Vessel Extraction from Retinal Images

Ariane ALIX - Baptiste DEHAINE
Master MVA École Normale Supérieure Paris-Saclay

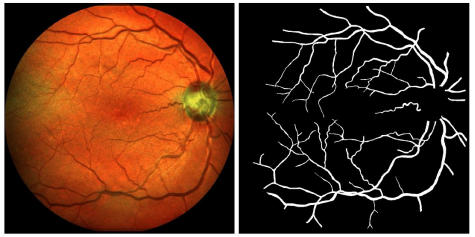
Summary

- Introduction
- Problem definition
- Related work
- Methodology
- Evaluation
- Discussion

Introduction

- **Problem**: segmentation of retinal blood vessels (DRIVE dataset, 40 images).
- **Our work**: development of a new method combining Shin's "Deep vessel segmentation by learning graphical connectivity" (2018), and Li's "IterNet" network (2019).
- **Why it is important**: the study of the length, width, tortuosity etc. can help diagnose cardiovascular diseases like diabetes or hypertension.
- **Other possible applications**: segment all types of networks or vessel-likes structures in pictures, like roads on satellite images, or cracks in structures.

Problem definition



(a) Retinal fundus image (b) Corresponding vessel segmentation.

- **Notation**: Let us note *X* the input image, and $\{x_i, i = 1...N\}$ its *N* pixels.
- **Representation**: each pixel is labeled either ”vessel” (1, white) or ”non-vessel” (0, black
- **Final goal**: minimize the pixel-wise cross-entropy when predicting if a pixel is part of a vessel:

$$Loss(X) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p(x_i) + (1 - y_i)\log(1 - p(x_i)))$$

Related work

With Deep Learning:

- Ronneberger's "U-net" (state-of-the-art in 2015): introduced a U-shaped architecture (downward path then upward path) with connections between layers; it was re-used by numerous other approaches afterward,

- Seung Yeon Shin's "VGN" (state-of-the-art in 2018): novel approach combining CNN-based networks and Graph Neural Networks that learn the graphical connectivity of the vessels,

- Liangzhi Li's "IterNet" (state-of-the-art in 2019): consists of multiple iterations of a mini-UNet.

Our method is a combination of the last two

Methodology

- Architecture overview
- IterNet
- GNN
- Inference module

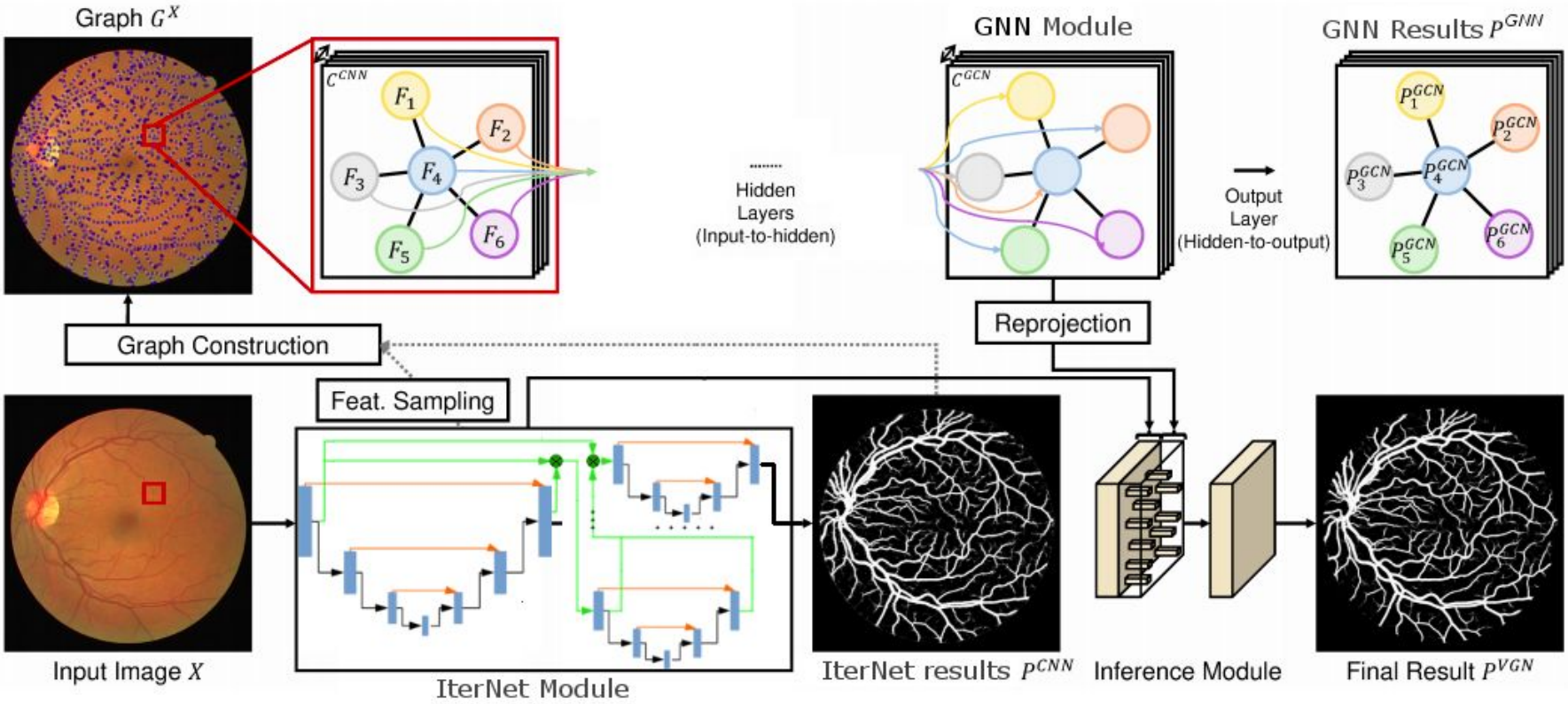
Architecture overview

3 main modules:

- IterNet (CNN-based): first segmentation optimizing on pixel-wise cross-entropy, replacing classical CNN in VGN's orginal version,
- Graph Neural Network (with attention): takes as input a graph G^X extracted from the IterNet segmentation results,
- Inference module (classical one-way downward CNN): takes the IterNet and the GNN results and deduce a final segmentation

Both VGN and IterNet available codes were in Tensorflow; we re-implemented everything in Pytorch.

Architecture overview

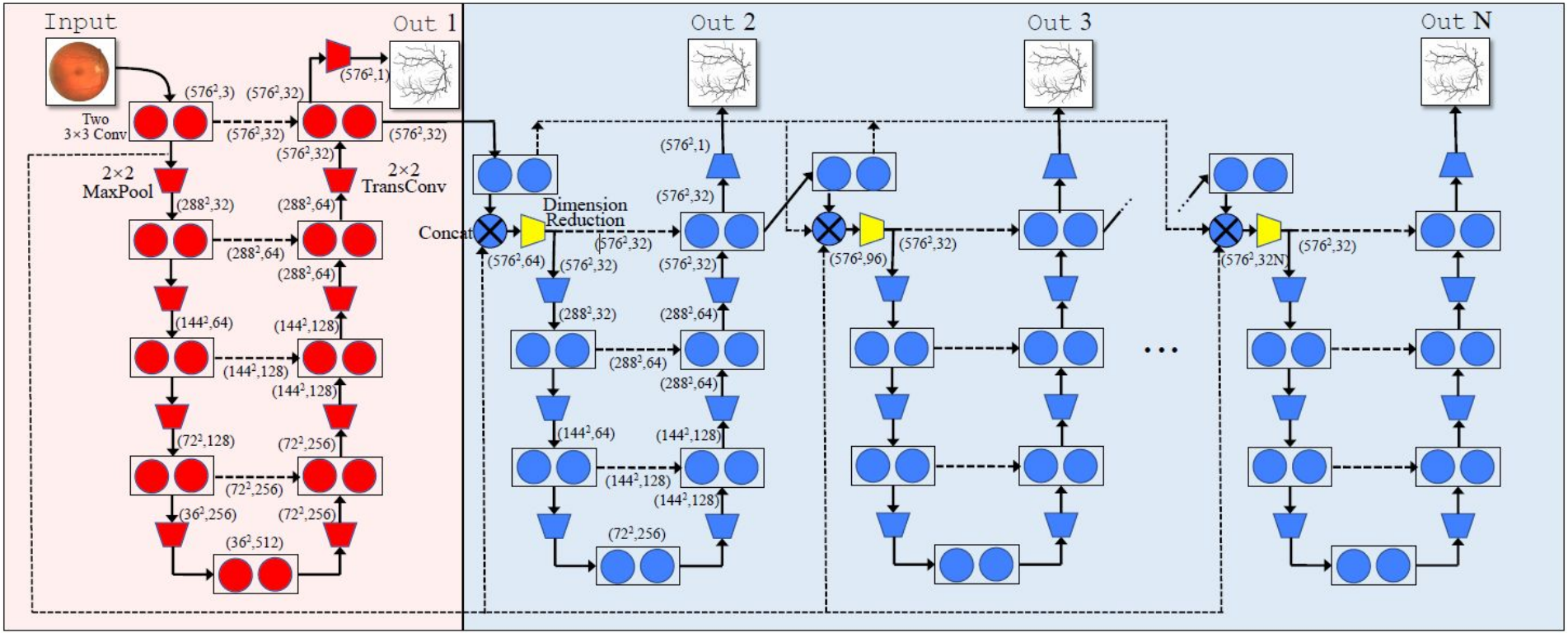


IterNet

Composed of a UNet, followed by N simplified versions called 'mini-UNets'.

- UNet: the base module (known for its great performances in segmentation),
- Mini-UNets: responsible for the refinement of small parts of the vessels, (ex: to infer missing pieces),
- Each mini-UNet uses the output of the previous module, and the concatenated features extracted after the first layers of all the previous modules (high-level information).

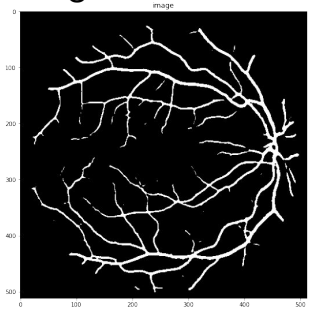
IterNet



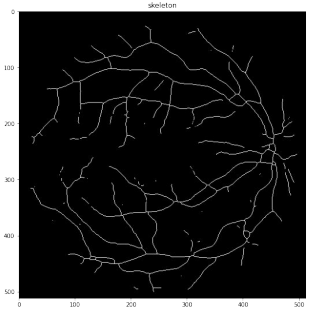
The structure of IterNet, which consists of one UNet and iterations of N-1 mini-UNets

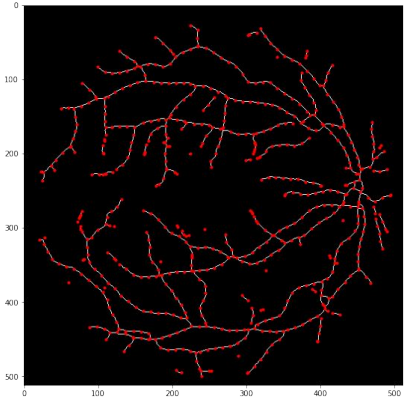
Graph Neural Network

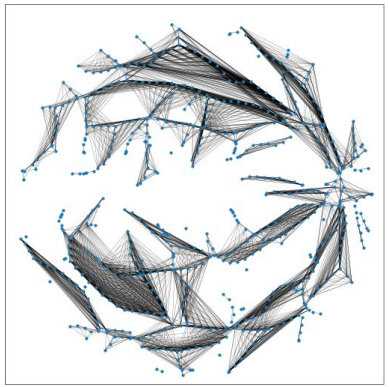
segmentation



skeleton







graph extracted

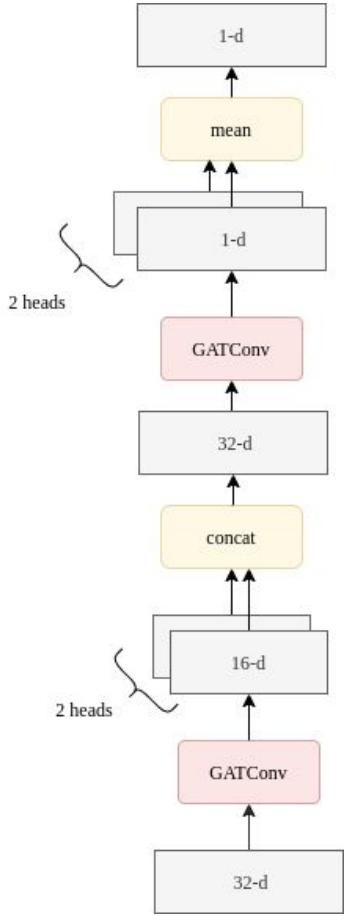


Diagram of the architecture.

Inference module

- Input: we concatenate CNN features and GNN features. Each features have 32 dimensions.
- Five convolutional layers, with kernel size 3*3. All intermediate layers have 32 hidden dimensions. Each layer except the last one is followed by a ReLu activation.

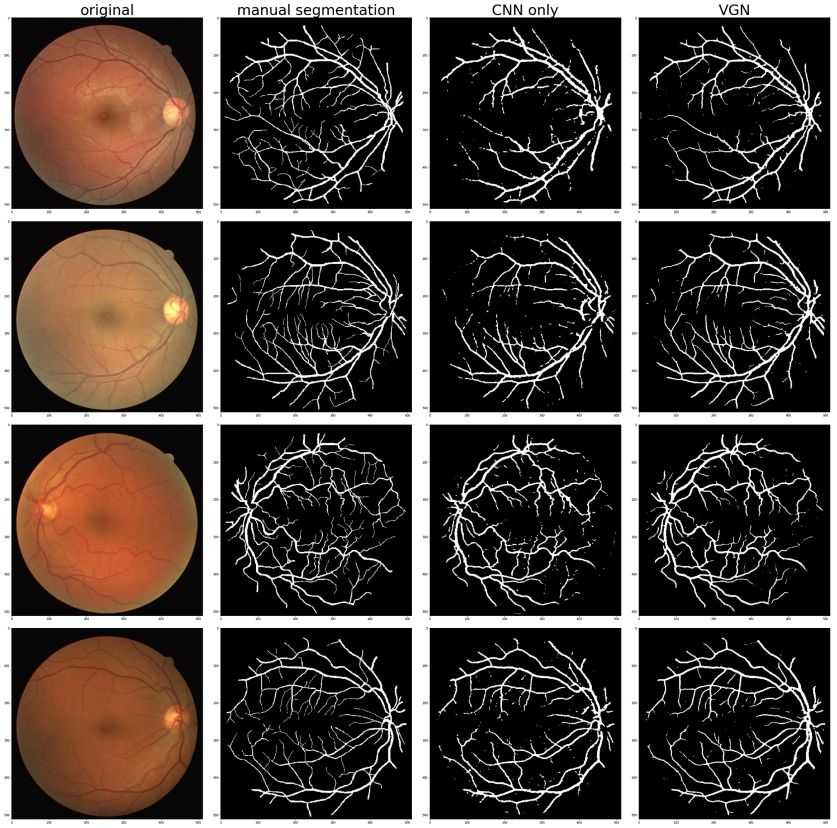
Final results and discussion

- Results on DRIVE dataset
- Discussion

Results on the DRIVE dataset

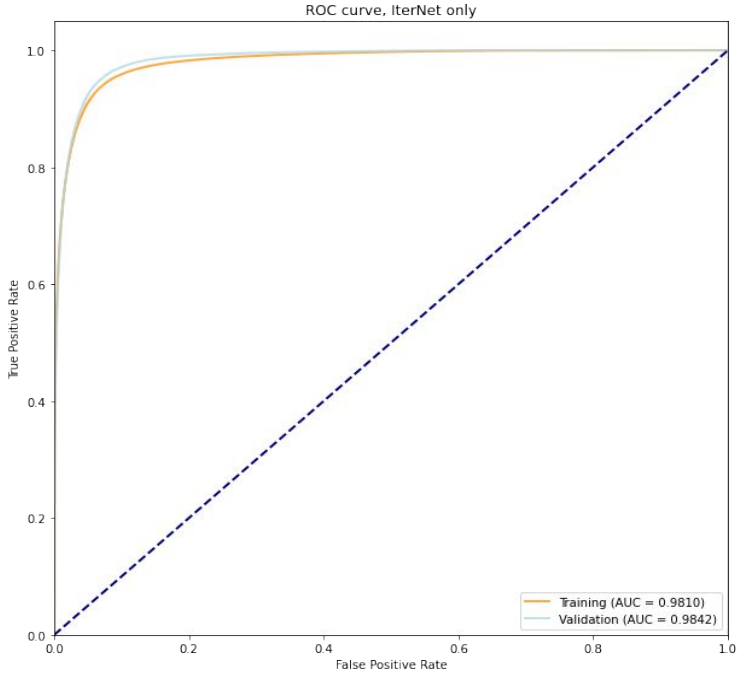
Our segmentation results on validation set, using only IterNet first and then with VGN.

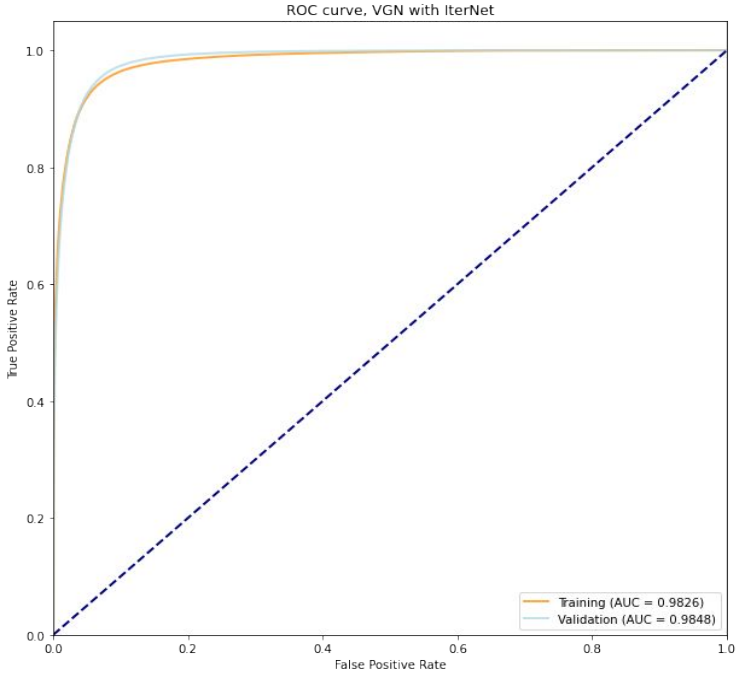
We can see that our VGN helps extract thinner vessel.



Results on the DRIVE dataset

	IterNet only		VGN with IterNet	
	Training	Validation	Training	Validation
AUC	0.9810	0.9842	0.9826	0.9848
F1-score	0.7914	0.7965	0.8062	0.7962





Potential improvements

- More computational power: more iterations of mini-UNet in the IterNet part (we were limited to 1)
- Data augmentation: as we have only 20 images, would help with overfitting (for ex: variations of color, shape, brightness, rotation...)

Thank you !