



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali  
Corso di Laurea Magistrale in Informatica  
Curriculum: Data Science

Tesi di Laurea

RESTAURO DI VIDEO E MIGLIORAMENTO  
DELLA QUALITÀ VISUALE USANDO RETI  
NEURALI BASATE SU U-NET

VIDEO RESTORATION AND VISUAL  
QUALITY ENHANCEMENT USING U-NET  
BASED NEURAL NETWORKS

MATTEO MARULLI

Relatore: *Professor Marco Bertini*

Anno Accademico 2020-2021

Matteo Marulli: *Restauro di video e miglioramento della qualità visuale usando reti neurali basate su U-Net*, Corso di Laurea Magistrale in Informatica  
Curriculum: Data Science, © Anno Accademico 2020-2021

---

## INDICE

---

1	Introduzione	9
2	Lavoro correlato	11
2.1	Introduzione alla letteratura	11
2.1.1	Rimozione degli artefatti e super risoluzione video	11
2.1.2	Rimozione degli artefatti e super risoluzione per immagini	12
2.1.3	Tecniche generative di sintesi video a video e di immagini	13
2.1.4	Self attention per i modelli GANs	13
2.2	Algoritmi per il video inpainting	14
3	Video inpainting e super risoluzione	17
3.1	Che cos'è il video inpainting	17
3.1.1	Motion field e optical flow	17
3.1.2	L'effetto della banda orizzontale nei video	18
3.1.3	Eliminazione della banda orizzontale con Flow-edge Guided Video Completion	20
3.1.4	Edge-guided flow completion	22
3.1.5	Temporal neighbor generation	23
3.1.6	Fusion e hallucination	25
3.2	Super risoluzione delle immagini con sr-unet	27
3.2.1	Generative Adversarial Network	27
3.2.2	Architettura U-Net	29
3.2.3	Metriche di percezione	30
3.2.4	Dataset: BVI-DVC	34
3.2.5	SR-unet	36
3.2.6	Modifiche a sr-unet	41
4	Esperimenti	45
4.0.1	Hardware e strumenti usati.	45
4.0.2	Detttagli di addestramento e divisione dati tra training set e test set	45
4.0.3	Risultati	46
4.0.4	Studio di ablazione	49
4.0.5	Confronto con sr-gan	49
4.0.6	Rimozione della banda orizzontale	51

**2 Indice**

<b>5 Conclusioni</b>	<b>55</b>
5.0.1 Sviluppi futuri	55
<b>5.1 Ringraziamenti</b>	<b>56</b>

---

## ELENCO DELLE FIGURE

---

- Figura 1      Effetto della distorsione delle barre orizzontali parte 1      18
- Figura 2      Effetto della distorsione delle barre orizzontali parte 2      18
- Figura 3      Rappresentazione del motion field      19
- Figura 4      Proiezione di un punto del mondo reale nel piano immagine      19
- Figura 5      Alcune maschere e relativi istogrammi      20
- Figura 6      Pipeline di lavoro di FGVC      21
- Figura 7      La figura di destra mostra una visualizzazione spazio-temporale per le linee di scansione evidenziate nelle immagini di sinistra. Le regioni verdi sono mancanti. La linea gialla, arancione e marrone nella figura di destra rappresenta la linea di scansione al primo frame non locale, il frame corrente e il terzo frame non locale, rispettivamente. La figura illustra i candidati al completamento per i pixel rossi e blu (i dischi grandi sulla linea arancione). Seguendo le traiettorie di flusso (le linee nere tratteggiate) fino al bordo della regione mancante, otteniamo i candidati locali per il pixel blu (i dischi piccoli), ma non per il pixel rosso, perché le gambe della persona formano barriere di flusso invalicabili. Con l'aiuto del flusso non locale che si collega ai fotogrammi temporalmente distanti, otteniamo dei vicini extra non locali per il pixel rosso (i dischi rossi sulla linea gialla e marrone). Come risultato, possiamo vedere il vero sfondo che è coperto dalle gambe che si muovono.      24

4 Elenco delle figure

Figura 8	Vicini temporali: i vicini temporali non locali (il secondo vicino non locale in questo caso) sono utili quando i pixel conosciuti corretti non possono essere raggiunti con catene di flusso locali a causa di barriere di flusso (il rosso indica vicini non validi). 25
Figura 9	I metodi precedenti operano direttamente nel dominio del colore, producendo delle cuciture (a). FGVC propaga nel dominio del gradiente (b), e poi ricostruisce i risultati con la ricostruzione di Poisson (c). 26
Figura 10	Schema di architettura GAN 28
Figura 11	Schema orinale della architettura U-Net 30
Figura 12	Valori di SSIM a confronto 32
Figura 13	Valori di SSIM a confronto. 33
Figura 14	Differenze percettive tra metodi tradizionali, reti neurali e umani. 34
Figura 15	Valori di LPIPS e SSIM a confronto 35
Figura 16	Estrazione di alcuni frame proveniente da diversi video che compongono il dataset BVI-DVC 36
Figura 17	Frame dal set 1 37
Figura 18	Frame dal set 2 37
Figura 19	Frame dal set 3 37
Figura 20	Frame dal set 4 37
Figura 21	Frame dal set 5 37
Figura 22	Frame originale 37
Figura 23	Schema di sr-unet. 39
Figura 24	Applicazione dell'operazione di pixel-suffle su un tensore 3D. 40
Figura 25	Esempio di confronto tra upsampling a interpolazione bicubica e pixel-shuffle. 40
Figura 26	Il receptive field di una convoluzione. 42
Figura 27	Il receptive-field di una convoluzione atrous 42
Figura 28	Lo schema del Atrous Spatial Pyramidal Pooling 43
Figura 29	Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato 49
Figura 30	Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato 49

- Figura 31 Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato 51
- Figura 32 Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con sr-gan 52
- Figura 33 Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con sr-gan 52
- Figura 34 Frame che presenta il problema della banda orizzontale. 53
- Figura 35 Maschera del frame a sinistra. 53



*"Take me down six underground  
the ground beneath your feet"*  
— *Sneaker Pimps*



# 1

---

## INTRODUZIONE

---

Molte aziende del settore televisivo sentono il bisogno di restaurare i loro contenuti audio-video, salvati su supporti analogici come le vecchie cassette. Con l'usura del tempo, le videocassette si rovinano, e viene introdotto rumore e/o difetti che abbassano la qualità visiva del video. In questo lavoro di tesi vengono mostrate delle tecniche di restauro per aumentare la qualità video, cercando di eliminare o riducendo tutti questi inestetismi con diverse tecniche di deep learning e di computer vision. La struttura della tesi parte da questa breve introduzione, e a seguire con il capitolo 2 che mostra una rassegna stampa della letteratura sul problema del video-inpainting e sul problema del miglioramento della qualità dell'immagine. Nel capitolo 3 affrontiamo la soluzione del problema delle barre orizzontali con il video-inpainting, mentre il problema della qualità dell'immagine verrà affrontato con una rete u-net basata sulla rete sr-unet discussa sempre nello stesso capitolo. Nel capitolo 4 verranno discussi i risultati ottenuti con la rete proposta e confrontati con la rete sr-unet precedente e con la rete sr-gan. Infine nel capitolo 5 faremo un resoconto finale per considerare sviluppi futuri. Il codice python e le istruzioni per far partire il progetto saranno disponibili sul mio github al seguente link: <https://github.com/MattBlue92>



# 2

---

## LAVORO CORRELATO

---

### 2.1 INTRODUZIONE ALLA LETTERATURA

In questo capitolo faremo una panoramica delle diverse soluzioni proposte dai ricercatori per quanto riguarda il problema del miglioramento della qualità delle immagini e dei video, attraverso l'uso di tecniche di super risoluzione e di aumento della qualità visiva. Molte di queste tecniche sono state ideate per restaurare contenuti multimediali che hanno subito l'applicazione di algoritmi di compressione lossy come jpeg, in quanto l'uso di questi algoritmi di compressione genera spesso l'introduzione di rumore e artefatti che abbassano la qualità visiva di una immagine. Vedremo anche alcune soluzioni per il video inpainting applicato alla rimozione di oggetti indesiderati su video e immagini.

#### 2.1.1 *Rimozione degli artefatti e super risoluzione video*

In molte applicazioni video come lo streaming come Netflix o Skype, l'applicazione di un algoritmo di compressione è fondamentale per permettere agli utenti, che hanno una connessione meno stabile di usare il servizio riducendo la qualità visiva. In [6] Galteri et al. propongono un metodo basato su architettura GAN<sup>1</sup> (*Generative Adversarial Network*) per la rimozione di artefatti generati dagli algoritmi compressione lossy come jpeg con codec H.264 o H.265/AVC, con l'obiettivo di ottimizzare i tempi e le risorse di addestramento della rete delegata alla generazione delle immagini, in quanto tale rete è molto profonda e anche per rispettare i vincoli temporali richiesti per l'elaborazione e la fruizione del materiale visivo.

In [5] Galteri et al. dimostrano che le reti GAN possono essere usate per restaurare con successo immagini e video di qualsiasi qualità, persino

---

<sup>1</sup> L'Architettura GAN (*Generative adversarial network*) verrà spiegata nel capitolo 3.

in presenza di rumore mosquito o ad alta frequenza. In [19] Mameli et al. presentano un' altro approccio basato su GAN per la rimozione di artefatti visivi per immagini e video causati dall'applicazione dell'algoritmo di compressione jpeg. La novità di questo lavoro è l'introduzione di una nuova strategia di allenamento chiamata *NoGAN* che prevede una fase di pre-training sia per il generatore e sia per il discriminatore, e poi una fase di fine-tuning per ricalibrare i pesi del generatore e del discriminatore usando l'addestramento classico delle GANs. Un altro approccio viene mostrato in [15] dove Li et al. propongono un modello ricorrente chiamato *Comisr* per la super risoluzione dei video compressi e senza introdurre artefatti, il Comisr per eseguire la super risoluzione l'utilizza tre moduli: modulo ricorrente bidirezionale di distorcimento, modulo di stima per il flusso di preservazione dei dettagli e un modulo per il miglioramento laplaciano.

### 2.1.2 Rimozione degli artefatti e super risoluzione per immagini

Quando usiamo i servizi di messaggistica come Telegram o Whatsapp, si solito a condividere con amici e parenti delle foto o video fatti con le camere dei nostri smartphone. Il servizio di messaggistica per risparmiare la banda di trasmissione dati, invia ai nostri amici una immagine compressa. Anche in questo caso, l'uso di un algoritmo di compressione genera artefatti che riducono la qualità visiva e in taluni casi anche la bellezza di una immagine. Molti sforzi sono stati fatti per eliminare gli artefatti e al tempo stesso eseguire la super risoluzione di un'immagine, per esempio in [14] Li et al. propongono il modello *Best-buddy GAN* per il compito restaurazione chiamato SISR (*Single image super-resolution*), la rete riesce a generare immagini in risoluzione 4K ad un dettaglio molto alto grazie al fatto che riesce a recuperare texture realistiche, e mantenendo la naturalezza delle immagini. In [33] Zang et al. propongono il modello RankSRGAN (*Super Resolution Generative Adversarial Networks with Ranker*) per il compito di aumentare la risoluzione di un'immagine (*Super resolution*) capace di ottimizzare il generatore nel minimizzare diverse metriche percettive<sup>2</sup> non differenziabili come NIQE (*Natural Image Quality Evaluator*), usando un Ranker capace di apprendere una classifica di metriche percettive per valutare la qualità delle immagini, il modello viene addestrato usando un set di addestramento costruito con le immagini

---

<sup>2</sup> Nel capitolo 3 parleremo delle loss percettive, ci concentreremmo sulle metriche SSIM e LPIPS.

ottenute da altri metodi progettati per risolvere il problema dell'aumento di risoluzione di un'immagine. In [30] Yeh et al. propongono un modello per rimuovere gli artefatti da un'immagine in jpeg con una rete CNN, che fonde immagini multiscala dello stesso input e formulando il problema della rimozione dei artefatti come la differenza o residuo tra l'immagine jpeg compressa o con l'immagine jpeg pulita dagli artefatti. Queste sono solo alcune delle tecniche possibili per rimuovere gli artefatti e aumentare la risoluzione delle immagini compresse, in [27] Wang et al. fanno un riassunto di diversi tecniche GAN per l'aumentare la risoluzione delle immagini e video.

### 2.1.3 Tecniche generative di sintesi video a video e di immagini

In alcune applicazioni di *Intelligent System* si ha l'esigenza di trovare una funzione di mapping che associa l'input, per esempio un video che descrive la geometria della scena, con un video di output che corrisponda fedelmente e realisticamente al video di input. In [25] Wang et al. propongono una GAN condizionale capace di sintetizzare video ad alta risoluzione, fotorealistici e temporalmente coerenti. Un dettaglio interessante di questa tecnica è che usa due discriminatori condizionali, una per le immagini e un altro per i video per prevenire il fenomeno del *mode collapse*<sup>3</sup>. In [26] Wang et al. propongono una rete GAN per la sintesi di video per i servizi di videoconferenza, la rete estrae in modo non supervisionato dei punti tridimensionali che usa per scomporre i tratti dei volti inquadrati dalla videocamera, e li slega da quelli legati dal movimento, in questo il modello riesce a generare un output coerente e risparmiando anche sulla banda di trasmissione dati.

In [16] Liu et al fanno una panoramica sull'utilizzo delle GAN per la generazioni di video di sintesi e di immagini.

### 2.1.4 Self attention per i modelli GANs

I modelli GAN che si occupano di applicazioni di computer vision sono realizzate principalmente con layer convoluzionali, dal momento che questi layer da anni sono la base delle più raffinate e potenti tecniche di deep learning legate alle immagini e ai video. Di recente per aumentare

---

<sup>3</sup> Il mode collapse è un fenomeno frequente che può verificarsi durante il processo di addestramento di modello GAN. Esso si verifica quando la rete generatrice genera dei dati che eludono facilmente la rete discriminatrice e continua per tutta la durata dell'addestramento a generare solo questi dati.

le prestazioni dei modelli si è fatta strada la possibilità di usare layer che calcolano l'*attenzione*. L'attenzione è una idea nata nel campo della *natural language processing*<sup>4</sup> con le reti *encoder-decoder*<sup>5</sup> per il compito di traduzione. Nella NLP, l'attenzione è una misura che permette alla rete neurale di cogliere il contesto di una parola nella frase in input, questa misura viene calcolata mettendo una parola in relazione con tutte le altre parole che compongono la frase, e assegnando a queste parole un peso che indica quanto quella parola è importante per capire il contesto della parola di cui vogliamo calcolare l'attenzione. Per calcolare l'attenzione vengono impiegati dei metodi chiamati meccanismi di attenzione, tra i più famosi o interessanti meritano sicuramente menzione il meccanismo di attenzione di Bahdanau [1], il meccanismo di attenzione di Vaswani [23] che ha fatto nascere il filone dei modelli Transformer, e il meccanismo del Longformer [2] di Beltagy et al. che usa un mix di attenzione locale e globale per ridurre la complessità spaziale del meccanismo di attenzione consentendo quindi l'elaborazione di sequenze dati molto lunghe.. In [31] Zhang et al. decidono di portare l'attenzione anche nel campo della Computer Vision, proponendo un nuovo modello di GAN chiamato SAGAN, dove sia il generatore che il discriminatore fanno uso di layer di self attention piuttosto che usare i tradizionali layer convoluzionari questo perché Zhang et al. hanno osservato che il loro modello i dettagli generati fanno uso delle feature dell'intera mappa, permettendo di ottenere delle immagini realissime e consistenti.

## 2.2 ALGORITMI PER IL VIDEO INPAINTING

Il bisogno di restaurare vecchi video e immagini ha spinto la ricerca a creare una nuova area della computer vision chiama *video inpainting*. Gli algoritmi di video inpainting tradizionali utilizzano approcci basati su patch di pixel, ovvero dei gruppi di pixel usati per riempire una regione corrotta di pixel. Le patch selezionate da questi algoritmi vengono dalle vicinanze della regione corrotta, e una volta trovata una patch che si adatta bene al vicinato di pixel della regione, non si fa altro che una

---

<sup>4</sup> La NLP (natural language processing) è un area dell'intelligenza artificiale che cerca di donare alle macchine la possibilità di analizzare e comprendere il testo scritto dall'uomo per compiti di: classificazione, domanda&risposta, generazione testo per scrivere riassunti ecc...

<sup>5</sup> Le reti encoder-decoder sono un architettura di reti neurali caratterizzate da due moduli: encoder e decoder. La forma di questa rete e dei suoi moduli, dipende dal tipo dei dati e dal compito da eseguire

copia della patch stessa [4] e [18]. Con l'avvento del deep learning si sono affacciate tecniche molto potenti come *Deep flow-guided video inpainting* di Xu et al. [29], dove viene usato il flusso ottico per propagare il colore dei pixel nelle regioni danneggiate/mancanti. Un altro approccio molto recente per il video inpainting basato su deep learning è il modello FuseFormer [17] di Liu et al. Il FuseFormer è un modello basato su Transformer progettato per il video inpainting, l'aspetto caratterizzante di questa tecnica è quella di fondere le feature map a grana fine basate sulle operazioni di soft split e soft composition. La soft split divide la feature map in molte patch che possono avere anche dei pixel in comune con altre patch, mentre la soft somposition ricuce le patch in un'intera feature map in cui vengono riassunti i pixel delle regioni sovrapposte. Con questo approccio Liu et al sono riusciti ad ottenere risultati allo stato dell'arte.



# 3

---

## VIDEO INPAINTING E SUPER RISOLUZIONE

---

In questo capitolo viene riportato il lavoro svolto per eliminare il problema delle barre orizzontali e il metodo proposto per la super risoluzione e l'aumento della qualità dell'immagine.

### 3.1 CHE COS'È IL VIDEO INPAINTING

Il *video inpainting* è un area della computer vision che mira a proporre delle tecniche per rimuovere oggetti o soggetti indesiderati in un'immagine o un video, oppure a ripristinare regioni corrotte di un'immagine o di un video. Nel nostro caso siamo interessati a restaurare vecchi video salvati su nastro, che hanno subito l'usura del tempo e con diversi problemi come le bande orizzontali; Le Figure 1 e 2 mostrano dei frame<sup>1</sup> afflitti da questo problema.

#### 3.1.1 Motion field e optical flow

Molte delle recenti tecniche sviluppate per il video inpainting usano l'*optical flow* (in italiano *flusso ottico*), definito come un'approssimazione del motion field (in italiano campo di moto). Per motion field si intende la proiezione del vettore velocità di un punto nello spazio tridimensionale sul piano immagine (bidimensionale). Le Figure 3 e fig:Projection mostrano la rappresentazione motion field e la proiezione di un punto nello spazio sul piano immagine.

Attraverso il flusso ottico gli algoritmi apprendono il movimento di oggetti, superfici e bordi tra frame adiacenti di un video.

---

<sup>1</sup> Il termine frame è usato come sinonimo di immagine in diverse discipline come la fotografia, il video-editing, la computer graphics e la computer vision.

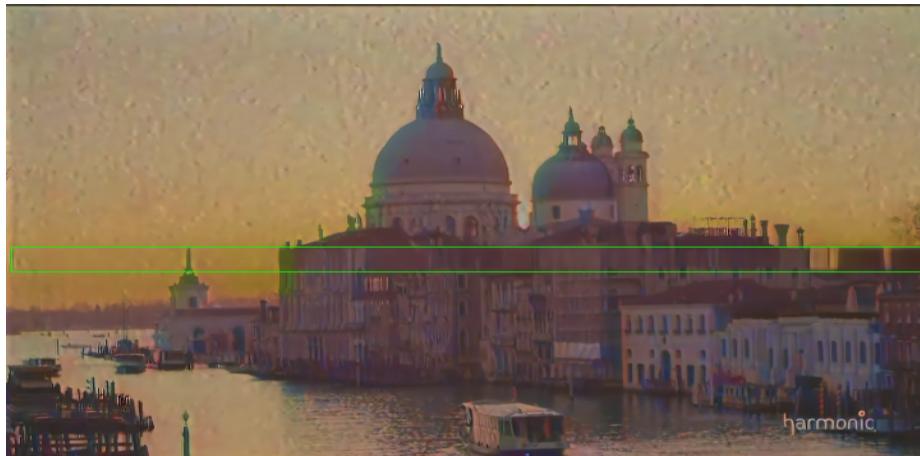


Figura 1: Effetto della distorsione delle barre orizzontali parte 1

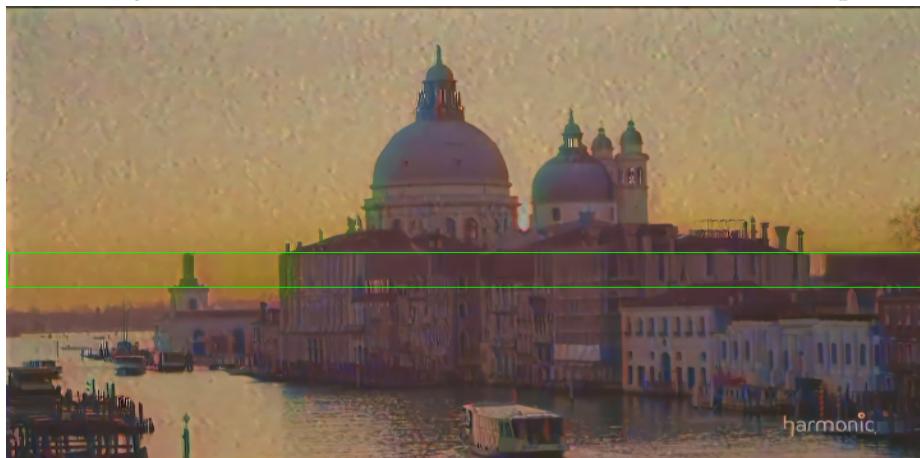


Figura 2: Effetto della distorsione delle barre orizzontali parte 2

### 3.1.2 *L'effetto della banda orizzontale nei video*

Come si vede nelle figure Figure 1 e 2 la banda orizzontale nei video si manifesta come un effetto ottico che distorce l'immagine. Per capire l'effetto di questo disturbo sui frame, è stato eseguito un pre-processing dei dati in python con la libreria OpenCv. Per prima cosa si è trasformato tutti i frame in tensori per permettere l'elaborazione e la manipolazione dei frame, poi si è convertito il canale RGB di tutti frame in scala di grigi, da queste elaborazioni si è ottenuto poi frame nuovi definiti come la differenza in valore assoluti del frame al tempo  $t+1$  e al tempo  $t$ , sui nuovi frame si è creato delle maschere con il metodo thresholding di

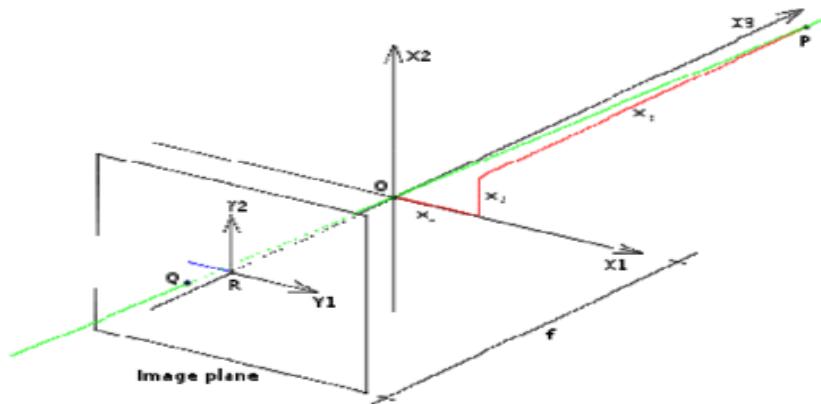


Figura 3: Rappresentazione del motion field

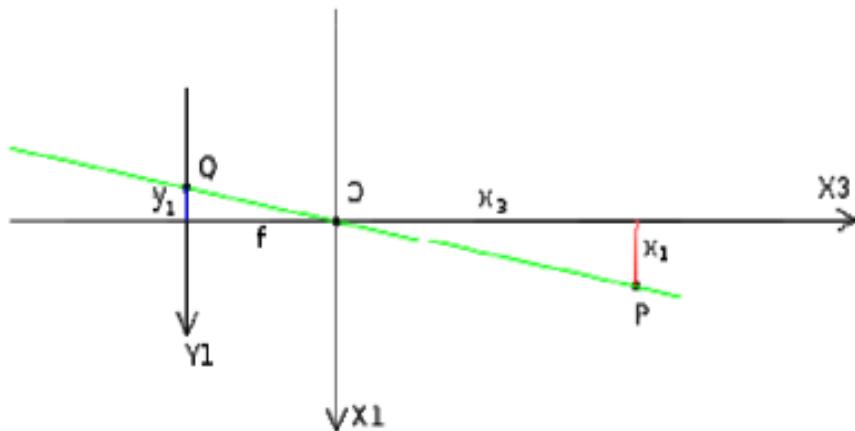


Figura 4: Proiezione di un punto del mondo reale nel piano immagine

Otsu<sup>2</sup> Per ogni maschera è stato creato un istogramma in cui ogni bin rappresenta una riga della maschera e l'altezza del bin rappresenta il numero di pixel accesi; dalla Figura 5 possiamo vedere come la banda orizzontale provochi uno spostamento dei pixel accesi.

---

<sup>2</sup> Il thresholding di Otsu è una tecnica di computer vision nata nel pre processing delle immagini per scopi di segmentazione. La tecnica prende in input una immagine in bianco e nero, e trova una threshold che verrà usata per mettere a 1 un pixel che supera la soglia e a 0 altrimenti, creando di fatto una maschera che può essere usata per vari contesti come la rivelazione un tumore mammario oppure per rilevare disastri ambientali in immagini satellitari.

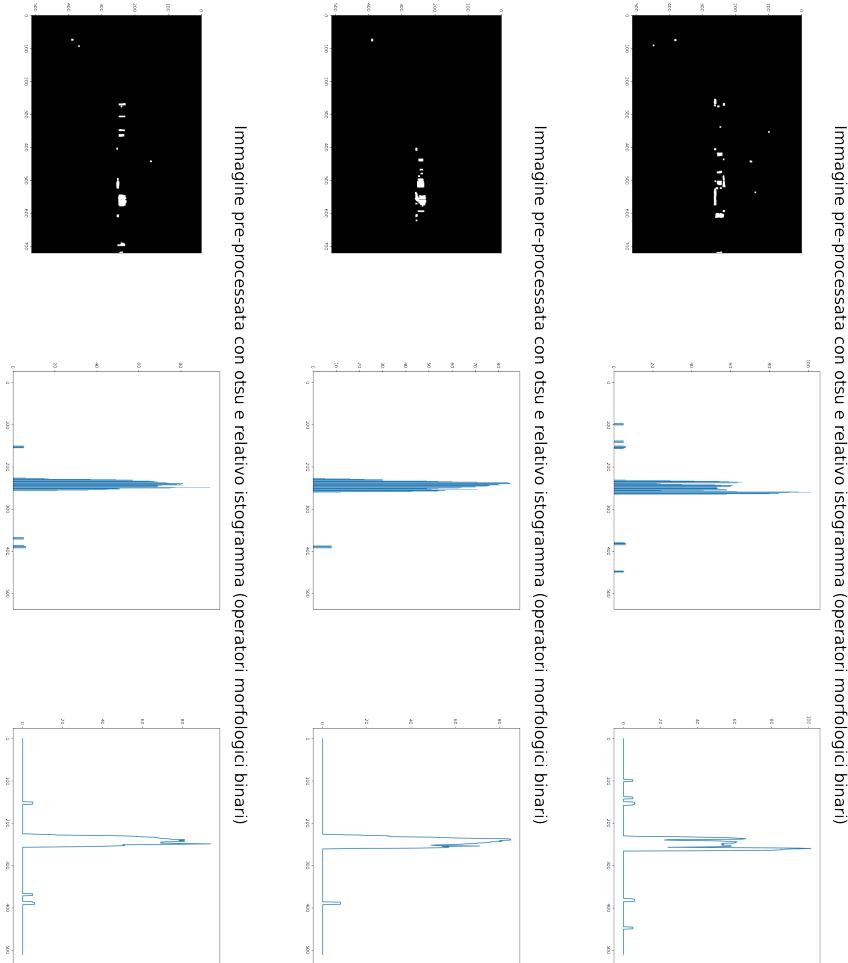


Figura 5: Alcune maschere e relativi istogrammi

### 3.1.3 Eliminazione della banda orizzontale con Flow-edge Guided Video Completion

In questo lavoro per eliminare il problema della banda orizzontale si è scelto l'algoritmo *Flow-edge Guided Video Completion* di Gao et al [7]. A differenza dei metodi visti nel capitolo precedente, FGVC è un metodo di video inpanting basato su deep learning che estrae e completa i bordi

di movimento e li usa per guidare il completamento del flusso. Questo metodo si basa su nuove idee che analizzeremo in dettaglio nelle sotto sezioni apposite. Per capire come lavora questo metodo partiamo dall'analizzare la Figura 6 che espone la pipeline di lavoro. Il metodo prende in ingresso una sequenza di frame<sup>3</sup> e una sequenza di maschere legate a quest'ultima<sup>4</sup>, gli input vengono passati alla prima fase della pipeline chiamata *edge-guided flow completion* che calcola l'optical flow in avanti e indietro con FlowNet2 tra frame adiacenti e non adiacenti in accoppiata con un operatore omografico, e usa ancora i frame non adiacenti per estrarre e calcolare i bordi; queste informazioni verranno poi usate per calcolare il flusso ottico *piecewise-smooth*. La seconda fase è chiamata *temporal neighbor generation*. In questa fase si seguono le traiettorie del flusso per calcolare un insieme di pixel candidati, tale calcolo viene eseguito utilizzando delle metriche di validità per ogni pixel mancante. Nella terza fase, chiamata *fusion e hallucination*, vengono fusi i pixel candidati nel dominio del gradiente per ogni pixel mancante, usando una media ponderata che usa le metriche di validità calcolate nella fase precedente; successivamente viene scelto un frame con la maggior parte dei pixel mancanti e riempito con l'inpainting calcolato. Quanto detto è una descrizione ad alto livello di FGVC; nei paragrafi seguenti analizzeremo tutte le fasi della pipeline di lavoro.

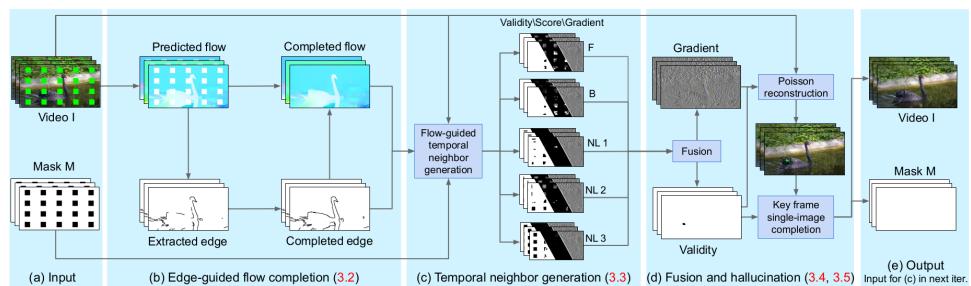


Figura 6: Pipeline di lavoro di FGVC

<sup>3</sup> per sequenza di frame si intende una sequenza di immagini prese in ordine cronologico da un video

<sup>4</sup> Una maschera è una matrice di pixel in cui gli unici valori assunti dai pixel sono 0 o 1, le maschere indicano quali regioni del frame devono essere sintetizzate per fare video inpainting

### 3.1.4 Edge-guided flow completion

Il primo passo di FGVC consiste nel quello di calcolare i flussi tra frame adiacenti ed estrarre i bordi da frame non adiacenti. identifichiamo con  $I_i$ ,  $M_i$ ,  $p$  rispettivamente il tensore<sup>5</sup>, la maschera dell'  $i$ -esimo frame e  $p$  un pixel qualsiasi e supponiamo che:

$$M_i(p) = \begin{cases} 1, & \text{se } p \text{ non è presente} \\ 0, & \text{se } p \text{ è presente} \end{cases}$$

Per calcolare il flusso tra due frame  $I_i$  e  $I_j$  viene usata la rete FlowNet2 [10] come segue:

$$F_{i \rightarrow j} = \mathcal{F}(I_i, I_j), \quad |i - j| = 1 \quad (3.1)$$

Tuttavia la 3.1 non riesce a fare una buona stima del flusso quando i frame non sono locali<sup>6</sup>. In questo caso il flusso viene calcolato con FlowNet2 riscrivendo il secondo input come output dell' operatore di deformazione omografica  $\mathcal{H}_{i \rightarrow j}$ , al output di FlowNet2 si somma l'inverso dell'operatore di deformazione omografica  $H_{i \rightarrow j}$  come mostrato nell'equazione 3.2.

$$F_{i \rightarrow j} = \mathcal{F}(I_i, \mathcal{H}_{j \rightarrow i}(I_j)) + H_{i \rightarrow j}, \quad |i - j| > 1 \quad (3.2)$$

Nell'equazione 3.2  $\mathcal{H}_{j \rightarrow i}$  viene usato per compensare i moti larghi che occorrono spesso nei frame non locali mentre aggiungiamo  $H_{i \rightarrow j}$  perchè non siamo interessati al flusso tra i frame allineati con l'omografia ma tra i frame originali. Per stimare l'omografia di allineamento viene impiegato l'algoritmo RANSAC<sup>7</sup> sulle feature locali calcolate con l'algoritmo ORB<sup>8</sup>. Per riempire le regioni mancanti con del contenuto plausibile, oltre ai flussi, la fase richiede di calcolare anche i bordi che serviranno a stimare il flusso delle regioni mancati. Per rilevare i bordi viene usato l'algoritmo di

<sup>5</sup> Il tensore di una immagine o frame RGB è una matrice 3D che ha le stesse dimensioni dell'immagine per ciascuna delle componenti dello spazio RGB

<sup>6</sup> I frame adiacenti si dicono locali; quelli non adiacenti si dicono non locali.

<sup>7</sup> RANSAC (RANdom SAMple Consensus) è un algoritmo di ottimizzazione per configurare un modello di regressione con un sottoinsieme di dati inlier, ovvero privi di dati anomali.

<sup>8</sup> ORB (Oriented FAST and rotated BRIEF ) è un algoritmo di detection di feature locali veloce e robusto presentato per la prima volta da Ethan Rublee et al. nel 2011 e spesso viene usato usato come alternativa a SIFT perchè più veloce.

Canny<sup>9</sup> per estrarre la feature map  $E_{i \rightarrow j}$  dei bordi del flusso. I bordi delle regioni mancanti vengono predetti invece attraverso una rete neurale per il completamento dei bordi usando come input le maschere e il frame. Poiché siamo interessati ad un completamento smooth tranne che ai bordi, calcoliamo una soluzione che minimizzi tutti i gradienti dei pixel (ad eccezione di quelli sui bordi); il flusso completo viene quindi calcolato come indicato nell'equazione 3.3:

$$\operatorname{argmin}_{\tilde{F}} \sum_{p|\tilde{E}(p)=1} \|\Delta_x \tilde{F}(p)\|_2^2 + \|\Delta_y \tilde{F}(p)\|_2^2 \quad (3.3)$$

Nell'equazione  $\Delta_x$  e  $\Delta_y$  indicano rispettivamente l'operatore di differenza in avanti orizzontale e verticale.

L'equazione 3.3 è soggetta al vincolo  $\tilde{F}(p) = F(p)|M(p) = 0$  e viene risolta usando il metodo dei minimi quadrati.

### 3.1.5 Temporal neighbor generation

La terza fase della pipeline viene chiamata *Temporal neighbor generation*, e impiega i flussi completati per guidare il completamento del colore nelle regioni mancanti. In questa fase, per ogni pixel di una regione mancante, troviamo un insieme di pixel vicini temporalmente, e calcoliamo un colore fondendo questi pixel mediante l'uso della media ponderata. I flussi stabiliscono una connessione tra pixel correlati attraverso i frame e vengono sfruttati per guidare il completamento propagando quindi i colori dai pixel noti attraverso le regioni mancanti lungo le traiettorie di flusso. Invece di "spingere" i colori propagandoli alla regione mancante è più desiderabile seguire transitivamente i collegamenti di flusso in avanti e indietro per un dato pixel mancante, fino a raggiungere pixel conosciuti, e "tirare" i loro colori. Controlliamo la validità del flusso misurando l'errore di coerenza del ciclo avanti-indietro con l'equazione 3.4

$$\tilde{D}_{i \rightarrow j}(p) = \|F_{i \rightarrow j}(p) + F_{j \rightarrow i}(p + F_{i \rightarrow j}(p))\|_2^2 \quad (3.4)$$

Il tracciamento si conclude quando incontriamo un errore maggiore di  $\tau = 5$  pixel. Chiamiamo i pixel che soddisfano questa condizione di tracciamento come *local temporal neighbors* (vicini locali-temporali), essi sono calcolati concatenando il vettore di flusso tra frame adiacenti.

---

<sup>9</sup> L'algoritmo di Canny è un operatore per il riconoscimento dei contorni ideato nel 1986 da John F. Canny

A volte capita che non siamo in grado di raggiungere un pixel locale, perché la regione mancante si estende fino alla fine del video a causa di un flusso non valido o perché incontriamo una barriera di flusso<sup>10</sup>, per visualizzare il problema delle barriere si guardi a tal proposito la Figura 7.

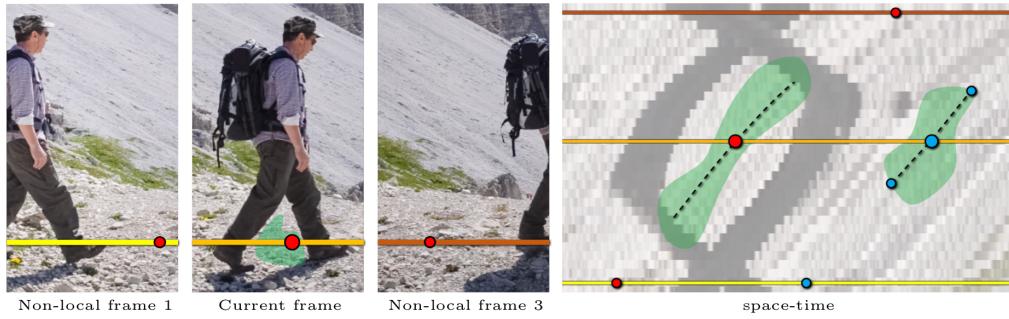


Figura 7: La figura di destra mostra una visualizzazione spazio-temporale per le linee di scansione evidenziate nelle immagini di sinistra. Le regioni verdi sono mancanti. La linea gialla, arancione e marrone nella figura di destra rappresenta la linea di scansione al primo frame non locale, il frame corrente e il terzo frame non locale, rispettivamente. La figura illustra i candidati al completamento per i pixel rossi e blu (i dischi grandi sulla linea arancione). Seguendo le traiettorie di flusso (le linee nere tratteggiate) fino al bordo della regione mancante, otteniamo i candidati locali per il pixel blu (i dischi piccoli), ma non per il pixel rosso, perché le gambe della persona formano barriere di flusso invalidabili. Con l'aiuto del flusso non locale che si collega ai fotogrammi temporalmente distanti, otteniamo dei vicini extra non locali per il pixel rosso (i dischi rossi sulla linea gialla e marrone). Come risulta-to, possiamo vedere il vero sfondo che è coperto dalle gambe che si muovono.

Per alleviare il problema delle barriere di flusso vengono calcolati dei pixel chiamati vicini temporali non locali. Il calcolo di questi pixel consiste nel calcolare il flusso verso in insieme di frame temporalmente distanti che tagliano le barriere del flusso, riducendo drasticamente il numero di pixel isolati e la necessità di allucinazioni. Per ogni fotogramma, calcoliamo il flusso non locale a tre frame aggiuntivi utilizzando il metodo dell'omografia allineata vista nell' equazione 3.2. Per semplicità,

<sup>10</sup> Le barriere possono portare a grandi regioni di pixel isolati senza vicini temporali locali e si verificano in ogni confine principale del movimento perché l'occlusione/disocclusione rompe la coerenza del ciclo avanti-indietro in quel punto.

selezioniamo sempre il primo, il medio e l'ultimo frame del video come vicini non locali. La Figura 8 mostra un esempio.

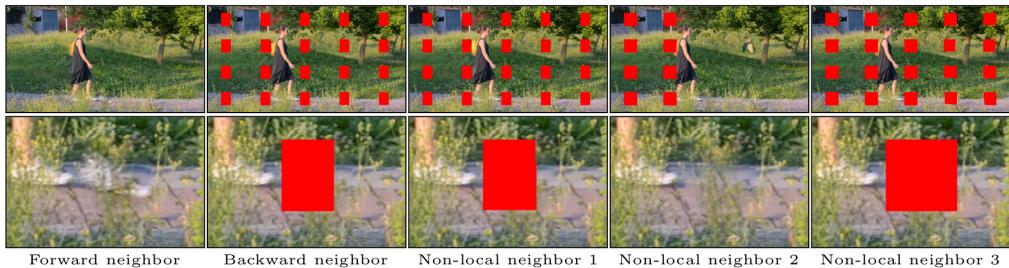


Figura 8: Vicini temporali: i vicini temporali non locali (il secondo vicino non locale in questo caso) sono utili quando i pixel conosciuti corretti non possono essere raggiunti con catene di flusso locali a causa di barriere di flusso (il rosso indica vicini non validi).

### 3.1.6 Fusion e hallucination

L'ultima fase della pipeline è chiamata *fusion e hallucination*. Essa prende in input i vicini temporali per i pixel mancanti calcolati nella fase precedente per fonderli e sintetizzarli in modo da ottenere il colore completo. Sia  $p$  un pixel mancante di una regione vuota e sia  $k \in N(p)$  l'insieme dei vicini temporali locali e non locali validi che riportano un errore di flusso inferiore a  $\tau$ . Calcoliamo il colore completato come media ponderata dei colori candidati  $c_k$  come riportato nell'equazione 3.5.

$$\tilde{I}(p) = \frac{\sum_k w_k c_k}{\sum_k w_k} \quad (3.5)$$

I pesi  $w_k$  sono calcolati come espresso nell'equazione 3.6

$$w_k = e^{\frac{-d_k}{T}} \quad (3.6)$$

Dove  $d_k$  indica l'errore di coerenza del ciclo avanti-indietro  $D_{i \rightarrow j}(p)$  per i vicini non locali. Impostiamo  $T$  al valore  $T = 0.1$ , per pesare i vicini che hanno un grande errore di flusso.

La Figura 9 mostra un esempio in cui il colore calcolato con il flusso presenta delle cuciture visibili quando il colore del frame cambia. Questo evento sfortunatamente è molto frequente e può essere provocato da diversi fattori, come: il cambiamento di illuminazione, di quello delle ombre, dalla vignettatura dell'obiettivo, dal bilanciamento del bianco, ecc...

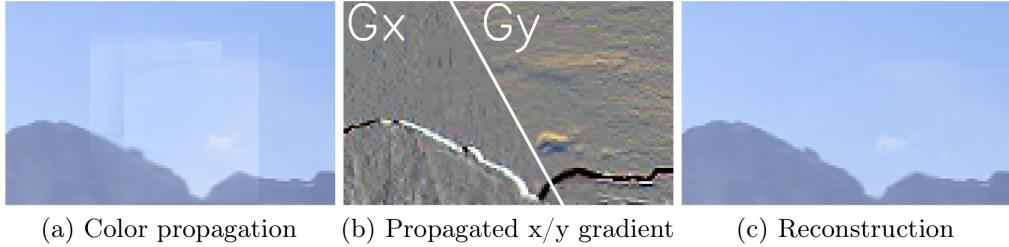


Figura 9: I metodi precedenti operano direttamente nel dominio del colore, producendo delle cuciture (a). FGVC propaga nel dominio del gradiente (b), e poi ricostruisce i risultati con la ricostruzione di Poisson (c).

Questo problema viene risolto riscrivendo l'equazione 3.5 nell'equazione 3.7 per calcolare la media ponderata dei gradienti di colore, piuttosto che calcolare i valori di colore:

$$\tilde{G}_x(p) = \frac{\sum_k w_k \Delta_x c_k}{\sum_k w_k}, \quad \tilde{G}_y(p) = \frac{\sum_k w_k \Delta_y c_k}{\sum_k w_k} \quad (3.7)$$

Le due quantità calcolate nella 3.7 verranno poi usate nella 3.8 che esprime un problema di ricostruzione di Poisson per calcolare l'immagine finale.

$$\operatorname{argmin}_{\tilde{I}} \|\Delta_x \tilde{I} - \Delta(G)_x\|_2^2 + \|\Delta_y \tilde{I} - \tilde{G}_y\|_2^2 \quad (3.8)$$

La 3.8 è soggetta ai vincoli  $\tilde{I}(p) = I(p)|M(p) = 0$  e può essere risolta attraverso i minimi quadrati. Come possiamo vedere dalla Figura 9b operando nel dominio del gradiente, le cuciture di colore vengono eliminate, Figura 9c.

Ad ogni iterazione, propaghiamo i gradienti di colore e otteniamo fino a cinque gradienti candidati, successivamente fondiamo tutti i gradienti candidati per ottenere il valore del colore dei pixel mancanti risolvendo la 3.8. Questo vale solo per tutti i pixel mancanti che hanno i local temporal neighbors validi, perché alcuni pixel mancanti potrebbero non avere alcun local temporal neighbors valido, anche con il flusso non locale, il che, per esempio, accade quando il pixel è occluso in tutti i frame non locali, o quando il flusso è stimato in modo errato. In questo caso il problema viene trattato come un compito di completamento di una immagine singola. Tuttavia se completassimo le restanti regioni mancanti in tutti i frame con questo metodo, il risultato non sarebbe temporalmente coerente. Invece selezioniamo solo un frame con il maggior numero di

pixel mancanti rimanenti e lo completiamo con il metodo dell’immagine singola. Alimentiamo quindi il risultato dell’inpainting come input per un’altra iterazione della nostra intera pipeline (con la notevole eccezione del calcolo del flusso, che non ha bisogno di essere ricalcolato). Nell’iterazione successiva, il frame completato a immagine singola è trattato come una regione nota, e i suoi gradienti di colore sono propagati coerentemente ai frame circostanti. Il processo di completamento iterativo termina quando non ci sono pixel mancanti e mediamente occorrono 5 iterazioni per riempire tutti i pixel mancanti di una regione.

### 3.2 SUPER RISOLUZIONE DELLE IMMAGINI CON SR-UNET

Oggi giorno siamo abituati a visualizzare film, documentari e videogiochi su schermi a risoluzioni altissime, come il 4K o 8K, ma negli anni 90 e primi 2000, quelle risoluzioni erano ancora impensabili, e al massimo ci si accontentava di risoluzioni come l’hdready o il fullhd con la tecnologia vhs<sup>11</sup>. Se facessimo un banale upscaling dei singoli frame a bassa risoluzione, per raggiungere le risoluzioni come i fullhd o il 4k avremmo sicuramente i frame a quella risoluzione, al costo però di introdurre rumori e artefatti che peggiorano la qualità visiva del frame. Da tempo nella computer vision si sono studiati algoritmi che potessero fare un upscaling di qualità di un frame, senza introdurre le problematiche precedentemente esposte. Perseguendo questo obiettivo, nella disciplina è nata una area di ricerca che mira a creare nuovi algoritmi per risolvere al meglio questo problema; quest’area ha visto grandissimi progressi grazie al deep learning con l’arrivo delle GANs e delle reti U-Net. Per questo lavoro abbiamo scelto di usare la rete *sr-unet* [22] di Vaccaro et. al. Prima di parlare della rete, introduciamo per dovere di chiarezza le architetture *GAN* e *u-net* al fine di comprendere meglio la rete scelta.

#### 3.2.1 Generative Adversarial Network

Fino a non molto tempo fa i ricercatori sono stati in grado di realizzare macchine che in qualche modo avessero delle capacità simili a quelle dell’uomo, come la visione e la comprensione dell’ambiente circostante

---

<sup>11</sup> Il VHS è caduto quasi totalmente in disuso nel corso del primo decennio degli anni 2000 per via della tecnologia basata su disco come dvd e blu-ray. Sebbene molto del materiale in vhs sia girato a risoluzioni molto basse, nel 1999 la JVC introdusse sul mercato le D-VHS, videocassette digitali ad alta definizione (720p - 1080i).

in termini di detection<sup>12</sup> e classificazione, per svolgere compiti come riconoscimento di oggetti o di persone. Tuttavia gli algoritmi di machine learning e deep learning hanno fatto molta fatica nel compito di generare nuovi dati, e che possiamo assimilare come capacità alla creatività umana, almeno fino all'arrivo delle **Generative Adversarial Network (GAN)**<sup>13</sup> ideate da Ian Goodfellow nel 2014 [8]. Le GAN sono un'architettura di deep learning che non usa una rete ma bensì due reti che vengono addestrate simultaneamente su compiti diversi. Inizialmente l'architettura fu proposta per risolvere problemi unsupervised, e in seguito venne impiegata con successo anche per problemi semi-supervised e supervised. La prima rete viene chiamata *generatore* e ha il compito di generare dei dati falsi che siano indistinguibili da quelli veri, l'altra rete viene chiamata *discriminatore* perché è un classificatore binario chiamato a distinguere i dati veri da quelli falsi; in Figura 10 è riportato uno schema generico di una GAN.

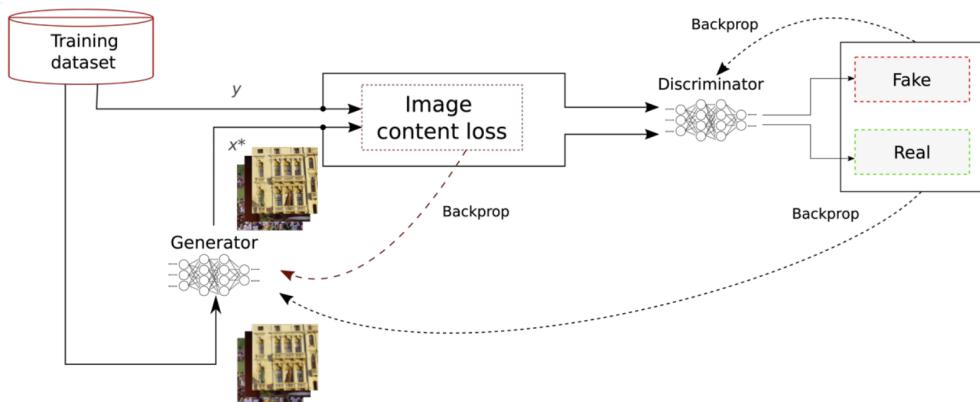


Figura 10: Schema di architettura GAN

Le due reti durante l'addestramento giocano una contro l'altra con l'obiettivo reciproco di battersi e l'addestramento termina quando viene raggiunto un *equilibrio di Nash*. L'equilibrio di Nash è una combinazione di strategie in cui nessuno dei due giocatori può migliorare la propria situazione o il proprio guadagno, e nel caso delle GAN questo equilibrio viene raggiunto quando valgono le seguenti condizioni:

- Il generatore produce dei dati falsi che sono indistinguibili dai dati reali presenti nel dataset di training;

<sup>12</sup> La detection è un compito di computer vision e di machine learning per identificare dei pattern nei dati.

<sup>13</sup> Le reti GAN sono conosciute in italiano come reti generative avversarie

- Il discriminatore può al massimo indovinare casualmente se un particolare dato è vero o falso.

Quando valgono queste condizioni, si dice che l’addestramento della GAN converge. Sfortunatamente è molto difficile raggiungere un equilibrio di Nash per le GAN, a causa di diverse complicanze che sono ancora una questione aperta e la comunità scientifica sta studiando diverse possibili soluzioni. Tuttavia, anche in mancanza di rigorose garanzie matematiche, le GAN hanno raggiunto dei risultati empirici di altissimo livello quando sono state applicate nei settori: della computer vision, della cyber-security, della musica e del natural language processing.

### 3.2.2 Architettura U-Net

Le reti *U-Net* sono nate per scopi di semantic segmentation<sup>14</sup> in ambito medico da Olaf Ronnerberger [20]. Le reti U-Net sono delle reti ad architettura encoder-decoder e devono il nome alla loro forma che ricorda una U; si guardi la Figura 11 dove è riportato lo schema originale della U-Net di Ronnerberger.

La prima metà della rete è la parte di *encoder*, ciascun blocco che lo compone è un layer composto da due layer convoluzionali con kernel di dimensione 3x3, alla cui feature map di uscita viene applicata una funzione non lineare, tipicamente viene usata la ReLU, e poi un’operazione di ritaglio. Alla feature map di uscita di ogni blocco del encoder viene poi applicata un’operazione di max-pooling con kernel di dimensione 2x2 per estrarre le informazioni più significative della feature-map. Lo scopo dell’encoder della U-net è quello di catturare la maggior parte dell’informazione contenuta in un’immagine, successivamente l’informazione calcolata passerà attraverso le skip connection<sup>15</sup> ai blocchi della seconda metà, chiamata *decoder*. Quest’ultima ha come scopo quello di espandere

---

<sup>14</sup> La semantic segmentation è un sotto compito di image segmentation. Il compito di image segmentation è un compito di computer vision che consiste nel dividere un’immagine in regioni di pixel, tali che ogni regione rappresenti una categoria (es alberi, cani, persone, ecc...). Un algoritmo addestrato a questo compito assegna a ciascun pixel dell’immagine una classe. Nella semantic segmentation, l’algoritmo cerca di mappare ogni pixel dell’immagine in una categoria.

<sup>15</sup> Le skip connection sono state introdotte nella letteratura per la prima volta con ResNet per aumentare i gradienti. Le skip connection possono essere viste come una specie di autostrada per il gradiente per raggiungere dei layer che hanno delle difficoltà. In pratica l’input  $x$  viene fatto passare al layer successivo senza che questo subisca delle trasformazioni.

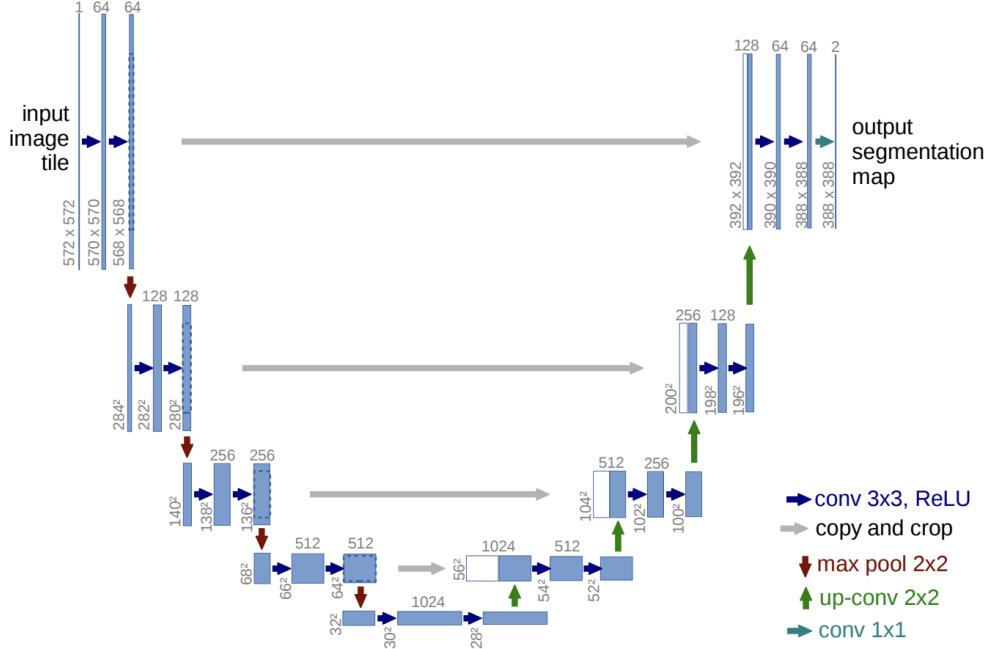


Figura 11: Schema orinale della architettura U-Net

le feature map di output dei propri blocchi. I blocchi del decoder presentano al loro interno dei blocchi convoluzionali in modo simile ai blocchi dell'encoder, ma non viene eseguita nessuna operazione di ritaglio e la feature map di input di ogni blocco del decoder viene sovra campionata con un operazione di upsampling.

### 3.2.3 Metriche di percezione

Per valutare il lavoro di super risoluzione svolto da sr-unet abbiamo deciso di utilizzare delle metriche di percezione. Le metriche di percezione sono utili per i compiti di restauro immagini e di super risoluzione, perché ci aiutano a capire come aggiustare un'immagine in bassa risoluzione e/o rumorosa, in modo tale che sia il simile il più possibile alla sua loro controparte in alta risoluzione e alla massima qualità. Le metriche di percezione si dividono principalmente in due categorie: *metriche full-reference* e *metriche no-reference*.

Le metriche full-reference misurano la differenza percettiva tra un'immagine di alta qualità e la corrispettiva immagine ricostruita, partendo dall'immagine in bassa risoluzione e/o rumorosa. Le metriche

full-reference sono simili alle metriche supervisionate, e si raggiungono dei risultati accurati per il compito di restauro immagini e di super risoluzione. Abbiamo scelto per valutare i nostri esperimenti queste metriche dal momento che possediamo i video in alta risoluzione e alla massima qualità. Le metriche no-reference misurano la ricostruzione dell'immagine in input senza l'uso della controparte in alta risoluzione e di qualità. Non usando quindi un'immagine di riferimento, i risultati che si possono ottenere con esse sono meno precisi ma comunque validi. Queste metriche sono quindi simili alle metriche interne che si usano tipicamente nell'apprendimento non supervisionato come il clustering. Tipicamente si usano in scenari dove non è possibile avere le fonti originali per fare il confronto.

### *Structural similarity (SSIM)*

La metrica *Structural similarity (SSIM)* è una delle due metriche di giudizio che abbiamo scelto per giudicare il lavoro svolto da sr-unet. La SSIM nasce nel 2001 dagli studi di Zhou Wang e Alan Bovik sotto il nome di Universal Quality Index (UQI), o anche di Wang–Bovik Index, per valutare la qualità percettiva delle immagini. Successivamente la metrica fu perfezionata attraverso la collaborazione Eero Simoncelli e Hamid Sheikh, e fu rilasciata nel 2004 con il nome Structural similarity (SSIM) come riportato in [28]. La SSIM considera la degradazione dell'immagine come un cambiamento nell'informazione strutturale, incorporando anche importanti fenomeni percettivi, inclusi i termini di mascheramento della luminanza e del mascheramento del contrasto. Questo concetto è basato sull'idea che i pixel di un'immagine hanno una forte inter-dipendenza tra di loro, specialmente quando sono vicini. A differenza delle metriche come Mean Square Error (MSE), in cui si calcolano gli errori assoluti per ogni pixel, il valore di distanza tra due immagini, espresso in SSIM, è basato sul cambiamento strutturale dell'intera figura. Il range di valori di SSIM è

$$-1 \leq \text{SSIM} \leq 1$$

dove:

- $-1$  è il valore minimo e indica che le due immagini valutate sono all'apposto;
- $0$  indica che le due immagini valutate non hanno alcuna somiglianza strutturale;

- 1 è il valore massimo e indica che le due immagini valutate sono molto somiglianti.

Nelle Figure 12 e 13 sono riportate degli esempi di immagini dove è stata applicata la SSIM, come possiamo vedere la metrica è molto sensibile alle applicazioni di algoritmi di compressione come jpeg, al rumore e agli effetti di sfocatura come il blur.

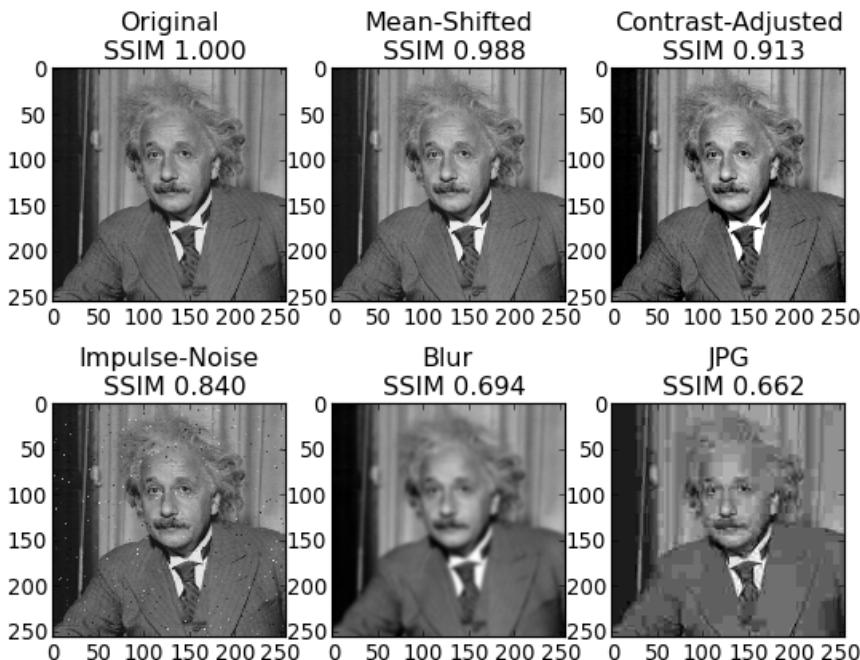


Figura 12: Valori di SSIM a confronto

### *Learned Perceptual Image Patch Similarty (LPIPS)*

L'acronimo LPIPS sta per *Learned Perceptual Image Patch Similarty* ed è una metrica che indica la distanza percettiva tra due immagini, fu proposta nel 2018 da Zhang et al [32]. L'obbiettivo della metrica LPIPS è quello di trovare una funzione che riesca ad interpretare al meglio i giudizi percettivi che stanno alla base della visione umana. Per farlo gli ideatori di questa metrica hanno raccolto un dataset molto grande di giudizi umani, sulla somiglianza tra immagini distorte che avessero problemi di: rumore, sfocatura, compressione e distorsione fotometrica. La ricerca di questa funzione viene compiuta su questo dataset attraverso il fine-tuning

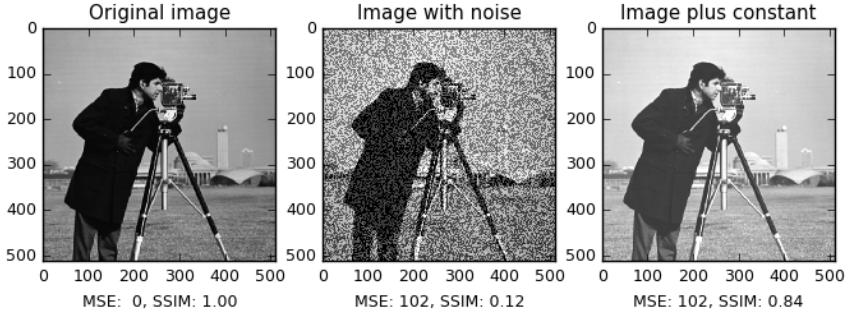


Figura 13: Valori di SSIM a confronto.

di una rete neurale convoluzionale pre-addestrata come VGG o alex-net. Negli ultimi layer convoluzionali di VGG (oppure alex-net), si trovano delle feature di alto livello che possono essere utilizzate per diversi scopi, come quello della creazione di una nuova metrica, perché sono piene di informazioni utili. LPIPS viene calcolata come distanza tra due patch<sup>16</sup>  $x$ ,  $x_0$  secondo l'equazione 3.9:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)\|_2^2 \quad (3.9)$$

dove:

- $x$  è la patch di riferimento,  $x_0$  è la patch distorta;
- $l$  è l' $l$ -esimo layer convolutivo presente nello stack dei layer della rete usata dalla metrica LPIPS;
- $H_l, W_l$  sono rispettivamente l'altezza e la larghezza della feature-map di output del  $l$ -esimo layer;
- $w^l \in \mathbb{R}^{C_l}$  è un vettore dalle dimensioni pari a quello del numero di filtri  $C_l$  del  $l$ -esimo layer;

<sup>16</sup> Una patch è una regione di pixel di una immagine. Tipicamente le patch sono o quadrate (più frequenti) o rettangolari (meno frequenti) e di dimensioni piccole, es 20x20. La loro forma dipende dal contesto o dall'algoritmo usato e a volte si usa il termine patch per riferirsi all'immagine nella sua interezza

- $\hat{y}^l, \hat{y}^l$  sono le feature map normalizzate della patch di riferimento e la l-esima feature map normalizzata della patch distorta.

Come tutte le distanze matematiche abbiamo che:

- il valore minimo è 0 ed indica che due immagini sono uguali;
- il valore massimo può tendere al infinito, il che indica che le due immagini non sono per niente simili.

In Figura 14, si visualizza un esempio pratico, ricavato dall’articolo di LPIPS [32], di come le reti neurali, siano più vicine alle percezione umana rispetto ai metodi standard ( $L_2$ /PSNR, SSIM [28] e FSIM [61]), usati per la valutazione di immagini.



Figura 14: Differenze percettive tra metodi tradizionali, reti neurali e umani.

In Figura 15, viene riportato un confronto diretto tra LPIPS e SSIM, usando l’immagine di Lenna come benchmark. Come possiamo vedere LPIPS sull’immagine originale LPIPS è molto suscettibile al rumore e agli effetti di sfocatura come il blur quando questi sono molto evidenti. Nel 2019 è stata proposta una versione ottimizzata di LPIPS chiamata E-LPIPS da Markus Kettunen e Erik Häkkinen [12]. Questa versione è capace di resistere ad attacchi avversari capaci di perturbare l’immagine in modo significativo ma a cui LPIPS assegnerebbe un giudizio valido.

### 3.2.4 Dataset: BVI-DVC

Per allenare i nostri modelli, abbiamo scelto come riferimento il dataset BVI-DVC. Questo dataset è stato creato per essere la fonte dati più grande ed esaustiva per l’addestramento di reti neurali che risolvessero il compito della compressione video profonda. Al suo interno troviamo 800 sequenze video a varie risoluzioni, si va da 270p fino a 2160p. Per i nostri scopi abbiamo scelto alcune sequenze video di BVI-DVC, e successivamente per ciascun video sono state creati cinque varianti alla risoluzione di

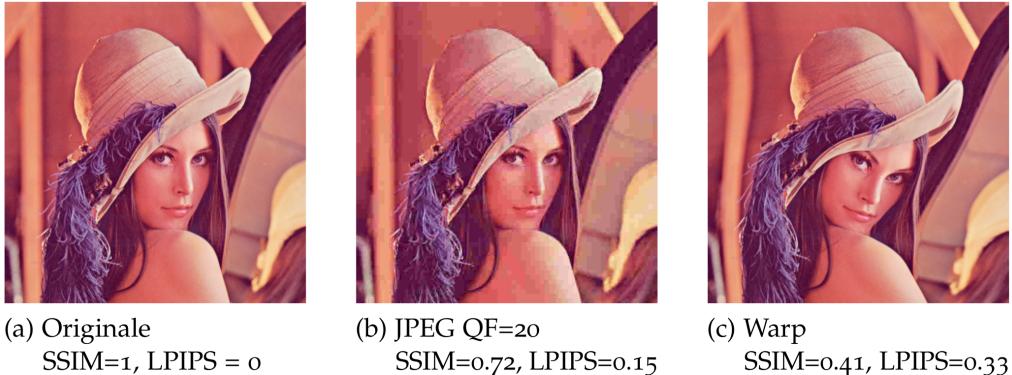


Figura 29: Valori di LPIPS e SSIM a confronto

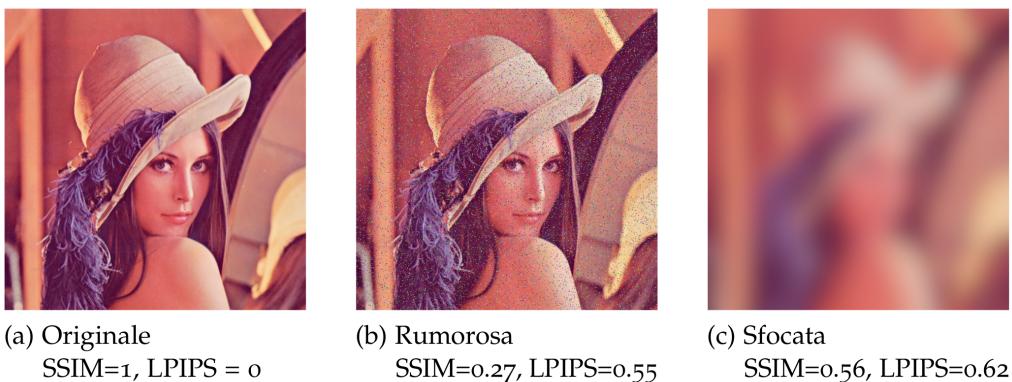


Figura 15: Valori di LPIPS e SSIM a confronto

720p e 1080p con il software *Adobe After Effects*, ciascuna variante è stata pesantemente rovinata con un tipo di rumore e varie manipolazioni sul colore e sull'illuminazione ed altri effetti tipici dei vhs rovinati, le Figure dalla 17 al 21 mostrano lo stesso frame estratto dal video di Venezia e modificato secondo uno dei preset che abbiamo definito per simulare un video su vhs rovinato. La tabella 1 riporta i valori dei parametri che abbiamo usato con After Effects per creare i video. Di questi video inoltre abbiamo preso anche gli originali alla risoluzione di 2160p da usare come ground truth, per valutare i risultati delle reti. La Figura 16 mostra alcuni frame estratti a caso dai diversi video che compongono il BVI-DVC.

Per creare il dataset di training e di test, abbiamo creato degli script bash che usano la libreria *ffmpeg* per campionare i frame dai video, e vengono poi salvati in formato jpeg per motivi di spazio.

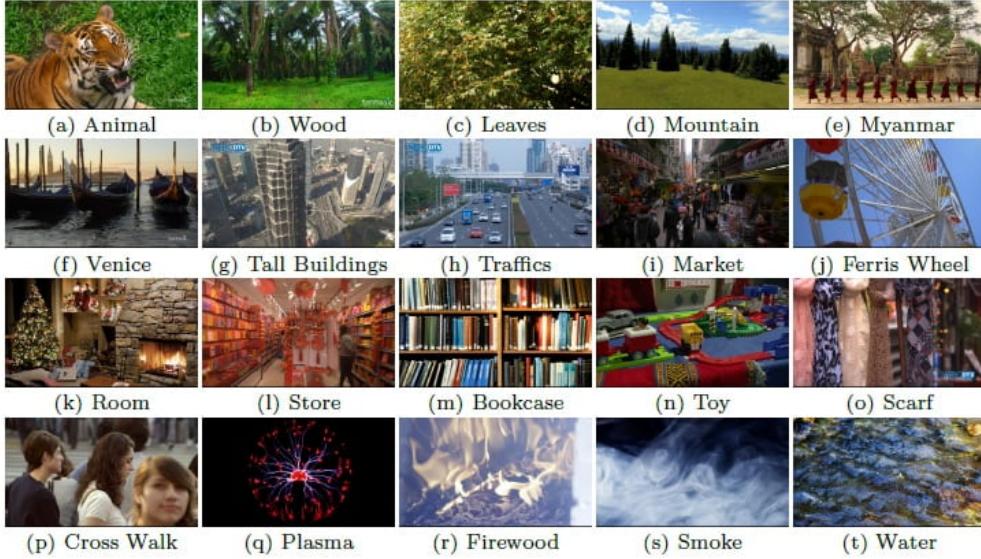


Figura 16: Estrazione di alcuni frame proveniente da diversi video che compongono il dataset BVI-DVC

### 3.2.5 SR-unet

Per la generazione delle immagini abbiamo scelto la rete sr-unet di Vacaro et al. [22] come rete generatrice. I principali contributi e motivi che ci hanno spinto nella scelta di questa rete sono: il miglioramento dell’architettura u-net per ridurre i tempi e le risorse per l’addestramento, e l’introduzione di una loss function che combina le metriche di percezione LPIPS e SSIM con la loss function delle rete discriminatrice. La rete è stata creata per migliorare la qualità visiva dei video di streaming decodificati in tempo reale sul device dell’utente finale, senza richiedere a quest’ultimo un aumento della banda di trasmissione. In Figura 23 è presente un’immagine dell’architettura di sr-unet, come possiamo vedere l’architettura è simile a quella della classica u-net, ma ci sono alcune differenze. La prima differenza sta nell’uso del numero di filtri per i diversi blocchi che compongono la rete, in sr-unet il numero di filtri  $F$  è costante in tutta la rete, mentre in una u-net standard no , questo perché il compito di super risoluzione non necessita di un numero di filtri crescente via via che si attraversa la rete; gli unici blocchi che non seguono questa impostazione sono il primo blocco dell’encoder e l’ultimo blocco del decoder di sr-unet, il numero di filtri per questi blocchi è  $\frac{F}{2}$ . La scelta di tenere costante il numero di filtri ha permesso di ridurre il numero di parametri da apprendere (meno memoria) e di conseguenza



Figura 17: Frame dal set 1



Figura 18: Frame dal set 2



Figura 19: Frame dal set 3



Figura 20: Frame dal set 4



Figura 21: Frame dal set 5



Figura 22: Frame originale

di velocizzare i tempi di addestramento. La seconda differenza è l'uso del metodo di upsampling chiamato pixel-shuffle [21] sul output dell'ultimo blocco del decoder; il metodo pixel-shuffle esegue velocemente l'operazione di upsampling attraverso l'uso di filtri convolutivi da apprendere nella fase di addestramento. La Figura 24 mostra l'applicazione di pixel-shuffle su un tensore 3D, mentre la Figura 25 confronta l'applicazione di pixel-shuffle su un immagine con altri metodi di upsampling come l'interpolazione bicubica.

Ora passiamo a descrivere i calcoli necessari al fine di generare delle immagini realistiche e credibili con sr-unet. L'immagine generata dal modello è definita dalla equazione 3.10:

$$x^{SR} = \text{HardTanh}(U(x^{LR}) + \text{upsample}(x^{LR})) \quad (3.10)$$

dove:

- $x^{SR}$  è l'output in super risoluzione;
- $x^{LR}$  è l'input in bassa risoluzione;

Parametri di Adobe After Effect per i vari set			
Set	Tape noise	Distorsion	Tape damage
1	52.00	1.20	Vertical Slip
2	2.00	0.20	Very bad tracking
3	4.00	0.30	Mixed Tracking
4	100.00	0.00	Bad Tracking
5	100.00	3.00	Vhs noise

Tabella 1: Tabella riepilogativa dei parametri ed effetti usati per creare i diversi set con Adobe after effect.

- $U$  è la rete sr-unet;
- $\text{upsample}$  è un filtro di upsample che può essere bilineare o bicubico;
- HardTanh è una funzione di attivazione non lineare definita come  $f(x) = \max(-1, \min(1, x))$ .

L'equazione 3.10 permette alla rete di concentrarsi sui pattern ad alta frequenza<sup>17</sup> nascosti nell'immagine in bassa risoluzione, e velocizza il processo di convergenza della rete. Per ottimizzare ulteriormente l'uso delle risorse e i tempi di addestramento, sono state modificate anche le skip connection, perchè questa tecnica consuma molta memoria ma non è possibile farne a meno dal momento che in fase di forward permette di conservare delle informazioni utili per i blocchi che compongono il decoder, e in fase di backpropagation garantisce dei gradienti sufficientemente grandi e stabili, utili a raggiungere la convergenza dell'addestramento. Le skip connection sono state riformulate come descritto dall'equazione 3.11

$$x' = \text{ReLU}(W_{3 \times 3} * x + W_{1 \times 1} * x + x) \quad (3.11)$$

dove abbiamo:

- $x'$  è la feature map di uscita;

<sup>17</sup> Per pattern ad alta frequenza nel contesto della computer vision, si intendono i dettagli ruvidi come i bordi o le linee di confine. I pattern a bassa frequenza sono i dettagli sfumati.

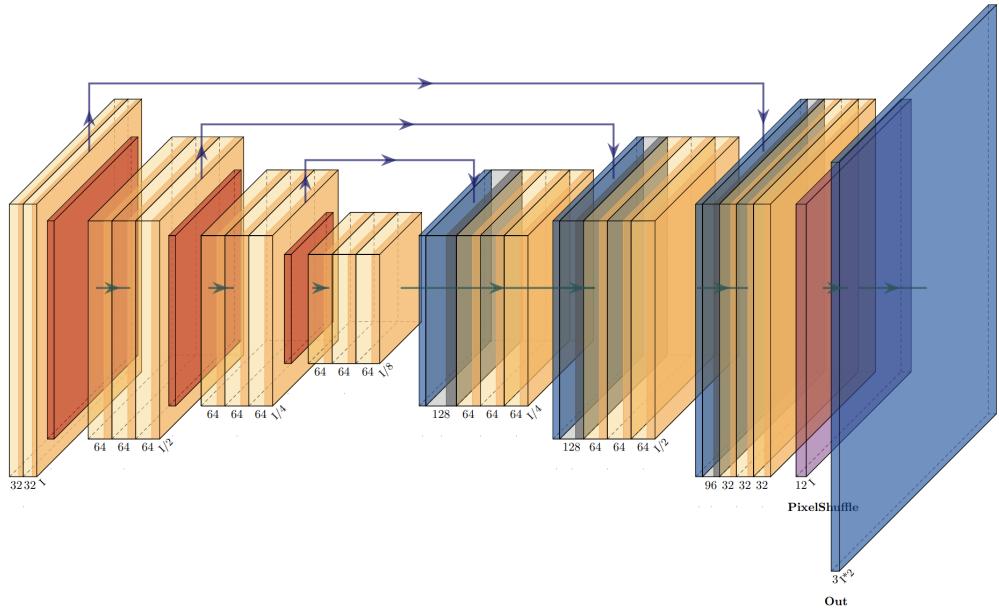


Figura 23: Schema di sr-unet.

- $x$  è l'input;
- $W_{3 \times 3}$  sono i pesi del layer convoluzionale ovvero i pesi della matrice kernel.
- $W_{1 \times 1}$  è la matrice kernel della convoluzione  $1 \times 1^{18}$

Per diminuire ulteriormente il numero di parametri della rete, possiamo eseguire una riparametrizzazione dei layer convolutivi in modo da semplificare l'argomento della ReLU. Le matrici  $W_{3 \times 3}$  e  $W_{1 \times 1}$  possono essere riscritte in unica matrice di dimensione  $3 \times 3$  così come definito nella 3.12

$$\hat{W} = W_{3 \times 3} + W_{1 \times 1}^{\text{pad}} + \text{diag}(\text{Id})_{3 \times 3} \quad (3.12)$$

dove abbiamo:

- $\hat{W}$  è la riscrittura dei parametri ;
- $\text{diag}(\text{Id})_{3 \times 3}$  è la matrice identità  $3 \times 3$ ;

<sup>18</sup> Le convoluzioni  $1 \times 1$  sono delle convoluzioni che usano filtri di dimensione  $1 \times 1$ , e vengono usate per avere un tensore di output mono canale che mantenga le stesse dimensione del tensore di input. Gli elementi del tensore in uscita sono una mappatura dei diversi valori presenti nella stessa posizione spaziale ma in canali diversi della feature map di input. Queste convoluzioni vengono usate per progettare reti in cui si vuole tenere sotto controllo il numero di parametri da apprendere.

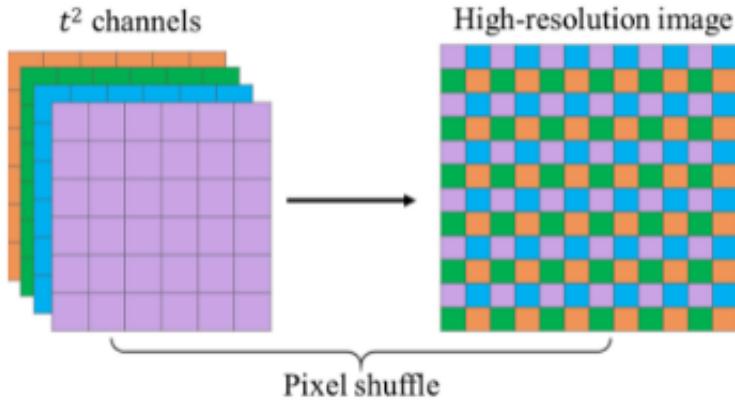


Figura 24: Applicazione dell'operazione di pixel-suffle su un tensore 3D.

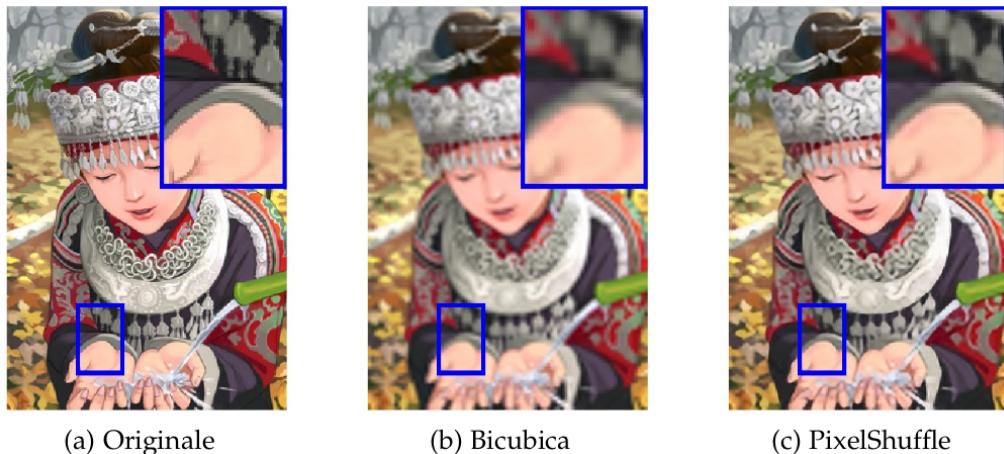


Figura 25: Esempio di confronto tra upsampling a interpolazione bicubica e pixel-shuffle.

- $W_{3 \times 3}$  è la matrice dei pesi dell' equazione 3.11.
- $W_{1 \times 1}^{\text{pad}}$  è la matrice kernel della convoluzione  $1 \times 1$  dell'equazione 3.11 a cui è stata aggiunta un padding di zeri per far si che abbia le stesse dimensioni della matrice  $W_{3 \times 3}$

#### *Loss di sr-unet*

La loss di fast sr-unet è descritta dalla equazione

$$L(y, \hat{x}) = \text{LPIPS}(y, \hat{x}) - \text{SSIM}(y, \hat{x}) - \lambda \log(D(\hat{x})) \quad (3.13)$$

dove:

- $y$  è l'immagine di ground truth;
- la  $\hat{x}$  è la corrispondente immagine ricostruita da sr-unet;
- $D(\hat{x})$  è loss della rete discriminatrice;
- $\lambda$  è un parametro di importanza posto a  $10^{-4}$ .

La loss di sr-unet è quindi una somma pesata dove sia SSIM e LPIPS hanno pari importanza dal momento che il loro peso è fissato ad 1.

### 3.2.6 Modifiche a sr-unet

Abbiamo detto nella sottosezione 3.2.5, che la rete sr-unet è una rete progetta per aumentare la qualità visiva dei video in streaming che hanno subito un processo di compressione dati. In questo lavoro di tesi, la rete lavora con dei video completamente diversi da quelli usati nel lavoro di Vaccaro [22]. I nostri video come descritto nella sottosezione 3.2.4 sono stati pesantemente modificati per sembrare dei video rovinati su cassetta, pertanto una rete piccola e veloce come quella di Vaccaro [22] potrebbe non riuscire ad ottenere dei risultati soddisfacenti. Dal momento che siamo interessati a mantenere i vantaggi della sr-unet di Vaccaro [22] e al tempo stesso di aumentare le performance della rete, il nostro contributo sarà quello di aggiungere dei moduli che aumentino le performance senza però far aumentare in modo considerevole le dimensioni della rete.

#### *Atrous Spatial Pyramidal Pooling*

I layer convoluzionali usati dai blocchi del encoder e decoder di sr-unet, sono delle convoluzioni semplici. Le convoluzioni semplici calcolano le feature map di output a partire da un'immagine di input, dove ciascun elemento proviene da un gruppo locale di pixel presenti nell'immagine di input chiamato *local receptive field*. La Figura 26 mostra questo processo di calcolo, si osservi che per determinare i valori della feature map di output, al local receptive field viene applicata una matrice di pesi chiamata *kernel* (*o filtro*) che può rilevare un certo pattern nell'input.

Il problema di questo approccio è che la feature map creata si basa solo su informazioni strettamente locali, e si otterrebbe di più se si riuscisse a considerare anche gruppi di pixel non locali in modo tale da allargare il receptive field e ottenere più informazioni. Per allargare il receptive field si usano delle convoluzioni chiamate *convoluzioni atrose* o più semplicemente *convoluzioni dilatate*, che espandono il receptive

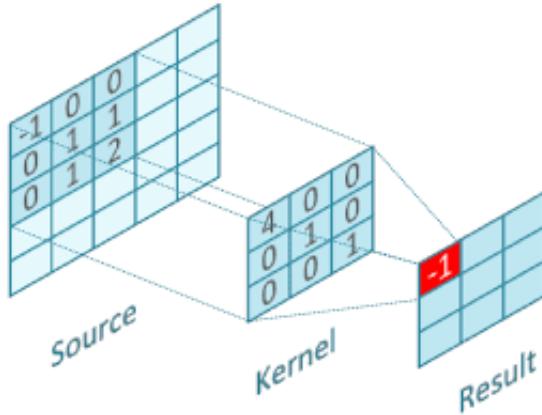


Figura 26: Il receptive field di una convoluzione.

field attraverso un fattore di dilatazione. La Figura 27 mostra come varia il gruppo di pixel scelto per formare il receptive field al variare del parametro di dilatazione. Seguendo i lavori di Jha et al. [11] e di Wang

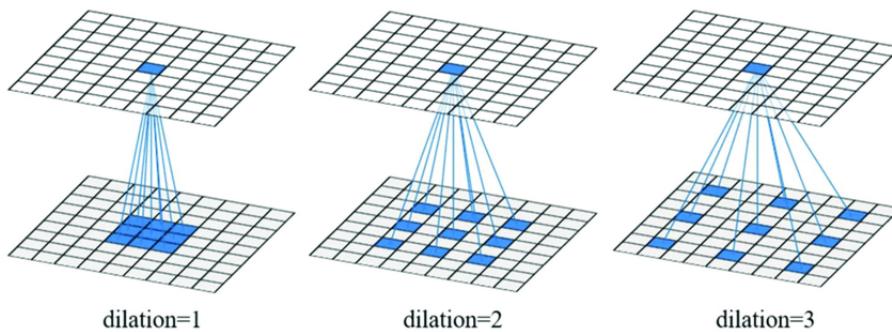


Figura 27: Il receptive-field di una convoluzione atrous

et al. [24], abbiamo introdotto un blocco *ASPP* (*Atrous Spatial Pyramidal Pooling*) tra l'encoder e il decoder con l'obiettivo di allargare il receptive field per includere più informazioni possibili. Il blocco ASPP è un insieme di convoluzioni atrose organizzate in modo piramidale, che estrae feature map in scala diverse, e in seguito le fonde per creare una feature map che tiene conto di tutte queste informazioni, la Figura 28 mostra lo schema di un blocco ASPP ed è stata presa dall'articolo di DeepLab di Chen e Papandreou [3] in cui il layer ASPP è stato introdotto per la prima volta in letteratura.

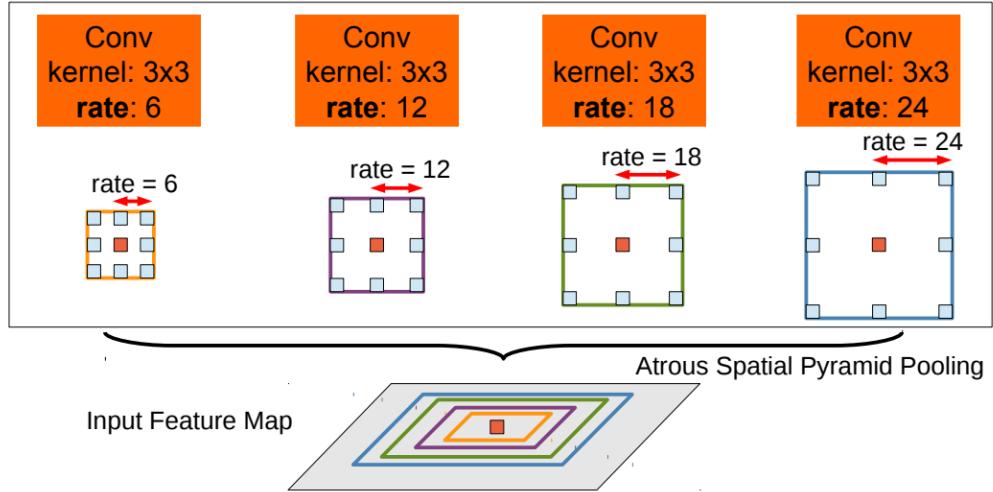


Figura 28: Lo schema del Atrous Spatial Pyramidal Pooling

### *Squeeze and Excitation Units*

La seconda modifica è quella di dotare l'encoder di blocchi di attenzione squeeze and eccitation introdotte nel 2018 da Hu et al [9]. Per spiegare il funzionamento dell'attenzione squeeze and eccitation, facciamo un richiamo sull'output di una convoluzionale. L'output di un layer di convoluzioni è una feature map composta da diversi canali, e tutte le matrici che ne fanno parte hanno le stesse dimensioni. Ciascun canale identifica una feature utile a descrive l'immagine, queste feature possono essere: bordi, linee, pattern in alta frequenza, tessitura, colore ecc.. Aumentando il numero di canali possiamo scoprire un maggior numero feature utili, ma anche il numero di quelle non utili cresce al crescere del numero di canali. In altre parole non tutti i canali hanno la stessa importanza, alcuni di questi contano poco altri invece contano tanto per svolgere un certo tipo di compito. Per esempio un canale che contiene delle informazioni sui bordi dell'immagine, potrebbe essere più utile rispetto a un canale che contiene informazioni sul cambiamento di tessitura dello sfondo, per il compito che stiamo svolgendo con la nostra rete. Nella sottosezione 2.1.4 del capitolo 2 abbiamo parlato del concetto di attenzione, e di come questo sia una misura che dice alla rete su cosa riporre l'attenzione per calcolare l'output giusto, i blocchi squeeze and eccitation sono dei meccanismi di attenzioni che dicono alla rete su quale canale della feature map di output concentrarsi per svolgere al meglio il proprio compito. Questa attenzione viene calcolata mediante due fasi. La prima fase si chiama squeeze, e ogni canale della feature map viene compresso

utilizzando il global average pooling per generare un vettore di statistiche. La seconda fase è chiamata excitation, e mira a generare dei pesi per ogni canale prendendo in input il vettore delle statistiche calcolato nella fase precedente. Nella nostra rete i blocchi di attenzione squeeze and excitation vengono usati dall'encoder prima di eseguire una operazione di max pooling.

#### *Attention block*

L'ultima modifica proposta è quello di inserire nel decoder dei blocchi di attenzione, che aiutino questa parte della rete a concentrarcisi su alcune parti delle feature map passata dalla skip connection e anche di quella calcolata dal blocco precedente, concentrandosi su queste parti si produce una nuova feature map con feature che aumentano i risultati senza avere impatti significativi sui tempi di esecuzione.

# 4

---

## ESPERIMENTI

---

In questo capitolo discutiamo dei risultati ottenuti con sr-unet e con il nostro modello, per quest'ultimo faremo anche uno studio di ablazione per valutare le modifiche apportate ad esso, i risultati verranno anche confrontati con una baseline rappresentato dal modello sr-gan. Infine mostriamo anche come abbiamo risolto il problema delle bande orizzontali.

### 4.0.1 *Hardware e strumenti usati.*

Per condurre gli esperimenti abbiamo usato una macchina equipaggiata con processore Intel® Xeon(R) CPU E5-2620 v3 @ 2.40GHz × 24 core, 126GB di memoria DDR4 @3200Ghz e un Nvidia geforce gtx titan Z con 12GB di memoria GDDR5 e 5760 cuda core. Il codice per sviluppare la rete è stato scritto in python 3.8 usando la libreria pytorch, per aumentare le prestazioni della gpu abbiamo installato nell'ambiente conda le librerie *cuDNN* (*The NVIDIA CUDA Deep Neural Network library*). Gli esperimenti che seguono in questo capitolo sono stati monitorizzati con *wandb* (*weights&biases*). Wandb è una piattaforma cloud che mette a disposizione per i propri utenti servizi e strumenti di MLOps<sup>1</sup> per sviluppare e analizzare meglio applicazioni che sfruttano il Machine Learning.

### 4.0.2 *Dettagli di addestramento e divisione dati tra training set e test set*

Per l'addestramento dei modelli abbiamo usato come algoritmo di ottimizzazione Adam, con learning rate fissato a  $10^{-4}$ , e abbiamo addestrato i modelli per 80 epoche con batch size fissato a 16 elementi. Gli elementi

---

<sup>1</sup> La MLOps è una metodologia nata negli ultimi anni per supportare lo sviluppo di modelli di Machine Learning a livello industriale. Possiamo pensare alla MLOps come all'applicazione della DevOps al Machine Learning

dei batch sono patch di dimensioni 96x96 campionate a caso dalle immagini che compongono il dataset di training. L'unica strategia di data augmentation<sup>2</sup> applicata è stata la riflessione orizzontale. La suddivisione dei dati usati è 80% per il training set e 20% per il test set. Le metriche usate per la valutazione dei modelli sono SSIM e LPIPS. Fissato una risoluzione tra HDReady e FullHD, tutti i modelli sono stati addestrati solo su un tipo di set di video per volta.

#### 4.0.3 *Risultati*

Nelle tabelle 2 e 3 sono riportati i nostri esperimenti con il modello modificato e il modello SR-unet. Per entrambi i modelli abbiamo osservato che aumentare il numero di filtri delle convoluzioni porta a risultati migliori in termini di SSIM e LPIPS. Entrambi i modelli hanno segnato i loro migliori risultati sui set in risoluzione FullHD. Il nostro modello si è comportato leggermente meglio in tutti casi rispetto al modello sr-unet di Vaccaro et al., tuttavia non abbiamo raggiunto valori elevati per SSIM e valori bassi per LPIPS perchè i video del nostri dataset erano molto rovinati, con alcuni difetti che le reti non riescono ad eliminare.

Nonostante questi risultati, siamo comunque soddisfatti dai risultati ottenuti, i nostri modelli sono riusciti a generare delle immagini in alta risoluzione, nitide e credibili se comparate alle immagini in bassa risoluzione che compongono i nostri dataset. Per visualizzare i risultati ottenuti con la nostra rete, si guardi le Figure: 29, 30 e 31, le Figure mostrano un confronto tra l'immagine in bassa risoluzione (a sinistra) e la corrispondente immagine generata (a destra), come possiamo vedere l'immagine generata è sensibilmente migliore in termini di qualità visiva, la rete è riuscita a:

- eliminare gran parte del rumore dalle immagine;
- migliorare la risoluzione delle immagine;
- migliorare il colore delle immagini che risultano più vivide;
- enfatizzare i bordi;
- enfatizzare la maggior parte dei dettagli ad alta frequenza.

---

<sup>2</sup> La data augmentation (in italiano aumento dei dati) è una pratica che consiste in un insieme di tecniche utilizzate per aumentare la quantità di dati, aggiungendo copie leggermente modificate dei dati già esistenti oppure dati sintetici creati da quelli in possesso.

Modello proposto				
Risoluzione	Set	filtri	SSIM $\uparrow$	LPIPS $\downarrow$
FullHD	1	64	0.5941	0.3664
FullHD	1	128	0.6144	0.3316
FullHD	2	64	0.6570	0.3072
FullHD	2	128	0.6591	0.2828
FullHD	3	64	0.6168	0.3285
FullHD	3	128	0.6079	0.3396
FullHD	4	64	0.5650	0.3938
FullHD	4	128	0.5937	0.3722
FullHD	5	64	0.5734	0.3722
FullHD	5	128	0.5891	0.3517
HDReady	1	64	0.5321	0.4743
HDReady	1	128	0.5801	0.4254
HDReady	2	64	0.5623	0.4464
HDReady	2	128	0.6362	0.3622
HDReady	3	64	0.5442	0.4673
HDReady	3	128	0.6209	0.3899
HDReady	4	64	0.5635	0.4815
HDReady	4	128	0.5687	0.4424
HDReady	5	64	0.5856	0.4486
HDReady	5	128	0.5872	0.3928

Tabella 2: Tabella riepilogativa del modello proposto valutato in termini di SSIM e LPIPS sui vari set nelle risoluzioni FullHD e HDReady.

SR-Unet				
Risoluzione	Set	filtri	SSIM ↑	LPIPS ↓
FullHD	1	64	0.5919	0.3456
FullHD	1	128	0.5748	0.3460
FullHD	2	64	0.6121	0.3146
FullHD	2	128	0.6030	0.3466
FullHD	3	64	0.5818	0.3466
FullHD	3	128	0.6079	0.3396
FullHD	4	64	0.5650	0.3938
FullHD	4	128	0.5937	0.3882
FullHD	5	64	0.5892	0.3574
FullHD	5	128	0.6165	0.3221
HDReady	1	64	0.5321	0.4743
HDReady	1	128	0.5801	0.4363
HDReady	2	64	0.5525	0.4628
HDReady	2	128	0.6076	0.3868
HDReady	3	64	0.5442	0.4673
HDReady	3	128	0.5500	0.4409
HDReady	4	64	0.5390	0.4722
HDReady	4	128	0.5494	0.4605
HDReady	5	64	0.5189	0.4818
HDReady	5	128	0.5850	0.4136

Tabella 3: Tabella riepilogativa del modello SR-unet valutato in termini di SSIM e LPIPS sui vari set nelle risoluzioni FullHD e HDReady.



Figura 29: Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato



Figura 30: Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato

#### 4.0.4 *Studio di ablazione*

Conduciamo ora uno studio di ablazione per capire il contributo dei moduli di cui abbiamo parlato nelle sottosezioni: 3.2.6, 3.2.6 e 3.2.6 del capitolo 3. Toglieremo un modulo alla volta e registreremo le prestazioni in termini di SSIM e LPIPS. Abbiamo ristretto lo studio solo sui video in FullHD e ai set 1 e 2 con il numero di filtri F posto a 128. Dai risultati ottenuti, si guardino le tabelle 4, 5, 6, possiamo concludere che le modifiche apportate al modello sr-unet sono significative, in quanto la loro rimozione porta ad un degrado prestazionale delle metriche SSIM e LPIPS.

#### 4.0.5 *Confronto con sr-gan*

Per confrontare i risultati ottenuti dai nostri modelli, abbiamo scelto come baseline il modello sr-gan di Ledig et al. [13]. Sr-gan è uno dei modelli generativi avversari più utilizzati e famosi per eseguire la super risoluzione e il miglioramento della qualità visiva di immagini in bassa

Modello proposto senza ASPP				
Risoluzione	Set	filtri	SSIM ↑	LPIPS ↓
FullHD	1	128	0.5717	0.3658
FullHD	2	128	0.6127	0.3110

Tabella 4: Tabella riepilogativa del modello proposto valutato in termini di SSIM e LPIPS sui set 1 e 2 per valutare l'impatto delle prestazioni dopo aver rimosso il blocco ASPP.

Modello proposto senza SE-block				
Risoluzione	Set	filtri	SSIM ↑	LPIPS ↓
FullHD	1	128	0.5506	0.3734
FullHD	2	128	0.6219	0.3175

Tabella 5: Tabella riepilogativa del modello proposto valutato in termini di SSIM e LPIPS sui set 1 e 2 per valutare l'impatto delle prestazioni dopo aver rimosso i blocchi squeeze and excitation.

Modello proposto senza attention-block				
Risoluzione	Set	filtri	SSIM ↑	LPIPS ↓
FullHD	1	128	0.5877	0.3641
FullHD	2	128	0.6007	0.3238

Tabella 6: Tabella riepilogativa del modello proposto valutato in termini di SSIM e LPIPS sui set 1 e 2 per valutare l'impatto delle prestazioni dopo aver rimosso i blocchi di attenzione.



Figura 31: Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con il modello modificato

SR-Gan				
Risoluzione	Set	filtri	SSIM ↑	LPIPS ↓
FullHD	1	128	0.5877	0.3641
FullHD	2	128	0.6360	0.3094
FullHD	3	128	0.5997	0.3335
FullHD	4	128	0.6027	0.3559
FullHD	5	128	0.6137	0.3390

Tabella 7: Tabella riepilogativa del modello proposto valutato in termini di SSIM e LPIPS su tutti i set alla risoluzione FullHD per valutare l'impatto delle prestazioni dopo aver rimosso i blocchi di attenzione.

risoluzione. Abbiamo eseguito i nostri esperimenti su sr-gan, si guardi la tabella 7, le prestazioni in termini di SSIM e LPIPS non sono molto distanti dai nostri modelli, ma il modello ha avuto delle difficoltà nel restaurare alcuni frame, in particolare in presenza di oggetti bianchi o su oggetti come statue d'oro, sr-gan non ha ben capito come ripristinate alcune regioni di pixel dell'immagine, che appaiono come sfere fosforescenti, nelle Figure 32 e 33 abbiamo riportato il fenomeno.

#### 4.0.6 Rimozione della banda orizzontale

Abbiamo risolto il problema della barra orizzontale attraverso il modello discusso nella sottosezione 3.1.3 del capitolo 3. Per ogni video in cui era presente la barra orizzontale, abbiamo ricavato i frame in cui essa compariva, e per ciascun di questi frame abbiamo poi creato manualmente delle maschere che identificassero la presenza della banda, a tal proposito si guardi le Figure 34 e 35 dove sono raffigurate un frame



Figura 32: Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con sr-gan

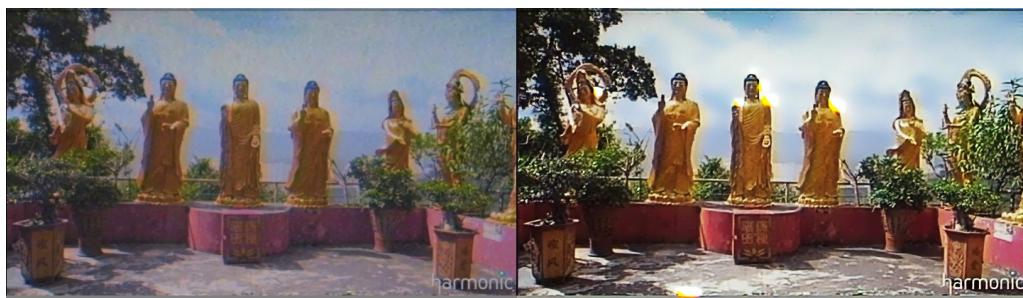


Figura 33: Confronto tra il frame in bassa risoluzione a sinistra e frame restaurato a destra con sr-gan

afflitto dal problema e la relativa maschera che rileva la presenza della banda orizzontale.

Una volta ottenuti i frame corretti dall'algoritmo FGVC, abbiamo corretto i video usando questi frame, successivamente i video sono stati restaurati con il nostro modello.



Figura 34: Frame che presenta il problema della banda orizzontale.



Figura 35: Maschera del frame a sinistra.



# 5

---

## CONCLUSIONI

---

In questa tesi, abbiamo messo al centro il restauro di video salvati in vhs. In primis è stato affrontato il problema della banda orizzontale risolvendolo con la rete FGVC di Gao et al. [7]. In secundis abbiamo affrontato il problema del miglioramento della qualità visiva dei video, proponendo un miglioramento della rete sr-unet proposta da Vaccaro et al. [22]. La nuova rete include: l'inclusione dei blocchi di Squeeze&Excitation nei blocchi dell'encoder, l'inclusione di blocchi di attenzione per i blocchi del decoder, in fine è stato aggiunto un blocco ASPP come ponte tra l'encoder e il decoder della rete. Per testare la rete, abbiamo eseguito una serie di esperimenti su diversi video, presenti in diverse risoluzioni e set che includevano diversi livelli di rumore e difetti, abbiamo osservato delle prestazioni leggermente superiori secondo le metriche di percezione visiva SSIM e LPIPS. Tuttavia nessuno dei modelli coinvolti in questo lavoro è riuscito ad ottenere dei buoni risultati in termini di SSIM e LPIPS, questo perchè i video in bassa risoluzione usati per l'addestramento soffrono di alcuni difetti, che le nostre reti non sono in grado di togliere, pertanto l'immagine generata se pur di qualità superiore (nei termini di colore, bordi, dettagli in alta frequenza) possiede ancora questi difetti che nell'immagine di ground truth non sono presenti. Nonostante ciò possiamo comunque ritenerci soddisfatti dai risultati ottenuti dal momento che il nostro modello elabora correttamente i video e li restituisce visivamente migliorati.

### 5.0.1 *Sviluppi futuri*

Per migliorare la qualità visiva dei video con i modelli visti in questo lavoro, sarebbe opportuno eliminare quanti più difetti possibili con le tecniche di video inpainting come FGVC, perchè sono responsabili dell'abbassamento delle metriche SSIM e LPIPS. Questo richiederebbe diversi sforzi per creare maschere che identificano la presenza di questi

difetti possibilmente in modo automatico, sviluppando dei metodi di segmentazione di pixel appositi, magari attraverso reti u-net.

### 5.1 RINGRAZIAMENTI

Siamo arrivati in fondo a questa tesi e anche del mio percorso universitario. Vorrei innanzitutto ringraziare il Professor Marco Bertini per avermi proposto questa tesi, in cui nonostante non avessi nessuna conoscenza di computer vision o esperienza in machine learning applicato alla computer vision, mi ha permesso di scoprire questo mondo e di appassionarmi alla disciplina. Lavorare a questo tesi è stata una sfida, ed attualmente è il lavoro che più mi inorgoglisce, tra tutti quelli che ho fatto in passato. Vorrei ringraziare e dedicare questa tesi alla mia famiglia che mi ha sempre sostenuto e creduto nel mio percorso, senza di loro molto probabilmente non sarei arrivato dove sono ora, quindi grazie Papà, Mamma, Emanuela e Marco. Oltre alla mia famiglia, vorrei ringraziare Matteo, detto il Ghera, con cui ho condiviso i migliori mal di testa della mia vita (ogni riferimento a Big Data Architectures, al diabolico Apache Mesos e a Machine Learning è puramente casuale) e risate, senza di lui forse non avrei mai completato il mio percorso di studi. Ci tengo a ringraziare Federica che mostra sempre interesse in quello che faccio e dico. Vorrei ringraziare un gruppo di amici che quest'estate mi ha cambiato la vita: Silvio, Leonardo e Luca, in particolare vorrei ringraziare Silvio e suo fratello Marco con cui sarò sempre in debito, perché mi hanno salvato la vita nella famosa notte dal ritorno dalla vacanze in Sardegna. Un altro gruppo di amici storici che vorrei ringraziare è quello formato da: Federico, Ar-j e Manuel che storicamente mi supporta e tollera ogni mia stranezza, crisi nevrotica e voglia di andare contro corrente. Infine sebbene non la senta da un pezzo a causa della pandemia da covid-19, vorrei ringraziare Nicole che mi è sempre stata simpatica e messo di buon umore.

Matteo Marulli

---

## BIBLIOGRAFIA

---

- [1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. (Cited on page 14.)
- [2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. (Cited on page 14.)
- [3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. (Cited on page 42.)
- [4] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003. (Cited on page 15.)
- [5] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep universal generative adversarial compression artifact removal. *IEEE Transactions on Multimedia*, 21(8):2131–2145, 2019. (Cited on page 11.)
- [6] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Towards real-time image enhancement gans. In *International Conference on Computer Analysis of Images and Patterns*, pages 183–195. Springer, 2019. (Cited on page 11.)
- [7] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. In *European Conference on Computer Vision*, pages 713–729. Springer, 2020. (Cited on pages 20 and 55.)
- [8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. (Cited on page 28.)

- [9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. (Cited on page 43.)
- [10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. (Cited on page 22.)
- [11] Debesh Jha, Pia H Smedsrud, Michael A Riegler, Dag Johansen, Thomas De Lange, Pål Halvorsen, and Håvard D Johansen. Resunet++: An advanced architecture for medical image segmentation. In *2019 IEEE International Symposium on Multimedia (ISM)*, pages 225–2255. IEEE, 2019. (Cited on page 42.)
- [12] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. E-lpisps: robust perceptual image similarity via random transformation ensembles. *arXiv preprint arXiv:1906.03973*, 2019. (Cited on page 34.)
- [13] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. (Cited on page 49.)
- [14] Wenbo Li, Kun Zhou, Lu Qi, Liying Lu, Nianjuan Jiang, Jiangbo Lu, and Jiaya Jia. Best-buddy gans for highly detailed image super-resolution. *arXiv preprint arXiv:2103.15295*, 2021. (Cited on page 12.)
- [15] Yinxiao Li, Pengchong Jin, Feng Yang, Ce Liu, Ming-Hsuan Yang, and Peyman Milanfar. Comisr: Compression-informed video super-resolution. *arXiv preprint arXiv:2105.01237*, 2021. (Cited on page 12.)
- [16] Ming-Yu Liu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *arXiv preprint arXiv:2008.02793*, 2020. (Cited on page 13.)
- [17] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenyi Sun, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fuseformer: Fusing fine-grained information in transformers for video inpainting.

- In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14040–14049, 2021. (Cited on page 15.)
- [18] Luca Lorenzi, Farid Melgani, and Grégoire Mercier. Inpainting strategies for reconstruction of missing data in vhr images. *IEEE Geoscience and remote sensing letters*, 8(5):914–918, 2011. (Cited on page 15.)
  - [19] Filippo Mameli, Marco Bertini, Leonardo Galteri, and Alberto Del Bimbo. A nogan approach for image and video restoration and compression artifact removal. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9326–9332. IEEE, 2021. (Cited on page 12.)
  - [20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. (Cited on page 29.)
  - [21] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. (Cited on page 37.)
  - [22] Federico Vaccaro, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Fast video visual quality and resolution improvement using sr-unet. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1221–1229, 2021. (Cited on pages 27, 36, 41, and 55.)
  - [23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. (Cited on page 14.)
  - [24] Jinke Wang, Peiqing Lv, Haiying Wang, and Changfa Shi. Sar-u-net: squeeze-and-excitation block and atrous spatial pyramid pooling based residual u-net for automatic liver ct segmentation. *arXiv preprint arXiv:2103.06419*, 2021. (Cited on page 42.)

- [25] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018. (Cited on page 13.)
- [26] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10039–10049, 2021. (Cited on page 13.)
- [27] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020. (Cited on page 13.)
- [28] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. (Cited on pages 31 and 34.)
- [29] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2019. (Cited on page 15.)
- [30] Chia-Hung Yeh, Chu-Han Lin, Min-Hui Lin, Li-Wei Kang, Chih-Hsiang Huang, and Mei-Juan Chen. Deep learning-based compressed image artifacts reduction based on multi-scale image fusion. *Information Fusion*, 67:195–207, 2021. (Cited on page 13.)
- [31] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. (Cited on page 14.)
- [32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. (Cited on pages 32 and 34.)
- [33] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Super resolution generative adversarial networks with learning to rank. *arXiv preprint arXiv:2107.09427*, 2021. (Cited on page 12.)