End-to-End Learning of Polygons for Remote Sensing Image Classification

Nicolas Girard and Yuliya Tarabalka

Inria Sophia-Antipolis Méditerranée, Titane team,
email: nicolas.girard@inria.fr

24th of July, 2018

Outline

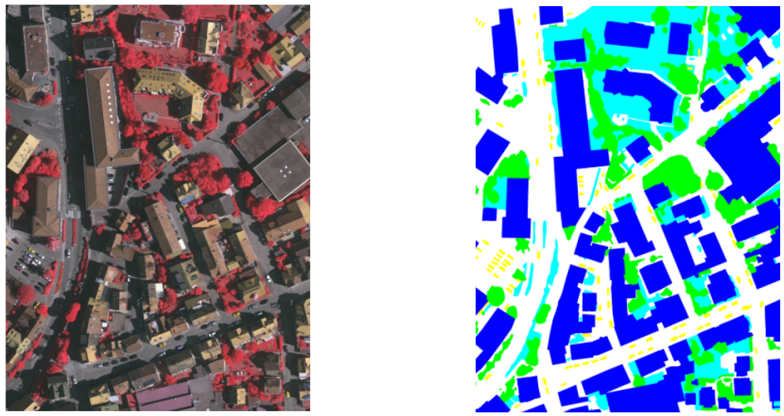
Context

Remote sensing applications:

- Urban planning
- Telecommunications
- Disaster management
- Search and rescue
- Surveying (animals, plants, etc)

(gisgeography.com)

Context



(ISPRS Vaihingen dataset)

- One big challenge in remote sensing:
  - Assignment of a thematic class to every pixel in a satellite or aerial image
  - Referred to as pixel-wise classification
- Important application of classification:
  - Integrate the data into geographic information systems (GIS)
  - Requires to represent the detected objects as polygons

Current Deep Learning approach to pixel-wise classification

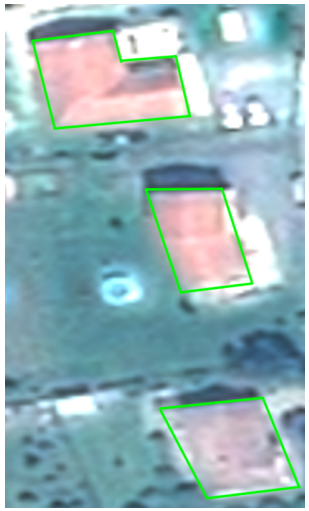


Image & GT

Class. map[1]

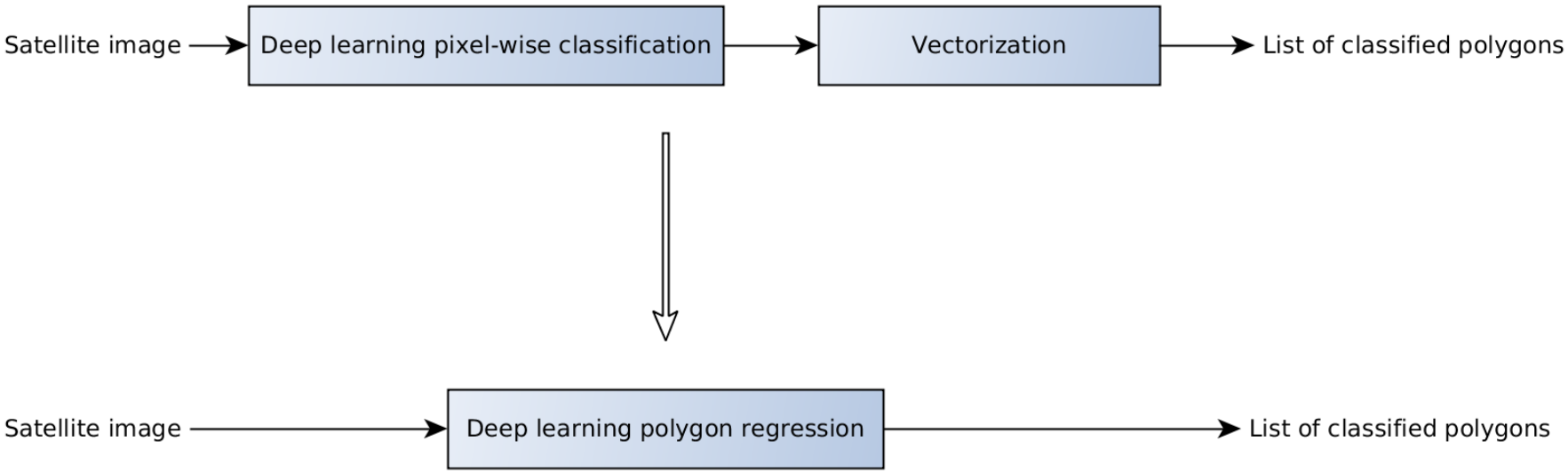
DP, thr. 3

DP, thr. 7

- Deep learning methods for pixel-wise classification
- Bitmap classification masks cannot be directly used to update geographic information systems (GIS)
- Vectorial maps are needed:
  - Object instances outlined by a polygon

[1] E. Maggiori et al., High-resolution aerial image labeling with convolutional neural networks (2017)

Can a deep learning method learn to directly regress polygons?



Analyzing this new paradigm

Several tasks:

- Object detection
- Object recognition
- Object polygon outline regression

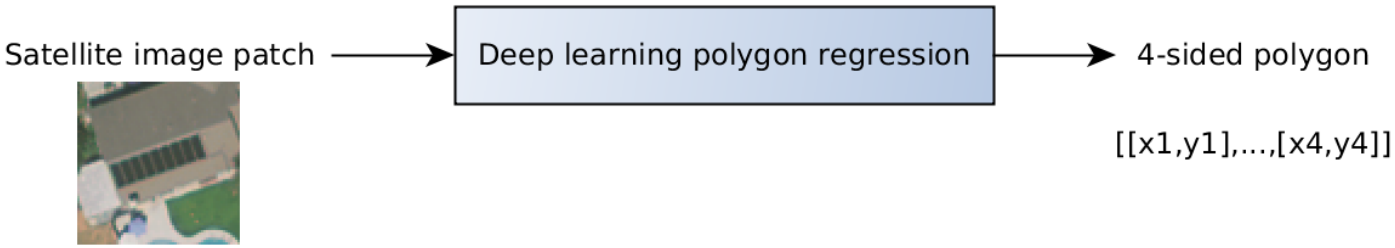
Difficulties:

- Thousands of objects per satellite image
- Variable amount of objects across images
- Variable amount of vertices across polygons
- Overlapping objects

A first problem statement

Reduced goal:

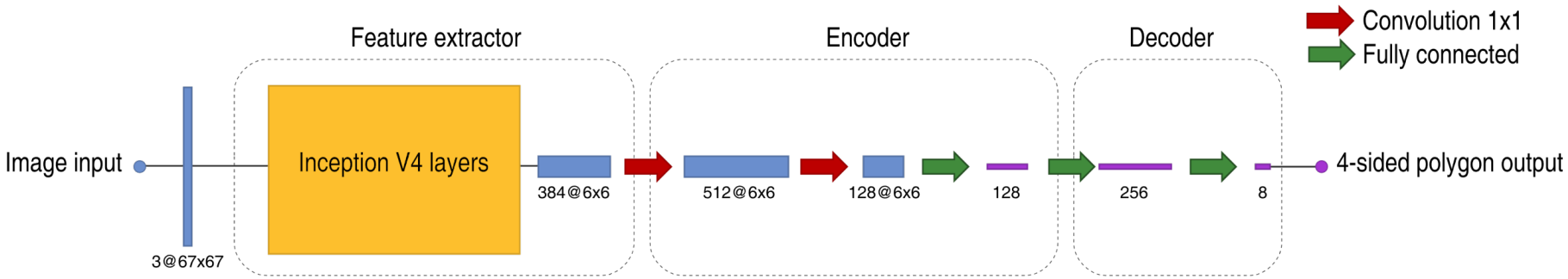
- Task: object polygon outline regression
- Input: image patch centered on a detected object of one class
  - Output of an object detector like Faster-RCNN[2]:
- Output: one polygon outlining the object
  - Number of vertices fixed to 4



[2] S. Ren et al., Faster R-CNN: towards real-time object detection with region proposal networks (2015)

Methodology

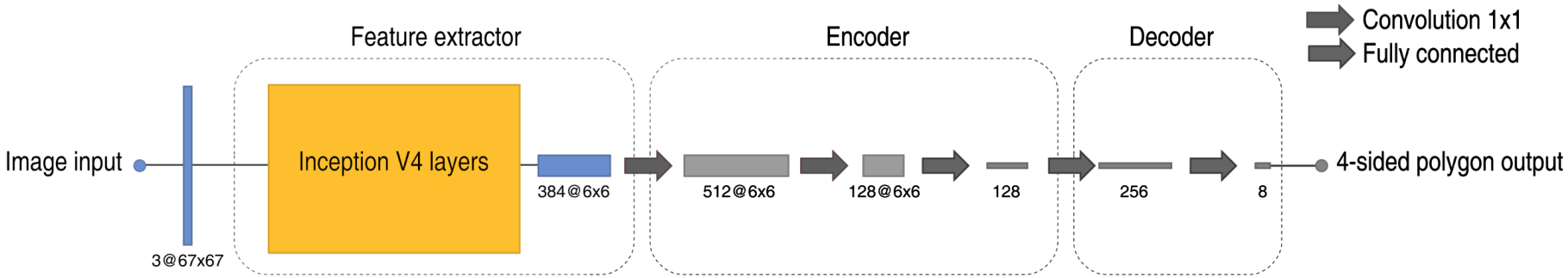
Polygon regression neural network



Three blocks:

1. Feature extractor
2. Encoder
3. Decoder

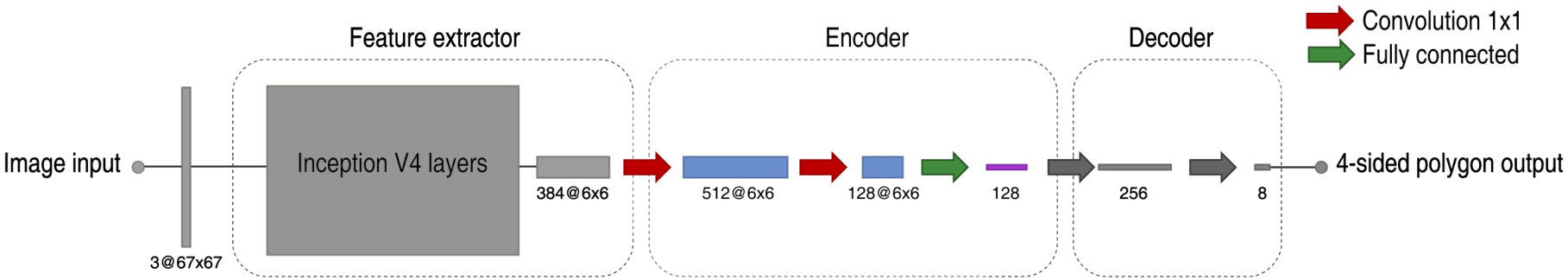
1. Feature extractor



Pre-trained Inception V4[3] layers used:

[3] C. Szegedy et al., Inception-v4, inception-resnet and the impact of residual connections on learning (2016)

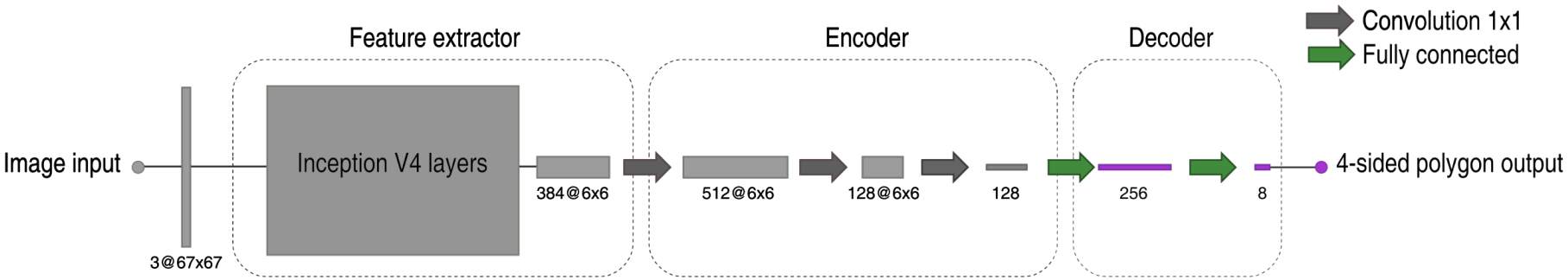
2. Encoder



Encodes object outline in a vector of 128 dimensions:

- A point in a latent space
- Represents the object shape

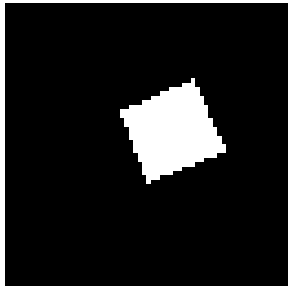
3. Decoder



Decodes the 128-dimensions vector into polygon coordinates:

- 4 vertices in $2D = 8$ scalars

Finding the right architecture for the encoder and decoder



Creation of an artificial dataset:

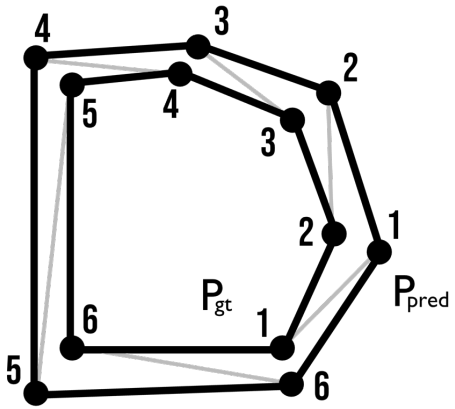
- Random 4-sided polygons as ground truth
- Rasterized polygons as image input

Infinite amount of simple example:

- Fast training, only a simple feature extractor is needed
- Test architectures to find the smallest encoder and decoder that work
- Pre-train decoder

Loss

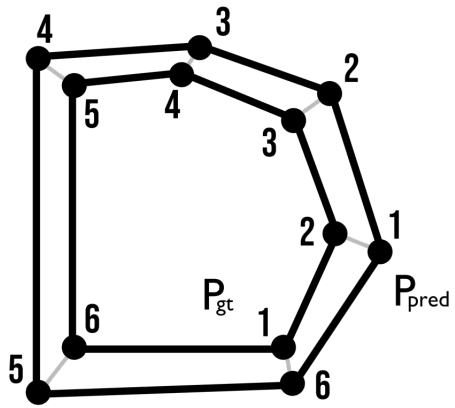
Network trained by supervised learning, naive loss:



$$L = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{P_{gt}}(\mathbf{i}, .) - \mathbf{P_{pred}}(\mathbf{i}, .)\|_2 \quad (1)$$

Loss

Network trained by supervised learning, corrected loss:



$$L = \min_{\forall s \in [0, n-1]} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{P_{gt}}(\mathbf{i}, .) - \mathbf{P_{pred}}(\mathbf{i} + \mathbf{s}, .)\|_2 \qquad (2)$$

Experimental setup and training

Dataset

Solar panel dataset from Bradbury et al.[4] from Duke University:

- 601 aerial images of 5000×5000 px
- Ground truth polygons of photovoltaic arrays
- Polygons precisely annotated manually
- Over 19000 solar panels
- Over 6000 4-sided ground truth polygons
- 256 polygons for validation and another 256 for testing

[4] K. Bradbury et al., Distributed solar photovoltaic array location and extent dataset for remote sensing object identification (2016)

Training

1. Feature extractor pre-trained on ImageNet[5]
2. Encoder initialized randomly
3. Decoder pre-trained on the artificial dataset

Learning rate schedule:

learning rate up to iteration	500	1000	90000
Feature extractor	0	0	$1e^{-5}$
Encoder	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$
Decoder	0	$1e^{-5}$	$1e^{-5}$

- Random weights produce big gradients at the start of training
- Avoid rapid distortion of pre-trained weights by freezing them in the beginning

[5] J. Deng et al., ImageNet: A Large-Scale Hierarchical Image Database (2009)

Results

Visual results

Test on image patches never seen by the network:

PolyCNN

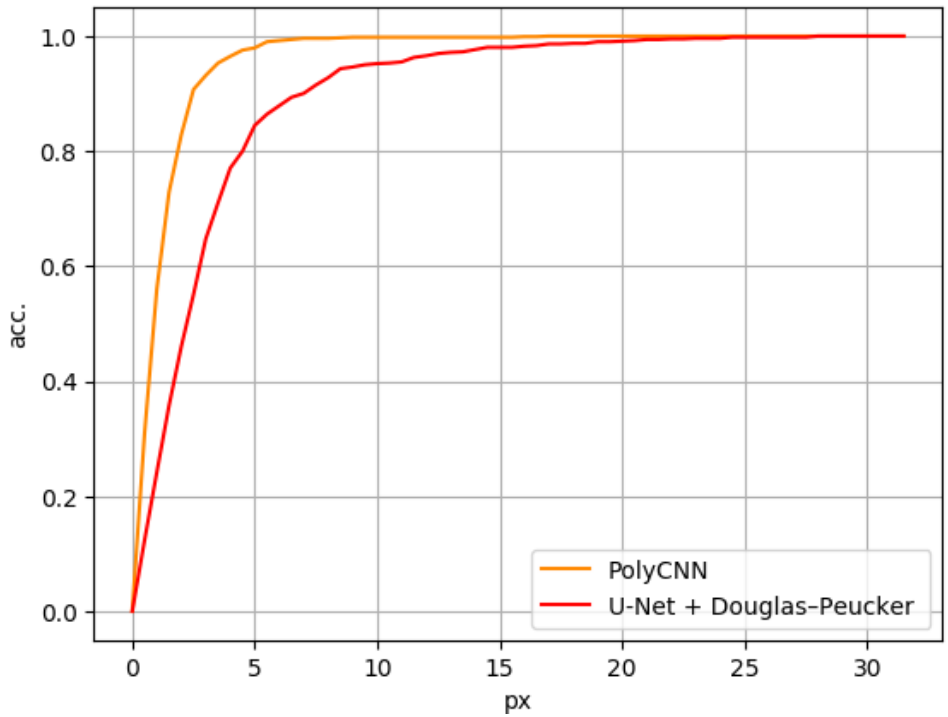
U-Net
+ Douglas-Peucker

Green: ground truth, orange: PolyCNN output and red: U-Net + Douglas-Peucker output

Quantitative results

	PolyCNN	U-Net + Douglas-Peucker
mIoU	79.5%	62.4%

For any threshold τ we compute the fraction of vertices whose ground truth point distance is less than τ:



Concluding remarks

Conclusions

- Learning directly in vectorial space is possible in a end-to-end fashion
- We get better results than a 2-step process involving a U-Net followed by vectorization
- Our method can better predict the geometric shape of the object (orientation and close-to-right angles)

Future works

- Learn *n*-sided polygons (keep the same feature extractor and encoder but train a different decoder)
- Learn to outline other types of objects
- Try using a fully-convolutional architecture in the future

Thank you for your attention !

Any questions ?