



Modifica del modello EGVSR

Damiano Giani

Dipartimento di Ingegneria Informatica

Relazione del progetto di Visual And Multimedia Recognition

Esame: Visual And Multimedia Recognition

Professore: Alberto Del Bimbo

16th September, 2022

Abstract

Per questo elaborato è stata implementata una modifica del modello EGVSR presentato nell'articolo **Real-Time Super-Resolution System of 4K-Video Based on Deep Learning** il cui scopo è la super risoluzione video. La modifica consiste inizialmente nel creare due distinte sequenze di frame, una rappresentante il background e l'altra il foreground dei frame originali. Successivamente queste sequenze di frame vengono processate da un modello EGVSR modificato. Questo modello cerca di sfruttare la grande differenza fra l'optical flow di pixel appartenenti al background e pixel appartenenti al foreground al fine di raggiungere una super risoluzione migliore.

1 Introduzione

Lo scopo di questo elaborato è quello di testare una modifica del modello EGVSR proposto nell'articolo **Real-Time Super-Resolution System of 4K-Video Based on Deep Learning** scritto da Yanpeng Cao ed altri. Questo articolo tratta di un metodo di Super risoluzione Video (VSR) realizzato attraverso l'utilizzo di una GAN.

La super risoluzione video, estensione della super risoluzione delle immagini, ha lo scopo di generare video di alta qualità partendo da video di bassa qualità cercando di evitare la creazione di rumori ed effetti blur che sono solitamente creati dalle tecniche basate esclusivamente su algoritmi di interpolazione.

Per questo elaborato sono stati presi in analisi video di partite di calcio. L'idea alla base della modifica del modello EGVSR è quella di trattare separatamente background e foreground di ogni frame. Questo viene fatto perché EGVSR come vedremo in seguito, utilizza la stima dell'optical flow durante il processo di super risoluzione. L'optical flow è però molto diverso fra pixel riguardanti il foreground e il background e quindi l'idea è che, trattando separatamente questi due elementi, l'optical flow dovrebbe essere stimato più accuratamente e di conseguenza la super risoluzione dovrebbe migliorare. In questa relazione descriverò in generale le GAN, poi effettuerò un approfondimento sul modello EGVSR ed infine andrò a spiegare le modifiche apportate al modello ed i risultati ottenuti da queste.

2 General Adversarial Network (GAN)

GAN è un particolare tipo di Modello Generativo. Lo scopo dei modelli generativi di immagini è quello di stimare la distribuzione di probabilità di vedere una certa immagine e generare nuove immagini andando a prendere dei campioni da questa distribuzione. Spesso si parte dall'utilizzo di una variabile latente per poi successivamente trasformala in un'immagine utilizzando una funzione deterministica che nel nostro caso è una neural network di cui vogliamo imparare i pesi. La variabile latente è solitamente una Gaussiana multivariata. Quindi partendo da una variabile latente z il *Generatore* crea un sample $G(z)$ che sarà poi dato in input a un *Discriminatore* che avrà lo scopo di dire se il sample generato è un'immagine vera o falsa. Il discriminatore è addestrato a predire se una foto è vera o falsa partendo da un dataset di immagini vere.

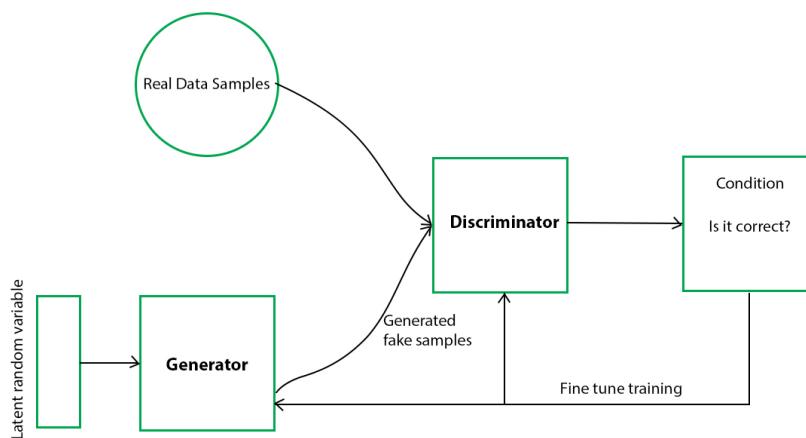


Fig. 1. GAN

Le reti che formano una GAN sono quindi due il **Generatore** e il **Discriminatore** come è possibile vedere nell'immagine 1.

La chiave per raggiungere l'obiettivo, ovvero generare una sample che sembrerà molto reale e andrà ad ingannare il discriminatore, è l'addestramento competitivo che viene effettuato addestrando una rete alla volta. Durante l'addestramento il discriminatore prende in input immagini vere e false e, minimizzando una loss (Binary cross entropy), aumenta la sua capacità di predizione.

Durante l'addestramento del discriminatore il generatore viene freezato.

Successivamente viene addestrato il generatore freezando il discriminatore. Il generatore cerca di imparare i pesi che trasformano una variabile latente random in un'immagine in grado di ingannare il discriminatore. Il Generatore è addestrato per minimizzare la probabilità che immagini false siano riconosciute come false. L'idea è che alla fine la funzione del generatore converga alla vera distribuzione dei dati reali che devono essere imitati, arrivando quindi a massimizzare la probabilità che il discriminatore consideri come veri i campioni prodotti dalla rete generativa.

2.1 GAN Per la Super Risoluzione

Le Gan possono essere utilizzate con lo scopo di creare un'immagine o più in generale un video a risoluzione maggiore rispetto alla risoluzione dell'immagine/video di input.

In questo caso come input del generatore non viene fornita una variabile gaussiana multivariata ma un'immagine a risoluzione $N*M$ e lo scopo è di generare in output la stessa immagine di input a risoluzione $sN*sM$ dove s è un fattore di scala.

L'utilizzo delle Gan per questo tipo di problemi riesce solitamente a generare immagini più realistiche di quelle create con le sole tecniche di interpolazione. Questo succede perché il discriminatore è in grado di riconoscere alcuni pattern nascosti difficili da imparare, spingendo il generatore ad adattarsi per ingannarlo,e quindi generando immagini più realistiche.

Un aspetto ulteriore che deve essere tenuto in considerazione è il fatto che i pixel appartenenti a frame temporalmente successivi sono legati da un optical flow. La stima di questo optical flow è utilizzata sia per ricostruire immagini di più alta qualità sia per preservare consistenza nel movimento.

Il mantenimento della consistenza del movimento viene effettuato quindi stimando l'optical flow e utilizzando questa informazione per effettuare un warping che allinea i fotogrammi temporalmente adiacenti.

3 Efficient and Generic VSR System

Efficient and Generic VSR System(EGVSR) è il modello che è stato modificato per questo elaborato.

Questo modello è basato su un precedente modello per la super risoluzione chiamato TecoGAN (TEmporally COherent GAN) che utilizza anche le informazioni spazio temporali per aiutare il discriminatore a imparare a capire la distribuzione di queste informazioni ovvero la relazione fra pixel di frame successivi.

L'utilizzo di queste informazioni aggiuntive porta ad avere meno instabilità nel dominio del tempo rispetto a una generica GAN utilizzata per la super risoluzione di immagini.

Ciò che caratterizza il modello EGVSR rispetto a TecoGAN è la particolare attenzione nel creare un modello che possa essere usato in real-time anche con risoluzioni molto alte come il 4k. Per fare questo gli autori hanno seguito le linee guida per creare un efficiente modello CNN riuscendo a costruire una rete particolarmente leggera.

Nell'articolo **Real-Time Super-Resolution System of 4K-Video Based on Deep Learning** in cui è stato introdotto questo modello, non è stato analizzato il *Discriminatore*, che ha mantenuto la stessa struttura di quello precedentemente implementato nel modello TecoGAN.Anche le mie modifiche al modello riguarderanno, come vedremo in seguito, solo la parte del *Generatore*.

3.1 Generatore

La parte del generatore è riportata in figura 2 ed è divisa in due moduli principali.

3.1.1 FNet

Il primo modulo è quello chiamato **FNet** ed ha lo scopo di stimare l'optical flow necessario per effettuare una ricostruzione coerente dei frame. Questo modello stima il Dense Optical Flow,ovvero per ogni pixel viene calcolato il corrispettivo optical flow. La struttura di FNet è basata su un'architettura encoder-decoder in RNN e fornisce informazioni sulla compensazione

del movimento le quali sono utilizzate per effettuare il warp del fotogramma adiacente. La parte di *Encoding* utilizza tre unità encoder ognuna formata in questo modo:

$$\text{Encoder} = \{\text{Conv2d} \rightarrow \text{LeakyReLU} \rightarrow \text{Conv2d} \rightarrow \text{LeakyReLU} \rightarrow \text{MaxPool2}\}$$

La parte di *Decoding* utilizza tre unità decoder ognuna strutturata in questo modo:

$$\text{Decoder} = \{\text{Conv2d} \rightarrow \text{LeakyReLU} \rightarrow \text{Conv2d} \rightarrow \text{LeakyReLU} \rightarrow \text{BilinearUp} \times 2\}.$$

3.1.2 SRNet

Il modulo **SRNet** ha invece lo scopo di effettuare la super risoluzione dei frame basandosi sull'immagine ricevuta a bassa risoluzione e andando anche ad utilizzare le informazioni fornite da FNet. SRNet si basa sul modello ResNet ed utilizza dei blocchi chiamati ResBlock la cui struttura è:

$$\text{ResBlock} = \{\text{Conv2d} \rightarrow \text{ReLU} \rightarrow \text{Conv2d}\}.$$

In particolare per questo modulo sono stati utilizzati dieci ResBlock e infine come metodo di upsampling è stato utilizzato un layer convoluzionale sub-pixel la cui struttura è:

$$\text{ConvSubPixel} = \{\text{PixelShuffle} \times 4 \rightarrow \text{ReLU} \rightarrow \text{Conv2d}\}.$$

Quindi andando a vedere il funzionamento del Generatore possiamo dire che : riceve in input due frame successivi a bassa risoluzione e ne stima il flusso ottico a bassa risoluzione utilizzando il modulo FNet.

Successivamente effettua un resize ad alta risoluzione del flusso stimato con una funzione di upsampling, ed effettua un'operazione di warping del frame precedente ad alta risoluzione (ritornato dal modulo SRNet) utilizzando il flusso ottico stimato. Il risultato del warping viene quindi concatenato con il frame a bassa risoluzione e questa concatenazione viene data in input al modulo SRNet che genera il nuovo frame ad alta risoluzione.

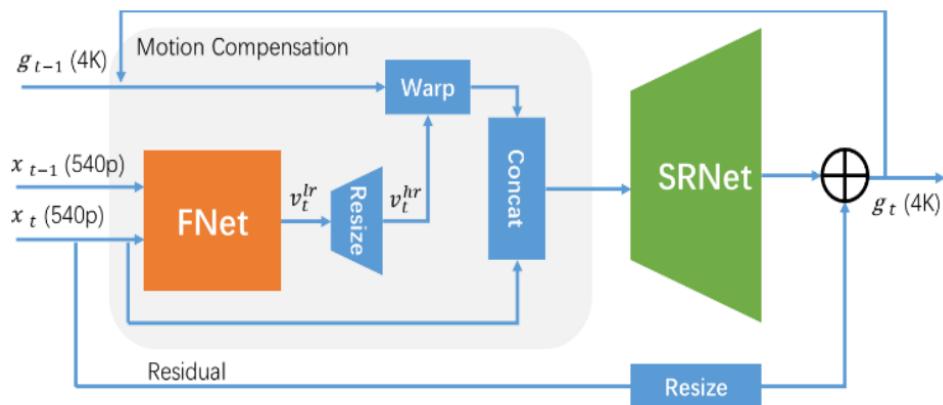


Fig. 2. EGVSR

4 Separazione background e foreground dei frame originali

Il primo lavoro che è stato necessario fare è stato separare il background dal foreground dei frame originali. Per farlo ho utilizzato YOLO (you only look once) un detector di immagini molto veloce.

Lavorando su immagini legate al calcio è stato possibile usare la versione più leggera e veloce di YOLO fra quelle disponibili in quanto era necessario che il modello riconoscesse solo i giocatori e il pallone. Una volta avviato YOLO sui frame del video, il modello è in grado di riconoscere gli oggetti di nostro interesse (che appartengono al foreground) e le loro relative bounding box. Risalendo alle coordinate delle bounding box è possibile ricavare le immagini di background settando a zero tutti i pixel all'interno delle bounding box. Per ottenere i frame relativi al foreground è sufficiente sottrarre all'immagine originale l'immagine con il background lasciando così solo i pixel presenti all'interno delle bounding box. Nell'immagine 3 possiamo vedere il frame originale e i due frame ricavati con il metodo sopra descritto. Il tempo necessario per eseguire questa operazione sui frame la cui dimensione è **3840x2160** e per salvarli è di circa **0.69 secondi** per ogni frame. Per elaborare frame di dimensione **1280x720** il tempo necessario è di circa **0.097 secondi** per ogni frame.

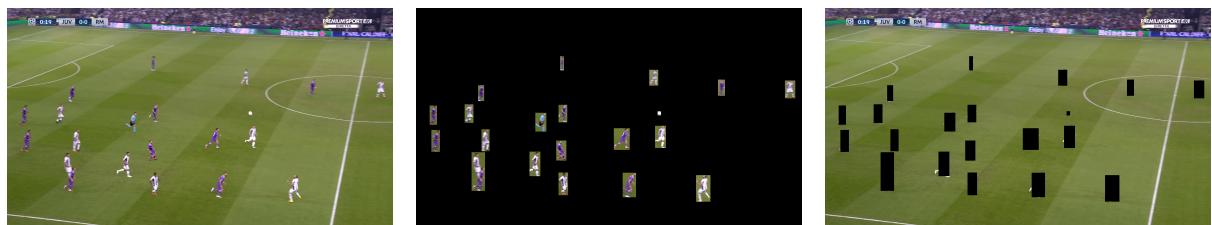


Fig. 3. Sulla sinistra è mostrato il frame originale, al centro il frame di foreground e sulla destra il frame di background generati usando YOLO

4.1 YOLO

La versione di YOLO utilizzata in questo elaborato non è la versione ufficiale ma è basata sugli stessi principi ed è implementata in PyTorch. YOLO è un detector che predice le bounding box e la loro probabilità di appartenere alla classe predetta direttamente sull'immagine in un unico passaggio attraverso una singola rete convoluzionale.

Rispetto allo stato dell'arte dei detector multi-stage è molto più veloce e leggermente meno accurato.

YOLO partiziona l'immagine di input in una griglia di dimensioni 7×7 pixel e il suo scopo è quello di predirre un tensore che contenga per ogni cella della griglia le informazioni riguardanti le bounding box e le probabilità relative a tutte le classi da lui imparate.

Se il centro dell'oggetto individuato cade all'interno di una cella della griglia, quella cella diventa responsabile del rilevamento dell'oggetto.

Ogni cella predice un numero fisso di bounding box, solitamente 2, a cui sono associate le relative probabilità delle classi ma successivamente vengono mostrate sull'immagine solo le bounding box correlate alle classi che hanno ottenuto uno score più alto.

Per quanto riguarda le bounding box YOLO utilizza alcuni modelli pre-definiti con forma e dimensione e questi modelli vengono adattati e modificati in fase di training. Il modello per predirre il tensore sopra descritto è quello rappresentato in figura 4

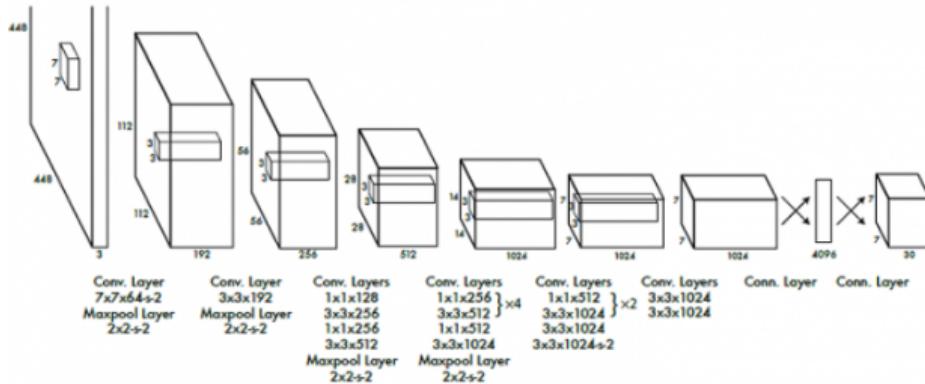


Fig. 4. YOLO

5 Modifica del generatore del modello EGVSR

Prima di implementare le mie modifiche ho effettuato un addestramento del modello EGVSR.

Per addestrare il modello ho utilizzato video in 4k di partite di calcio da cui sono stati estratti i frame di interesse andando a creare sia i frame per il ground truth in 4k che la loro versione in dimensione ridotta. Dai frame ad alta risoluzione ho successivamente generato un file *.lmdbs* per velocizzare la procedura di addestramento della rete, come richiesto dall'articolo.

Prima di effettuare l'addestramento ho inizializzato il modello partendo dai pesi ricavati dagli autori dell'articolo e successivamente ho utilizzato i parametri e la procedura da loro indicati per addestrare il modello.

Il Learning rate scelto per addestrare il modello è stato quello consigliato dall'articolo ovvero 5e-5 sia per l'addestramento del generatore che per il discriminatore.

Per effettuare l'addestramento ho eseguito 12000 iterazioni salvando i pesi del modello ogni 3000 iterazioni. Per quanto riguarda il *Discriminatore* non sono state apportate modifiche né da parte mia né da parte degli autori dell'articolo che hanno utilizzato lo stesso discriminatore precedentemente implementato nell'articolo riguardante TecoGAN.

Una volta estratti i frame ho effettuato i test su due modifiche del generatore del modello EGVSR.

5.1 Prima modifica

La prima modifica è riportata in figura 5. Il concetto di questa modifica è quello di processare separatamente i frame di background e di foreground fino alla generazione della loro corrispettiva versione ad alta risoluzione.

Per ottenere l'intera immagine ad alta risoluzione vengono poi sommati i due risultati ottenuti andandosi così a sovrapporre background e foreground.

La speranza è che, separando il processo di stima dell'optical flow e andandolo così a specializzare solo su elementi di foreground o background, le singole stime saranno più accurate e la loro unione renderà un risultato migliore.

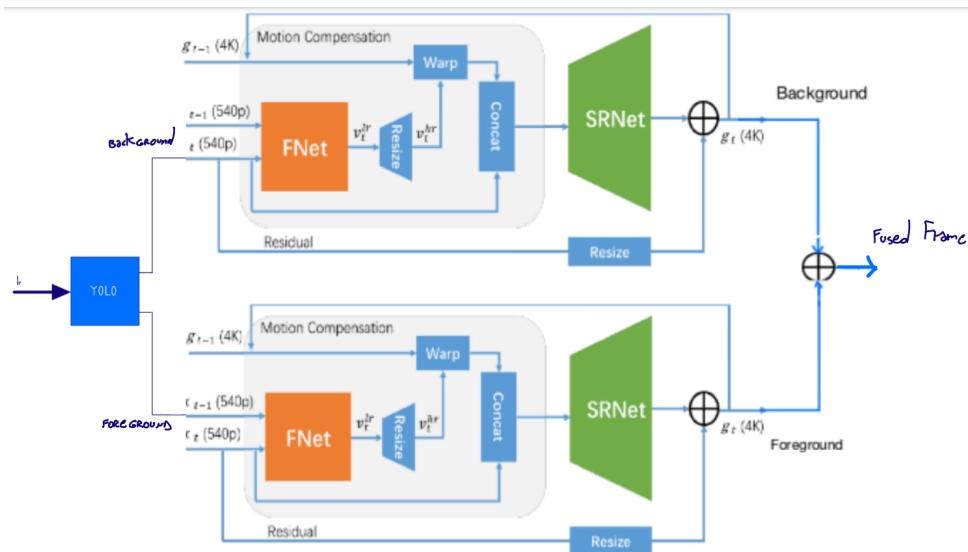


Fig. 5. Prima modifica del Generatore del modello EGVSR

Come è possibile vedere dalla sezione riguardante i risultati ,dal punto di vista quantitativo questo metodo ottiene valori leggermente inferiori a quelli ottenuti con il metodo baseline. Il frame risultante dal primo metodo è quello rappresentato nell'immagine 6 e si può confrontare con quello risultante dal metodo baseline riportato nell'immagine 10. Dal punto di vista qualitativo è possibile notare che i colori sono rimasti coerenti ma è presente una chiara separazione fra quello che prima era la parte di background e quella di foreground.

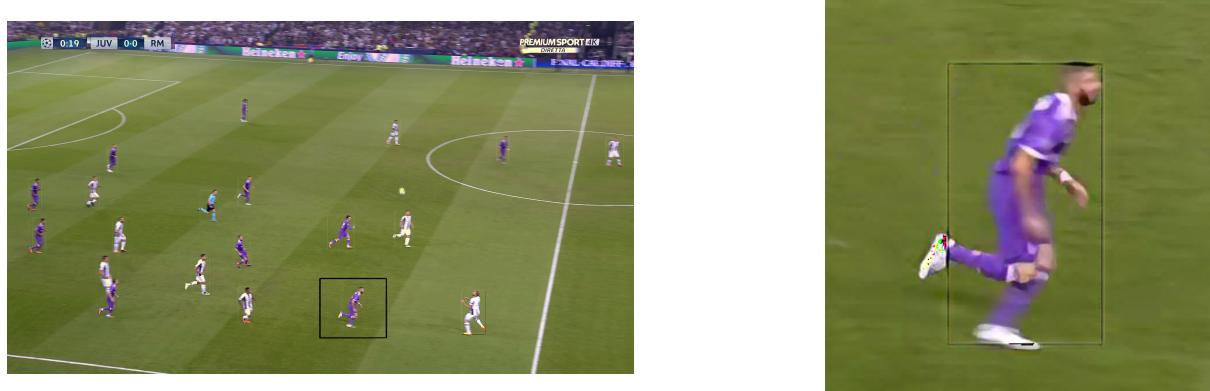


Fig. 6. Frame risultante dalla prima modifica

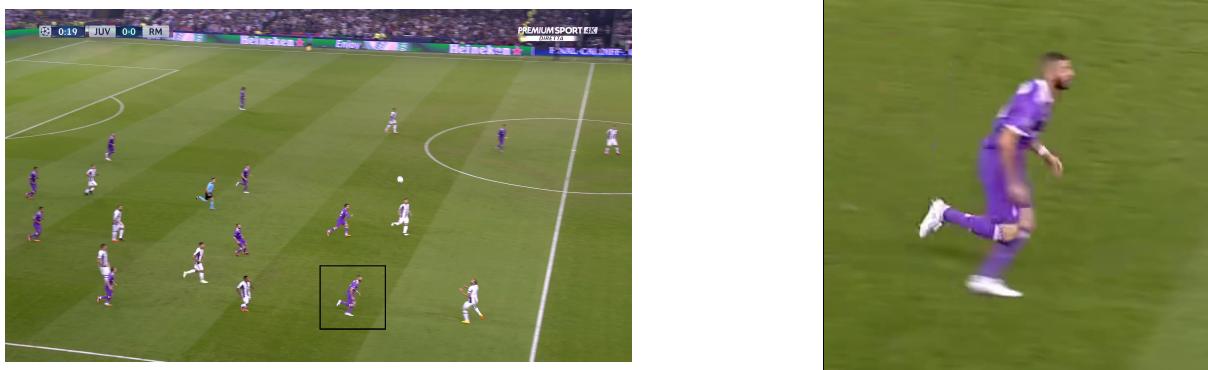


Fig. 7. Frame risultante dal metodo Baseline

5.2 Seconda modifica

La seconda modifica è mostrata in figura 8 e consiste nel processare separatamente i frame di background e di foreground. Durante questo processo vengono stimati separatamente gli *optical flow* del background e del foreground da due differenti moduli FNet. Queste due stime vengono poi utilizzate per effettuare il warp dei corrispettivi frame precedenti ad alta risoluzione generati da due differenti moduli SRNet.

I risultati dei due warp vengono successivamente sommati fra loro e concatenati con la somma dei frame a bassa risoluzione di background e foreground. Il tensore risultante da questa concatenazione viene infine dato in input a un terzo modulo SRNet che rende in output il frame ad alta risoluzione.

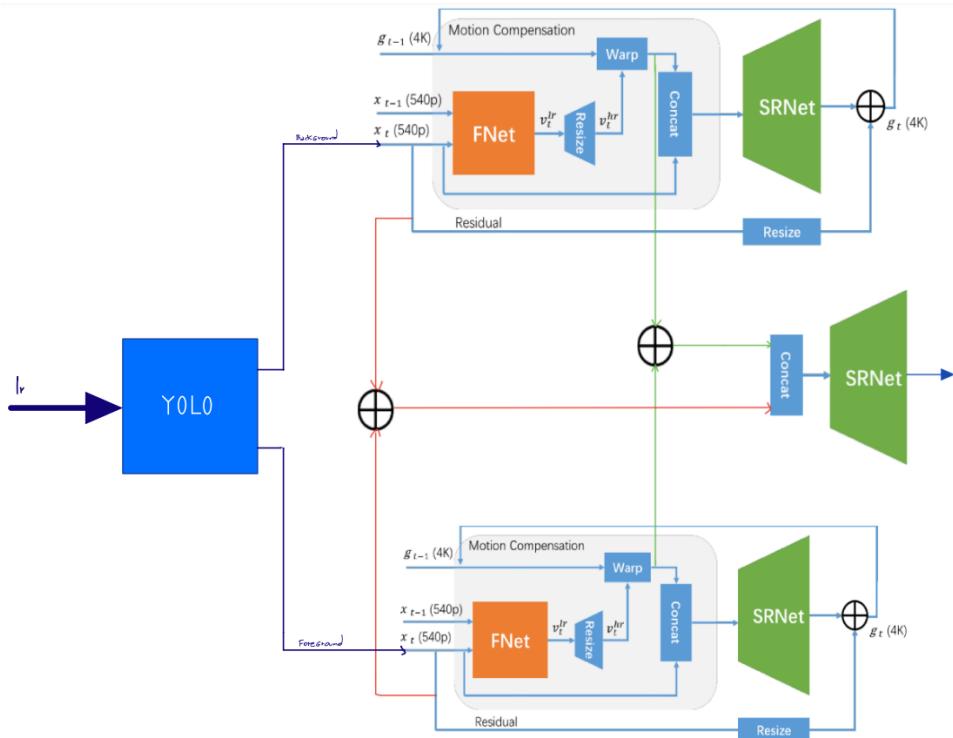


Fig. 8. Seconda modifica del Generatore del modello EGVSR

Nell'immagine 9 è riportato il frame ad alta risoluzione ottenuto utilizzando questo metodo con uno zoom sul dettaglio di un giocatore.

Nell'immagine 10 è invece riportato lo stesso frame processato con il metodo baseline.

È possibile notare da un analisi qualitativa dei due frame che la ricostruzione dei colori è la stessa e, utilizzando questo metodo, non vengono lasciati artefatti nel punto di giunzione fra il background e il foreground andando ad unire queste due parti in modo coerente.

Il risultato quantitativo ottenuto da questa modifica, come vedremo nella sezione risultati, è leggermente superiore a quello ottenuto con il metodo baseline per quanto riguarda PSNR e SSIM.

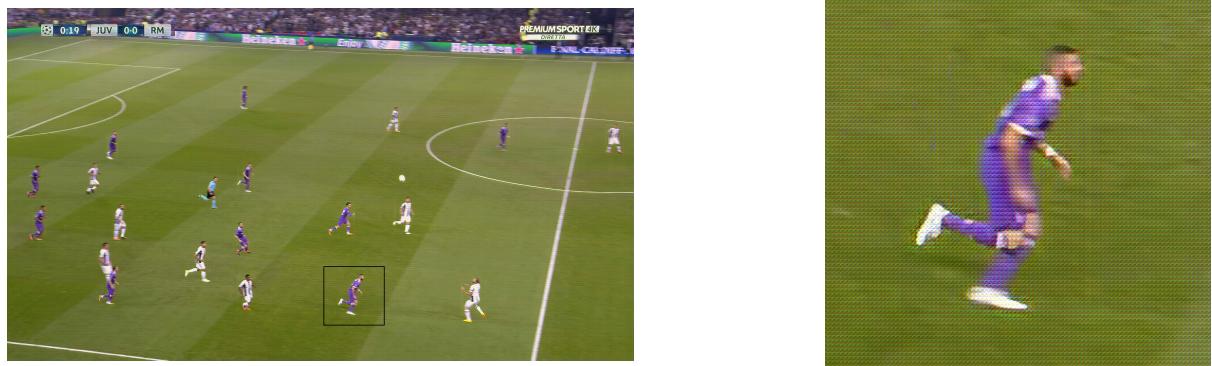


Fig. 9. Frame risultante dalla seconda modifica

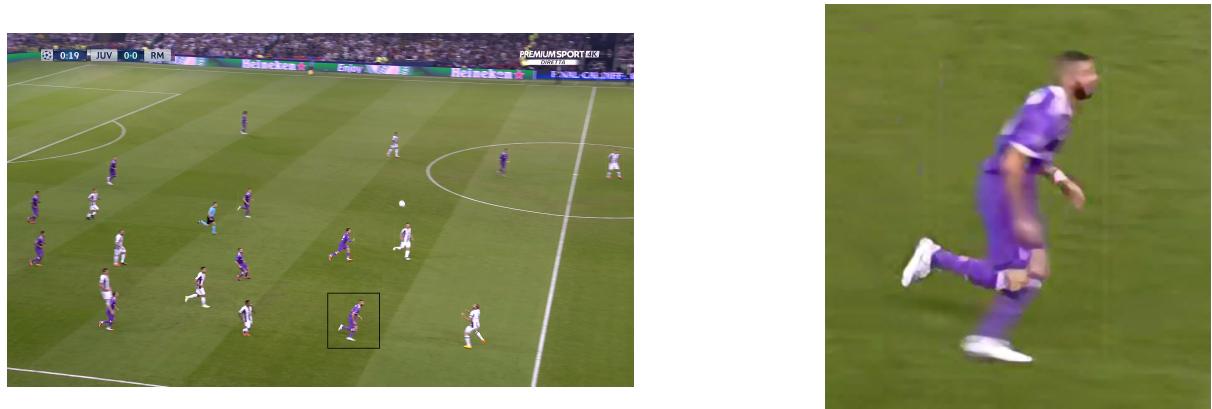


Fig. 10. Frame risultante dal metodo Baseline

6 Risultati

In questa sezione vengono descritte le metriche utilizzate per calcolare i risultati ed i relativi valori ottenuti analizzando i frame generati dalle due modifiche sopra descritte.

Questi valori sono poi messi a confronto con quelli ottenuti analizzando i frame generati con il metodo baseline.

6.1 Metriche utilizzate per i risultati

Le metriche utilizzate per effettuare i confronti sono:

1. PSNR (Peak signal-to-noise ratio)
2. SSIM (Structural similarity index measure)
3. LPIPS (Learned Perceptual Image Patch Similarity)

Queste metriche sono state calcolate sia sull'intero frame che solo sulla parte del frame relativa al foreground.

Questo è stato fatto allo scopo di vedere se i miei metodi, che si concentrano separatamente su background e foreground, hanno dato qualche beneficio nella ricostruzione dei dettagli dei calciatori e del pallone.

Per effettuare il calcolo solo sulla patch di foreground è stato nuovamente utilizzato YOLO per estrarre solo i pixel di interesse su cui poi sono state effettuate le misurazioni.

Oltre alle metriche sopra riportate ho effettuato anche un confronto sui tempi di elaborazione necessari per la generazione dei frame con le due differenti modifiche.

6.1.1 PSNR

Il **Peak signal to noise ratio** è una misura solitamente adottata per effettuare una valutazione della qualità di un'immagine compressa rispetto all'originale. Questo valore indica il rapporto fra la massima potenza di un segnale e la potenza del rumore che altera la qualità dell'immagine. Più alto è il valore del PSNR migliore è il risultato. La formula per effettuare il calcolo del PSNR è :

$$PSNR = 10 \log_{10} \frac{(L - 1)^2}{MSE}$$

dove L è il valore del pixel con il più alto livello di intensità e MSE è il **Mean squared error** ed così definito:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (O(i, j) - D(i, j))^2$$

Dove O e D sono le due immagini confrontate.

6.1.2 SSIM

Structural similarity index measure è una misura per calcolare la qualità percettiva di immagini digitali.

SSIM stima la degradazione di un'immagine rispetto all'immagine originale andando ad analizzare i cambiamenti delle informazioni strutturali della luminanza e del contrasto. Anche in questo caso più è alto il valore migliore è il risultato. La formula per calcolare SSIM è la seguente:

$$SSIM(x, y) = [l(x, y)]^\alpha * [c(x, y)]^\beta * [s(x, y)]^\gamma$$

dove le tre funzioni comparano rispettivamente la luminanza ,il contrasto e la struttura delle due immagini.

Le costanti α , β e γ denotano l'importanza relativa di ogni metrica.

6.1.3 LPIPS

Learned Perceptual Image Patch Similarity è utilizzato per misurare la percentuale di similarità fra due immagini.

Per calcolare questo valore le due immagini vengono date in input a un modello che confronta le patch delle immagini fra loro. In questo caso valori più bassi significano immagini più simili. Questo metodo di analisi delle immagini è stato introdotto nel 2018 da Richard Zhang ed altri. È stato provato che questa misura è molto attinente con la percezione umana.

6.2 Valori dei risultati

Nella tabella 1 sono riportati i risultati relativi alle metriche sopra descritte ottenuti analizzando i frame generati dalle mie due modifiche e dal metodo baseline. I valori nella prima sezione sono riferiti al frame analizzato per intero, mentre quelli nella seconda sezione si riferiscono ai valori ottenuti analizzando solo il foreground.

Evaluation Method	Baseline	Method1	Method2
Full Frame			
PSNR	29.605	28,260	29.655
SSIM	0.940	0.930	0.944
LPIPS	0.0162	0,043	0.017
Foreground			
PSNR	25.192 (-14.9%)	24.359 (-13,8%)	25.394 (-14.37%)
SSIM	0.841 (-10.5%)	0.812 (-12.6%)	0.846 (-10,3%)
LPIPS	0.040 (-146.92%)	0.042 (+2.33%)	0.045 (-164,7%)

Table 1
Valori di PSNR, SSIM e LPIPS ottenuti dagli esperimenti

Analizzando la tabella possiamo notare che i risultati migliori per quanto riguarda PSNR e SSIM sono quelli ottenuti con la seconda modifica.

Per quanto riguarda la metrica LPIPS invece il metodo baseline ottiene risultati migliori. I risultati ottenuti con la prima modifica sono leggermente peggiore poichè il metodo non è stato in grado di unire foreground e background in modo coerente.

Nella tabella 1, nella sezione riferita al foreground, è riportata anche la differenza in percentuale dei valori ottenuti sul foreground rispetto quelli ottenuti analizzando l'intero frame con la stessa metrica.

È possibile notare che sia con il metodo baseline che con il secondo metodo abbiamo un calo dei risultati andando ad analizzare il foreground.

In particolare, per quanto riguarda le metriche PSNR e SSIM il calo è maggiore in percentuale nel metodo baseline. Per quanto riguarda la metrica LPIPS il calo maggiore si ha con la mia seconda modifica.

Per quanto riguarda la metrica LPIPS possiamo notare che i risultati ottenuti con il primo metodo sul foreground sono migliorati rispetto a quelli ottenuti sull'intera immagine.

Nella tabella 2 sono riportati i tempi di elaborazione dei tre differenti metodi.

Possiamo notare che il tempo di elaborazione dei metodi si riduce con il ridursi della dimensione delle immagini.

Il rapporto fra i tempi di elaborazione dei tre metodi rimane comunque circa lo stesso cambiando la dimensione delle immagini ovvero il metodo originale richiede 1/3 del tempo rispetto al primo metodo e 1/4 del tempo rispetto al secondo metodo.

Image Dimension	Baseline	Method1	Method2
3840x2160	23.41s	66,42s	93,66s
1280x720	5.53s	15,75s	22.67s

Table 2

Tempo necessario alla generazione dei frame ad alta risoluzione
rispetto al metodo utilizzato e alla dimensione dei frame

6.3 Conclusioni

Per questo progetto sono state implementate due modifiche al modello EGVSR descritto nell'articolo **Real-Time Super-Resolution System of 4K-Video Based on Deep Learning** scritto da Yanpeng Cao ed altri.

L'idea alla base di queste modifiche è stata quella di separare in due fasi il processo di generazione dei frame ad alta risoluzione processando separatamente background e foreground dei frame a bassa risoluzione e unendo infine i risultati ottenuti da queste elaborazioni.

I frame risultanti dai due metodi e dal metodo baseline sono stati analizzati sia con un'analisi qualitativa che con misure quantitative.

I risultati ottenuti analizzando i frame generati dalla prima modifica sono stati leggermente peggiori rispetto a quelli ottenuti analizzando i frame generati con il metodo baseline, mentre con la seconda modifica ho ottenuto un leggero miglioramento per quanto riguarda PSNR e SSIM.

Per quanto riguarda i tempi di elaborazione sono per entrambe le mie due modifiche aumentati rispetto ai tempi del metodo originale in quanto le operazioni da effettuare sono aumentate. In questo link è contenuto il codice con le modifiche descritte in questa relazione

<https://github.com/DamianoGiani/EGVSR>

7 Bibliografia

- [Original EGVSR Model](#)
- [YOLOv5](#)
- [YOLO](#)
- [LPIPS](#)
- [TecoGAN](#)