaborazione frame per frame, il codec processa gruppi di fotogrammi in parallelo, comprimendoli in una rappresentazione latente comune e propagando il contesto temporale tra chunk successivi.

Il DCVC-UF è progettato per ridurre notevolmente la complessità computazionale e aumentare la velocità di codifica e decodifica, ma non affronta direttamente il problema della perdita o della corruzione dei dati. Risulterebbe, pertanto, particolarmente interessante applicare il framework sviluppato al DCVC-UF, così da valutarne la resilienza in condizioni di rete non ideali. In particolare, sarebbe possibile analizzare gli effetti della corruzione del bitstream sui fotogrammi appartenenti allo stesso chunk e verificare l'eventuale propagazione dell'errore ai chunk successivi attraverso il contesto temporale condiviso. Inoltre, questo consentirebbe di confrontare il comportamento del nuovo paradigma di codifica con quello delle architetture RT e FM analizzate nel presente lavoro.

Un secondo sviluppo riguarda l'introduzione di tecniche di error concealment

più evolute. La tecnica del *Frame Freeze* si è dimostrata una soluzione semplice ed efficace, senza introdurre overhead di rete, ma non è in grado di ricostruire l'evoluzione della scena durante la latenza necessaria al ripristino. Una possibile estensione consisterebbe nell'impiego di reti neurali leggere, in grado di stimare i fotogrammi mancanti a partire dalle informazioni temporali disponibili nel DPB. In una prima implementazione, i fotogrammi generati potrebbero essere utilizzati esclusivamente come sostituzione temporanea dell'output durante l'attesa del recupero, senza intervenire sullo stato interno del codec. Un'estensione più avanzata potrebbe invece mirare alla ricostruzione delle rappresentazioni latenti associate ai fotogrammi persi, permettendo al decodificatore di utilizzare l'informazione ricostruita nel successivo processo di decodifica e mantenere così la coerenza del contesto temporale.

Infine, un'ulteriore direzione di ricerca riguarda l'architettura stessa del codec. Gli esperimenti condotti in questo lavoro hanno affrontato la perdita e la corruzione dei dati intervenendo principalmente in fase di decodifica. Un approccio alternativo consiste nell'integrare le condizioni di un canale inaffidabile direttamente nella fase di addestramento, introducendo nella pipeline di training simulazioni di perdita e corruzione dei dati. Un approccio di questo tipo è stato recentemente esplorato anche da architetture sperimentali come il framework *Grace* [9] e permetterebbe al codec di apprendere rappresentazioni latenti più robuste alle perturbazioni del canale. In questo modo, la resilienza agli errori non verrebbe affidata esclusivamente a meccanismi di recupero applicati in fase di decodifica, ma diventerebbe parte integrante del processo di apprendimento del codec.

Bibliografia

[1] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai e Zhiyong Gao. «DVC: An End-To-End Deep Video Compression Framework». In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10998–11007. DOI: 10.1109/CVPR.2019.01126 (cit. a p. 4).

[2] Jiahao Li, Bin Li e Yan Lu. «Deep Contextual Video Compression». In: *Advances in Neural Information Processing Systems*. A cura di M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang e J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 18114–18125. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/96b250a90d3cf0868c83f8c965142d2a-Paper.pdf (cit. a p. 5).

[3] Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li e Yan Lu. «Towards Practical Real-Time Neural Video Compression». In: *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 12543–12552. DOI: 10.1109/CVPR52734.2025.01170 (cit. a p. 5).

[4] Z. Wang, E.P. Simoncelli e A.C. Bovik. «Multiscale structural similarity for image quality assessment». In: *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216 (cit. a p. 6).

[5] Christos G. Bampis, Zhi Li e Alan C. Bovik. «Spatiotemporal Feature Integration and Model Fusion for Full Reference Video Quality Assessment». In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.8 (2019), pp. 2256–2270. DOI: 10.1109/TCSVT.2018.2868262 (cit. a p. 7).

[6] David Flynn, Karl Sharman e Christopher Rosewarne. *Common Test Conditions and Software Reference Configurations for HEVC Range Extensions*. Rapp. tecn. JCTVC-N1006. Joint Collaborative Team on Video Coding (JCT-VC), 2013 (cit. a p. 11).

[7] Jiahao Li, Bin Li e Yan Lu. «Neural Video Compression with Feature Modulation». In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 26099–26108. DOI: 10.1109/CVPR52733.2024.02466 (cit. a p. 18).

[8] Jiahao Li, Wenxuan Xie, Zhaoyang Jia, Bin Li, Zongyu Guo, Xiaoyi Zhang e Yan Lu. *Ultra-Fast Neural Video Compression*. 2026. arXiv: 2606.04410 [cs.CV]. URL: https://arxiv.org/abs/2606.04410 (cit. a p. 48).

[9] Yihua Cheng et al. «GRACE: loss-resilient real-time video through neural codecs». In: *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation*. NSDI'24. Santa Clara, CA, USA: USENIX Association, 2024. ISBN: 978-1-939133-39-7 (cit. a p. 49).