Sviluppo di sistemi per compressione video semantica

Relatore: Bertini Marco

~~Pistoia~~, 4/08/2020

Schema da seguire:

2-3 lucidi di introduzione al problema

4-5 lucidi di descrizione della soluzione (H.265, CNN per segmentazione, vista del sistema, no diagrammi UML, esempi di segmentazione)

3-4 lucidi di esempi ed esperimenti

0-1 lucidi di conclusioni

0-1 lucidi di lavoro futuro

Introduzione

Di cosa si parla

Scopo di questa tesi è lo sviluppo di un sistema di codifica video basato sulla semantica, ovvero un sistema nel quale le zone dell'immagine più interessanti per l'utente sono codificate con qualità migliore delle altre… altri dettagli, citare H.265… riprendere concetti dal lucido successivo… In particolare il sistema è stato valutato per l'applicazione alla compressione di video di calcio.

Si vuole realizzare un codec H.265 esteso partendo da HEVC (encoder hevc_nvenc) utilizzando una quantizzazione per Regioni di Interesse, ovvero rendendo la qualità video meno alta dove non ci sono zone potenzialmente interessanti per l'utente che visualizza il video e lasciando la qualità ottimale con perdita di informazione trascurabile dove sono presenti quelle zone. Questo codec è utilizzabile limitatamente ai casi in cui il video da codificare sia un video di calcio HD.

Ambito

Inquadratura scientifica

I concetti successivi riportarli al lucido di introduzione. Qui descrivere ancora il problema se il testo precedente è troppo fitto.
Il nostro encoder si colloca nel campo della computer vision e con più precisione si vuole affermare come alternativa semantica tra gli esistenti codec, comprimendo una quantità maggiore di dati in ingresso. Questo tipo di codec semantico si colloca tra quelli che utilizzano segmentazione di immagini prima dell'inizio della codifica, quantizzano in modo da scartare le alte frequenze e successivamente entrano nella fase di codifica vera e propria utilizzando un codec già esistente, possibilmente il più performante.

Strumentazione

Questi dettagli tecnici lasciarli verso la fine

Cosa abbiamo usato

Una prima distinzione tra ciò di cui abbiamo fatto uso si può suddividere più dettagliatamente in:

- strumenti software
- strumenti hardware.

- Gli strumenti hardware, sono tutte le risorse accedibili che risiedono in una macchina server Linux con 2 GPU Titan X all'indirizzo pubblico solaris.unifi.micc.it; da notare la recentezza delle schede per una maggior performance nel fine tuning (allenamento) della nostra rete neurale.

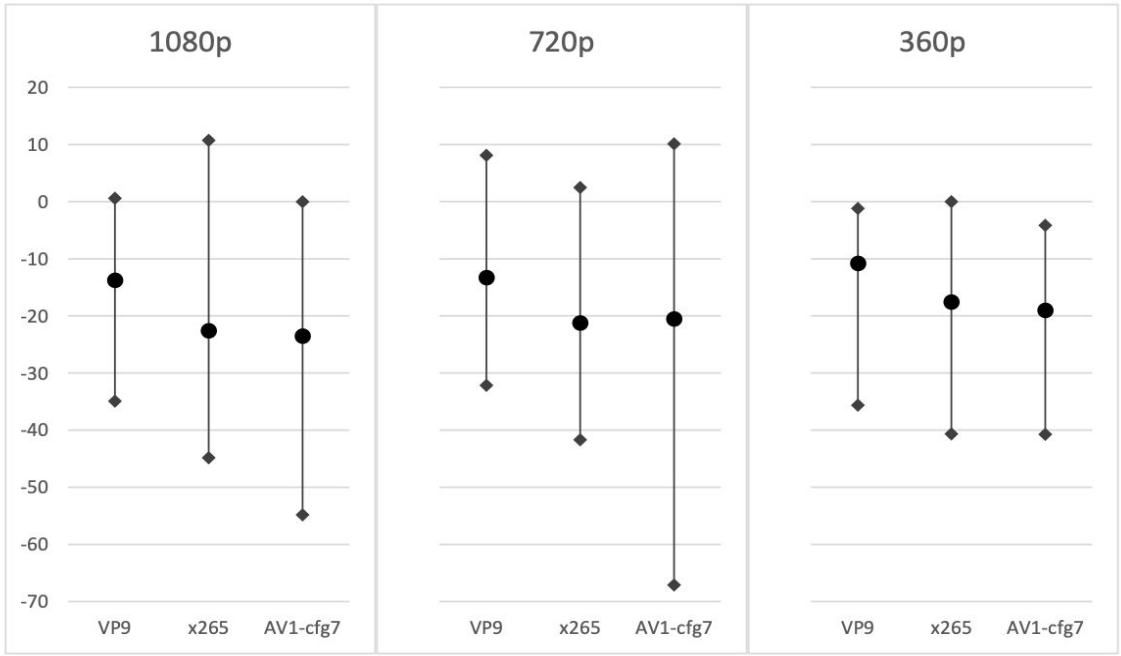
- Gli strumenti software invece sono elencabili come qui di seguito:
- Piattaforma utilizzata: Linux + Pycharm 3.6
- Miniconda: package/environment manager con ambienti:

1) tf_cpu : per tutte le piattaforme anche senza una GPU
2) tf_gpu : per solo piattaforme con GPU avanzata

- Codec usato: H.265 di nvidia chiamato hevc_nvenc
- Rete neurale Mask-RCNN
- Libreria FFMPEG per codifica video

Codec

Base iniziale



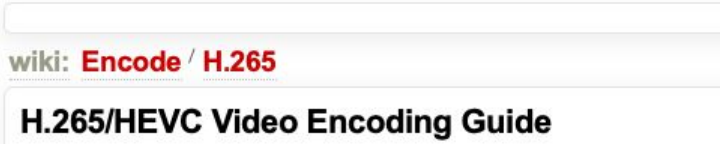
Cosa c'è sulla Y ? Che metrica ?

- Come si vede sopra, prima della scelta del codec hevc ci siamo trovati a visionare un paragone tra diversi codec, che come possiamo vedere si è sempre risolto con la vittoria o il pareggio di x265, motivo per il quale abbiamo scelto hevc_nvenc che è un implementazione dello stesso standard H.265.
- Il motivo per cui il codec è stato cambiato in hevc_nvenc da x265, è da ricercarsi nei canali Miniconda sui quali siamo andati a cercare gli encoder che seguono lo stesso standard H.265.

FFMPEG

Libreria per codifica

Dettagli non adatti per una presentazione di tesi, OK per una relazione tecnica



Nelle slide precedenti si è parlato di questo strumento software ignorandone la descrizione della sua utilità e il motivo della sua presenza. Questa è innanzitutto una libreria che consente a noi di effettuare una codifica con svariati standard, tra i quali x264 e x265. Nella slide debug vedremo perché abbiamo scelto un codec x265 alternativo chiamato **hevc_nvenc.**

Con questa libreria è possibile impostare svariati parametri per la codifica di un video, e qui sotto ne elenchiamo alcuni:

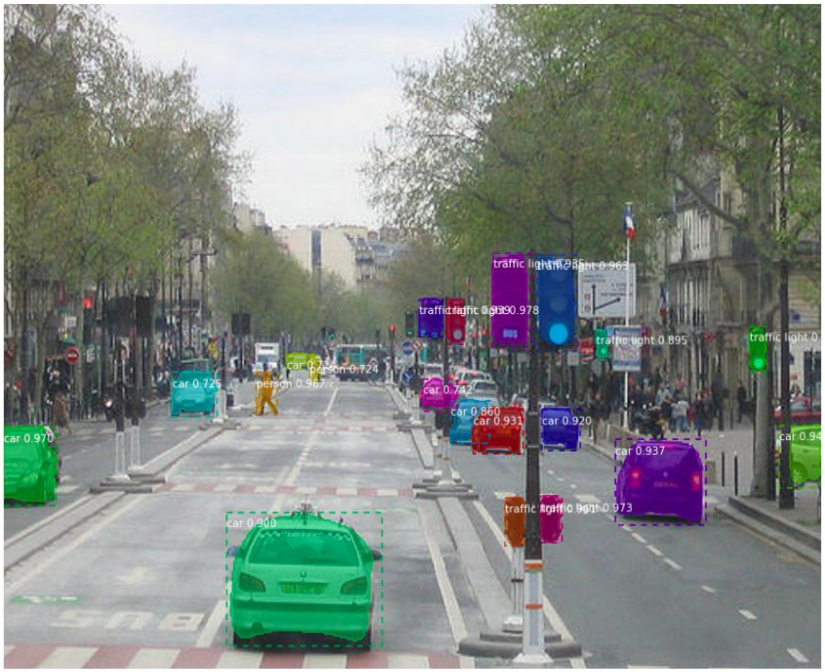
- -r: frame rate
- -vcodec: nome del codec usato
- -qmin/-qmax: costante minima e massima di qualità
- -cq: costante di qualità. Ci soffermiamo su ciò poiché il metodo che abbiamo usato è stato quello di adottare una qualità costante, invece di utilizzare i bitrate obbiettivo in 1 o 2 passi.
- Questo parametro è l'equivalente al CRF per la libreria x265.
- -rc: tipo di controllo del rate, necessario utilizzare vbr per i nostri scopi.
- -b:v: bitrate obbiettivo, dev'essere posto a 0 per causa di un bug per ottenere qualità costante.

Mask-RCNN

Rete neurale adottata

Mask R-CNN for Object Detection and Segmentation

This is an implementation of Mask R-CNN on Python 3, Keras, and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Feature Pyramid Network (FPN) and a ResNet101 backbone.



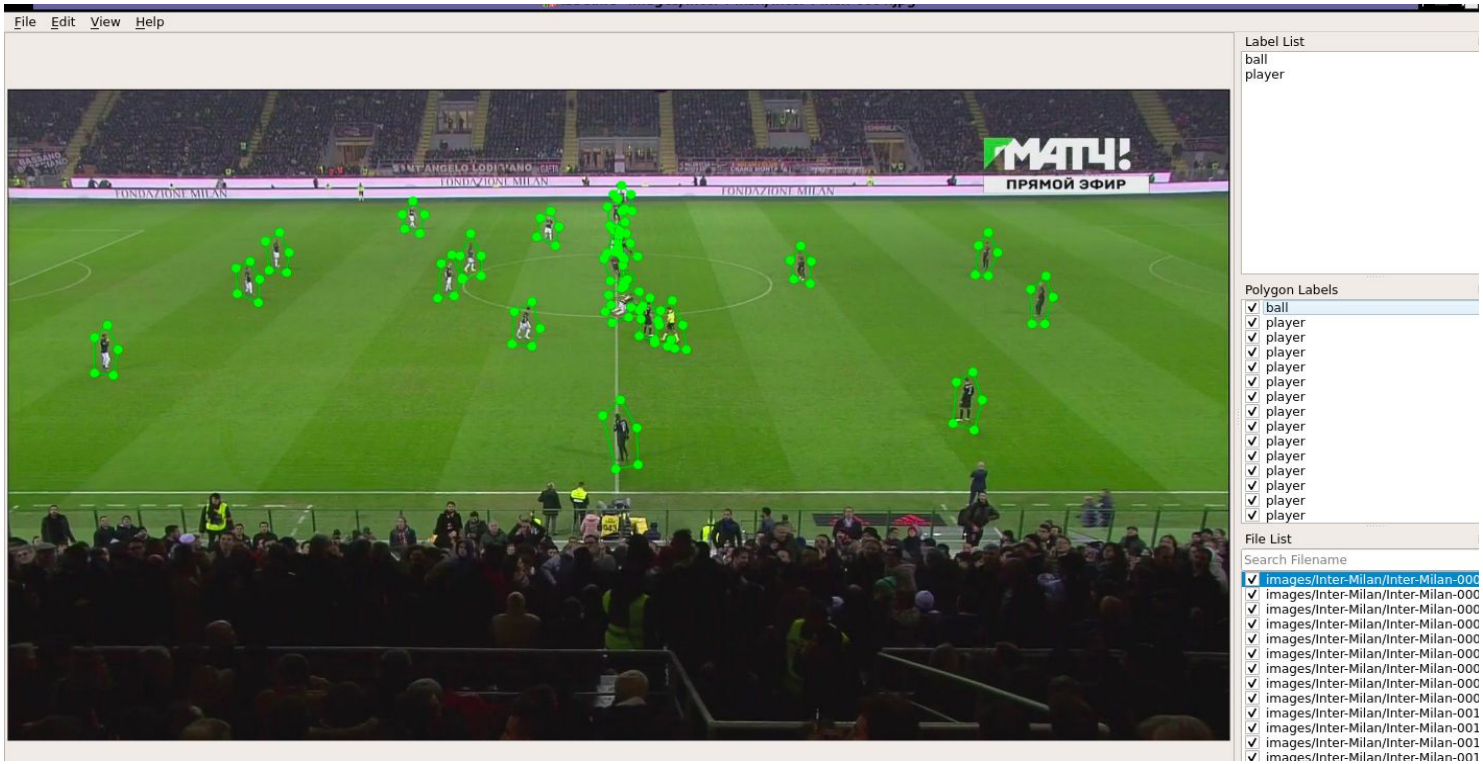
Come possiamo vedere, la nostra rete ha il perché della sua esistenza documentati in figura. Viene utilizzata per effettuare un object detection e quella che in italiano si chiama una segmentazione, molto discreta. Mask-RCNN è il seguito delle reti Faster-RCNN, che avevano come caratteristica il RoI Pooling, mentre questa rete innovativa porta perfino un'altra miglioria chiamata RoI Align, che effettua un altro passo significativo nella precisione.
Come tutte le reti, pur essendo preaddestrata per un numero limitato di classi, per i nostri scopi calcistici andrà riallenata perché riconosca i giocatori all'interno del campo e la palla. Per fare questo bisogna effettuare un fine tuning, e bisogna utilizzare un software di labeling per tracciare i contorni di giocatori e palla affinché la rete capisca quali regole deve usare per riconoscerli.

Primo lucido utile per spiegare il sistema implementato. Dire cos'è la segmentazione semantica.

Labeling

Etichettatura immagini per l'allenamento

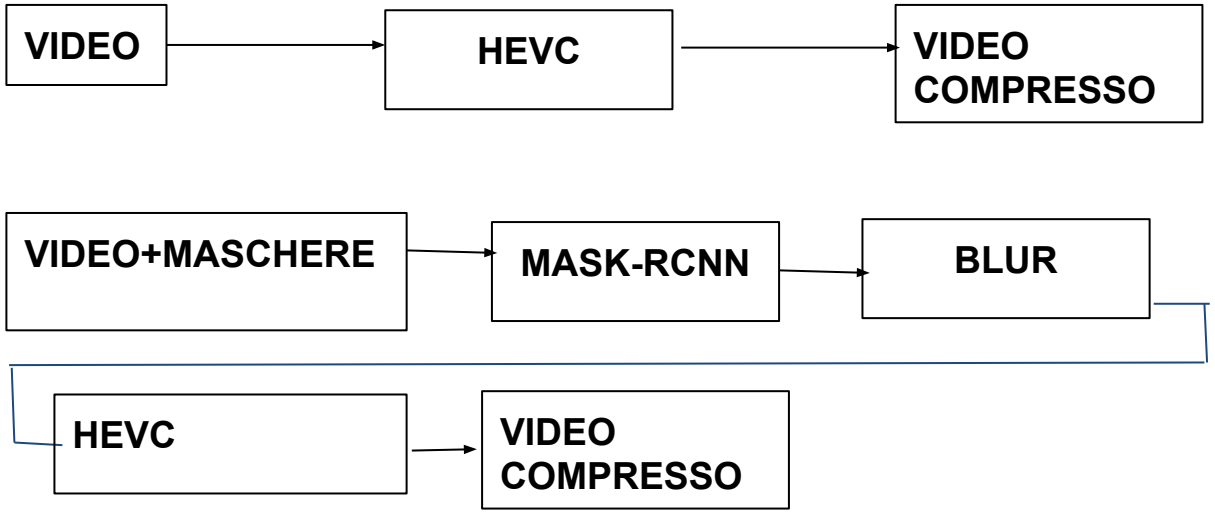
OK bene. Nel lucido precedente indicare la necessità di fine tuning



- Il software di labeling sopra rappresenta un requisito fondamentale per il nostro training, e cioè produrre un file risultato chiamato annotazione. Questo è possibile utilizzando uno di questi software, dove possiamo fare il seguente lavoro frame per frame: tracciare il contorno di giocatori e la palla, elencare se si tratta di uno o l'altra grazie a delle etichette caricate, e infine salvare il risultato dell'annotazione del frame in uno o più file di annotazione .JSON, che verrà aperto dal nostro codice all'inizio del training; questo per capire le regole per trovare giocatori e palla per poi allenare la rete a riconoscerli.
- Il software di cui sopra si chiama LABELME.

Labeling

Etichettatura immagini per l'allenamento

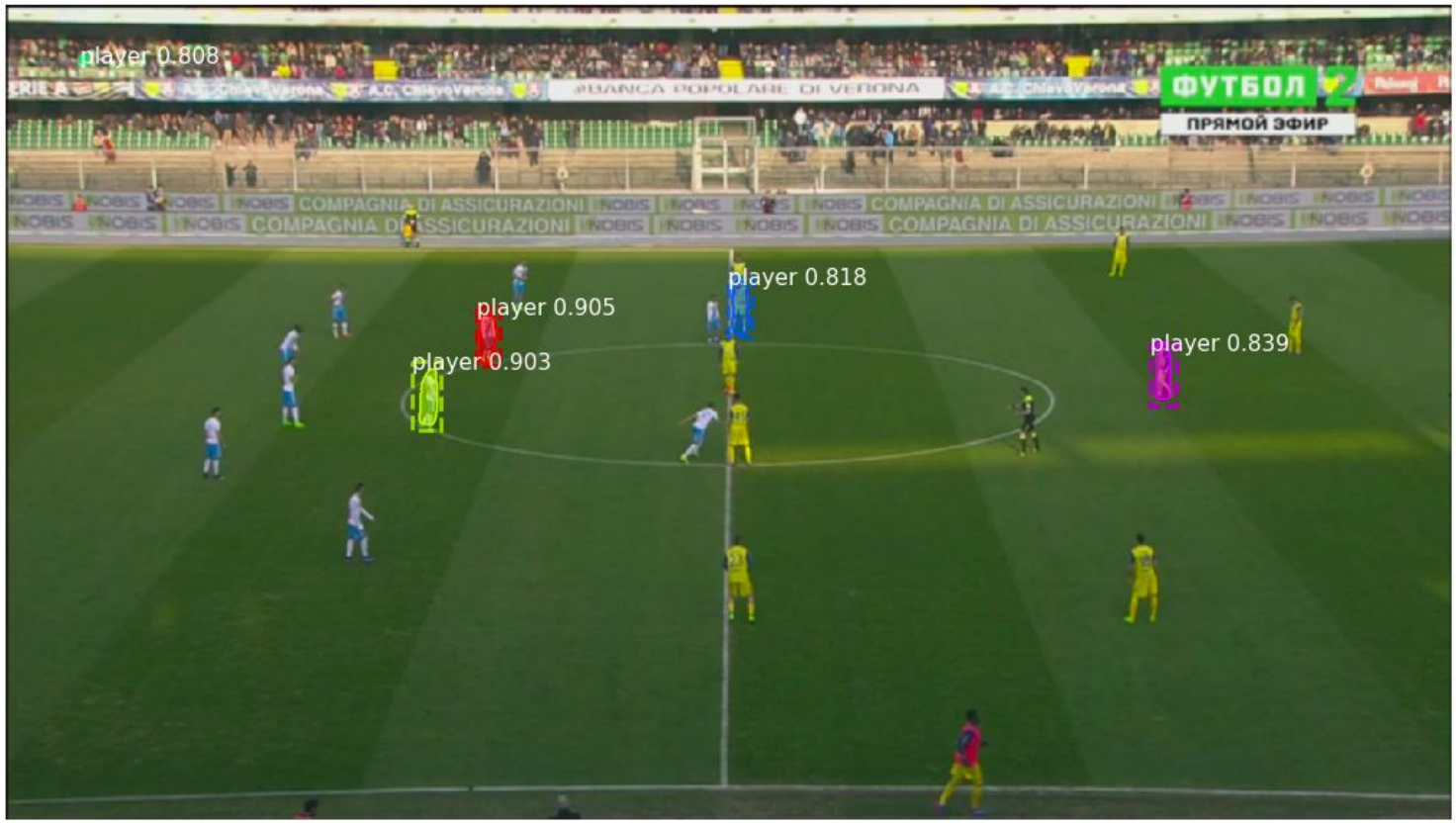


OK bene. Nel lucido precedente indicare la necessità di fine tuning

- Il software di labeling sopra rappresenta un requisito fondamentale per il nostro training, e cioè produrre un file risultato chiamato annotazione. Questo è possibile utilizzando uno di questi software, dove possiamo fare il seguente lavoro frame per frame: tracciare il contorno di giocatori e la palla, elencare se si tratta di uno o l'altra grazie a delle etichette caricate, e infine salvare il risultato dell'annotazione del frame in uno o più file di annotazione .JSON, che verrà aperto dal nostro codice all'inizio del training; questo per capire le regole per trovare giocatori e palla per poi allenare la rete a riconoscerli.
- Il software di cui sopra si chiama LABELME.

Funzionalità

Previste e fornite



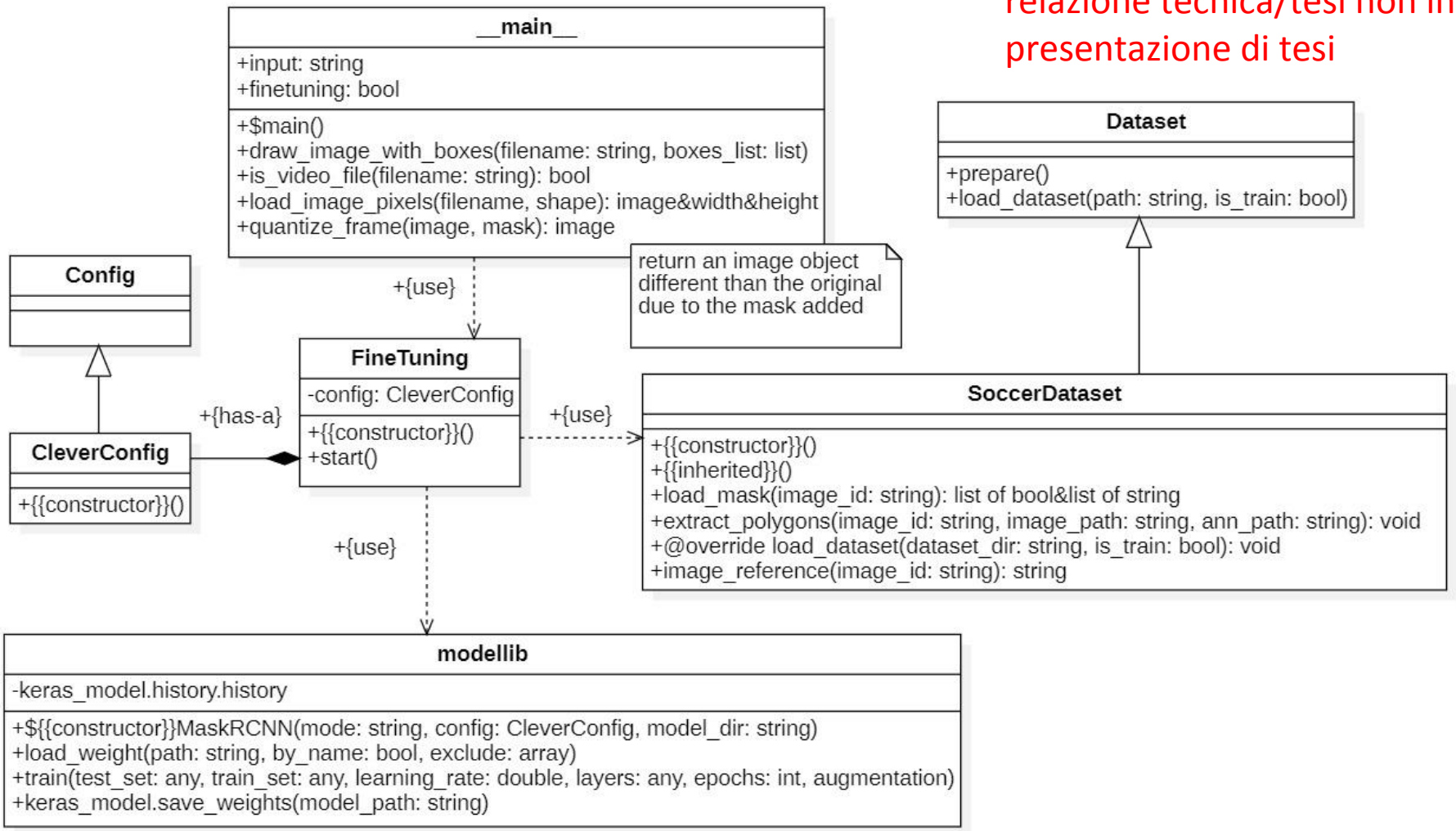
Questo esempio non è buono: sembra che non riconosca i giocatori tranne rari casi. Cosa accade abbassando la soglia di confidenza ?

- L'immagine sopra mostra un esempio di come la rete Mask-RCNN riconosce i giocatori dopo il training ovvero effettua la segmentazione(instance segmentation): in evidenza le RoI dei giocatori a cui sarà tolta un'informazione trascurabile, mentre le regioni di non interesse saranno quantizzate e quindi private delle alte frequenze.
- La funzionalità messe a disposizione dal nostro software si baseranno su un tasto per una risoluzione HD "sofisticata" dal codec oppure in un estensione browser che chiederà ad ogni tentativo di visione o scaricamento di un video se si desidera utilizzare il codec spiegato fin'ora.

Schema tecnico

Qui va messo uno schema del sistema non del software. Questo dettaglio va bene in relazione tecnica/tesi non in presentazione di tesi

Diagrama della soluzione adottata



Modulo main

Frammenti schema tecnico

- Il Modulo main.py è il modulo essenziale per eccellenza, in quanto senza di esso non si possono utilizzare le istruzioni a barra di comando per richiedere un servizio. In questo caso il servizio richiesto sarà una codifica di un video in input che risulterà in un video codificato in modo semantico con metodo progressivo in output.
-
- Le istruzioni a barra di comando fruibili sono le seguenti: main.py [-h] (-i INPUT | -ft)
  In altre parole, si può utilizzare sia l'istruzione **main.py -ft** per effettuare il training della rete che chiamiamo fine tuning poiché è una rete preaddestrata, il quale utilizza le informazioni che abbiamo salvato nel software di labeling, il file .JSON contenente le annotazioni, per giungere all'obbiettivo. Oppure, possiamo utilizzare l'istruzione **main.py -i INPU**T per caricare un file video da codificare con il nostro encoder ed ottenerne una versione codificata in modo semantico. Si noti che **INPUT** è una parola che significa che dopo il parametro **-i** dev'esserci il nome di un video esistente.
- Infine, c'è l'opzione facoltativa **main.py -h** che ha l'obbiettivo di spiegare quali e quanti sono i parametri a barra di comando utilizzabili con una breve descrizione.

Il dettaglio va bene in relazione tecnica/tesi non in presentazione di tesi. Nella presentazione si deve stare a più alto livello indicando le funzioni di fine tuning collegate alla funzione di addestramento che sarà mostrata nello schema di sistema del lucido precedente

Modulo model

Troppo densa, va bene come contenuto a livello concettuale, togliere i dettagli dalla presentazione di tesi.

Frammenti schema tecnico

Il modulo model, chiamato nel codice con alias modellib, è la classe wrapper fondamentale dove risiede il costruttore della rete e tutti i metodi ad essa associati. Viene utilizzato infatti nel modulo fine_tuning in 3 diversi punti:

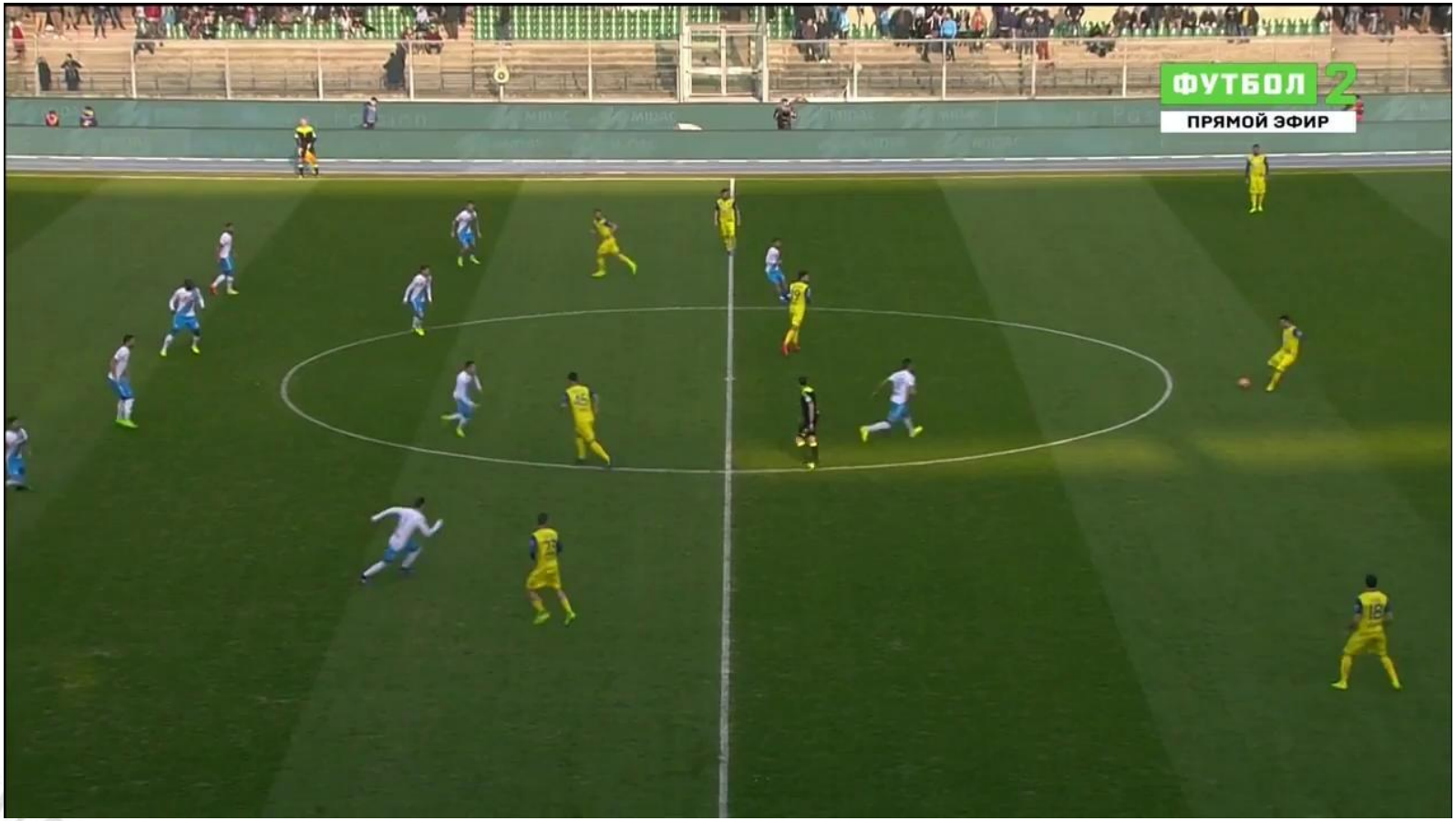
- Per creare un costruttore di MaskRCNN : grazie ad esso si possono effettuare tutte le operazioni riguardanti i pesi (weights) e il training stesso.
- Per caricare gli weights: layers della rete con il quale essa effettua l'object detection, ovvero trova le entità obbiettivo che durante la configurazione delle annotazioni abbiamo impostato che debba riconoscere. Il caricamento degli weights è limitato a tutti i layers meno gli ultimi 4, poiché la rete era stata addestrata a riconoscere circa una sessantina di oggetti mentre rimuovendo questi layers noi ci assicuriamo che trovi solo calciatori e palla.
- Per addestrare effettivamente la rete con il metodo **train()**: questo poiché si tratta di un fine tuning di una rete preaddestrata si effettuerà in 2 passi: il primo addestrando solo i top layers chiamati **heads**, in modo da far riconoscere ai layers di coda le nostre entità; il secondo passo consiste nello "scongelare" alcuni layers della rete ResNet101, in particolare dal livello 4 in su, e addestrare la rete che era già stata preaddestrata per altri scopi.
  Come parametri per un buon addestramento va specificato un **train_set** e un **validation_set**, un **learning_rate** che prendiamo direttamente dal modulo config; il numero di **epoch** durante cui ripetere l'addestramento, il nome dei layers bersaglio (**heads** nel primo caso,**4+** nel secondo caso) e un **augmentation** facoltativa, che consiste nell'applicare trasformazioni desiderate alle immagini per fare l'object detection con le immagini anche non originali dando un approfondimento nella ricerca.
- Per salvare infine i nuovi weights addestrati utilizzando sul modello **model.keras_model** il metodo **save_weights()**.

2-3 lucidi in cui si riportano gli esperimenti, descrivendo prima le metriche usate

Paragoni su risultati della codifica

Metrica SSIM

Il risultato della codifica visibile nel video seguente è stato analizzato con la metrica di qualità SSIM concludendo con i risultati seguenti:



Paragoni su risultati della codifica

Metrica LPIPS

Il risultato della codifica visibile nel video seguente è stato analizzato con la metrica di qualità LPIPS e sotto una visione del paragone tra il video codificato e non, abbiamo tratto i risultati seguenti

Codec alternativo

Se non c'è una comparazione con h.264 è da togliere. Cmq andr4bbe fatto vedere su un esempio di calcio. Finire la presentazione con un video buono di esempio

X264 con Salienza

Il codec di cui vogliamo dimostrare le prestazioni stavolta è uno di quelli che metta in evidenza invece che alcune zone di interesse come personaggi o palla, una zona centrale di interesse dove l'utente andrà a posizionare lo sguardo, detto metodo con salienza. Conferma dall'immagine sottostante si può vedere che la qualità video viene perduta leggermente ai lati del binario, mentre al centro di esso si ha ancora un'alta definizione.