ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI FISICA E ASTRONOMIA - DIFA

PhD Thesis seminary

Implementation and optimization of algorithms in Biomedical Big Data Analytics

Nico Curti

Supervisor:

Prof. Daniel Remondini

Co-Supervisor:

Prof. Gastone Castellani

Prof. Armando Bazzani

January 27, 2020

Purposes

Biomedical Big Data Analytics

Feature Selection
DNetPRO

- **Topic:** Gene Expression Analysis;
- **Characteristics:** Novel algorithm & optimization on distributed computing;
- **Applications:**
  - Synthetic dataset;
  - Benchmark TCGA datasets;
  - Cytokinoma;
  - Bovine Signature;
  - Pedestrian mobility in Venice;
- **URL:** https://github.com/Nico-Curti/DNetPRO;

Deep Learning
Byron

- **Topic:** Neural Network models;
- **Characteristics:** Code Optimization and multi-threading parallelization;
- **Applications:**
  - Super Resolution;
  - Object Detection;
  - Image Segmentation;
- **URL:**
  - https://github.com/Nico-Curti/NumPyNet;
  - https://github.com/Nico-Curti/Byron;

Big Data
CHIMeRA

- **Topic:**
  - Biomedical Big Data;
  - Database service;
  - Network analysis;
  - Natural Language Processing;
  - Web scraping;
  - Data harmonization;
- **Characteristics:** Merging of public available knowledge into network structure;
- **Applications:** Leukemia;
- **URL:** Not available yet;

Deep Learning

Neural Network models

Deep Learning model is becoming equivalent to Neural Network architecture/model, i.e a more or less complex pipeline of functions which takes in input a sample and it applies a series of transformations to obtain the desired result.

Objectives

- Optimization and extension of state-of-art Neural Network libraries;
- Development of two novel libraries, for education and analytic purposes, respectively;
- Improve parallelization strategies to reach the best computational performances in cluster machine without GPUs supports;

All the algorithm and codes developed are **tested**, **documented** and **public available**!

Reference: https://nico-curti2.gitbook.io/phd-thesis (GitBook documentation).

What is a Deep Learning model?

Neural Network models

- Neural Networks are mathematical models **commonly used** in data analysis.
- From a theoretical point-of-view we can define a Neural Network as **a series of non-linear multi-parametric functions**.
- Neural Networks are considered as **Universal Function Approximators**.

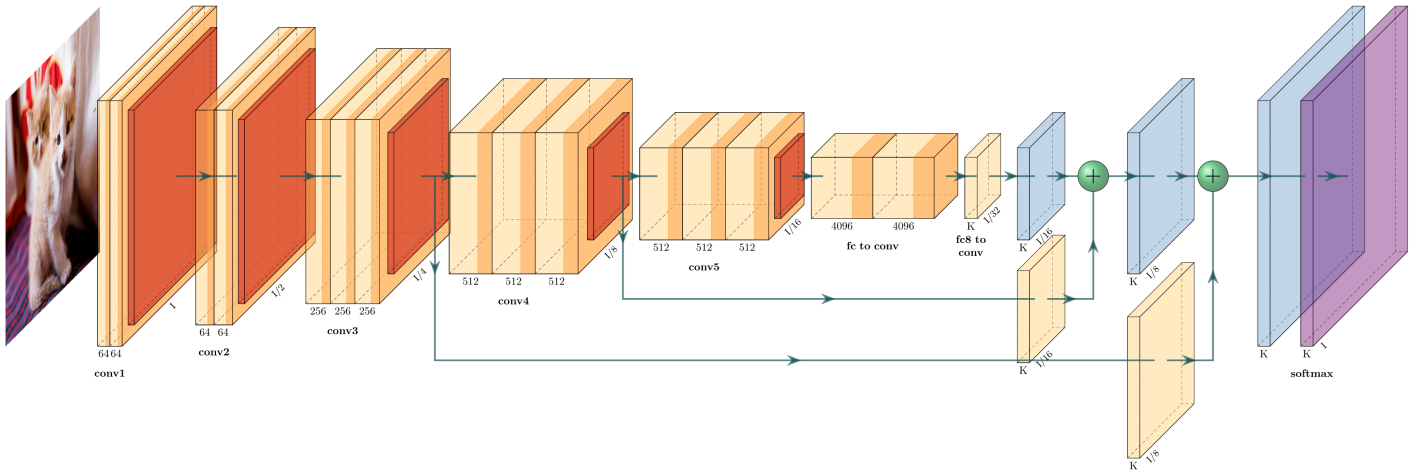


Figure: Neural Network model scheme. The model is composed by a series of layers made by interconnected computational units. The connections between layers can be serials or parallels and each layer models a pre-determined mathematical function

State-of-art libraries

Why other two libraries?

A wide range of documentations and implementations have been written on Deep Learning and it is more and more hard to move around the different sources. Leader on this topic are became the multiple open-source 'Python' libraries available on-line as Tensorflow, Pytorch and Caffe.

Pros

- Simplicity in writing complex models in a minimum number of code lines;
- Simplicity of the Python language;
- Extremely efficient in GPU environments;

Cons

- Steep learning curve for complex applications or algorithm modifications;
- Hard to manage as services;
- They are not designed for CPU performances;

Only a small part of the research community uses more deeper implementation in C++ or other low-level programming languages. About them it should be mentioned the darknet project of Redmon J. et al. which created a sort of standard in object detection applications using a pure Ansi-C library.

NumPyNet & Byron

Why other two libraries?

NumPyNet

- **LANGUAGE:** **Python**;
- **INSTALLATION:** **Setup** (**PyPI** upload wip);
- **OS:** Linux & MacOS & Windows;
- **DOCUMENTATION:** fully documented;
- **PURPOSE:** educational / study;
- **CI:** Travis & Appveyor;
- **USAGE:** model design / algorithm improvements;
- **MISCELLANEOUS:** all functionality are tested against **Keras** counterpart;
- **URL:** https://github.com/Nico-Curti/NumPyNet

Byron

- **LANGUAGE:** **C++** Standard 17;
- **INSTALLATION:** **CMake** & **Make** & **Docker**;
- **OS:** Linux & MacOS & Windows;
- **DOCUMENTATION:** wide set of usage examples;
- **PURPOSE:** multi-threading CPU optimization;
- **CI:** Travis & Appveyor & CircleCI;
- **USAGE:** Super Resolution & Object Detection & Image Segmentation;
- **MISCELLANEOUS:** extension and optimization of the **darknet** project;
- **URL:** https://github.com/Nico-Curti/Byron

From the union of this two project **Pyron** is born, i.e the completely wrap of the Byron library in Python using Cython.
* Both libraries are already used in different works of thesis and research projects!

NumPyNet

Neural Networks in Pure NumPy

NumPyNet

- The library is written in pure Python and the only **external library** used is Numpy (a based package for the scientific research).
- Despite all common libraries are correlated by a wide documentation is often difficult for novel users to move around the many hyper-links and papers cite in it. NumPyNet tries to overcome this problem with a **minimal mathematical documentation associated to each script** and a **wide range of comments inside the code**.
- Libraries like Tensorflow are efficient by a computational point-of-view and they have an extremely simple user interface. On the other hand the **deeper functionalities** of the code and the **implementation strategies** used **are unavoidably hidden behind tons of code lines**. In this way the user can performs complex computational tasks using the library as **black-box package**. NumPyNet wants to overcome this problem using **simple Python codes with extremely readability also for novel users** to better understand the symmetry between mathematical formulas and code.
- NumPyNet uses Python threads to easily manage independent tasks and optimize the computation.

* The library was developed in collaboration with Dott. Ceccarelli.

Byron

Build YouR Own Neural Network

Byron

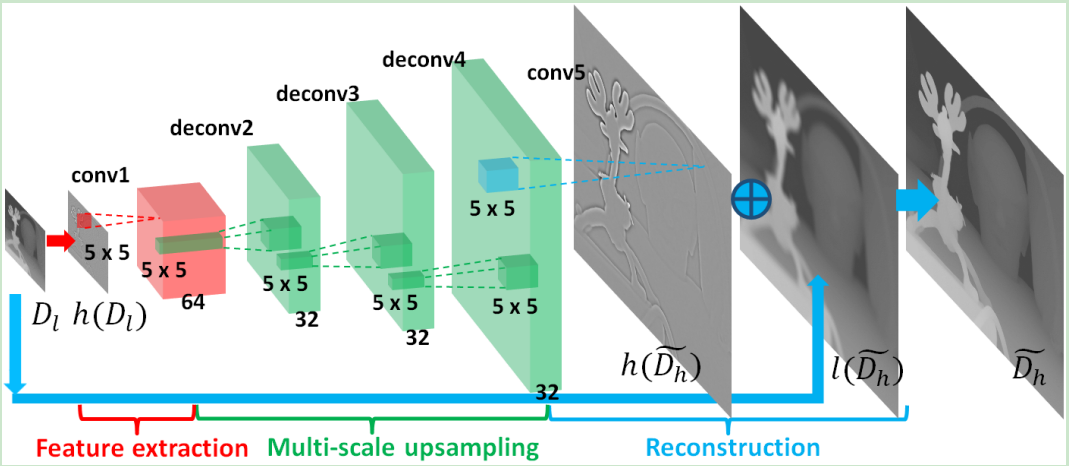
- The library is written in **pure C++** with the support of the modern standard 17.
- The library is **optimized for image processing** (probably the most common task in biomedical research).
- Starting from the darknet project backbone, Byron proposes **numerous improvements and fixes**.
- Byron works in a **fully parallel section** (OpenMP) in which each single computational function is performed using the full set of available cores. To further reduce the time of thread spawn and so optimize as much as possible the code performances, the library **works using a single parallel section** which is opened at the beginning of the computation and closed at the end.
- The library is also **completely wrapped** using Cython to enlarge the range of users also to the Python ones.

* The library was developed in collaboration with Dott. Baroncini.

Applications

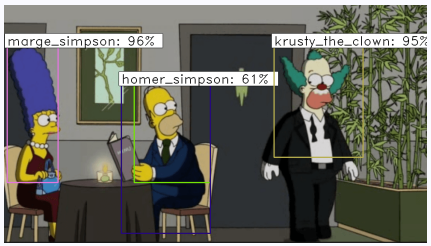
Deep Learning model tasks

Single Image Super Resolution



Improve spatial image resolution and reconstruction from low-resolution (LR) to high-resolution (HR).

Object Detection



Identify single or multiple objects into a picture or video stream.

Image Segmentation

Extract the exact pixels which belong to an object into a given image.

Single Image Super Resolution

What is Super Resolution?

We commonly think about Neural Network models as tools to **reduce** the problem dimensionality, i.e starting from high-dimensional data (e.g $w \times h \times c$ image) the model predicts a class (single number).
In the Super Resolution problem we have no classes but the desired output is a image.
Single Image Super Resolution tasks starts from a image aiming to reconstruct its HR counterpart.

Super resolution procedure

- We start from a prior-known HR image;
- We rescale this image obtaining its LR counterpart;
- We feed a super-resolution model with the LR image aiming to obtain in output its HR version;
- With the trained model we can re-apply the up-sampling procedure and further increase the HR quality*.

If this pipeline is performed feeding a Neural Network model using one-by-one a series of images, we talk about Single Image Super Resolution (SISR) algorithm.

* This is even more efficient if we think about a **zoom** procedure.

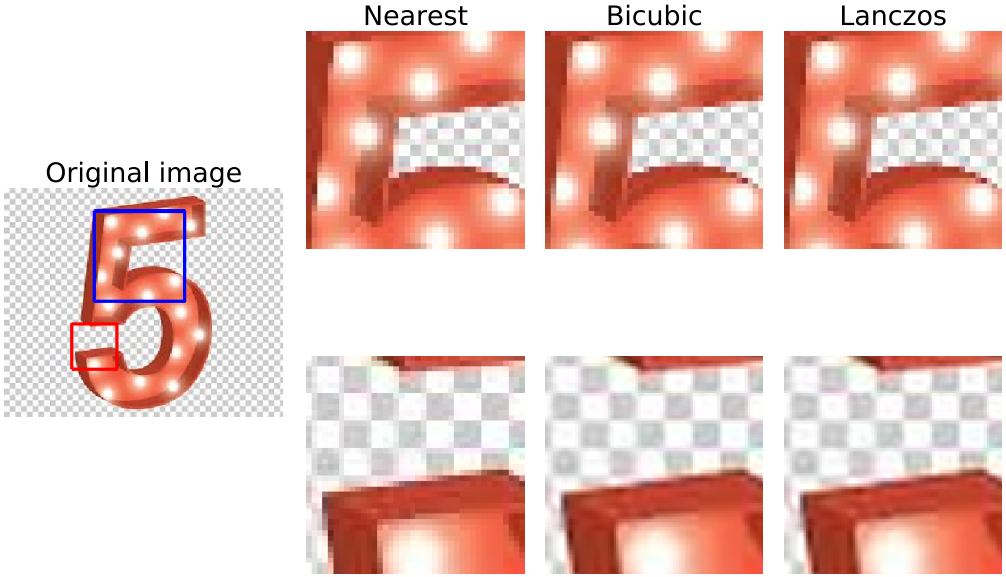
Standard up/down scaling algorithms

Single Image Super Resolution

Interpolation algorithms:

- **Nearest**;
- **Bicubic**;
- **Lanczos**;

The best compromise between computation cost and efficiency is given by the Bicubic interpolation algorithm. Given a pixel, the interpolation function evaluates the 4 pixels around it applying a filter given by the equation:



$$k(x) = \frac{1}{6}\begin{cases} (12-9B-6C)|x|^3 + (-18+12B+6C)|x|^2 + (6-2B) & \text{if} |x| < 1 \\ (-B-6C)|x|^3 + (6B+30C)|x|^2 + (-12B-48C)|x| + (8B+24C) & \text{if} 1 \leq |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

where x identifies each pixel below the filter. Commonly values used for the filter parameters are $B = 0$ and $C = 0.75$ (used by OpenCV library) or $B = 0$ and $C = 0.5$ used by Matlab.
NOTE: the nearest interpolation could be achieved also using *Pooling* algorithm.

Image Quality

Single Image Super Resolution

The most common image quality evaluator is given by our eyes.
Quantitative evaluations:

PSNR - Peak Signal to Noise Ratio

- It establishes the compression lossless of an image;
- Equation:

$$PSNR = 20 \cdot \log_{10}\left(\frac{\max(I)}{\sqrt{(MSE)}}\right)$$

where $\max(I)$ is the maximum value which can be taken by a pixel in the image and MSE is the Mean Square Error evaluated between the original and reconstructed images.

SSIM - Structural SIMilarity index

- It evaluates the structural similarity between two images taking into account also the visible improvement seen by human eyes.
- Equation:

$$SSIM(I,K) = \frac{1}{N}\sum_{i=1}^{N}\frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{({\mu_x}^2 + {\mu_y}^2 + c_1)({\sigma_x}^2 + {\sigma_y}^2 + c_2)}$$

where N is the number of arbitrary patches which divide the image and c_1 and c_2 parameters are fixed to avoid mathematical divergence.

	PSNR (dB)	SSIM ($\in [-1,1]$)
Nearest	25.118	0.847
Bicubic	27.254	0.894
Lanczos	26.566	0.871

EDSR - Enhanced Deep Super Resolution

Super Resolution Models

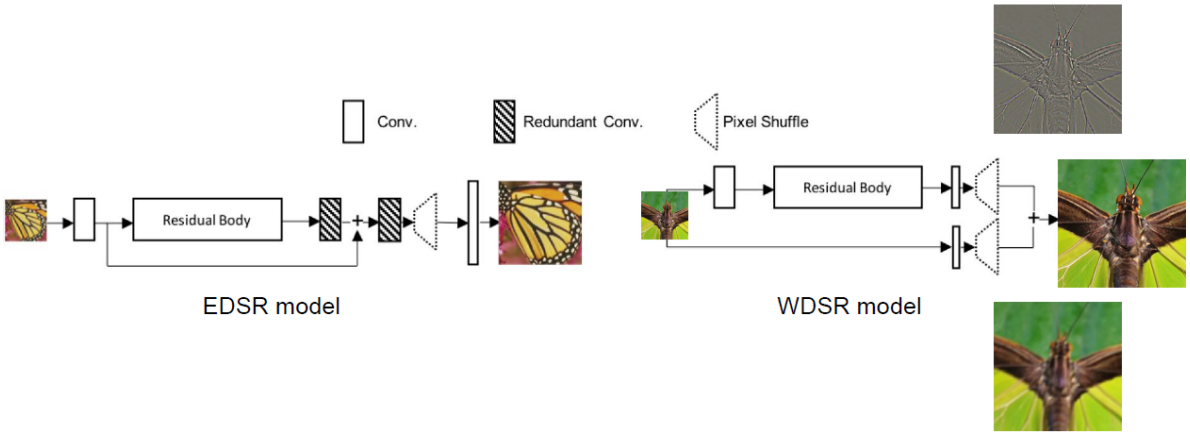
Architecture

Layer	Channels input/output	Filter dimensions	Number of Parameters
Conv. input	3/256	3×3	6912
Conv. (residual block)	256/256	3×3	589824
conv. (pre-shuffle)	256/256	3×3	589824
Conv. (upsample block)	256/1024	3×3	2359296
Conv. output	256/3	3×3	6912

Average Time on 510×339 image: 576.92 s.
Winner of the **NTIRE 2017**. More than **3 million** of parameters!

WDSR - Wide Deep Super Resolution

Super Resolution Models



Architecture

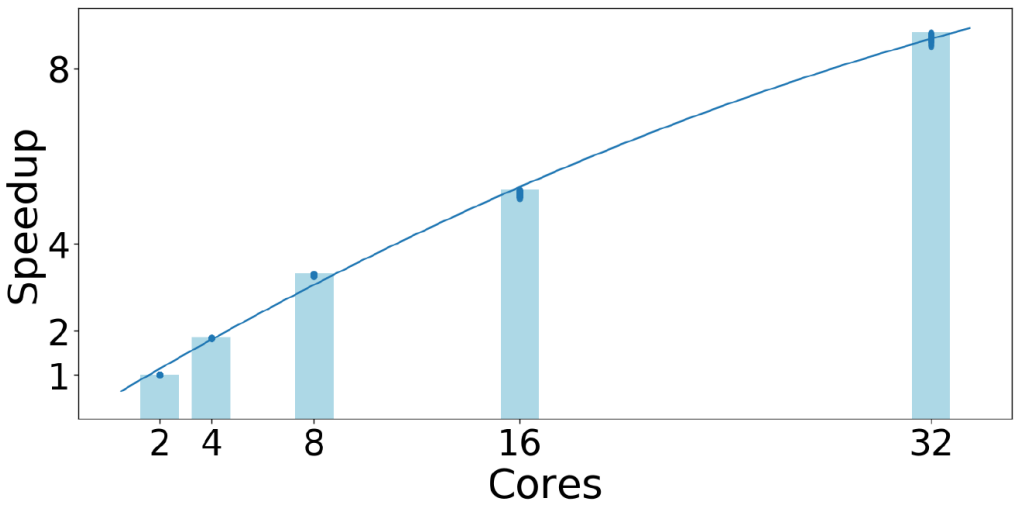
Layer	Channels input/output	Filter dimensions	Number of Parameters
Conv. input 1	3/32	3×3	864
Conv. 1 (residual block)	32/192	3×3	55296
conv. 2 (residual block)	192/32	3×3	55296
Conv. (pre-shuffle)	32/48	3×3	13824
Conv. input 2 (pre-shuffle)	3/48	5×5	3600

Average Time on 510×339 image: 46.35 s.
Winner of the **NTIRE 2018**. $\sim$ 100K parameters, **less than** 10% **of EDSR model**!

Computational Performances

Super Resolution Models

Computational performances of WDSR model on 510×339 images (100 runs). Environment: Bioinformatics server grade machine (128 GB RAM memory and 2 CPU E5-2620, with 8 cores each.



N° threads	Average Time (s)
2	403.478
4	219.268
8	122.986
16	78.571
32	46.348

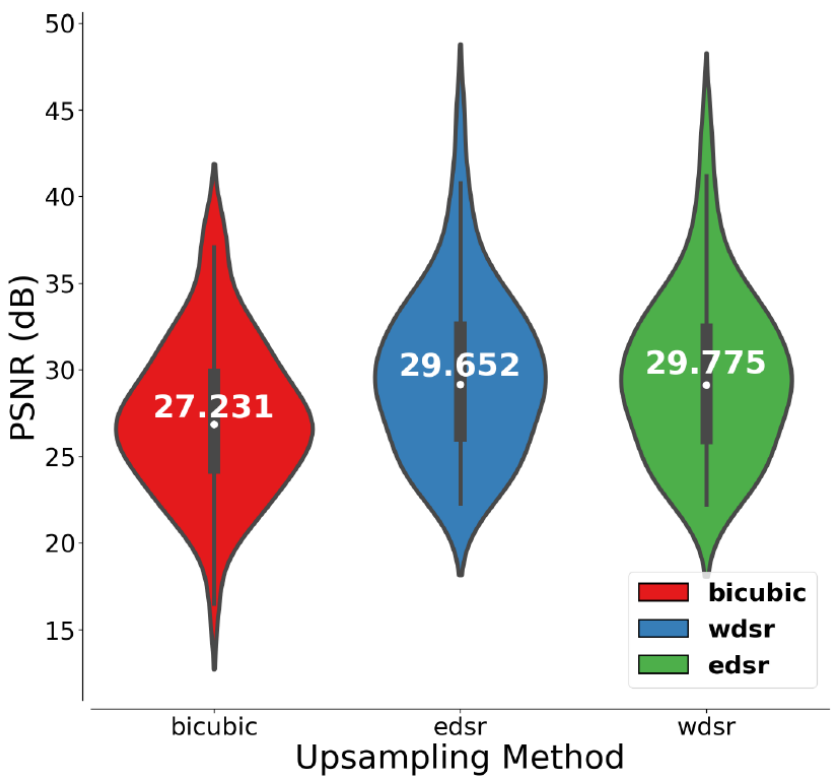
Sub-linear trend:

- Hyperthreading;
- NO affinity;

Quality Performances

Super Resolution Models

Evaluation of PSNR and SSIM on DIV2K validation set.



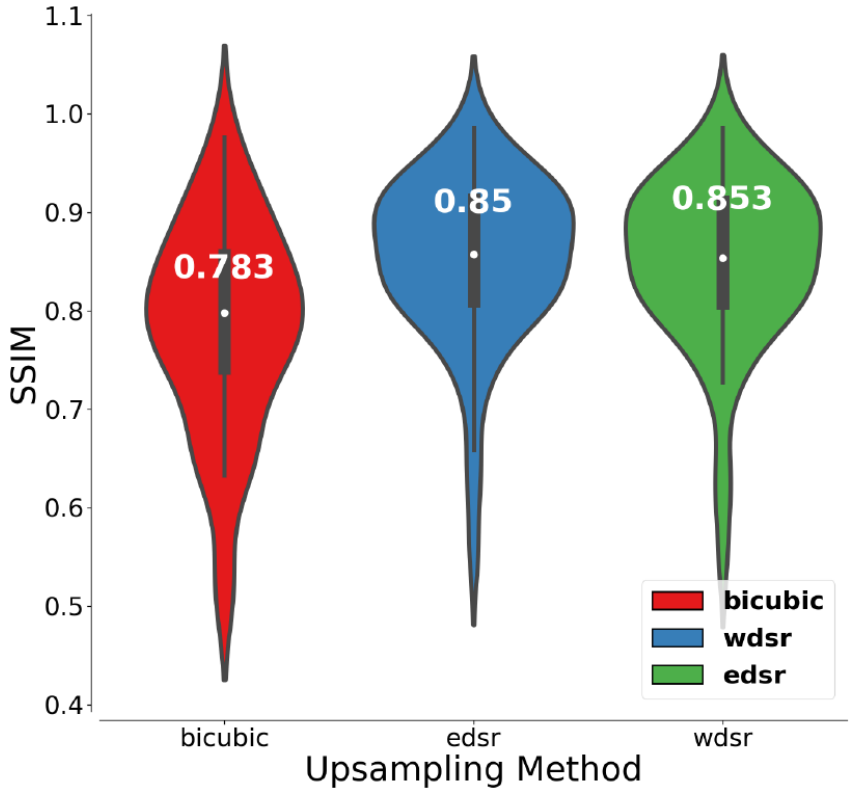
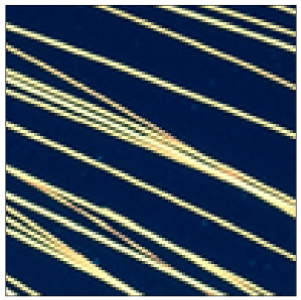


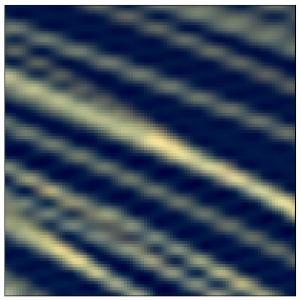
Image Results

Single Image Super Resolution

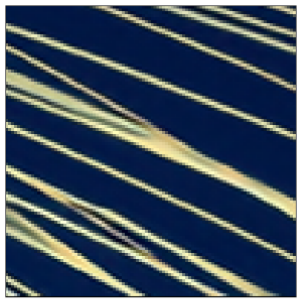
Original image 0828 from DIV2K



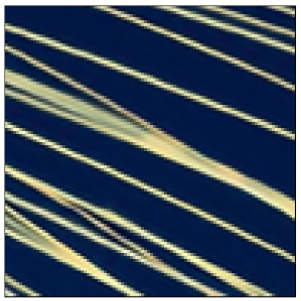
HR
(PSNR/SSIM)



Bicubic
(16.5 dB/0.656)



WDSR
(22.72 dB/0.926)



EDSR
(23.0 dB/0.926)

The model is trained on the **DIV2K**(*DIVerse 2K resolution high quality images*) dataset.
The dataset contains 800 high-resolution images as training set and their corresponding low-resolution ones, obtained by different down-sampling methods and different scale factors (2, 3, and 4).

Image Results

Single Image Super Resolution

Original image 0861 from DIV2K

HR
(PSNR/SSIM)

Bicubic
(21.23 dB/0.648)

WDSR
(23.17 dB/0.776)

EDSR
(23.36 dB/0.783)

The model is trained on the **DIV2K**(*DIVerse 2K resolution high quality images*) dataset.
The dataset contains 800 high-resolution images as training set and their corresponding low-resolution ones, obtained by different down-sampling methods and different scale factors (2, 3, and 4).

Image Results

Single Image Super Resolution

Original image 0845 from DIV2K

HR
(PSNR/SSIM)

Bicubic
(21.5 dB/0.669)

WDSR
(24.58 dB/0.801)

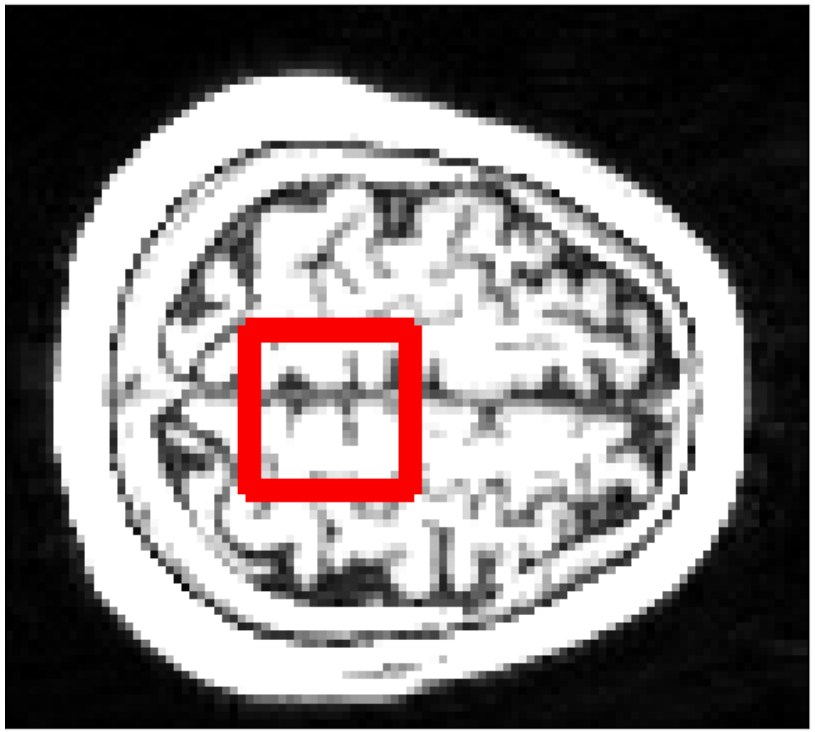
EDSR
(25.04 dB/0.811)

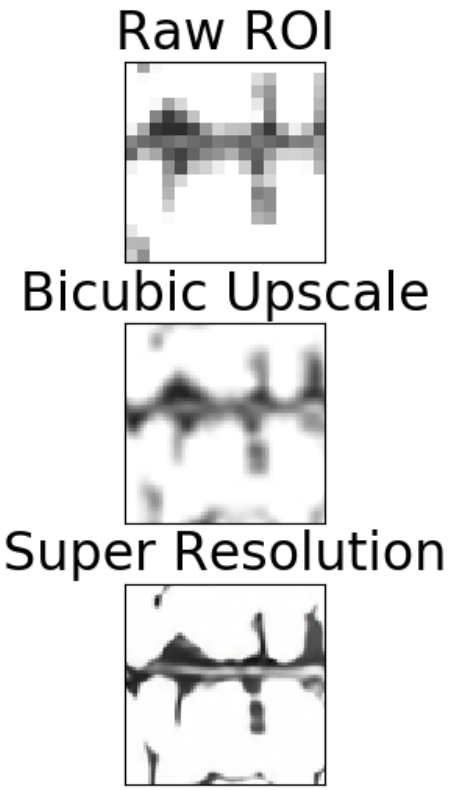
The model is trained on the **DIV2K**(*DIVerse 2K resolution high quality images*) dataset.
The dataset contains 800 high-resolution images as training set and their corresponding low-resolution ones, obtained by different down-sampling methods and different scale factors (2, 3, and 4).

Model Extrapolation

NMR Application

Test on T1 weighted NMR images ($256 \times 256 \rightarrow$ **REAL TIME**).
The images were down-sampled ($128 \times 128 \rightarrow$ 2x and $64 \times 64 \rightarrow$ 4x) and then re-up-sampled.

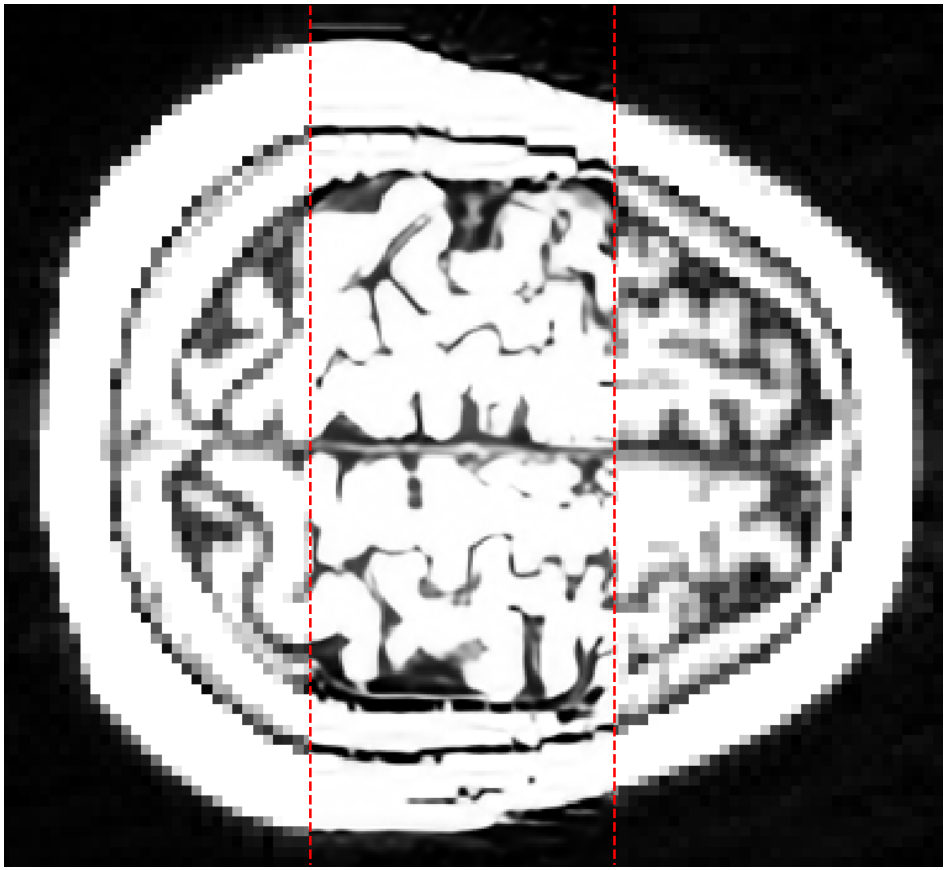




Model Extrapolation

NMR Application

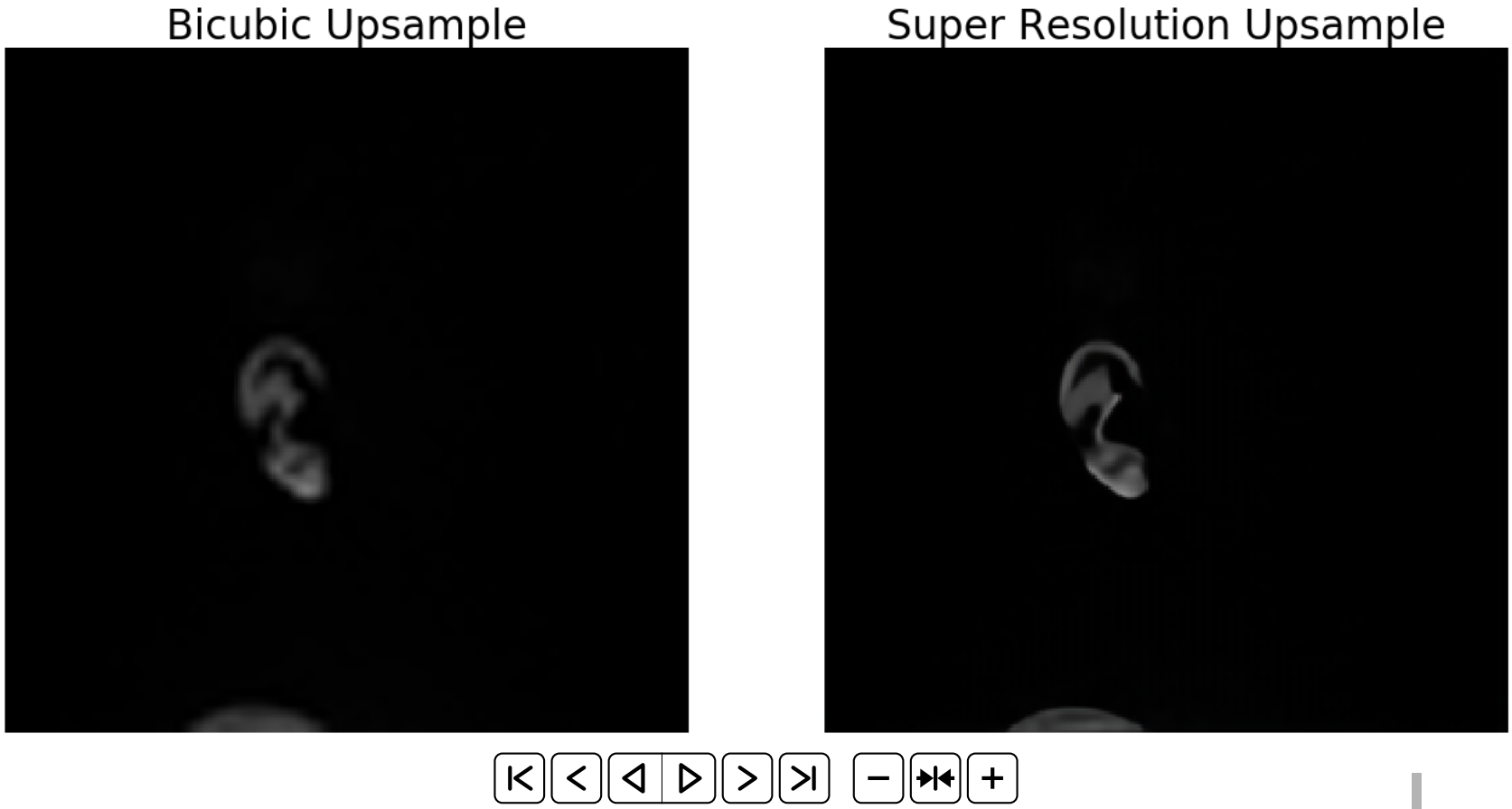
Test on T1 weighted NMR images ($256 \times 256 \rightarrow$ **REAL TIME**).
The images were down-sampled ($128 \times 128 \rightarrow$ 2x and $64 \times 64 \rightarrow$ 4x) and then re-up-sampled.



Visual score

NMR Application

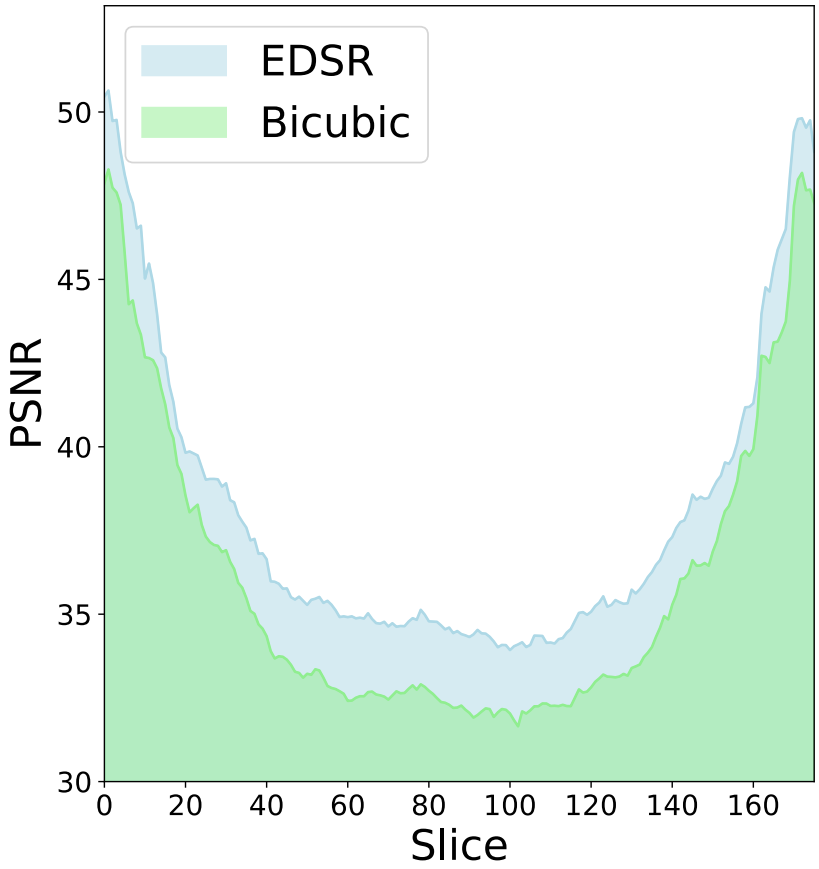
Test on T1 weighted NMR images ($256 \times 256 \rightarrow$ **REAL TIME**).
The images were down-sampled ($128 \times 128 \rightarrow$ 2x and $64 \times 64 \rightarrow$ 4x) and then re-up-sampled.

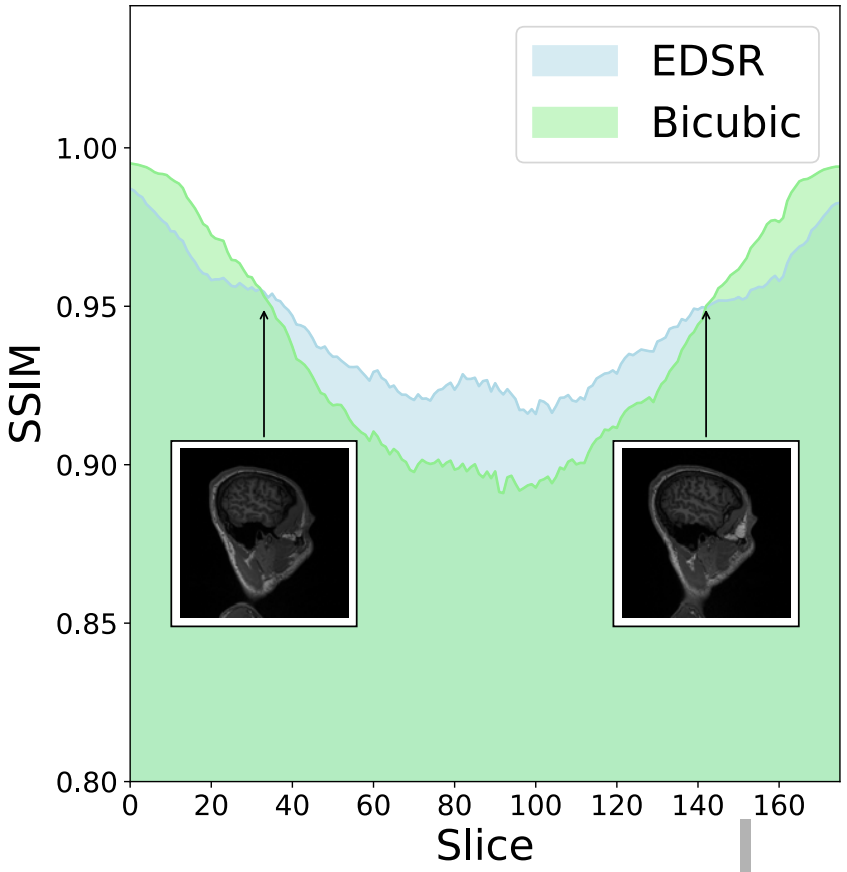


Quality score

NMR Application

Test on T1 weighted NMR images ($256 \times 256 \rightarrow$ **REAL TIME**).
The images were down-sampled ($128 \times 128 \rightarrow$ 2x and $64 \times 64 \rightarrow$ 4x) and then re-up-sampled with SR models.

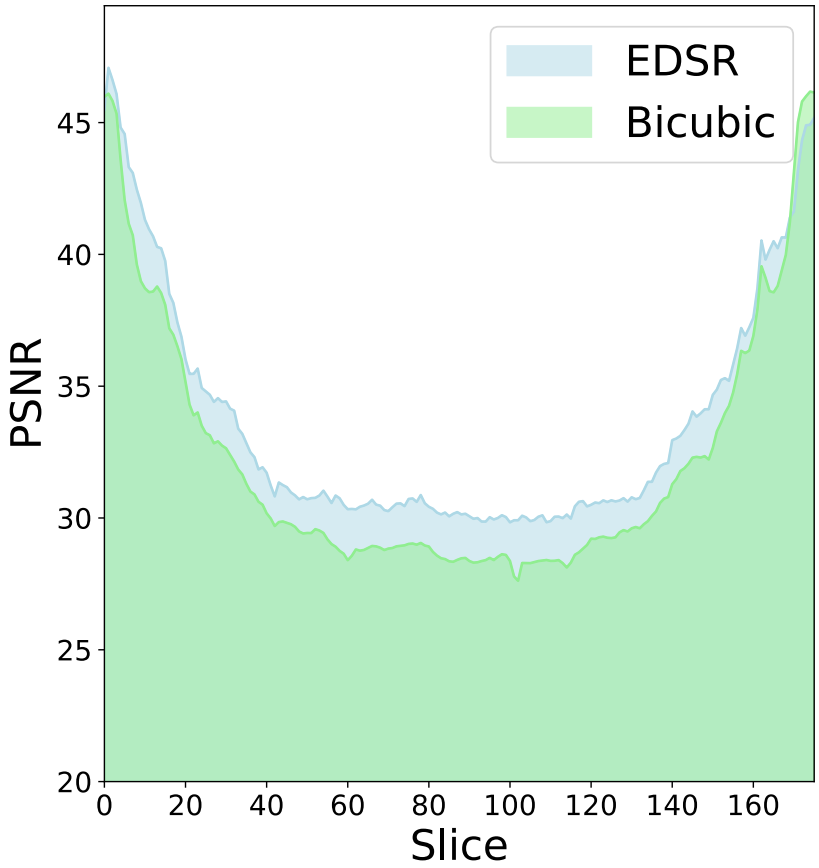


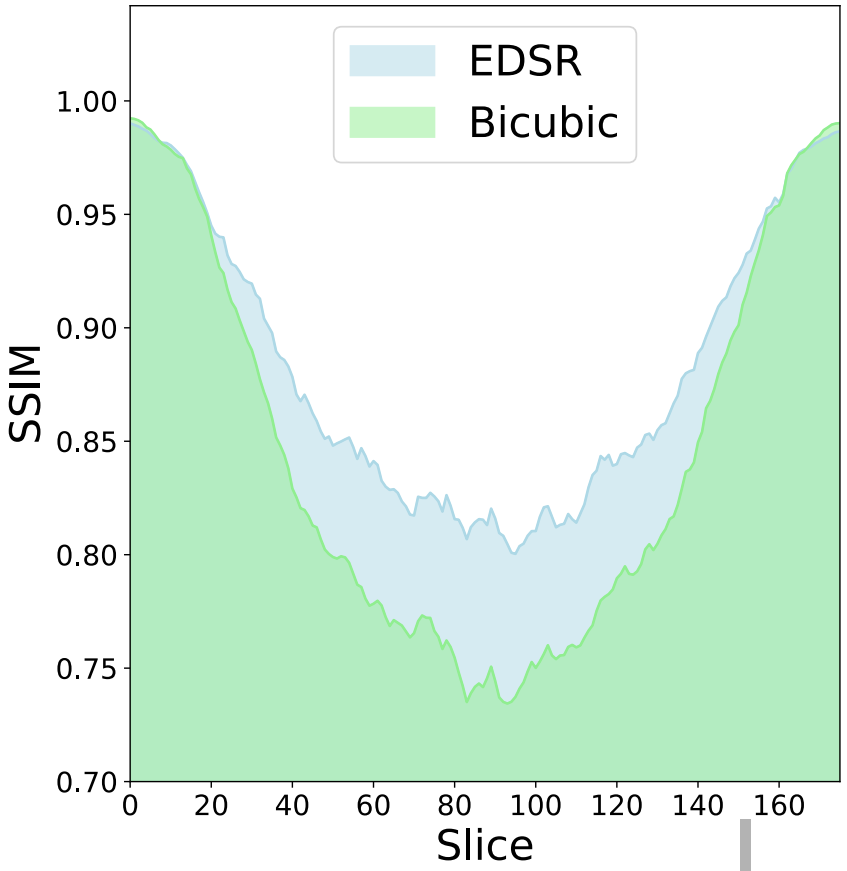


Quality score

NMR Application

Test on T1 weighted NMR images ($256 \times 256 \rightarrow$ **REAL TIME**).
The images were down-sampled ($128 \times 128 \rightarrow$ 2x and $64 \times 64 \rightarrow$ 4x) and then re-up-sampled with SR models.

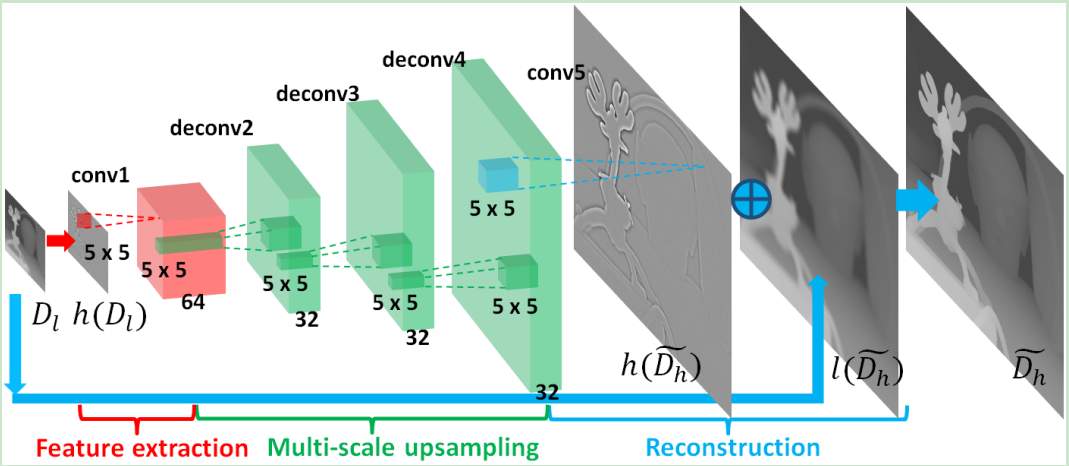




Applications

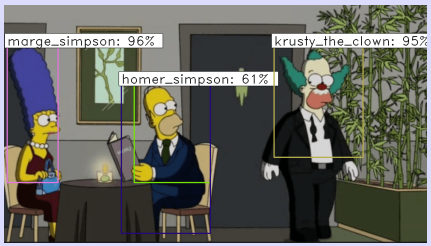
Deep Learning model tasks

Single Image Super Resolution



Improve spatial image resolution and reconstruction from low-resolution (LR) to high-resolution (HR).

Object Detection



Identify single or multiple objects into a picture or video stream.

Image Segmentation

Extract the exact pixels which belong to an object into a given image.

Object Detection

What is Object Detection?

Object detection is one of the larger deep learning sub-discipline, especially when we talk about Neural Network models.
This kind of problems aim to identify single or multiple objects into a picture or video stream.

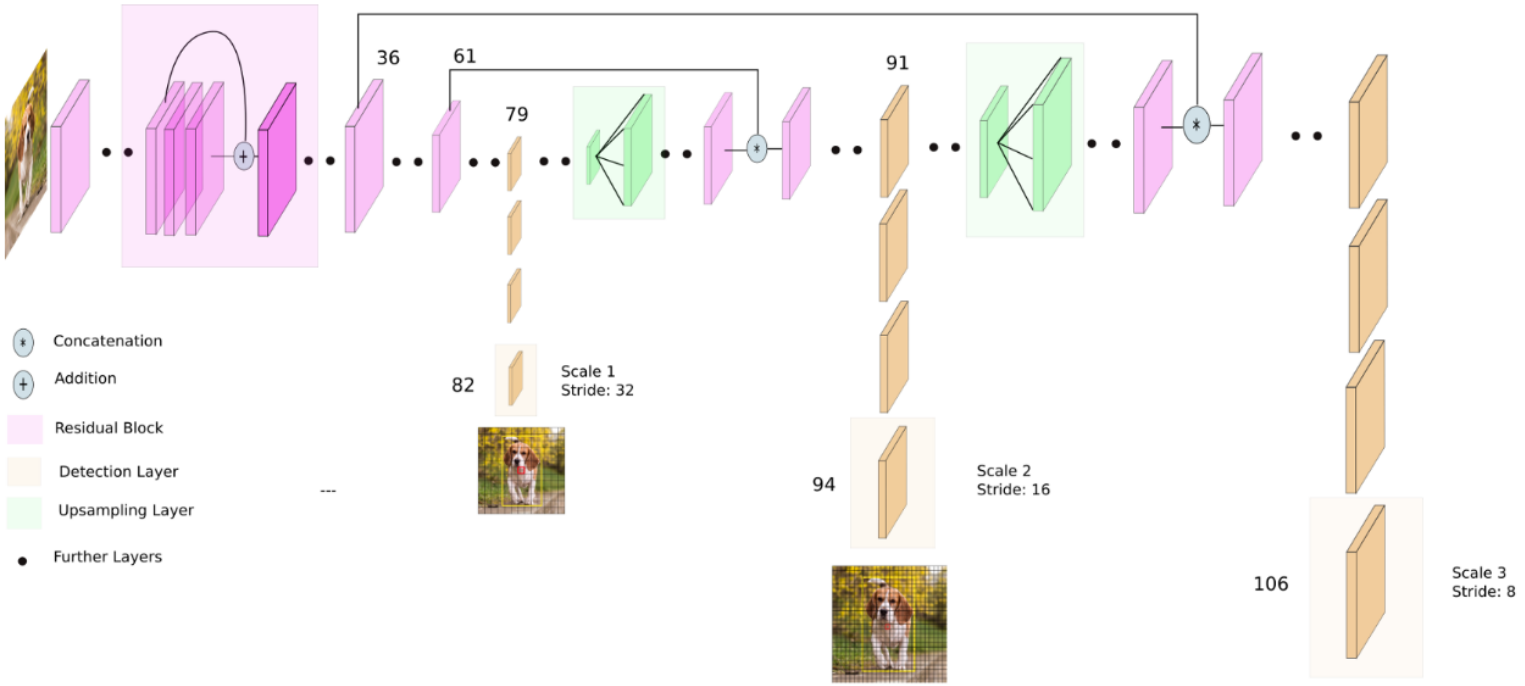
Possible applications:

- **Object tracking**;
- Video surveillance;
- **Pedestrian detection**;
- Anomaly detection;
- **People counting**;
- Self-driving cars;
- Face detection;
- . . . ;

YOLO - You Only Look Once

Object Detection Model

YOLO is a deep Neural Network model with more than 100 layers and more than 62 million of parameters.
The original implementation of the model is provided by Joseph Redmon (https://github.com/pjreddie/darknet) into the darknet project (ANSI C).

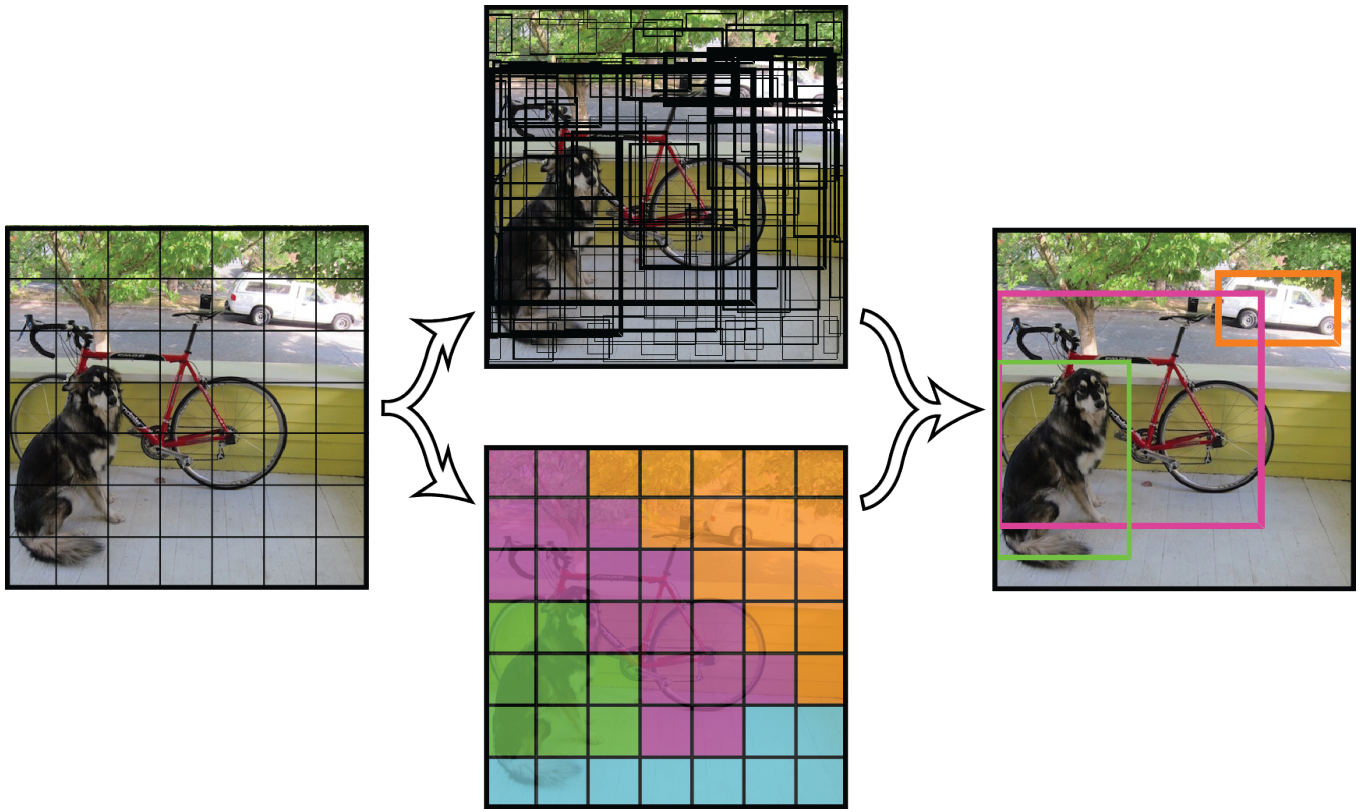


YOLO v3 network Architecture

YOLO - You Only Look Once

Object Detection Model

YOLO is a deep Neural Network model with more than 100 layers and more than 62 million of parameters.
The original implementation of the model is provided by Joseph Redmon (https://github.com/pjreddie/darknet) into the darknet project (ANSI C).

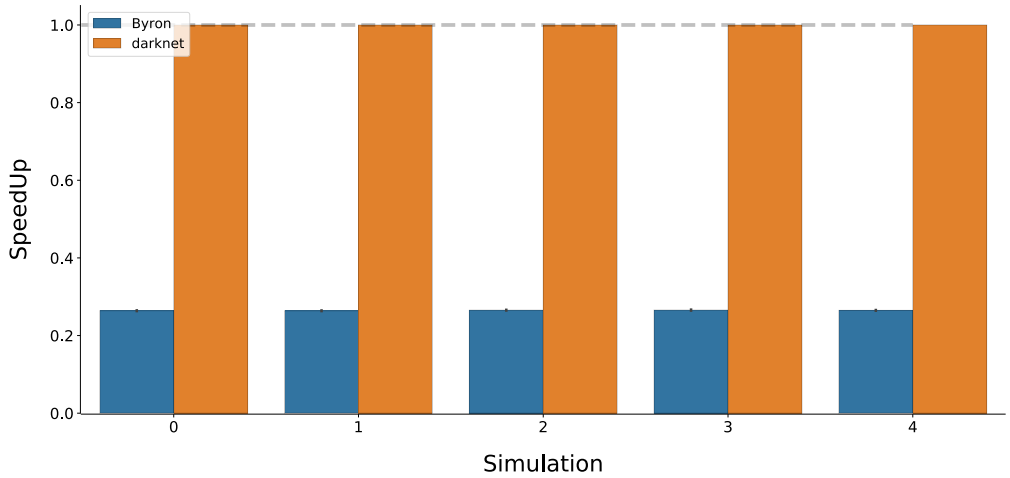


Computational Performances

Object Detection Models

Byron is inspired to darknet and its first implementation was about object detection task.
darknet → benchmark → but its efficiency is based on GPU support!

Evaluation performed on 100 runs without multi-threading!



Framework	Average Time (s)
Byron	5.18
darknet	19.58

Time evaluated on 416×416 images.
Environment: Bioinformatics server grade machine (128 GB RAM memory and 2 CPU E5-2620, with 8 cores each.

Model Extrapolation

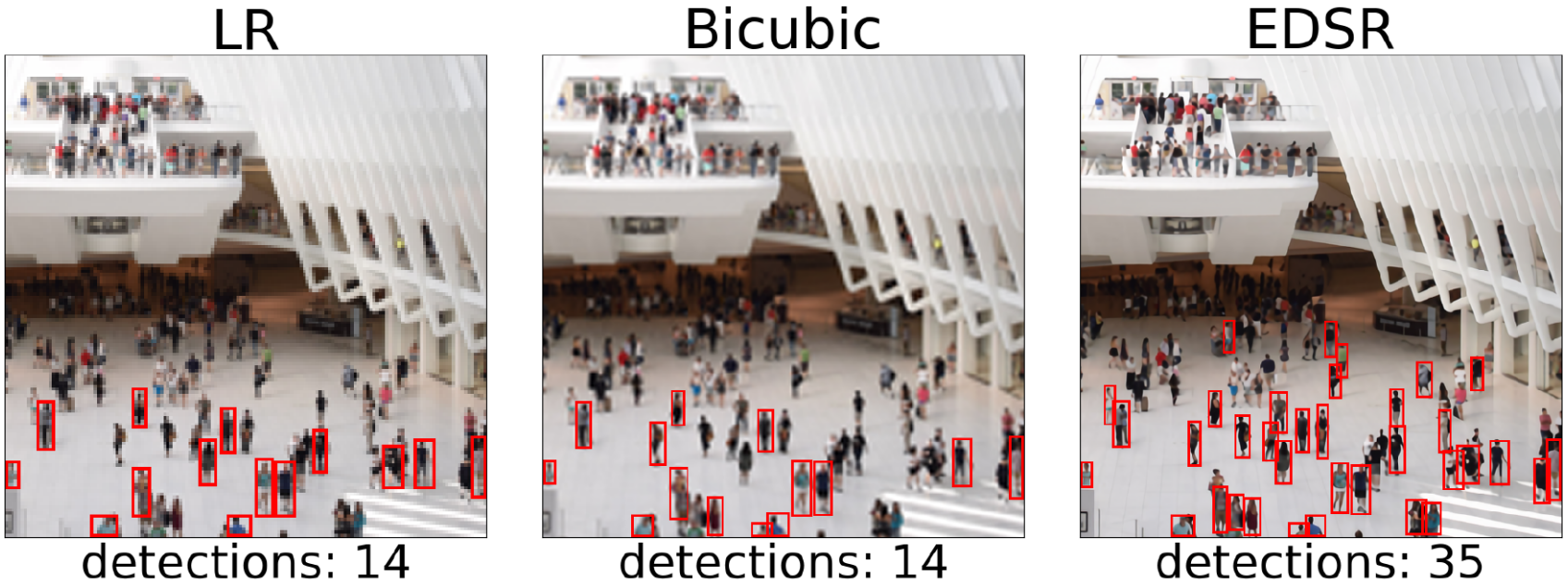
People counting Application

Despite the model is incredibly efficient in object detection also with low quality images, there is a sort of **limit** in the number of pixels needed for object identification. We had the opportunity to empirically verify its limit working on a people tracking project for real time applications.
The project was developed in collaboration with the Complex Systems (**PhySyCom**) group.

Model Extrapolation

People counting Application

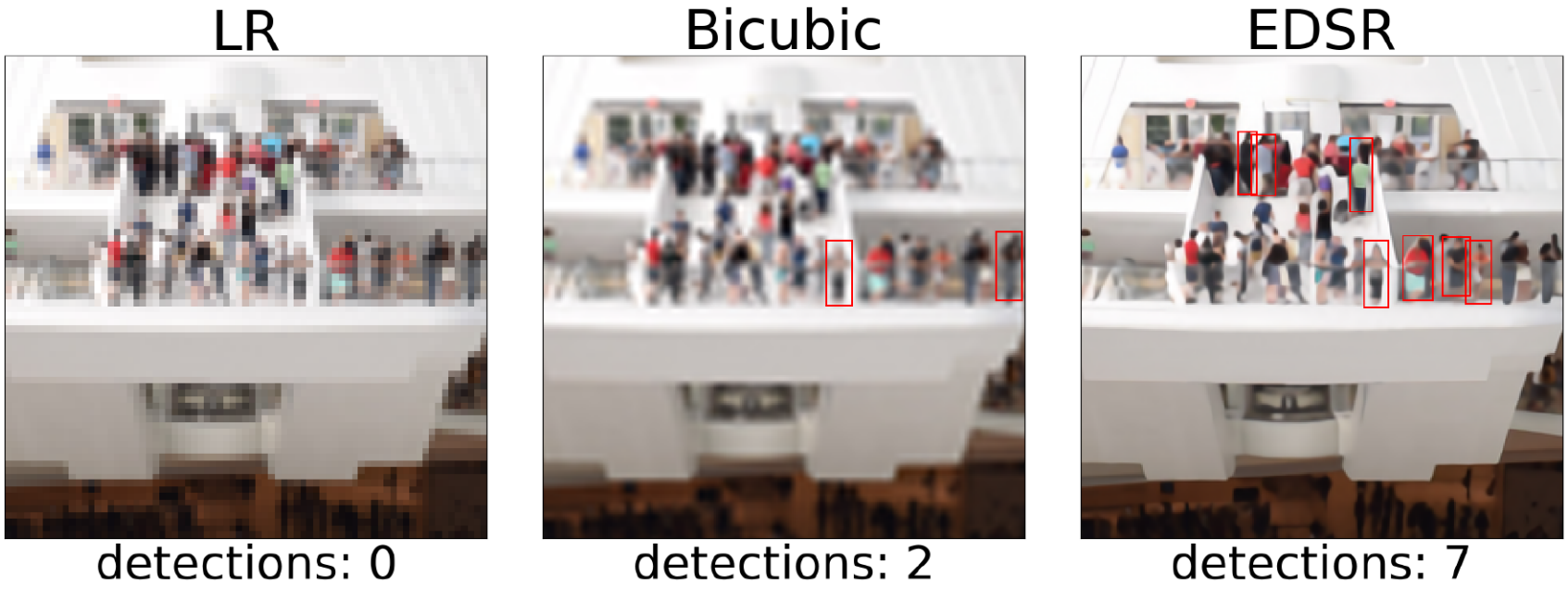
Despite the model is incredibly efficient in object detection also with low quality images, there is a sort of **limit** in the number of pixels needed for object identification. We had the opportunity to empirically verify its limit working on a people tracking project for real time applications.
The project was developed in collaboration with the Complex Systems (**PhySyCom**) group.



Model Extrapolation

People counting Application

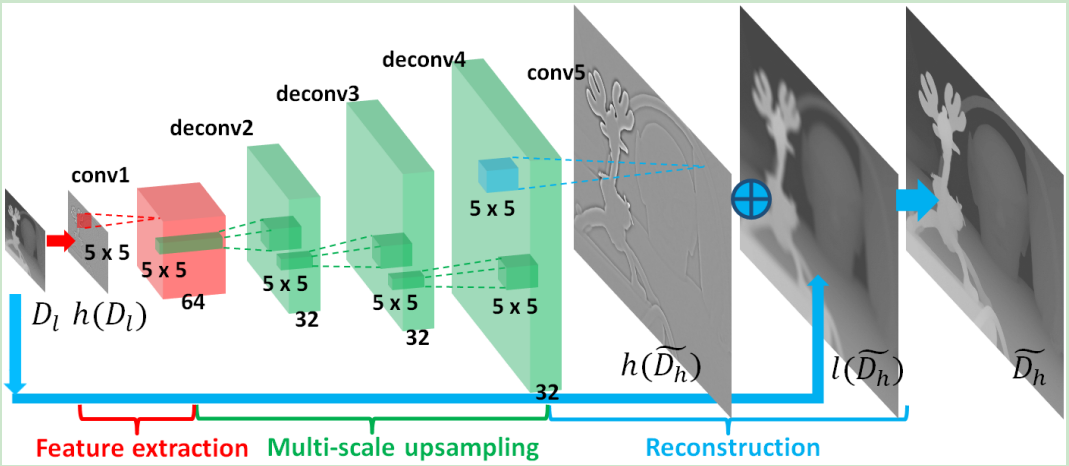
Despite the model is incredibly efficient in object detection also with low quality images, there is a sort of **limit** in the number of pixels needed for object identification. We had the opportunity to empirically verify its limit working on a people tracking project for real time applications.
The project was developed in collaboration with the Complex Systems (**PhySyCom**) group.



Applications

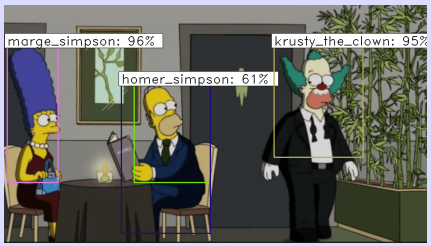
Deep Learning model tasks

Single Image Super Resolution



Improve spatial image resolution and reconstruction from low-resolution (LR) to high-resolution (HR).

Object Detection



Identify single or multiple objects into a picture or video stream.

Image Segmentation

Extract the exact pixels which belong to an object into a given image.

Image Segmentation

What is Image Segmentation?

Object Detection: give a label to **ROIs** of the input image.
Image Segmentation: give a label to **each pixel** of the input image.

3D Femur Structure

Rizzoli Hospital project

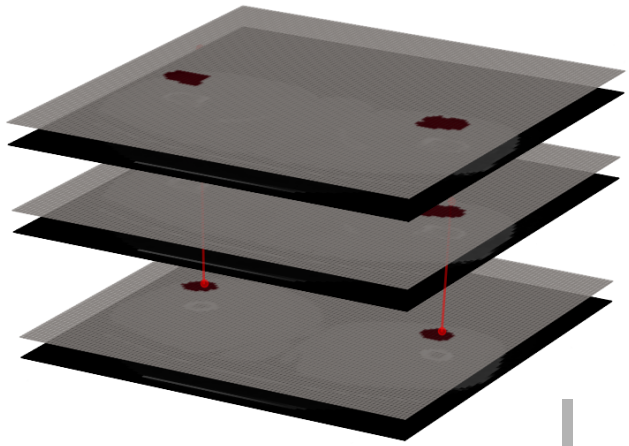
Improve the identification and segmentation of the femoral head, trying to discriminate this part of the bone from the articular cartilage and moreover from the acetabular fossa.

Data type: CT (*Computer Tomography*) images.

Dataset: 4 (not-annotated) patients.

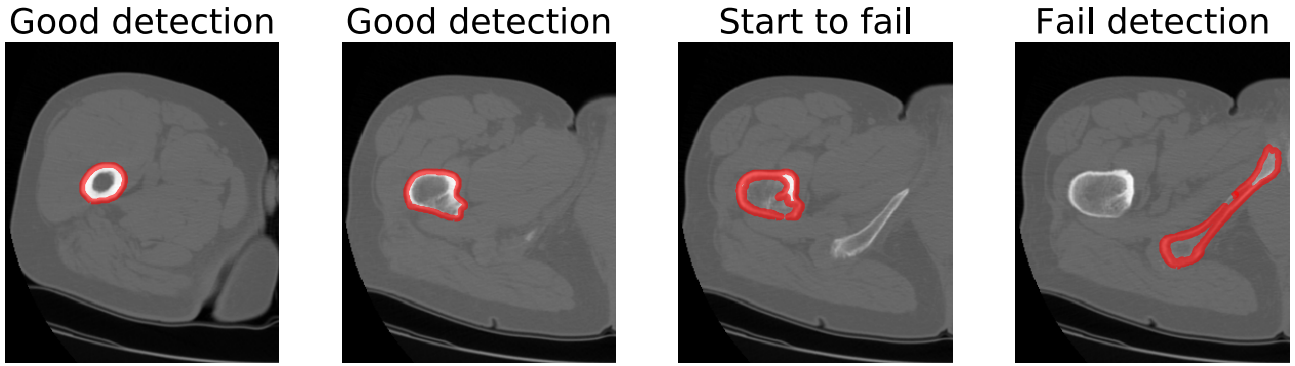
Semi-automatic pipeline:

- Edge enhancement;
- Thresholding;
- Morphological operations;
- Connected Components;
- Interpolation between consecutive frames;

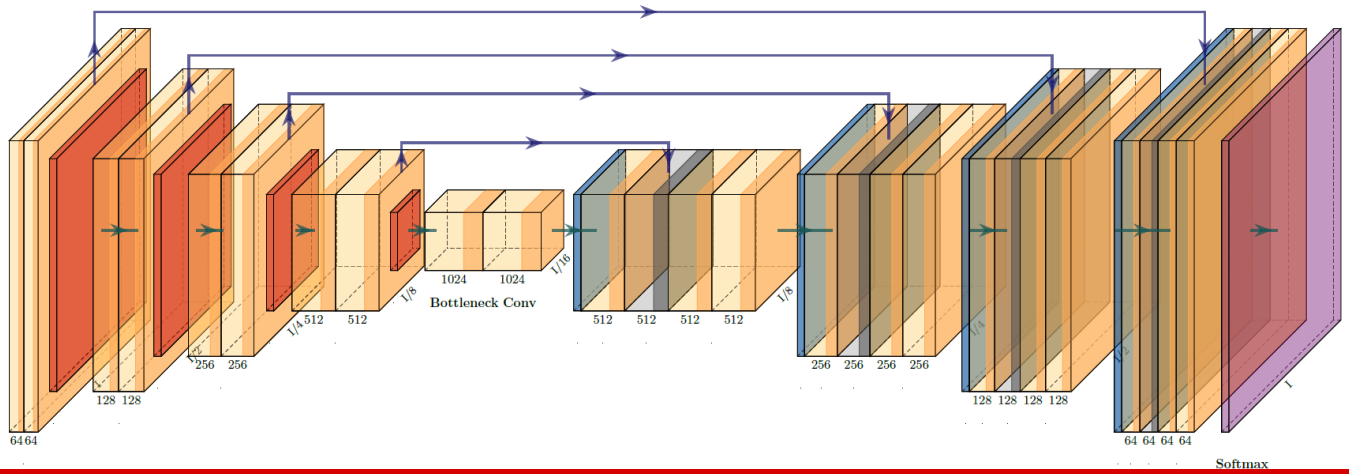


Segmentation Models

The method is too naive to perform a good segmentation on the full set of slices. However, it can be useful to **reduce** the quantity of slices to annotate manually ($\sim$ 400 slices per patient)!



Thus, we moved to a **Neural Network model**

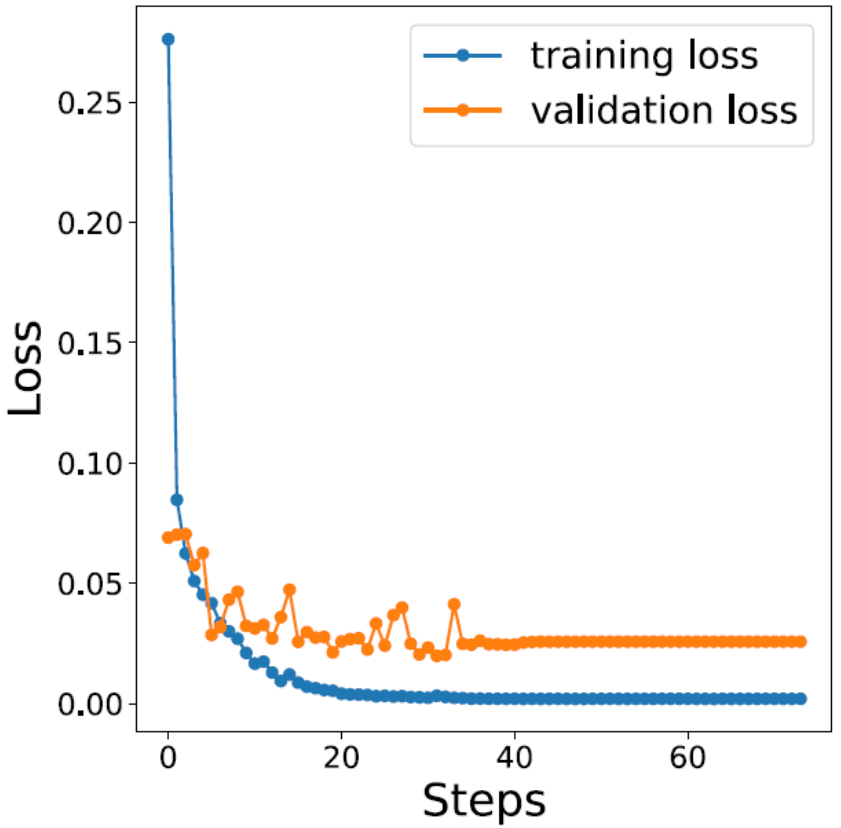


U-Net Model

Segmentation Models

Training parameters:

- Training Set: 96 images;
- Test Set: 40 images;
- Data augmentation: Rotated, shifted, mirrored.
- 40 epochs;
- loss: binary cross-entropy
- metrics: IoU (*Intersection over Union*)

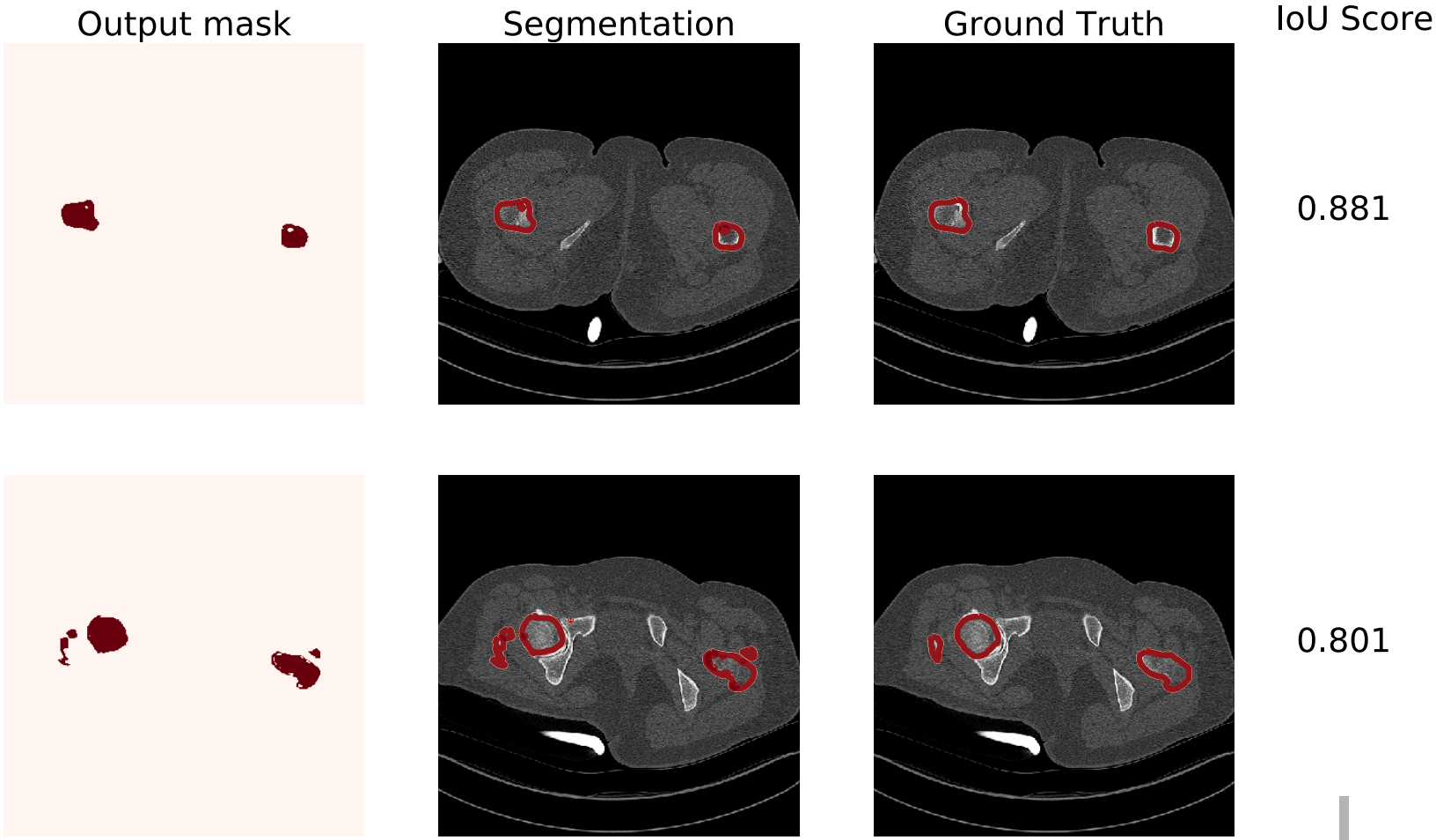


IoU Validation Set:
The 80% of the test set has obtained a IoU score greater than 0.8 and thus a good correspondence between our results and the ground truth.

IoU score

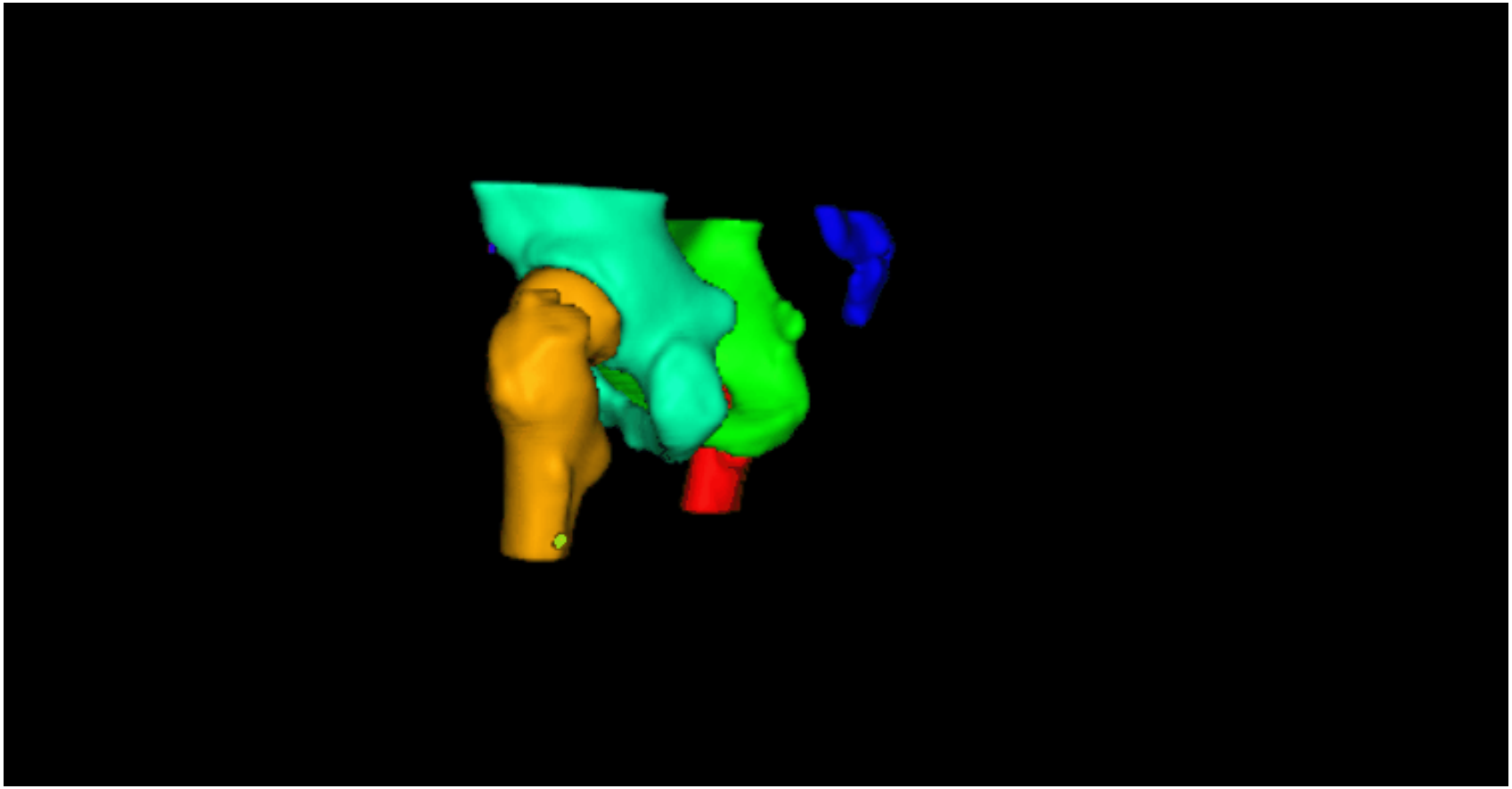
Segmentation Models

The model output were filtered using a thresholding of 10^{-2}, i.e values $\leq$ were turned off.



3D Reconstruction

Segmentation Models



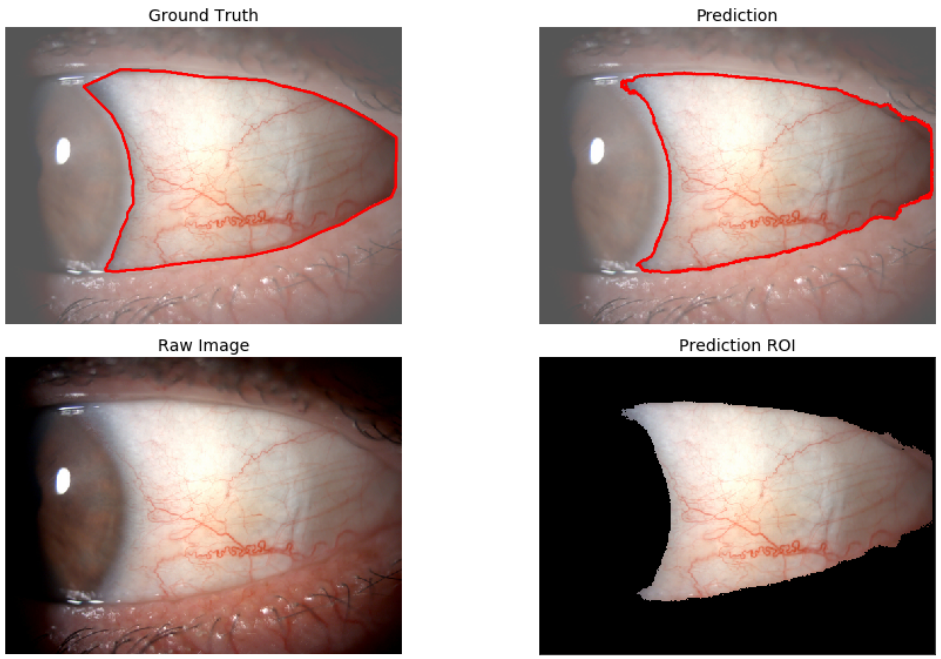
ROSSO project

ROSSO - Robotic Ocular Semantic Segmentation for Ophthalmology

Sant'Orsola Hospital project: Extract the pixels belonging to sclera for the quantification of the red part of the image.

All the images have been manually annotated by experts. Dataset:

- Training: 150 images;
- Test: 18 images;



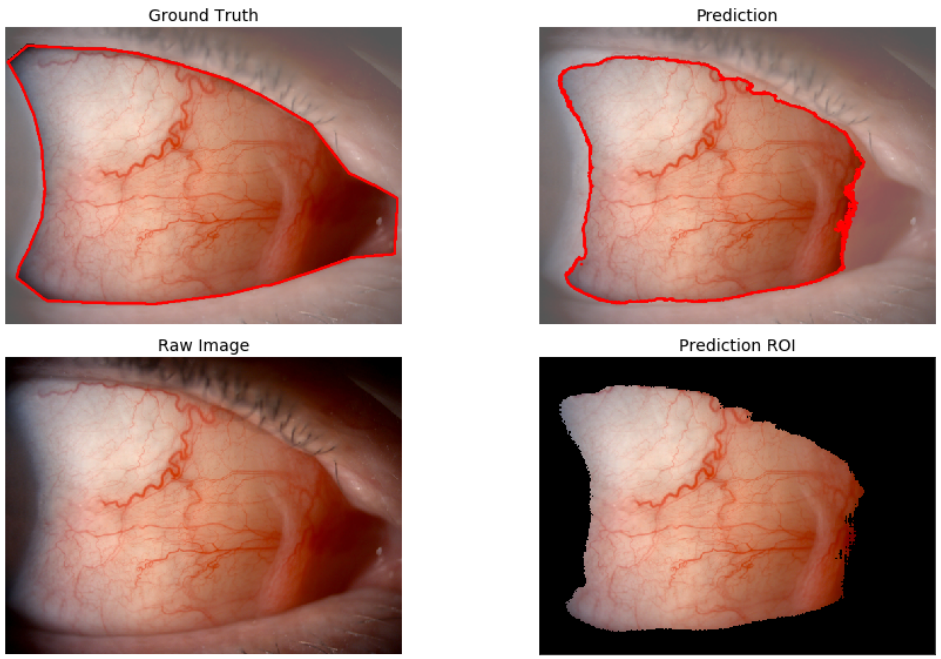
ROSSO project

ROSSO - Robotic Ocular Semantic Segmentation for Ophthalmology

Sant'Orsola Hospital project: Extract the pixels belonging to sclera for the quantification of the red part of the image.

All the images have been manually annotated by experts. Dataset:

- Training: 150 images;
- Test: 18 images;



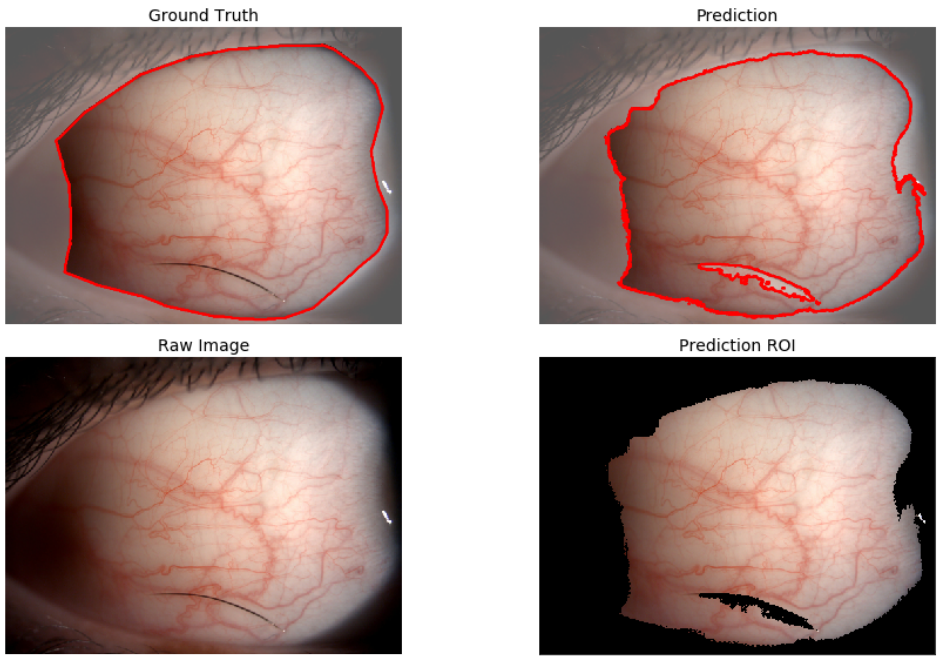
ROSSO project

ROSSO - Robotic Ocular Semantic Segmentation for Ophthalmology

Sant'Orsola Hospital project: Extract the pixels belonging to sclera for the quantification of the red part of the image.

All the images have been manually annotated by experts. Dataset:

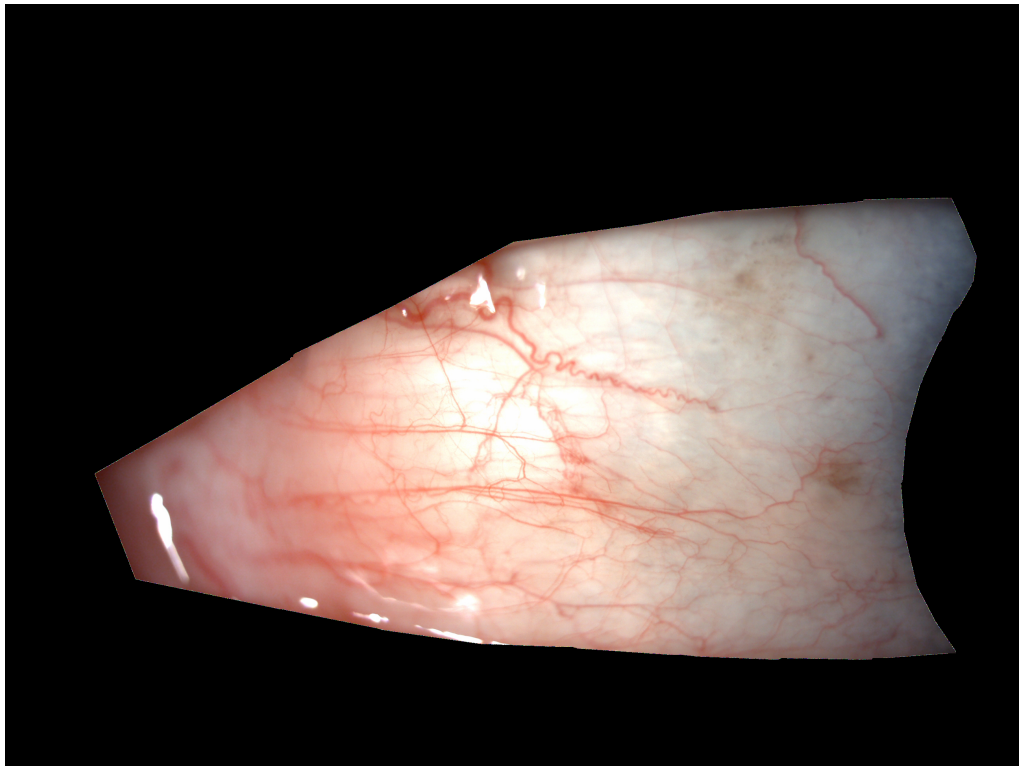
- Training: 150 images;
- Test: 18 images;



ROSSO project

ROSSO - Robotic Ocular Semantic Segmentation for Ophthalmology

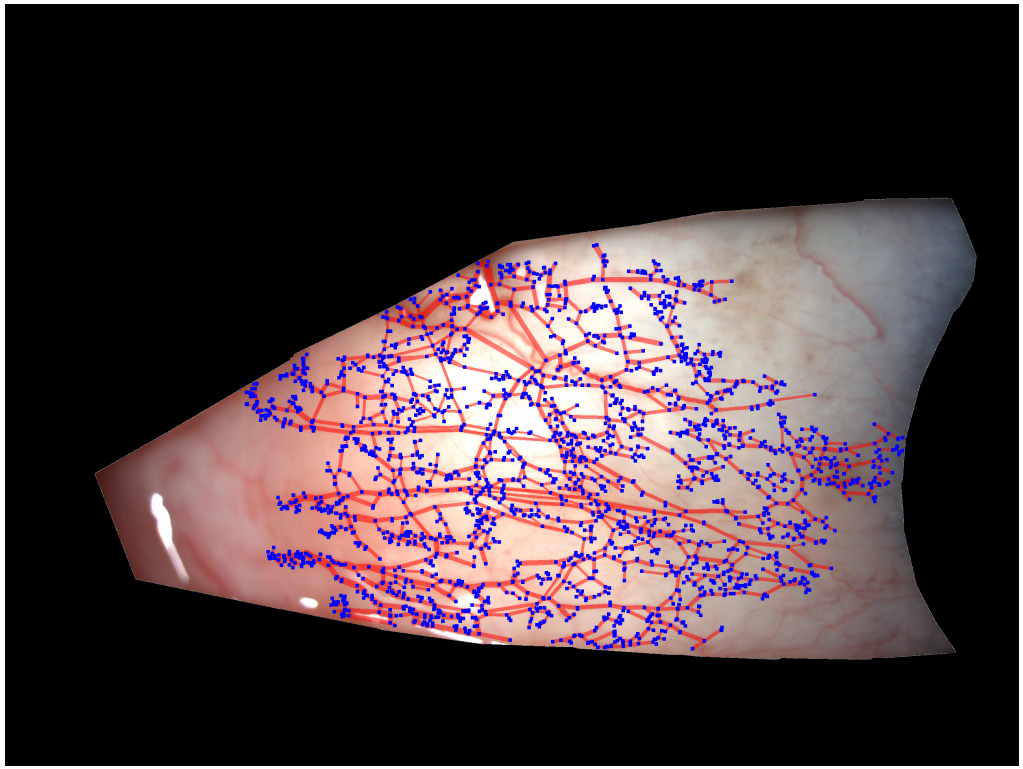
Sant'Orsola Hospital project: Extract the vascular network from the eye's sclera component.



ROSSO project

ROSSO - Robotic Ocular Semantic Segmentation for Ophthalmology

Sant'Orsola Hospital project: Extract the vascular network from the eye's sclera component.



Conclusion

- Two new libraries for Deep Learning applications were proposed:
  - NumPyNet: a tool for educational practice and modeling;
  - Byron: a tool for an efficient implementation of these models;
- Both libraries implement cutting edge numerical techniques to maximize computation efficiency;
- Byron optimize and extend the darknet project with several fixes and computational improvements.
- Byron is able to join in a single framework either super resolution and object detection tasks.
- The results obtained using Super Resolution and Segmentation models are very promising for the analysis of biomedical images.
- We have shown that deep learning models are capable of a very efficient generalization due to their vast amount of parameters and a well-programmed training section.
- The Byron YOLO version is approximately 3.8x faster than darknet in all the simulations;

Project References

https://github.com/Nico-Curti

Byron: Build YouR Own Neural Network, http://github.com/Nico-Curti/Byron

NumPyNet: Neural Network library in Pure Numpy, http://github.com/Nico-Curti/NumPyNet

DNetPRO: Discriminant Analysis with Network PROcessing, http://github.com/Nico-Curti/DNetPRO

rFBP: Replicated Focusing Belief Propagation algorithm, http://github.com/Nico-Curti/rFBP

Scorer: Machine Learning Scorer Library with parallel DAG support, http://github.com/Nico-Curti/scorer

FiloBluService: FiloBlu project service manager for text message processing, http://github.com/Nico-Curti/FiloBluService

ShUt: Shell Utilities and Installers for no root users, http://github.com/Nico-Curti/shut

genetic: Examples about genetic algorithms for parallel and distributed computing, http://github.com/Nico-Curti/genetic

BlendNet: Network viewer with Blender support, http://github.com/Nico-Curti/BlendNet

CryptoSocket: TCP/IP Client Server with RSA cryptography, http://github.com/Nico-Curti/CryptoSocket

easyDAG: Simple template DAG scheduler in C++, http://github.com/Nico-Curti/easyDAG

Cardio: Pulse oximetry data processing and classification, http://github.com/Nico-Curti/cardio

SysDyn: System Dynamics script utilities, http://github.com/Nico-Curti/SysDyn

rSGD: Replicated Stochastic Gradient Descent algorithm, http://github.com/Nico-Curti/rSGD

Walkers: Random Walk and Optimizer Simulator, http://github.com/Nico-Curti/Walkers

Data-Analysis: Data Analysis Utilities (from Statistics to Machine Learning), http://github.com/Nico-Curti/Data-ANalysis

Thanks to the research group