

Can we classify the world?

Where Deep Learning Meets Remote Sensing

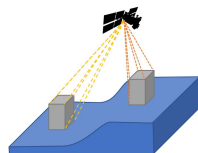
Yuliya Tarabalka, Emmanuel Maggiori, Nicolas Girard, Onur Tasar,
Armand Zampieri, Andrew Khalel, Guillaume Charpiat, Pierre Alliez

Inria Sophia Antipolis-Méditerranée - TITANE team
Université Côte d'Azur - École Doctoral STIC



Context

- Continuous proliferation & improvement of remote data sensors
- Huge volume of unstructured satellite images

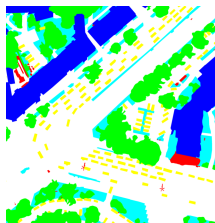


Context

- **Crucial need:** automatize the analysis of remote sensing data
- **Remote sensing image classification:** assign a semantic class to every pixel



Input



Output

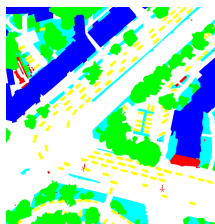
- Impervious surf.
- Building
- Low veget.
- Tree
- Car
- Clutter

Context

- **Crucial need:** automatize the analysis of remote sensing data
- **Remote sensing image classification:** assign a semantic class to every pixel



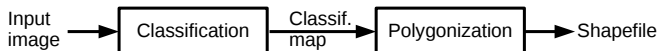
Input



Output

- Impervious surf.
- Building
- Low veget.
- Tree
- Car
- Clutter

- **Applications:** Automatic update of GIS maps, Earth monitoring, ...



Challenges: Large-scale data sources

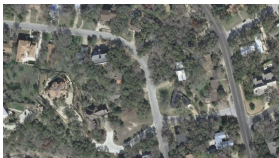
- Increasing amount & openness of data, e.g.:
 - Pléiades: entire earth every day (< 1 m resolution)
 - Free Copernicus satellite data

⇒ Scalability: temporal/space complexity

- Intra-class variability:



Chicago



Austin



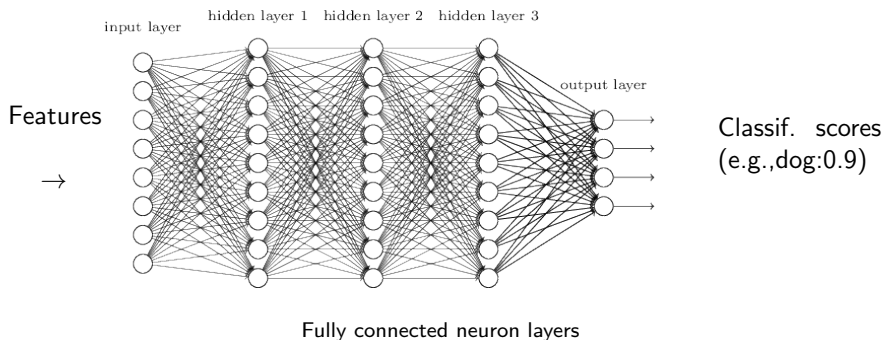
Vienna

- Interest in semantic classes (e.g., *building*, *road*, *lane*)
 - ⇒ Need for high-level contextual reasoning (shape, patterns,...)
 - ⇒ Generalization to different locations

Artificial neural networks

Multilayer perceptron (MLP)

Learn a function that maps input features to desired classification scores:



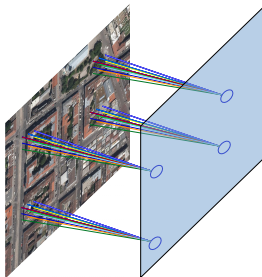
Deep learning for large-scale image classification

Convolutional neural networks (CNNs) [LeCun et al., 1998]

- Jointly learn to extract contextual features & conduct classification
- Input: the image itself
- *Convolutional* layers & *pooling* layers

Convolutional layers:

Learned convolution filters \rightarrow feature maps



Special case of fully connected layer:

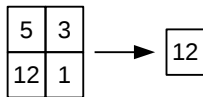
- Only local spatial connections
 - Location invariance
- \Rightarrow Meaningful in image domain (or text, time series,...)

Deep learning for large-scale image classification

Pooling layers

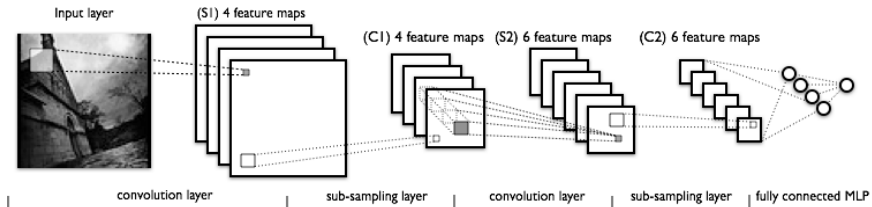
Subsample feature maps

- Increase *receptive field* 😊
- Downgrade resolution
 - Robustness to spatial variation 😊
 - Not good for *pixelwise* labeling ☹️



Max pooling

Overall image categorization CNN



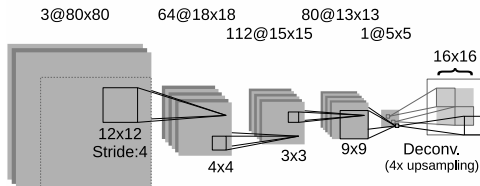
Source: deeplearning.net

Remote sensing: *dense* classification with CNNs?

Fully convolutional networks (FCNs) [Long et al., CVPR 2015]

- Interpolation with a learned kernel (“deconvolutional” layer)
- Lost resolution is upsampled

Proposed FCN for remote sensing



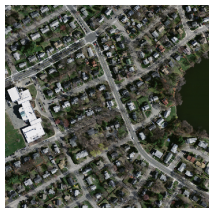
- Adapted from previous work (Mnih, 2013) and made it fully conv.
- 10x faster and more accurate

E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez. “Convolutional neural networks for large-scale remote sensing image classification”, IEEE TGRS, 55 (2), 2017.

Classification with FCNs: some results

Massachusetts dataset

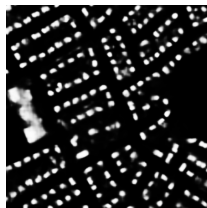
[Dataset: Mnih, 2013]



Color input



Reference



FCN



Pixelwise SVM

- Classification of 22.5 km² (1 m resolution): 8.5 seconds (2.7 GHz 8-core, Quadro K3100M GPU)

E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez. "Convolutional neural networks for large-scale remote sensing image classification", IEEE TGRS, 55 (2), 2017.

Yielding high-resolution outputs

Recognition/localization (RL) trade-off

Subsampling:

- increases the receptive field (improving recognition)
- reduces resolution (hampering localization)



Input



Ref.



FCN

First architectures for high-resolution labeling

- *Dilation* (Chen et al., 2015; Dubrovina et al., 2016,...)
- *Unpooling/deconv.* (Noh et al., 2015; Volpi and Tuia, 2016,...)
- *Skip networks* (Long et al., 2015; Badrinarayanan et al., 2015,...)

Yielding high-resolution outputs

Premise

- CNNs do not need to “see” everywhere at the same resolution
- E.g., to classify central pixel:



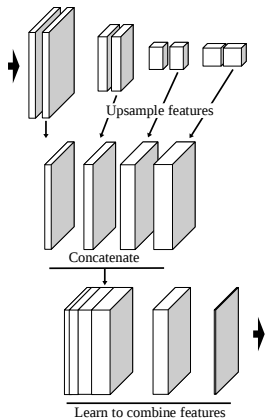
Full resolution context



Full resolution only near center

⇒ Combine resolutions in a flexible way to address trade-off

To address RL trade-off: MLP network

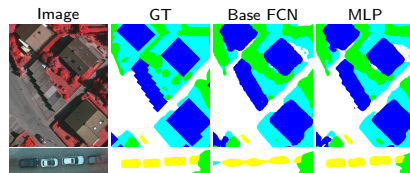


1. Base FCN
2. Extract intermediate features
⇒ Pool of features
3. Multi-layer perceptron (1 hidden layer)
learns how to combine those features
⇒ Output classification map

E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez. "High-Resolution Aerial Image Labeling with Convolutional Neural Networks", IEEE TGRS, 55 (12), 2017.

Experiments

Vaihingen & Postdam ISPRS datasets:



Impervious surface (white), Building (blue), Low veget. (cyan), Tree (green), Car (yellow)

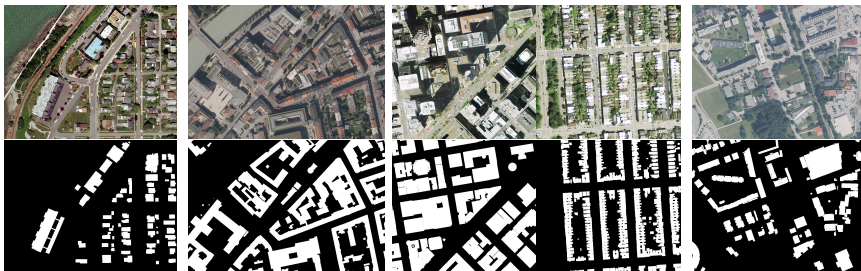
<i>Vaihingen</i>	Imp. surf.	Build.	Low veg.	Tree	Car	Acc.
CNN+RF	88.58	94.23	76.58	86.29	67.58	86.52
CNN+RF+CRF	89.10	94.30	77.36	86.25	71.91	86.89
Deconvolution						87.83
Dilation	90.19	94.49	77.69	87.24	76.77	87.70
Dilation + CRF	90.41	94.73	78.25	87.25	75.57	87.90
MLP	91.69	95.24	79.44	88.12	78.42	88.92

Submission to ISPRS server

- Overall accuracy: 89.5%
- **Second place** (out of 29) at the time of submission
- Significantly simpler and faster than other methods

Classifying cities over the earth: can CNNs generalize?

Inria Aerial Image Labeling Dataset (810 km², 30 cm resolution, 3 bands):



Bellingham

Innsbruck

San Francisco

Tyrol

- Images over US and Austria with open images and building footprints
- Different cities in training and test sets


















⇒ project.inria.fr/aerialimagelabeling

E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez. "Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark". IGARSS 2017.

Inria Aerial Image Labeling Benchmark

- Dec. 2016 - present: > 2000 downloads, > 60 submissions

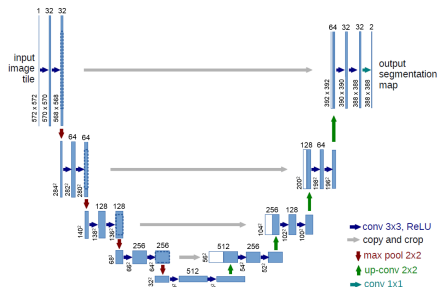
Leaderboard

Method	Date	Bellingham		Bloomington		Innsbruck		San Francisco		East Tyrol		Overall	
		IoU	Acc.	IoU	Acc.	IoU	Acc.	IoU	Acc.	IoU	Acc.	IoU	Acc.
Inria1 	3-Jan-17	52.91	95.14	46.08	94.95	58.12	95.16	57.84	86.05	59.03	96.40	55.82	93.54
Inria2 	3-Jan-17	56.11	95.37	50.40	95.27	61.03	95.37	61.38	87.00	62.51	96.61	59.31	93.93
TeraDeep 	5-May-17	58.08	95.88	53.38	95.61	59.47	95.26	64.34	88.71	62.00	96.57	60.95	94.41
RMIT 	16-July-17	57.30	95.97	51.78	95.60	60.70	95.69	66.71	89.23	59.73	96.59	61.73	94.62
Raisa Energy 	8-Sep-17	64.46	96.04	56.63	95.38	66.99	95.97	67.74	87.48	69.21	96.92	65.94	94.36
DukeAMLL 	6-Nov-17	66.90	96.69	58.48	96.15	69.92	96.37	75.54	91.87	72.34	97.42	70.91	95.70
NUS 	13-Nov-17	65.36	96.34	58.50	95.95	68.45	96.21	71.17	90.08	71.58	97.32	68.36	95.18
Onera 1 	14-Nov-17	63.42	96.11	62.74	96.20	63.77	95.44	66.53	89.18	65.90	96.76	65.04	94.74
Onera 2 	14-Nov-17	68.92	96.94	68.12	97.00	71.87	96.72	71.17	89.74	74.75	97.78	71.02	95.63
NUS 	22-Nov-17	70.74	97.00	66.06	96.74	73.17	96.75	73.57	91.19	76.06	97.81	72.45	95.90
DukeAMLL 	29-Nov-17	67.14	96.64	65.43	96.73	72.27	96.66	75.72	91.80	74.67	97.70	72.55	95.91
Raisa Energy 	30-Nov-17	68.73	96.79	60.83	96.23	70.07	96.31	70.64	89.52	74.76	97.64	69.57	95.30
NUS 	11-Dec-17	66.93	96.54	61.42	96.34	71.06	96.47	74.34	91.47	73.21	97.51	70.99	95.67
NUS 	15-Dec-17	70.73	97.06	64.87	96.69	73.64	96.86	73.04	91.19	76.36	97.88	72.18	95.94
DukeAMLL 	15-Dec-17	68.75	96.89	59.17	96.27	71.82	96.66	69.28	89.90	75.83	97.85	69.15	95.51
Koki Takahashi 	22-Dec-17	69.71	96.89	67.10	96.49	78.41	97.47	79.79	93.29	80.15	98.26	76.42	96.48
Tao Hu 	2-Jan-18	65.92	96.78	62.05	96.54	72.37	96.91	69.78	90.35	70.21	97.46	68.72	95.61

Classifying cities over the earth: can CNNs generalize?

Inria Aerial Image Labeling Dataset: first outcomes

- Intersection over union improved from 55.82% to **78.39%**
- The most commonly used successful architecture: **U-Net**
 - AMLL, Duke University: original U-Net with half as many filters

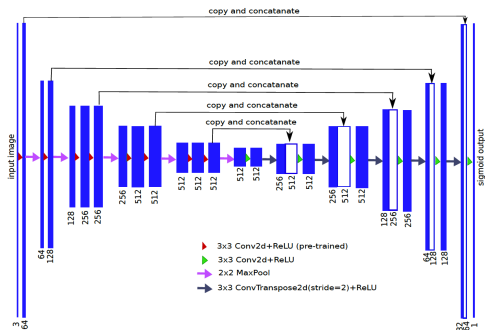


B Huang et al. "Large-scale semantic classification: Outcome of the first year of Inria aerial image labeling benchmark". IGARSS 2018.

Classifying cities over the earth: can CNNs generalize?

Inria Aerial Image Labeling Dataset: first outcomes

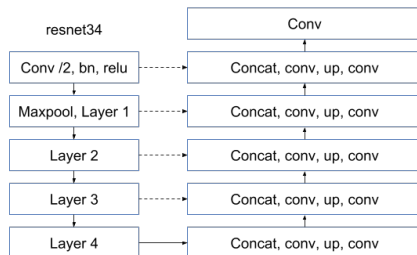
- Intersection over union improved from 55.82% to **78.39%**
- The most commonly used successful architecture: **U-Net**
 - Vladimir Iglovikov: TeraNet with VGG11-like encoder



Classifying cities over the earth: can CNNs generalize?

Inria Aerial Image Labeling Dataset: first outcomes

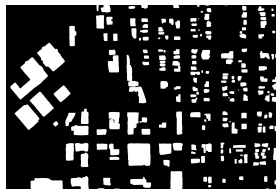
- Intersection over union (IoU) improved from 55.82% to **78.39%**
- **Winning architecture:**



Inria Aerial Image Labeling Dataset outcomes (IoU, %)



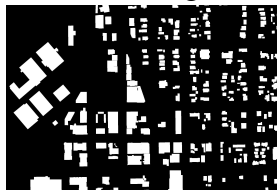
RGB image



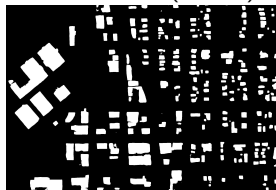
A. Buslaev (78.39)



V. Iglovikov (75.28)



Ground truth



AMLL (72.55)



Onera (71.02)

B Huang et al. "Large-scale semantic classification: Outcome of the first year of Inria aerial image labeling benchmark". IGARSS 2018.

Classifying cities over the earth: can CNNs generalize?

How to achieve good results?

- Right choice of architecture: U-net for semantic labeling
- Right choice of loss function: combination of cross-entropy & IoU
- Right choice of training strategy
- Right choice of other parameters: learning rate, size of batch, ...

Common loss functions

Loss function quantifies the misclassification by comparing the target label vectors $\mathbf{y}^{(i)}$ and the predicted label scores $\hat{\mathbf{y}}^{(i)}$, for n training samples

- **Cross-entropy loss:**

$$L_{CE} = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^{|\mathcal{L}|} y_k^{(i)} \log \hat{y}_k^{(i)}$$

- has fast convergence rates when training neural networks
- numerically stable when coupled with softmax normalization
- Differentiable soft IoU loss*:

$$L_{IoU} = \frac{1}{|\mathcal{L}|} \sum_{\mathcal{L}} \frac{\sum_i \hat{y}_k^{(i)} \cdot y_k^{(i)}}{\sum_i \hat{y}_k^{(i)} + y_k^{(i)} - \hat{y}_k^{(i)} \cdot y_k^{(i)}}$$

- enforces network to push predictions to 0 and 1

Classifying cities over the earth: can CNNs generalize?

How to achieve good results?

- Right choice of architecture: U-net for semantic labeling
- Right choice of loss function: combination of cross-entropy & IoU
- Right choice of training strategy
- Right choice of other parameters: learning rate (e.g. cyclic), size of batch, ...

Classifying cities over the earth: can CNNs generalize?

How to achieve good results?

- Right choice of architecture: U-net for semantic labeling
- Right choice of loss function: combination of cross-entropy & IoU
- Right choice of training strategy
- Right choice of other parameters: learning rate (e.g. cyclic), size of batch, ...
- **Good training dataset!**

Where to get data?

- A lot of available aerial and satellite imagery!
 - Example: Sentinel
<https://sentinel.esa.int/web/sentinel/home>
 - Difficult and expensive to get ground-truth data
- Solutions?
 - Use available datasets/benchmarks
 - IARPA challenge: 3.5TB satellite multispectral data, 62 classes
 - CrowsAI challenge: > 300K 300 × 300 RGB satellite images & building annotations
<https://www.crowdai.org/challenges/mapping-challenge>
 - Use crowd-sourced maps
 - <http://www.openstreetmap.org>
 - <https://www.eea.europa.eu/data-and-maps/data/copernicus-land-monitoring-service-urban-atlas>

Where to get data?

- Solutions?
 - Use maps predicted with deep learning nets and released as open data?



Bing has made very significant investments in the area of deep learning, computer vision and artificial intelligence to support a number of different search scenarios. The Bing Maps team has been applying these techniques as well with the goal to increase the coverage of building footprints available for [OpenStreetMap](#). As a result, today we are announcing that we are releasing 124 Million building footprints in the United States to the OpenStreetMap community.

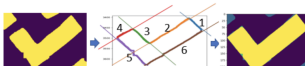
The Maps team has been relying on the Open Source [CNTK Unified Toolkit](#) which was developed by Microsoft. Using CNTK we apply our Deep Neural Networks and the ResNet34 with RefineNet up-sampling layers to detect building footprints from the Bing imagery.

First stage - Semantic Segmentation



We remove noise and suspicious data (false positives) from the predictions and then apply a polygonization algorithm to detect building edges and angles to create a proper building footprint.

Second stage - Polygonization

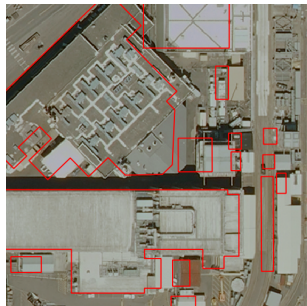


Dealing with imperfect training data

And if training dataset is not good enough?

Dealing with imperfect training data

- Frequent misregistration/omission in large-scale data sources
 - Example: OpenStreetMap data are mostly misaligned with satellite data



Dealing with imperfect training data



Pléiades image + OpenStreetMap (OSM)

Proposed method

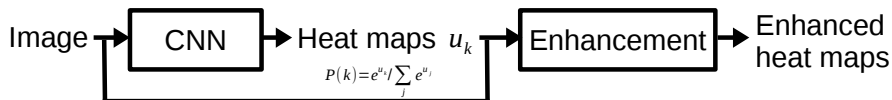
Two-step training process:

1. Pretrain on large amounts of imperfect data
→ Learn dataset generalities
2. Fine-tune on a small piece of manually labeled reference



E. Maggiori, Y. Tarabalka, G. Charpiat, P. Alliez. "Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification", TGRS 2017.

Enhancing CNNs' outputs



Recent approaches

- CNN + Fully connected CRF (Chen et al., ICML 2015)
- CNN + Fully connected CRF as RNN (Zheng et al., CVPR 2015)
- CNN + Domain transform (Chen et al., CVPR 2016)

In remote sensing:

- CNN + CRF (Paisitkriangkrai et al., CVPR Workshops 2015)
- CNN + fully connected CRF (Marmanis et al., ISPRS 2015; Sherrah 2016,...)

Goal

Learn iterative enhancement process

Partial differential equations (PDEs)

Given heat maps u_k , image I :

- Heat flow

(Smooths out u_k)

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(\nabla u_k(x))$$

- Perona-Malik

Edge-stopping function $g(\nabla I, x)$

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(g(\nabla I, x) \nabla u_k(x))$$

- Anisotropic diffusion

Diffusion tensor $D(I, x)$

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(D(\nabla I, x) \nabla u_k(x))$$

- Geodesic active contours

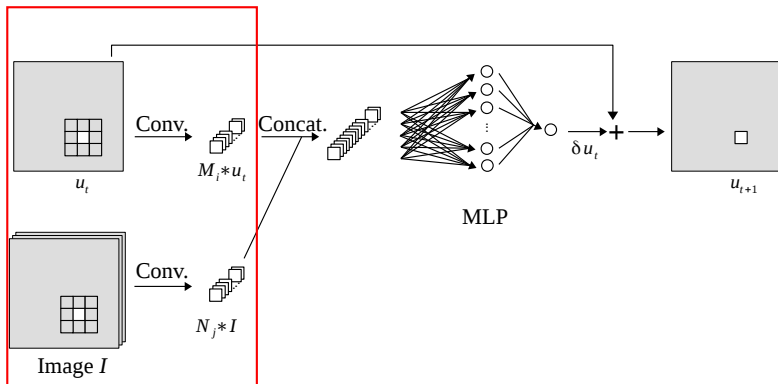
Edge-stopping function $g(\nabla I, x)$

$$\frac{\partial u_k(x)}{\partial t} = |\nabla u_k(x)| \text{div} \left(g(\nabla I, x) \frac{\nabla u_k(x)}{|\nabla u_k(x)|} \right)$$

- ...

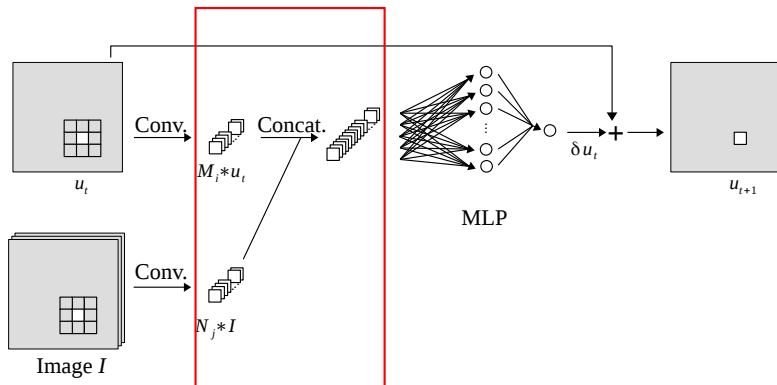
A generic enhancement process

- Differential operations ($\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, $\frac{\partial^2}{\partial x \partial y}$, $\frac{\partial^2}{\partial x^2}$, ...) applied on u_k and image I
- Implemented as convolutions: $M_i * u_k$, $N_j * I$
 $\{M_1, M_2, \dots\}$, $\{N_1, N_2, \dots\}$ conv. kernels (e.g., Sobel filters)



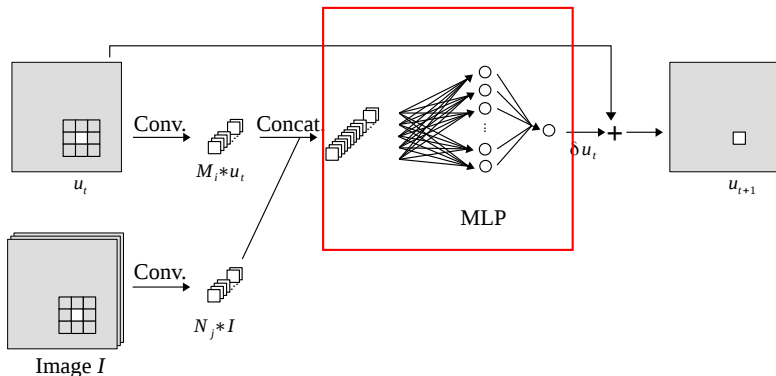
A generic enhancement process

- $\Phi(u_k, I) = \{M_i * u_k, N_j * I; \forall i, j\}$, set of responses



A generic enhancement process

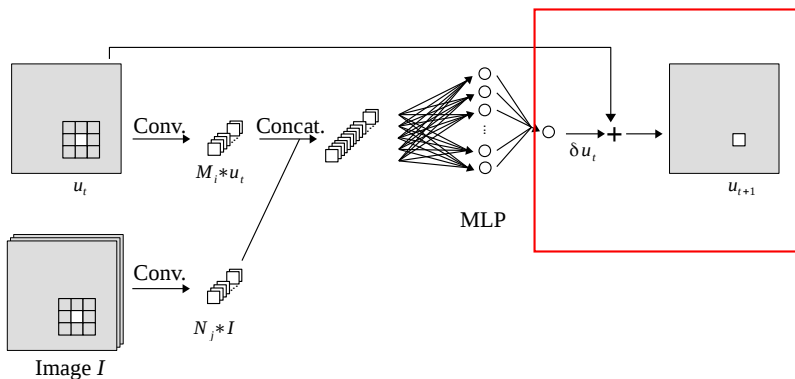
- Overall update on u_k at x : $\delta u_k(x) = f_k(\Phi(u_k, I)(x))$
- Class-specific f_k , implemented as multilayer perceptron
- M_i and N_j convey spatial reasoning (e.g., gradients), f_k their combination (e.g., products)



A generic enhancement process

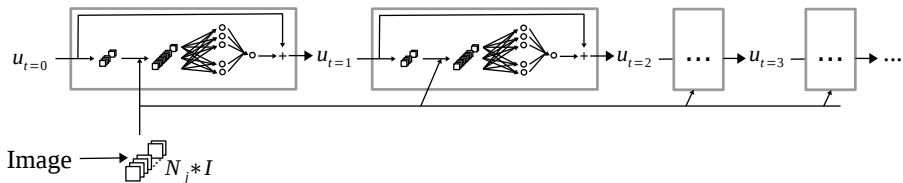
- Discretized in time:

$$u_{k,t+1}(x) = u_{k,t}(x) + \delta u_{k,t}(x), \text{ overall update } \delta$$



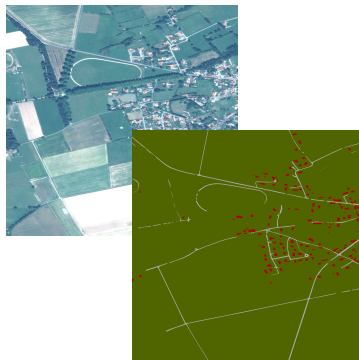
Iterative processes as recurrent neural networks (RNNs)

- “Unroll” iterations
- Every iteration is meant to progressively refine the classification maps
- Enforce weight sharing along iterations
- Train by backpropagation (“through time”)



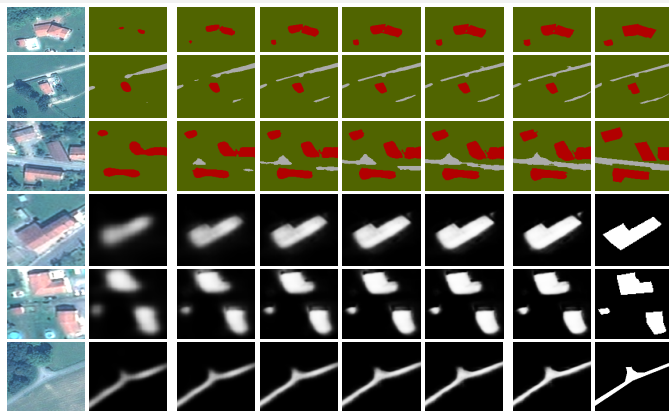
Experiments

- FCN trained on Pléiades + OSM data
- Manually labeled tiles for RNN training/testing
- Unroll 5 iterations
- 32 M_i and 32 N_j
- MLP: 1 hidden layer, 32 neurons



Building, Road, Background

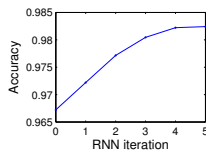
Experiments



Color CNN map
(RNN input)

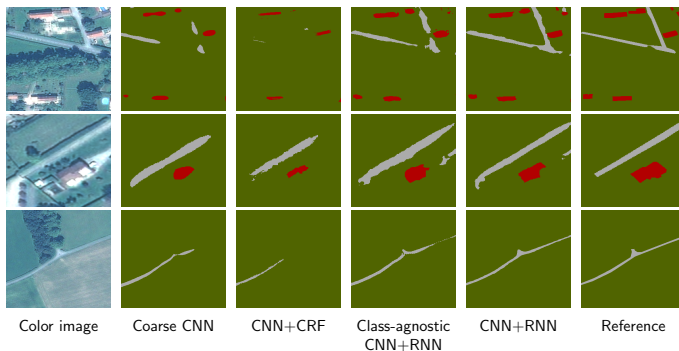
— Intermediate RNN iterations —

RNN output Reference



Experiments

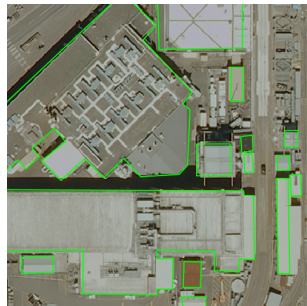
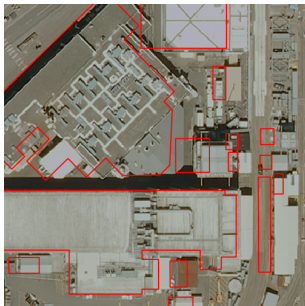
Comparison



Method	Overall accuracy	Mean IoU	Class-specific IoU		
			Build.	Road	Backg.
CNN	96.72	48.32	38.92	9.34	96.69
CNN+CRF	96.96	44.15	29.05	6.62	96.78
Class-agn. CNN+RNN	97.78	65.30	59.12	39.03	97.74
CNN+RNN	98.24	72.90	69.16	51.32	98.20

Fully-convolutional net for multimodal image registration

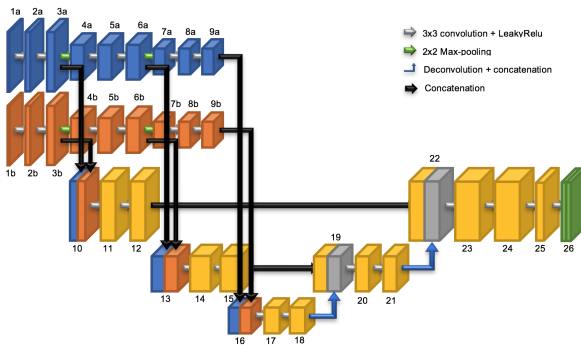
- Could we train a neural net to solve image-map alignment problem?



A. Zampieri, G. Charpiat, N. Girard, Y. Tarabalka. "Multimodal image alignment through a multiscale chain of neural networks with application to remote sensing", ECCV 2018.

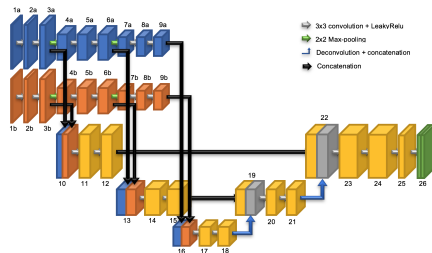
Fully-convolutional net for multimodal image registration

- Fully-convolutional neural net for image-map alignment



- Image is fed to 1a, rasterized map is fed to 1b
- Output: 2-dimensional vector map representing a deformation field

Fully-convolutional net for multimodal image registration



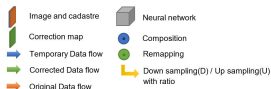
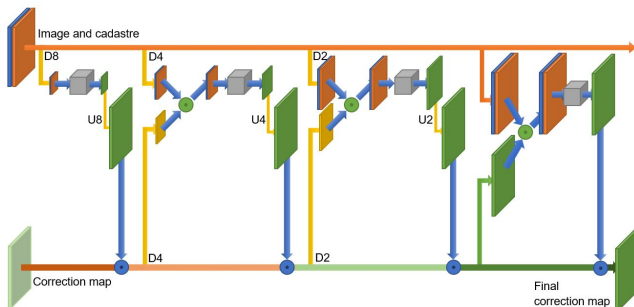
- Loss function: Euclidean norm of the prediction error

$$C = \mathbb{E}_{(l_1, l_2, \phi_{GT}) \in \mathcal{D}} \left[\sum_{\mathbf{x} \in \Omega(l_2)} \left\| \hat{\phi}_{(l_1, l_2)}(\mathbf{x}) - \phi_{GT}(\mathbf{x}) \right\|_2^2 \right],$$

i.e., expectation, over the ground truth dataset \mathcal{D} of triplet examples (RGB image l_1 , cadastral image l_2 , associated deformation ϕ_{GT}), of the sum, over all pixels \mathbf{x} in the image domain $\Omega(l_2)$, of the norm of the difference between the ground truth deformation $\phi_{GT}(\mathbf{x})$ and the one predicted $\hat{\phi}_{(l_1, l_2)}(\mathbf{x})$ for (l_1, l_2)

Fully-convolutional net for multimodal image registration

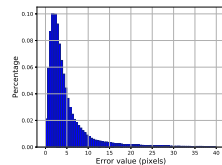
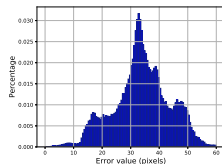
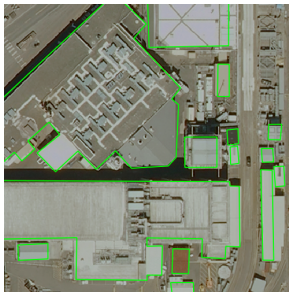
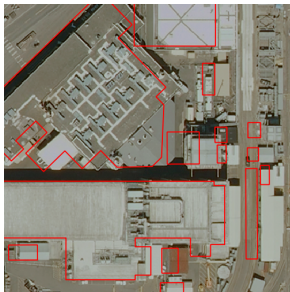
- Chain of scale-specific neural nets for image-map alignment



- Main idea: each scale-specific block:
 - downsamples the images to the right size,
 - applies the previously-estimated deformation,
 - refines it

Fully-convolutional net for multimodal image registration

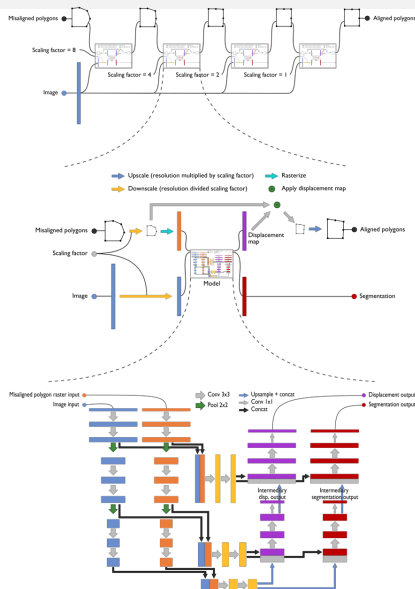
- Chain of scale-specific neural networks to solve alignment problem



Misalignment distribution
before and after processing

A. Zampieri, G. Charpiat, N. Girard, Y. Tarabalka. "Multimodal image alignment through a multiscale chain of neural networks with application to remote sensing", ECCV 2018.

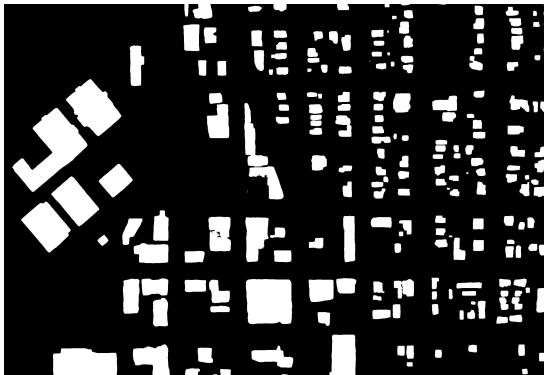
Align and update maps in one net?

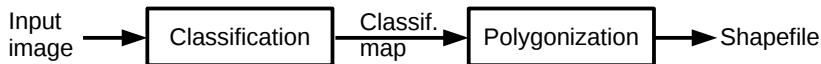


- Multi-resolution and multi-task deep learning
- Modified U-Net to have 2 image inputs and 2 image outputs
- Output: aligned and updated cadaster map
 - Each task helps training the other
 - Better performance on both tasks
 - Intermediate losses help training

N. Girard et al., "Aligning and updating cadaster maps with aerial images by multi-resolution, multi-task deep learning", ACCV 2018.

How to update GIS with the created maps?





- Polygonization of classification maps
→ incorporate objects into GIS

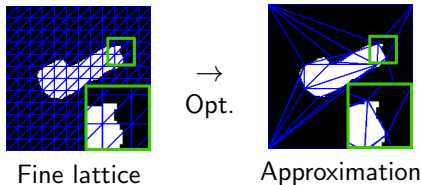


From raster to polygons

- Polygonization of classification maps → incorporate objects into GIS
- Typically (QGIS, GRASS, ArcGIS,...): greedy simplification algorithms

Proposed method

- Goal: approximate objects with (labeled) triangular mesh
- Integral formulation



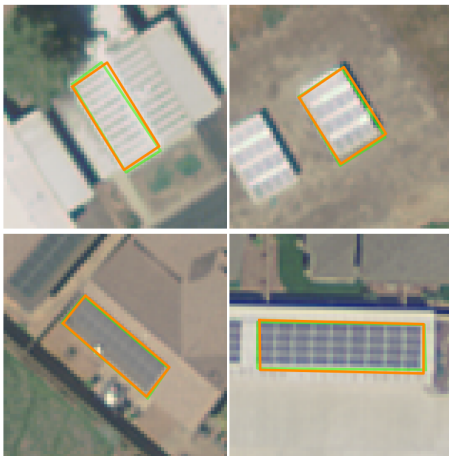
Optimization algorithm

- Discrete mesh operators (edge flip/collapse)
- Continuous vertex relocation
- Topology preservation & geometric regularity



O. Tasar, E. Maggiori, P. Alliez and Y. Tarabalka, "Polygonization of binary classification maps using mesh approximation with right angle regularity," IGARSS 2018.

Can we learn in a vector space?



Analyzing this new paradigm

Several tasks:

- Object detection
- Object recognition
- Object polygon outline regression

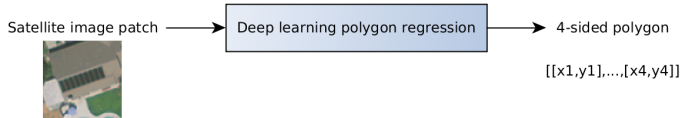
Difficulties:

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

Problem statement

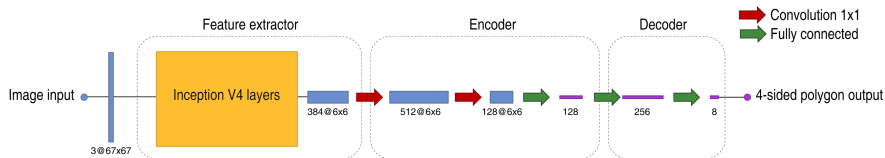
Reduced goal:

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



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

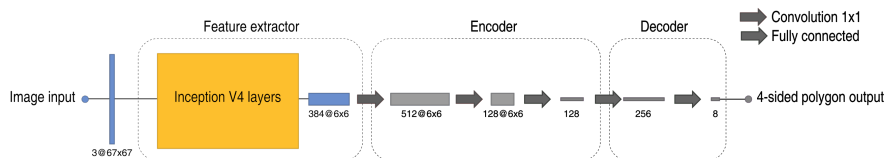
Polygon regression neural network *PolyCNN*



Three blocks:

1. Feature extractor
2. Encoder
3. Decoder

1. Feature extractor

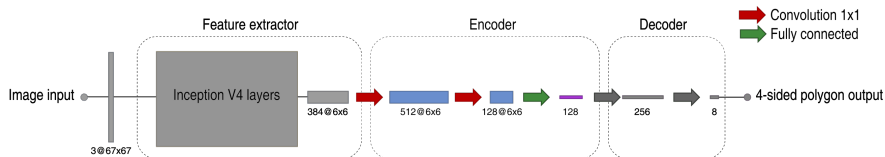


Pre-trained Inception V4 layers used:



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

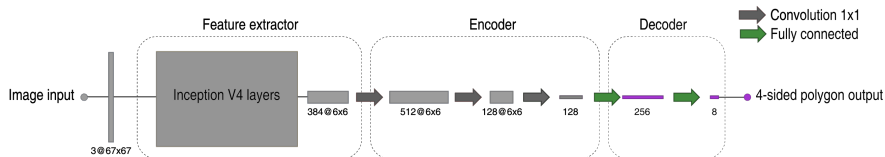
2. Encoder



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:

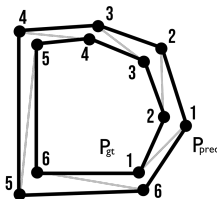
- Random 4-sided polygons as ground truth
- Rasterized polygons as image input

Infinite amount of simple examples:

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

Loss

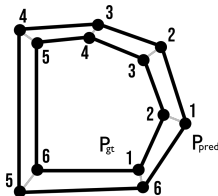
Network trained by supervised learning, naive loss:



$$L = \frac{1}{n} \sum_{i=1}^n \| \mathbf{P}_{gt}(\mathbf{i}, \cdot) - \mathbf{P}_{pred}(\mathbf{i}, \cdot) \|_2^2 \quad (1)$$

Loss

Network trained by supervised learning, corrected loss:



$$L = \min_{\forall s \in [0, n-1]} \frac{1}{n} \sum_{i=1}^n \| \mathbf{P}_{\text{gt}}(\mathbf{i}, \cdot) - \mathbf{P}_{\text{pred}}(\mathbf{i} + \mathbf{s}, \cdot) \|_2 \quad (2)$$

Dataset



Solar panel dataset from Bradbury et al. 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

Training

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

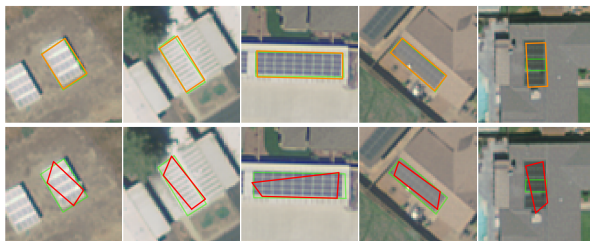
Learning rate schedule:

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

Visual results

Test on image patches never seen by the network:



PolyCNN

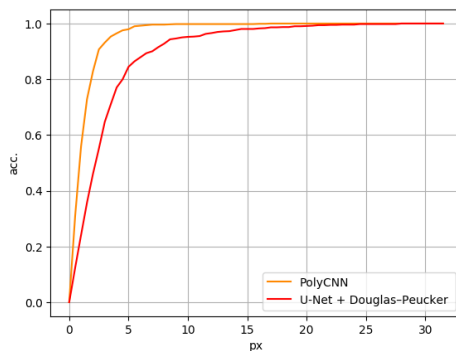
U-Net
+ Douglas-Peucker

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 :



Conclusions & Perspectives

There is no such thing as a universally better classifier

- To classify remote sensing images on a world-scale:
 - Learning methods must be **generic** and **highly scalable**
 - CNNs have shown a remarkable computational performance
 - Capable to learn expressive multi-scale contextual features
 - Succeed in classifying new unseen earth areas
- Current/future work?
 - Still many unsolved problems: domain adaptation, training from very noisy data, learning to predict shapefiles
 - New powerful models: adversarial networks, unsupervised & semi-supervised learning, capsule nets

Thank you for your attention!

Questions?

