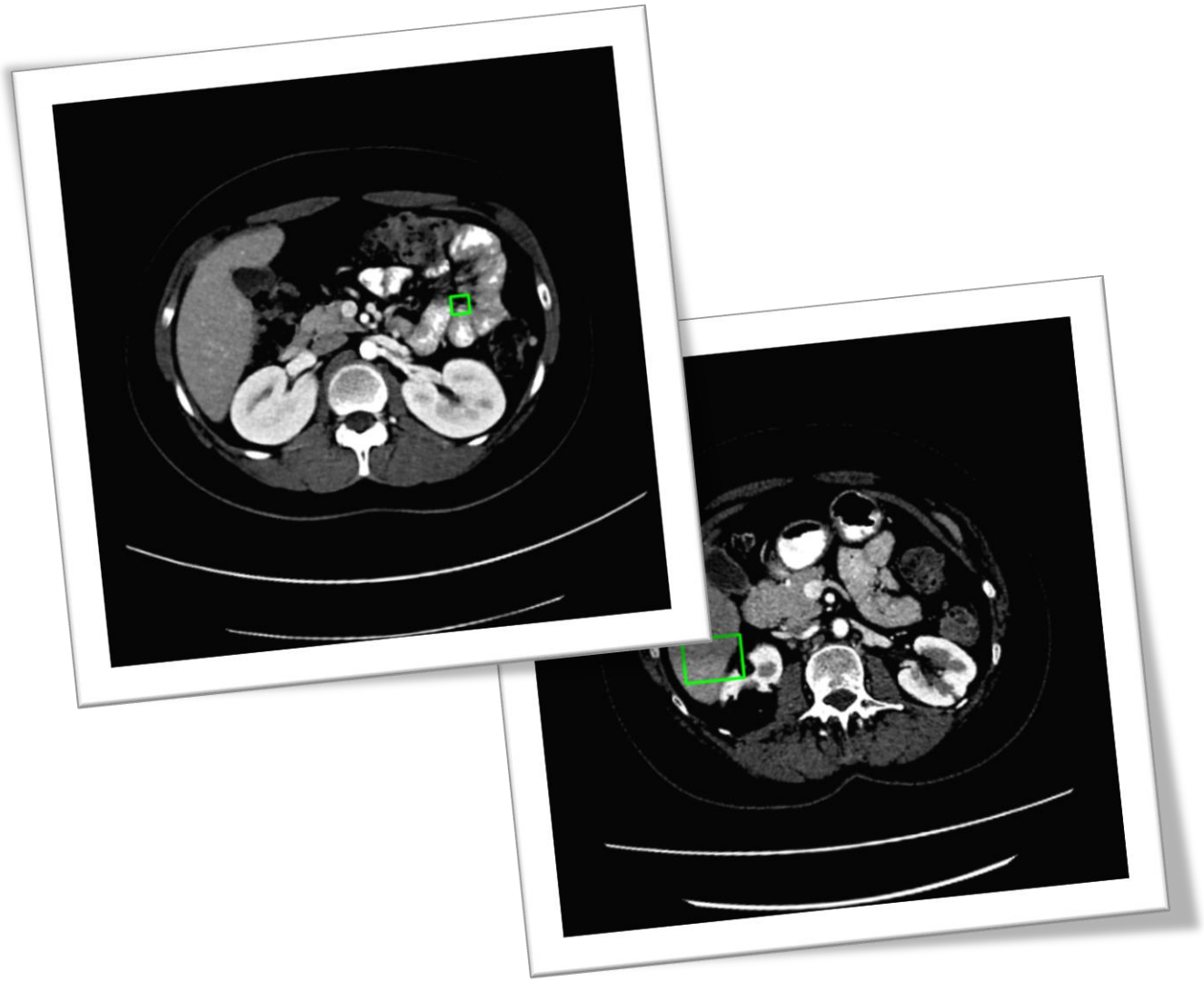
Lesion Detection

Computer Vision and Image Processing M Project



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

Index

Introduction

Neural Networks are, among all the machine learning techniques, the most used in many kinds of situations. Going a bit more in the details, NNs use as many examples as possible in the training phase in order to automatically infer the rules to be able to recognize the objects.

Computer vision is one of the most famous applications of the NNs, that is able to reproduce the cognitive path of a human that is able to recognize the surrounding environment through the images perceived by the eyes.

This document has the aim to show the steps achieved in order to build a neural network able to detect, through the bounding boxes technique, lesions on different parts of the human body.

In order to explain all the different techniques involved for the project, we will talk about Computer Vision and Machine Learning tools and techniques, in particular we will focus on Tensorflow using Python.

1 Background

1.1 Computer Vision

Computer Vision is a science that falls down the Artificial Intelligence field, and its aim is to understand how to build models representing the real world starting from bidimensional images.

In particular, it takes care to acquire, store and elaborate images coming from external supports (for example webcam) in order to recognize and determine specific characteristics of the image and being then able to control, classify and select them.

Computer Vision is a discipline strictly related to the theory of the artificial system able to extract information from the images. In fact, the data obtained from the images can be represented in a lot of different ways: for example, like sequences of videos coming from different cameras, or like multimedia data coming from the scanner.

From the technology point of view, Computer Vision takes care to build visual systems on the machine, for example machine vision, robot vision and visual-based multimedia system.

Image 1: Example of using Computer Vision

The images elaboration, that is the principal component of the computer graphics, relies on various operations that, based on the abstract operational level where they operate, can be defined as low-level (from the acquisition process of the image to the preprocessing), medium-level (from the image segmentation to their representation and description) and high-level (the research of an interpretation of the observed scene)

To conclude, thanks to Computer Vision it is possible to model a dynamic non structured environment, being able to plan trajectory, building tridimensional reconstruction of the scene and many more, keeping in mind some factors of criticity like quality and speed to acquire the images.

Computer Vision's system follows precises steps:

1. Images acquisitions from cameras
2. Transfer of data
3. Image processing, meaning the acquired image's analysis from the software
4. elaboration of data regarding the result obtained

1.2 CNN (Convolutional Neural Network)

The CNN, utilized in Computer Vision, can be seens as Deep Learning algorithms. They are mainly used to classify images, group them by similarity and recognize the object within the scene through the improvement of the weights of the network

Every weight is usually represented by a numeric value in float32, that can also be reduced to float16 or float8 in order to speed up the network. The weights initially assigned randomly are during the training step modified to get the best and most precise results.

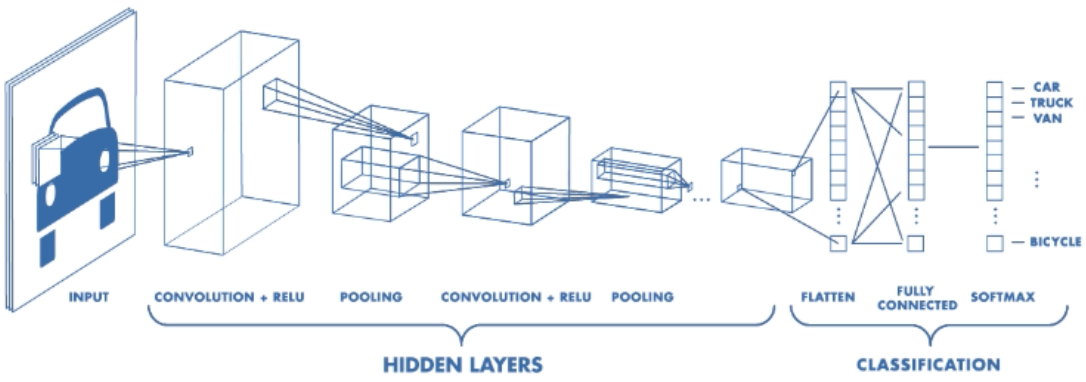


Image 2: Convolutional Neural Network

1.3 Object Detection

Artificial Intelligence combined with Computer Vision it's a powerful way to approach the elaboration of the images and there are many methods to do so. A classical example is the Object Detection problem that gives the possibility to obtain the bounding box (the area of the image that contains the object that we want to recognize) for each object detected in the image.

Another approach that we will not implement during the project, but it is worthed to be highlighted is Image Classification, which is a process that brings a trained model to the acquisition of the input and gives you back the most probable object's class or the percentage of how the input belongs to a specific class.

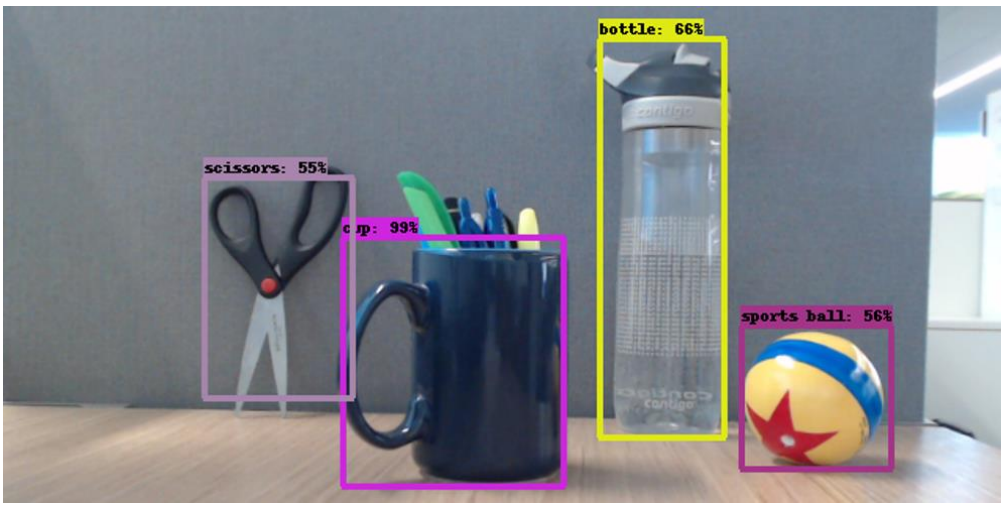


Image 3: Example of Object Detection

2 Tools

In order to develop the following project tools like, TensorFlow, Keras, OpenCV played a crucial role in order to let us train the networks on the dataset

2.1 TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning that uses a flow graph of the data approach. It has been developed in order to let the parallel computation available on more than 1 CPU or GPU, both on a single machine and on a distributed one. TensorFlow has been firstly deployed by Google Brain Team researchers and engineers with the aim of ML research and DL neural network but then the deployment switched to a more general approach in order to use it in many other applications.

The library has many API, one of the lower-level ones is the TensorFlow Core, that gives the possibility to have complete control on the programming part. The APIs are typically used in the ML field because they make the detailed control of all the model's elements possible.

TensorFlow is compatible with our main operative 64 bit (Windows, Linux and macOs) and Android system. Moreover, it provides several interfaces for the different languages, namely Python, C/C++, Java and Go. The specific case of this project Python is the language used to deploy it.

In general, TensorFlow allows the developers to make the following operations:

- Pre-processing of the data
- Build of the model
- Training and evaluation of the model

2.2 Keras

TensorFlow is a great framework for Deep Learning applications, but the realization of the models remains a complex task. In fact, the open-source framework Keras helps us in order to define the deep learning models in a more intuitive and modular way and with less lines of code. The main peculiarity of the library is the capability of being the interface of lower-level libraries like TensorFlow or Theano. Keras provides all the most common deep learning model's implementations losing less in performance with respect to

TensorFlow. Since 2007, TensorFlow's team decided to add modules in order to support Keras in their libraries.

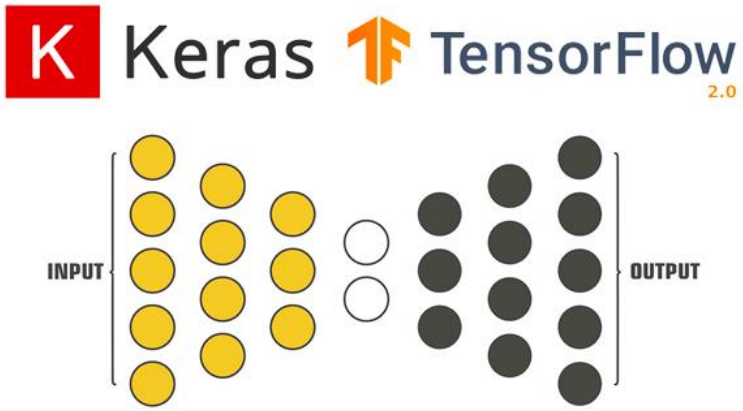


Image 4: Keras & TensorFlow

2.3 OpenCV

Image 5: OpenCV

OpenCV (Open-Source Computer Vision) is an open-source library, in C++, that contains all the functions used for real-time video streaming, namely Computer Vision. OpenCV framework was born from Intel's initiatives while working on bettening of their CPU for intensive applications, for example the real time ray-tracking and the 3D projection. Later on, some of the infrastructure for Computer Vision employees decided to start to project a new framework for the new Intel's processors. In 1999 started the project in collaboration with a Russian team, and the first official release of OpenCV happened in 2006. In 2011 a 2.2.0 version was released that allows the support of Android devices. Today the 4.5.1 release is available.

3 Models

In order to train our network, we used two different pretrained models to be able to compare the performances and to evaluate the main differences. The two models are:

- ResNet50
- VGG16

Both the models have been previously trained on the ImageNet dataset, that contains more than 14 Million images divided in 1000 classes and are indeed able to distinguish an object in the image among 1000 different categories. Then, these models have been trained again on our dataset that is composed of images that represent sections of computed tomography, in order to let them be specific for the problem that we want to address: The Lesion Detection.

For what concerning the training, the callbacks have been implemented, in particular the 'Early Stopping' that stops the training whenever the values don't get better after some epochs and the 'Dynamic Learning Rate' that starting from a predefined learning rate it decreases during the training process in order to reach a better precision.

In addition, it is important to highlight that we have introduced different significant parameters in order to perform our training phase and obtain reasonable results. In particular, we have the batch size parameter and different sizes of each of the 4 dense layers of our networks:

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

Both the models have been trained in two modalities and the difference depends on a specific parameter for each model, namely 'trainabe'. Setting this parameter equal to 'True' we are going to train not only our model, but we are also re-training the original model (VGG16 or ResNet50 in our case) modifying the pretrained weights. On the contrary, setting the parameter equal to 'False' we will keep the original pretrained weights on ImageNet for the models.

3.1 VGG16

VGG16 is a neural network model that trained on the ImageNet dataset is able to reach an accuracy of 92,7%. This model is used in many projects where the main task is a deep learning task for image classification; however, smaller architectures are very often preferable.

3.2 ResNet50

ResNet50 is an architecture that, using a new and innovative type of blocks (called residual block) and the residual learning method, allowed it to reach very high depth using the feed forward model due to the problem of gradient degradation.

Exist different depth implementations where the deepest one has 152 levels.There also exists a prototype with 1202 levels that reaches worse results due to overfitting.

4 Dataset

This project relies on the DeepLesion dataset (https://nihcc.app.box.com/v/DeepLesion) provided from the National Institutes of Health Clinical Center. The dataset contains 32120 digitalized tomography images and for the sake of the project we took in consideration just 15526 of them, which we then manually splitted in three sets: training set (12824), validation set (1534) and test set (1168).

In order to proceed with the split, we used three different excel files that supported our process (Train_info.csv, Val_info.csv, Test_info.csv) available together with the Dataset, these contain all the information needed for every single image. In particular we can find the bounding box coordinates that allow us to represent the boxes on the images through the OpenCV function.

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

Image 6: VGG16 training results (Non Trainable)

In this model, the validation loss value is equal to 0,0154.

Trainable

Image 7: VGG16 training results (Trainable)

In this model, the validation loss value is equal to 0,0156.

5.2 ResNet50

Non Trainable

Image 8: ResNet50 training results (Non Trainable)

In this model, the validation loss value is equal to 0,0148.

Trainable

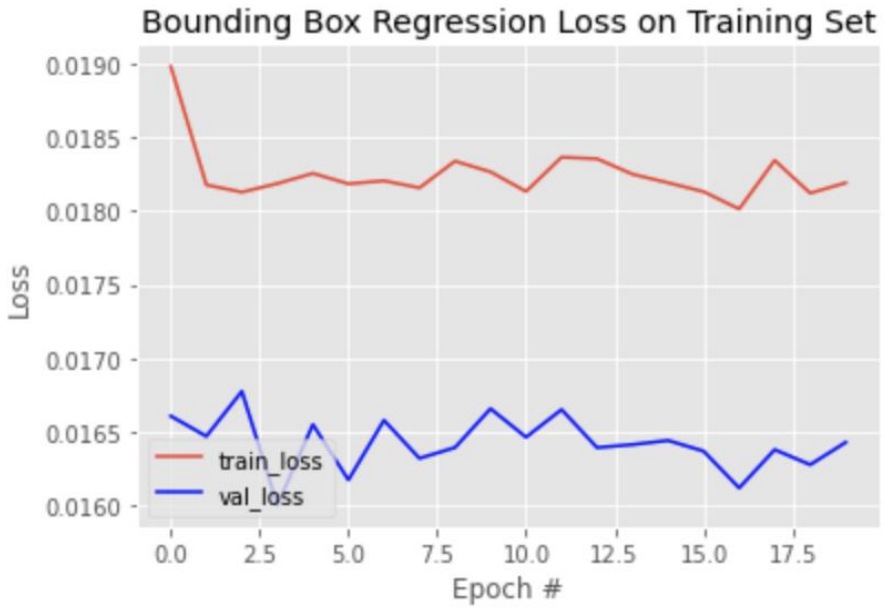


Image 9: ResNet50 training results (Trainable)

In this model, the validation loss value is equal to 0,0163.

6 Results

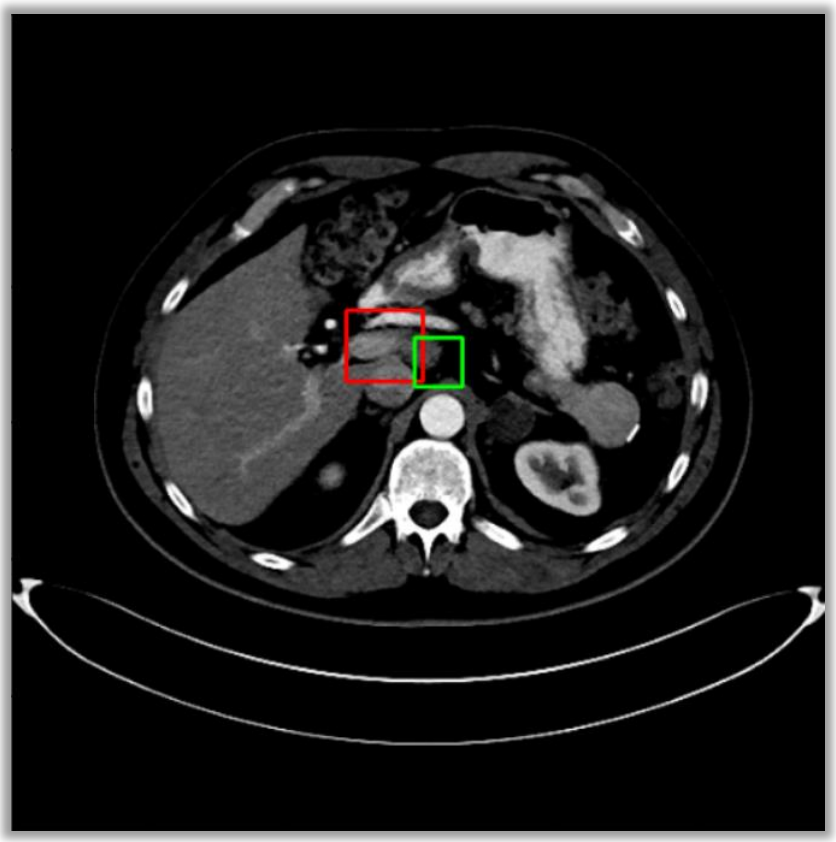
In the table reported below we have highlighted the loss values obtained on our test dataset using our different modalities.

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

Analyzing the table numbers we can notice that the model that shows the best and more precise value is VGG16 using the trainable modality set to 'False' meaning that it is not trainable. The loss value reached is 0.0172.

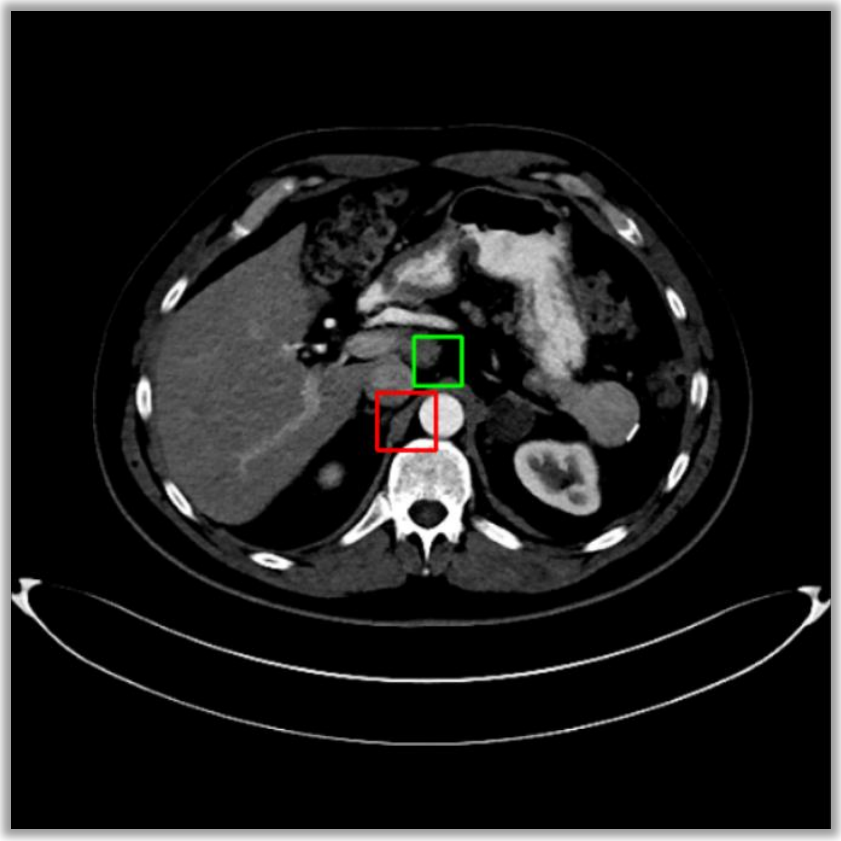
For what concerning VGG16 we can conclude that the most appropriate modality for this kind of task is the not trainable one that uses the already trained weights on the ImageNet dataset. Instead, considering the ResNet50 model we can notice that the best setting is when the trainable parameter is equal to 'True' that uses the new weights trained on the new dataset provided.

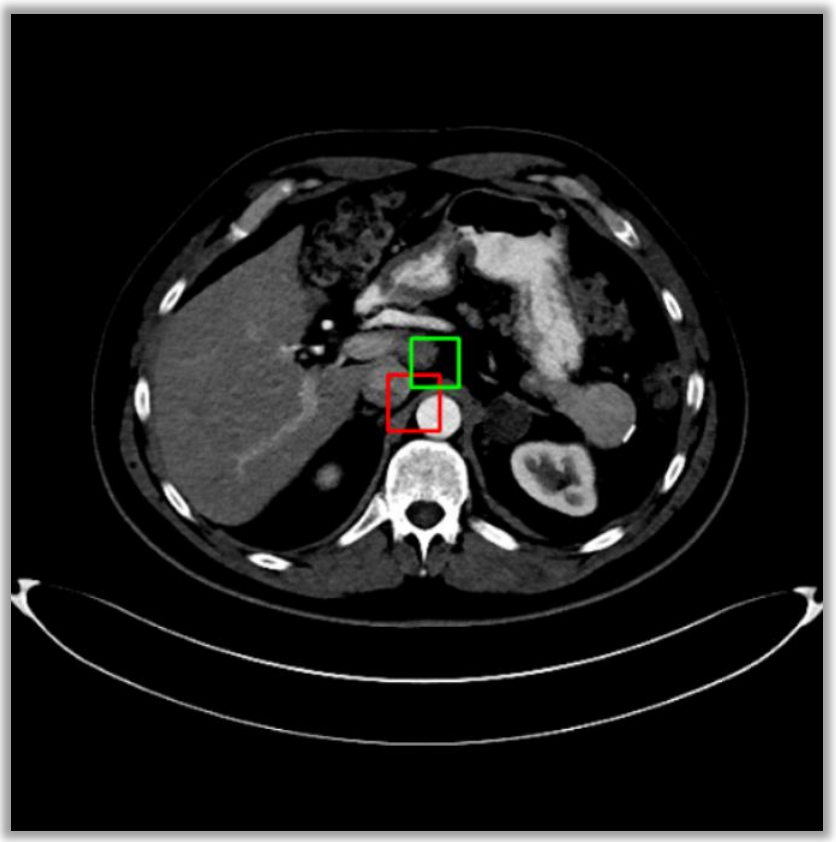
In the following images we will show the results, from a graphical point of view, of each of our applications using an image from the test split of our dataset, so never seen before from the network. In order to verify the precision of each model we have drawn on the image two different bounding boxes: the green one representing where the lesion really is and the red one that identifies the prediction obtained from our model.



VGG16 Non Trainable

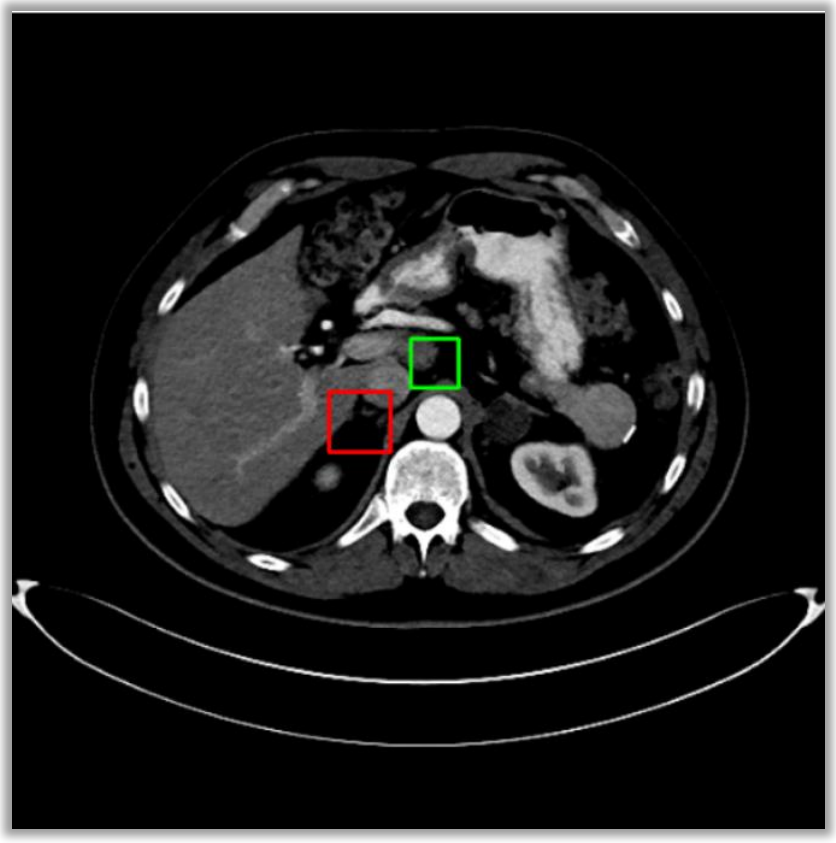
VGG16 Trainable





ResNet50 Non Trainable

ResNet50 Trainable



Conclusion and next steps

This project has the primary aim to use the Object Detection technique in order to identify the bounding boxes on the images that represent where the lesions are located on the human body.

The overall evaluation of the project gives nice results that can be absolutely improved in many ways. In particular would be a good start to investigate other pretrained models to be more suitable for the type of images that we used.

In addition, a great improvement of what we have done so far would be the insertion of the lesion's labels in order to not only find the lesion but also classify it with the correct label. The material we have used for our project contains all the needed information to make these improvements happen.

A valuable 'next step' in addition to the one just explained can be the introduction of the axis to detect the exact point where the lesion is located within the bounding box. Again here, the info about the coordinates of the axis are available in the project materials.

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