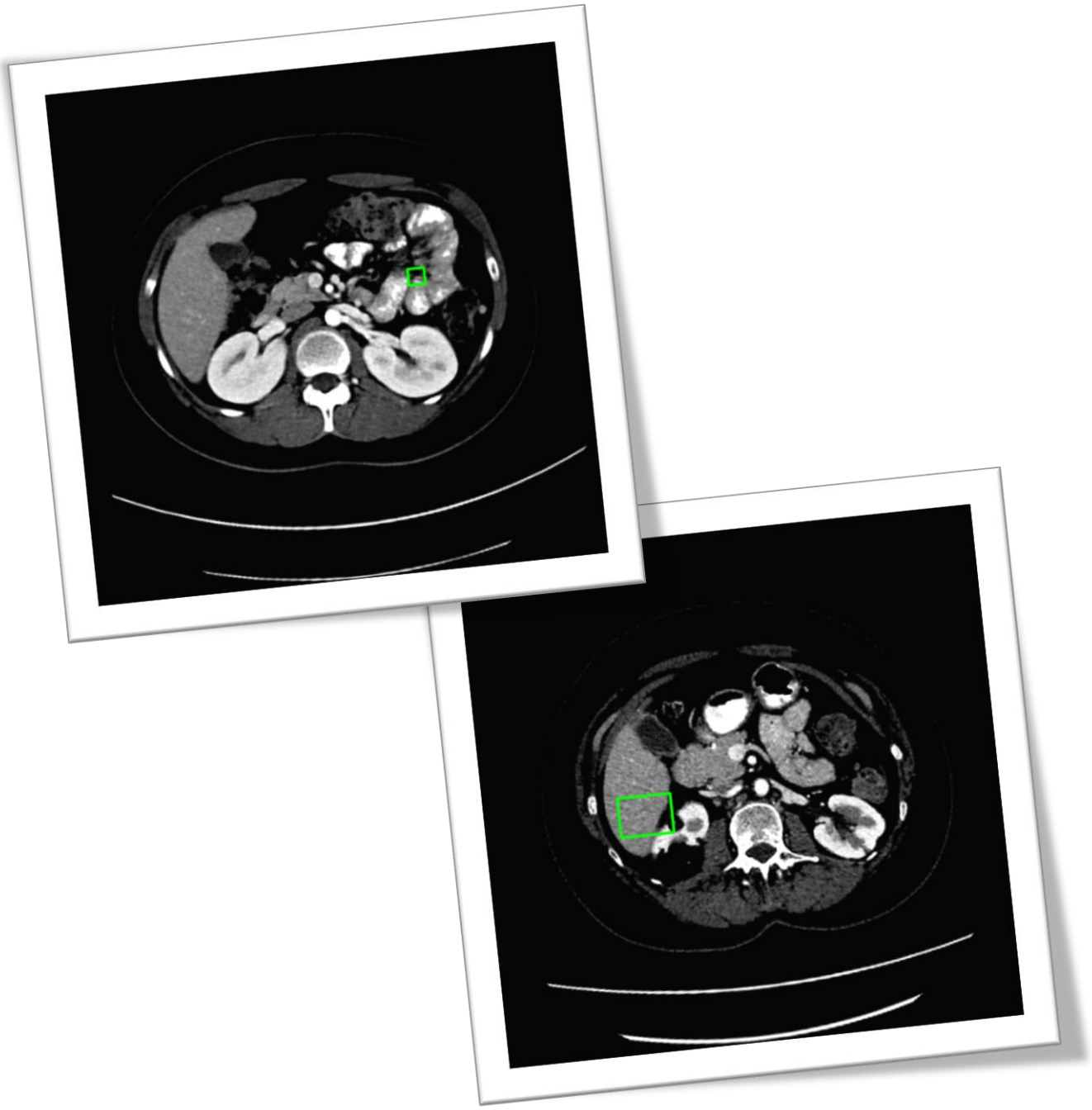
Lesion Detection

Attività progettuale **Fondamenti**

di Intelligenza Artificiale M



Federico Dal Monte federico.dalmonte3@studio.unibo.it

Indice

Introduzione

Tra le metodologie di apprendimento automatico, le reti neurali sono diffuse in diversi ambiti. Nello specifico, utilizzano degli esempi per inferire automaticamente le regole per il riconoscimento dell'oggetto preso in considerazione.

Uno degli ambiti di maggior applicazione è quello della Computer Vision che riproduce sui calcolatori elettronici il percorso cognitivo compiuto dall'essere umano nell'interpretazione della realtà che lo circonda attraverso le immagini che esso percepisce per mezzo degli occhi.

In questo documento verrà illustrato il percorso effettuato per realizzare una rete neurale capace di individuare le lesioni presenti nel corpo umano tramite l'utilizzo di bounding box.

A tal fine ci approcceremo a tecnologie riguardanti l'ambito della Computer Vision e del Machine Learning, in particolare approfondiremo l'utilizzo di TensorFlow in Python.

1 Background

1.1 Computer Vision

La Computer Vision è una disciplina inclusa nel campo dell'Intelligenza Artificiale e si occupa di studiare come possono essere realizzati i processi necessari a creare un modello approssimato del mondo reale partendo da immagini bidimensionali. In particolare, si occupa di acquisire, registrare ed elaborare immagini provenienti da un supporto elettronico (ad esempio una webcam) allo scopo di riconoscere determinate caratteristiche dell'immagine per varie finalità di controllo, classificazione e selezione. Come disciplina scientifica, la Computer Vision è strettamente legata alla teoria riguardante i sistemi artificiali che estraggono le informazioni dalle immagini. Infatti, i dati ricavati dall'immagine possono essere rappresentati in molte forme: come sequenze di video provenienti da diverse telecamere oppure come dati multimediali provenienti da scanner. Invece, come disciplina tecnologica si occupa della costruzione di sistemi di visione basati sul calcolatore, ad esempio machine vision, robot vision e visual-based multimedia systems.

Figura 1: Esempio di utilizzo della Computer Vision

L'elaborazione delle immagini, che rappresenta la parte principale della computer grafica, consiste in varie operazioni che, a seconda del livello di astrazione a cui operano, possono essere definite di basso livello (dal processo di acquisizione delle immagini al pre-processing), medio livello (dalla segmentazione delle immagini alla loro rappresentazione e descrizione) ed alto livello (in cui si ricerca una interpretazione della scena osservata).

In conclusione, tramite la Computer Vision è possibile modellare ambienti dinamici non strutturati, potendo così effettuare pianificazioni di traiettorie, ricostruzioni tridimensionali della scena ed altro ancora, tenendo presente la criticità di alcuni fattori come la qualità e la velocità di acquisizione delle immagini.

Il sistema di Computer Vision segue alcuni passi ben precisi:

1. l'acquisizione delle immagini da parte della telecamera;
2. il trasferimento dei dati;
3. l'image processing ovvero l'analisi dell'immagine acquisita da parte del software;
4. l'elaborazione dei dati riguardanti il risultato ottenuto.

1.2 CNN (Convolutional Neural Network)

Le CNN, utilizzate nell'ambito Computer Vision, possono essere viste come algoritmi di Deep Learning. Principalmente vengono utilizzate per classificare le immagini, raggrupparle per somiglianza ed eseguire il riconoscimento di oggetti all'interno delle scene tramite il perfezionamento dei pesi della rete.

Ogni peso è solitamente rappresentato da un valore numerico in float32, che può essere ridotto anche a float16 o float8 al fine di velocizzare la rete. Questi pesi all'inizio sono randomici e durante il training vengono modificati al fine di ottenere risultati migliori e più precisi.

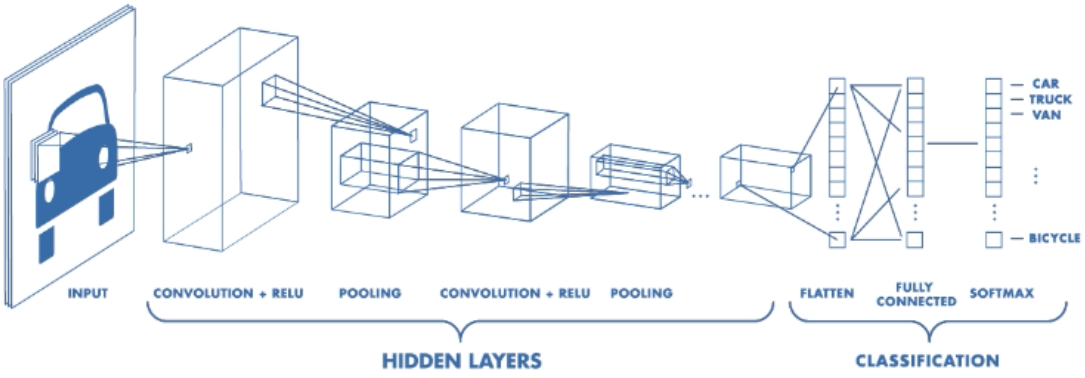


Figura 2: Convolutional Neural Network

1.3 Object Detection

Nell'ambito dell'Intelligenza Artificiale combinata alla Computer Vision esistono diversi modi di approcciarsi all'elaborazione delle immagini. Un classico esempio è il problema del Object Detection che consente di ottenere i bounding box (cioè l'area dell'immagine che contiene l'oggetto riconosciuto) per ogni oggetto rilevato nell'immagine.

Un altro approccio, non integrato in questo progetto, è quello dell'Image Classification, il quale consiste in un processo che porta un modello, correttamente allenato, all'acquisizione di un input ed alla restituzione della classe dell'oggetto che ritiene più probabile o della percentuale che l'input sia di una classe particolare.

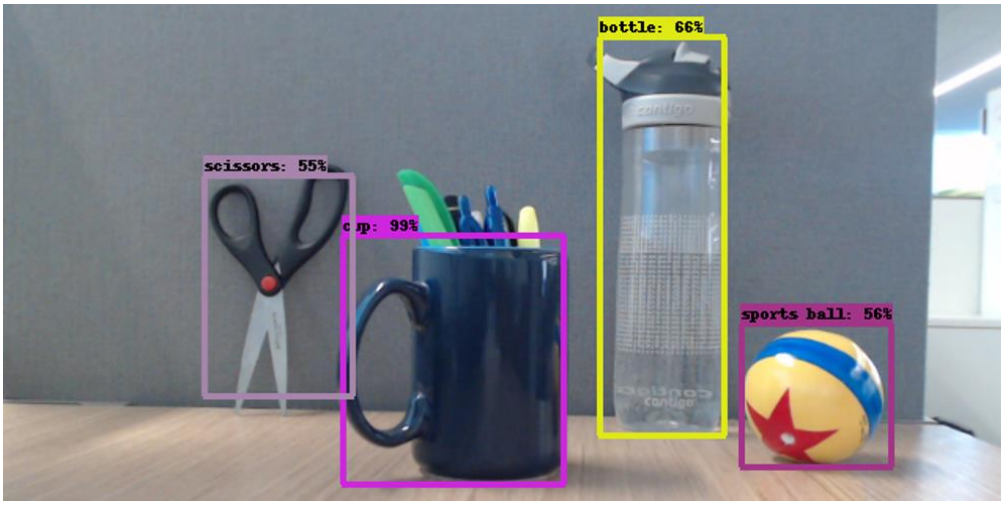


Figura 3: Esempio di Object Detection

2 Strumenti Utilizzati

Per lo sviluppo di questo progetto sono risultati necessari strumenti come TensorFlow, Keras, OpenCV, per l'allenamento delle reti sul dataset.

2.1 TensorFlow

TensorFlow è una libreria software open source per il calcolo numerico che utilizza un approccio a grafo del flusso di dati. È stato progettato per rendere disponibile la computazione parallela su più CPU o GPU, sia su singola macchina che nel distribuito. TensorFlow venne sviluppato originariamente da ricercatori e ingegneri del Google Brain Team per scopi di ricerca nel campo del machine learning e delle deep neural network, ma venne progettato con un approccio più generale in modo da poter essere utilizzato in molte altre applicazioni diverse di computazione.

Questa libreria presenta numerose API, tra cui quella di più basso livello, TensorFlow Core, che permette un controllo completo sulla programmazione. Queste API sono quelle tipicamente utilizzate nel campo del machine learning, poiché rendono possibile controllare nel dettaglio tutti gli elementi del modello che si sta implementando.

TensorFlow è compatibile con i principali sistemi operativi a 64 bit (Windows, Linux e macOS) e Android. Inoltre, fornisce interfacce per diversi linguaggi, tra cui Python, C/C++, Java e Go. Nel caso specifico di questo progetto è stato deciso di utilizzare l'interfaccia Python.

In generale TensorFlow consente agli sviluppatori di svolgere le seguenti operazioni:

- Pre-processamento dei dati
- Realizzazione di un modello
- Addestramento e valutazione del modello individuato

2.2 Keras

Per quanto TensorFlow risulti essere un ottimo framework per applicazioni di deep learning, la creazione dei modelli rimane un processo complesso. Il framework open source Keras risponde all'esigenza di definire modelli di deep learning in maniera intuitiva, modulare e con un numero contenuto di righe di codice. La principale peculiarità di questa libreria è la capacità di funzionare come interfaccia di librerie di più basso livello come TensorFlow o Theano. Keras fornisce tutte le più comuni implementazioni dei modelli di deep learning con una perdita di

prestazioni inferiore rispetto a TensorFlow. Dal 2017, il team di TensorFlow ha deciso di aggiungere dei moduli per il supporto di Keras nella proria libreria.

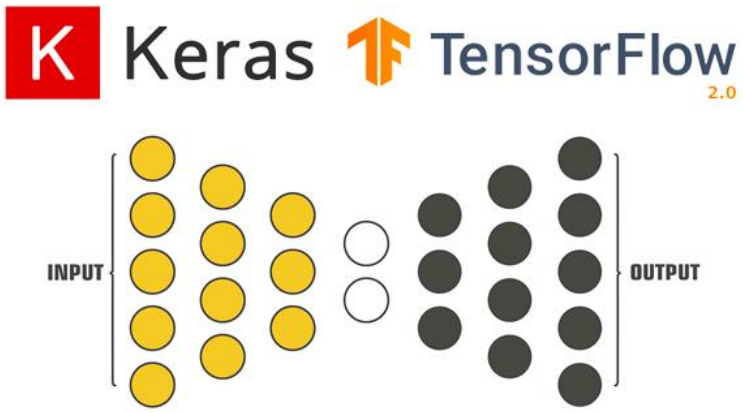


Figura 4: Keras & TensorFlow

2.3 OpenCV

Figura 5: OpenCV

OpenCV (Open Source Computer Vision) è una libreria open-source, scritta in C++, che contiene un insieme di funzioni utilizzate per lo streaming video real-time, ovvero la computer vision. Il framework OpenCV nasce da un'iniziativa di Intel mentre lavorava su miglioramenti delle loro CPU per applicazioni intensive, ad esempio ray-tracing in tempo reale e proiezione 3D. Successivamente diversi addetti della Intel notarono che in molte università erano nate delle infrastrutture per la computer vision, così decisero di iniziare a progettare un framework per i nuovi processori Intel. Nel 1999 si diede il via al progetto con la collaborazione di un team russo, ma la prima release ufficiale di OpenCV risale al 2006. Nel 2011 è stata rilasciata la versione 2.2.0 che permette il supporto da parte dei dispositivi Android e attualmente è disponibile la release 4.5.1.

3 Modelli

Per la fase di training abbiamo individuato due reti neurali con lo scopo di confrontarne le performance e valutarne le differenze. I modelli presi in considerazione da qui in avanti sono:

- VGG16
- ResNet50

Questi modelli sono stati precedentemente allenati sul dataset ImageNet, contenente più di 14 milioni di immagini suddivise in 1000 classi e sono perciò in grado di distinguere un oggetto presente in un'immagine tra 1000 categorie differenti.
In seguito, i modelli sono stati interamente ri-allenati sul dataset di immagini rappresentanti sezioni di tomografie computerizzate, in modo da renderli specifici per il problema di Lesion Detection.

Per quanto riguarda il training sono state implementate delle callbacks, in particolare 'Early Stopping' che interrompe l'allenamento della rete dopo che per 15 epoche non c'è stato un miglioramento, 'Model Checkpoint' che permette di salvare il modello migliore e 'Dynamic Learning Rate' che, a partire da un valore iniziale predefinito, decresce man mano che l'allenamento si protrae in modo tale da raggiungere una precisione migliore.

Inoltre, sono stati impostati altri parametri significanti per la fase di training, tra cui la batch e quattro layer di tipo dense con diverse dimensioni:

```
Batch = 32
Layer = 128, 64, 32, 4
```

Entrambi i modelli sono stati allenati in due modalità diverse ovvero andando a modificare il parametro 'trainable' della rete. Impostando questo parametro uguale a true si va a riallenare la rete originale modificando i pesi preimpostati. Viceversa, impostandolo uguale a false si mantengono i pesi originali allenati sul dataset ImageNet.

3.1 VGG16

VGG16 è un modello di rete neurale che riesce a raggiungere il 92,7% di accuratezza sul dataset ImageNet. Questo modello viene utilizzato in molti problemi di classificazione delle immagini di deep learning; tuttavia, le architetture di rete più piccole sono spesso desiderabili.

3.2 ResNet50

Si tratta di un'architettura che, mediante l'utilizzo di un nuovo ed innovativo tipo di blocchi (detti residual block) ed al concetto del residual learning, ha permesso di raggiungere profondità impensabili con il modello feed forward classico a causa del problema della degradazione del gradiente. Esistono implementazioni con profondità diverse, di cui la più profonda conta ben 152 livelli. Esiste anche un prototipo con 1202 livelli, che però ha raggiunto risultati peggiori a causa dell'overfitting.

4 Dataset

Il dataset preso in considerazione per questo progetto è DeepLesion (https://nihcc.app.box.com/v/DeepLesion) il quale è fornito dal National Institutes of Health Clinical Center e contiene 32120 immagini di tomografie computerizzate. Dopo averlo scaricato sono state prese in considerazione solamente 15526 immagini le quali sono state suddivise manualmente in tre set: training (12824), validation (1534) e test (1168).

Per effettuare questa suddivisione si sono presi in considerazioni tre file Excel (Train_info.csv, Val_info.csv, Test_info.csv) contenenti tutte le informazioni di ogni singola immagine. Tra queste troviamo anche le coordinate dei bounding box che sono state utilizzate per rappresentarli sull'immagine tramite la funzione OpenCV.

5 Training

I due modelli sono stati valutati considerando la loss ottenuta su un dataset di test completamente nuovo per la rete. Di seguito sono riportati i grafici che mostrano l'andamento della fase di training relativi alla funzione di loss per ogni singola rete.

Nei seguenti grafici possiamo notare l'andamento sia della training loss che della validation loss.

5.1 VGG16

Non Trainable

Figura 6: Risultati del training del modello VGG16 (Non Trainable)

In questo modello il valore della validation loss risulta ad essere uguale a 0,0154.

Trainable

Figura 7: Risultati del training del modello VGG16 (Trainable)

In questo modello il valore della validation loss risulta ad essere uguale a 0,0156.

5.2 ResNet50

Non Trainable

Figura 8: Risultati del training del modello ResNet50 (Non Trainable)

In questo modello il valore della validation loss risulta ad essere uguale a 0,0148.

Trainable

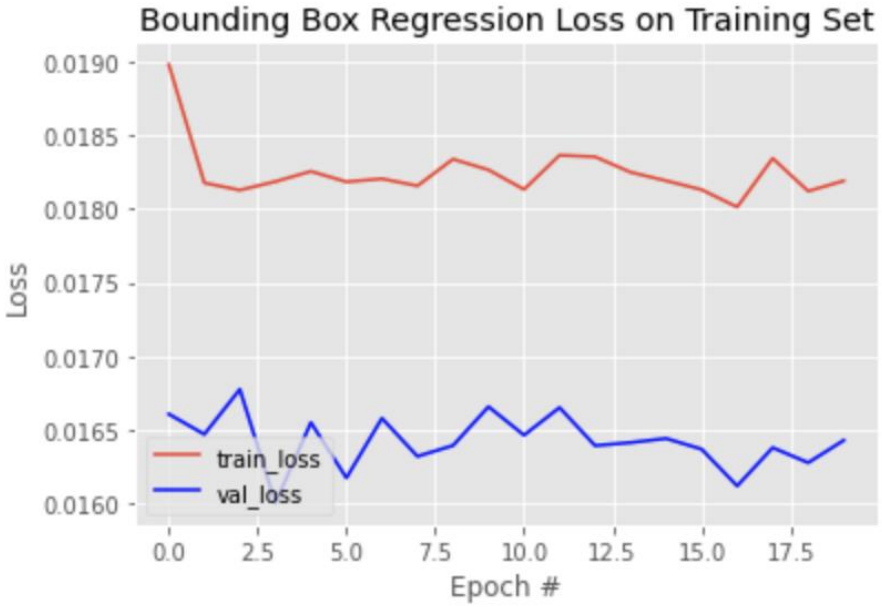


Figura 9: Risultati del training del modello ResNet50 (Trainable)

In questo modello il valore della validation loss risulta ad essere uguale a 0,0163.

6 Risultati

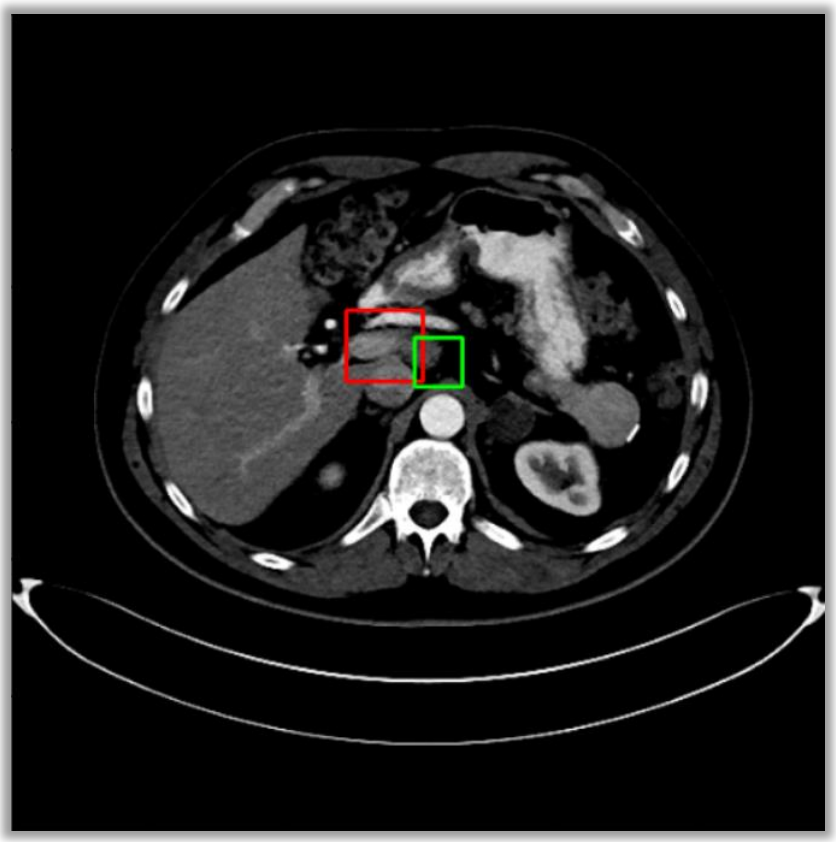
Nella tabella sottostante vengono riportati i valori della loss ottenuti sul dataset di test nelle diverse modalità.

	Non Trainable	Trainable
VGG16	0,0172	0,0186
ResNet50	0,0181	0,0176

Analizzando i dati della tabella possiamo notare che il modello che presenta il risultato più preciso è VGG16 nella modalità 'non trainable' con un valore di loss pari a 0,0172.

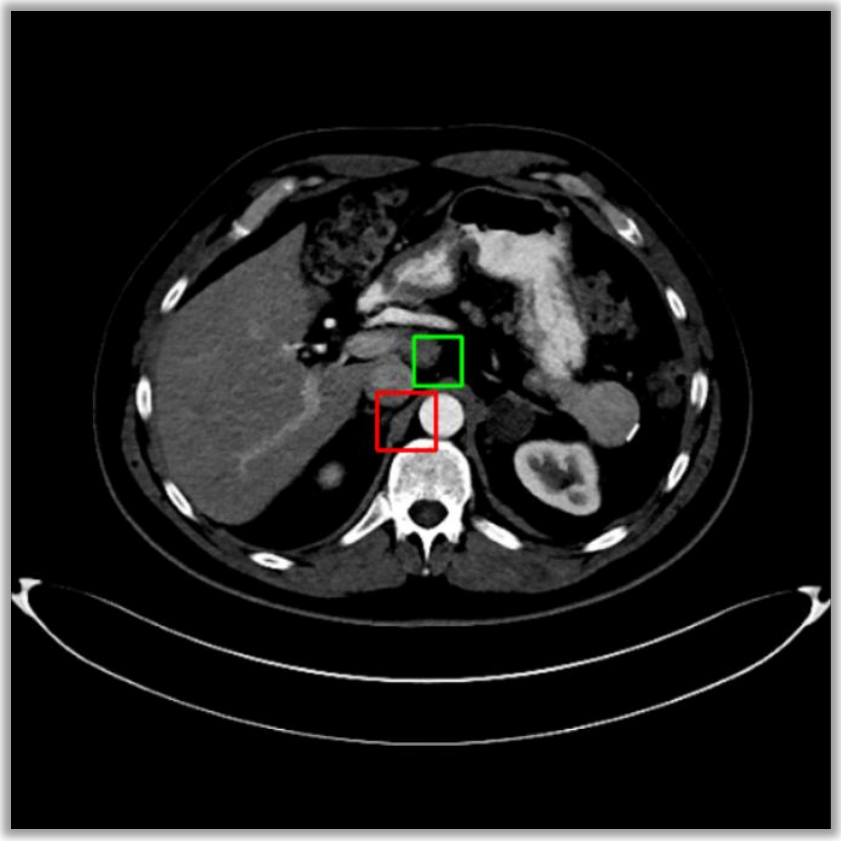
Per quanto riguarda il modello VGG16 si può concludere che la modalità più performante è quella 'non trainable', ovvero utilizzando i pesi allenati sul dataset ImageNet. Invece, se consideriamo il modello ResNet50 vediamo che è più performante la modalità 'trainable', la quale utilizza i pesi allenati sul nuovo dataset.

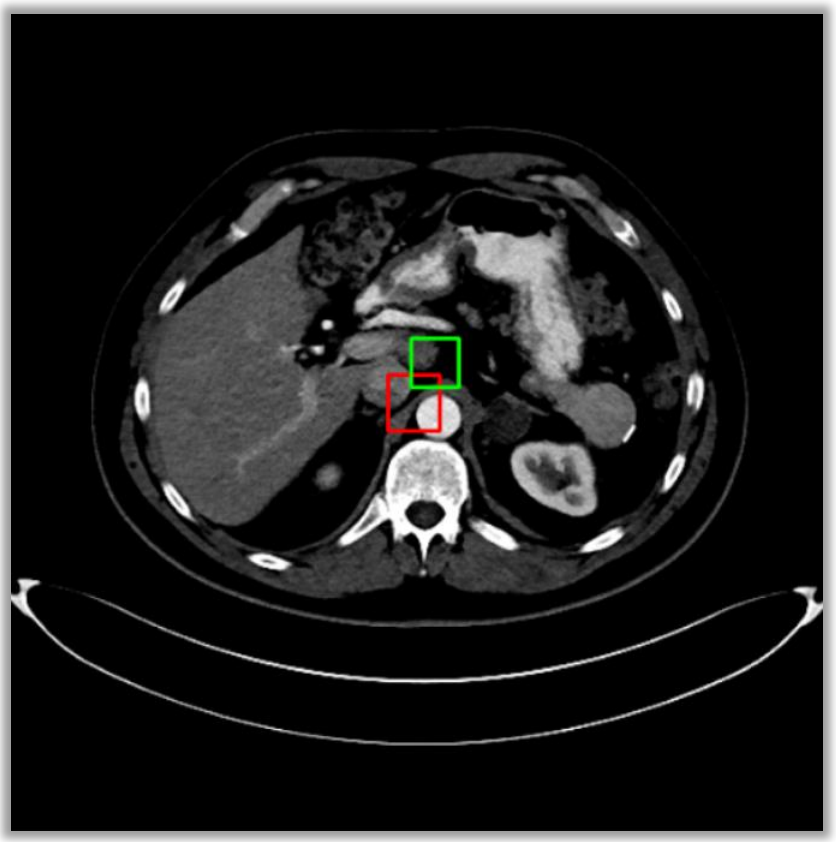
Nelle immagini seguenti si può vedere l'applicazione in esecuzione in tutte le sue modalità su un'immagine presa dal dataset di test, quindi mai vista in precedenza. Per verificare la precisione dei vari modelli sono stati disegnati sull'immagine due bounding box differenti: uno verde che identifica dove dovrebbe essere la lesione realmente e uno rosso che identifica la predizione della lesione da parte della rete.



VGG16 Non Trainable

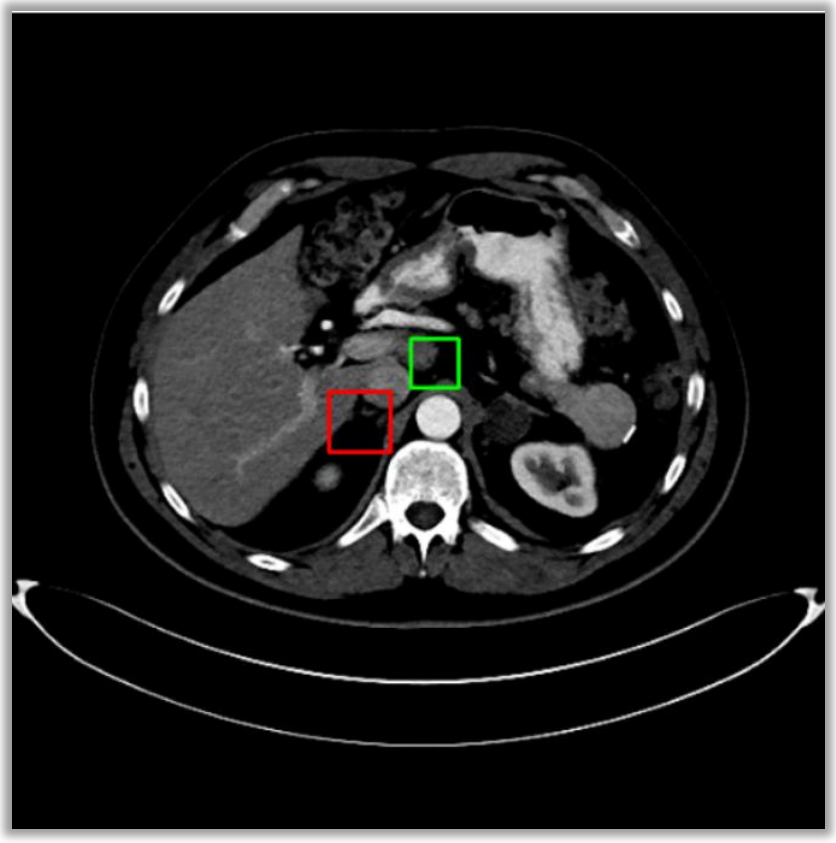
VGG16 Trainable





ResNet50 Non Trainable

ResNet50 Trainable



Conclusioni e Sviluppi Futuri

Questo progetto è stato improntato totalmente sull'utilizzo della tecnica di Object Detection che permette l'individuazione dei bounding box che rappresentano le diverse lesioni presenti sul corpo umano.

La valutazione complessiva del progetto da buoni risultati che possono essere migliorati tramite altre modalità. In particolare, può essere preso come base di partenza per provare altri modelli più adatti al tipo di immagini prese in considerazione.

Inoltre, un grande miglioramento per la rete attuale potrebbe essere l'inserimento delle labels che indicano la tipologia della lesione. In questo modo oltre a trovare la posizione si approfondirebbe con anche la specifica. Il materiale che è stato utilizzato per sviluppare questo progetto contiene tutte le informazioni necessarie per realizzare questi miglioramenti.

Un'ultima modifica che si potrebbe introdurre è l'utilizzo degli assi x e y che individuano il punto esatto in cui si trova la lesione all'interno del bounding box. Anche qui, le informazioni sulle coordinate degli assi sono disponibili nel materiale usato per lo sviluppo del progetto.

Link repository GitHub: https://github.com/Fedeee9/LesionDetection