Input Similarity from the Neural Network Perspective

Guillaume Charpiat

TAU team, LRI, Paris-Sud / INRIA Saclay

Journée Statistique et Informatique pour la Science des Données à Paris-Saclay

February 4th, 2021

Overview of this talk

- **I - Remote sensing image segmentation and registration**
- **II - Dataset self-denoising**
- **III - Notion of similarity from the neural network viewpoint**
- **IV - Back to dataset self-denoising**

Work in collaboration with Nicolas Girard, Loris Felardos & Yuliya Tarabalka

$\hookrightarrow$ TAU team, at INRIA Saclay & Titane team, at INRIA Sophia-Antipolis

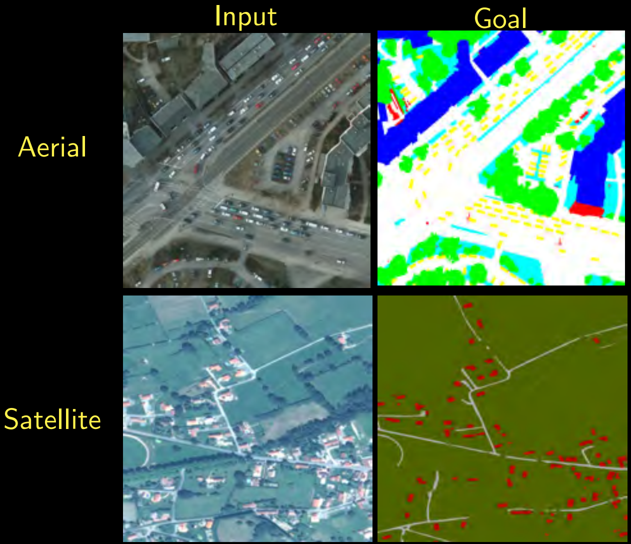
Part I

Remote sensing image segmentation and registration

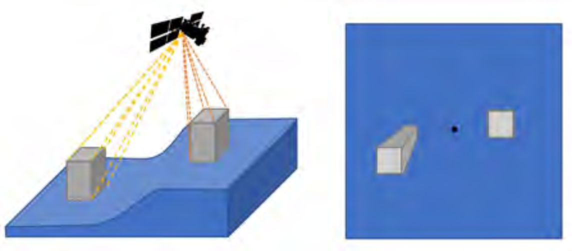
Task 1: Remote sensing image segmentation

- **Goal:** semantic segmentation of satellite images
  i.e.: each pixel $\mapsto$ *class* $\in$ {building, road, ...}
- **Tool:** neural networks with varied architectures
- **Obstacles:** no reliable dataset

Goal: semantic segmentation



Issue: misalignment



Deformations inherent to aerial or satellite photography

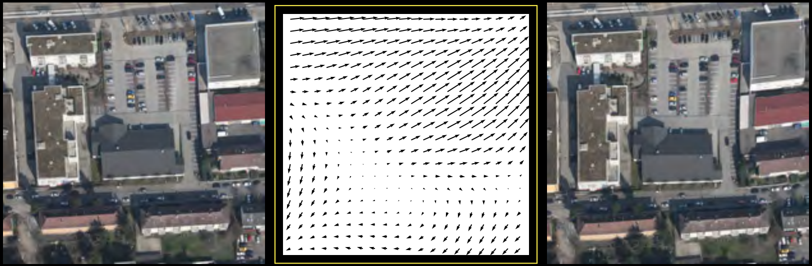
Issue: misalignment

- Registration: cadaster map (cyan) vs. photo RGB

Automatic realignment?


Multimodal pair of images: aerial RGB image / binary vector-format cadastral image (buildings in white)

Task 2: Multimodal Registration



Example of deformation. Image I ; a deformation ϕ, i.e. a $\mathbb{R}^2$ vector field ; associated deformed image $I \circ \phi$.

Approach

Optimization criterion: Euclidean norm of the prediction error

$$C(\mathrm{w}) = \mathop{\mathbb{E}}_{(I_1,I_2,\phi_{\mathrm{GT}})\in\mathcal{D}} \left[\sum_{\mathrm{x}\in\Omega(I_2)} \left\| \widehat{\phi}_{(\mathrm{w})(I_1,I_2)}(\mathrm{x}) - \phi_{\mathrm{GT}}(\mathrm{x}) \right\|_2^2 \right]$$

Approach

Optimization criterion: Euclidean norm of the prediction error

$$C(\mathrm{w}) = \mathop{\mathbb{E}}_{(I_1,I_2,\phi_{\mathrm{GT}})\in\mathcal{D}} \left[\sum_{\mathrm{x}\in\Omega(I_2)} \left\| \widehat{\phi}_{(\mathrm{w})(I_1,I_2)}(\mathrm{x}) - \phi_{\mathrm{GT}}(\mathrm{x}) \right\|_2^2 \right]$$

- Issue: the network doesn't learn
- for prediction: each pixel $\mapsto$ deformation ± 25 px is too hard
- Idea: deformation ± 1 px is easy

Approach

Optimization criterion: Euclidean norm of the prediction error

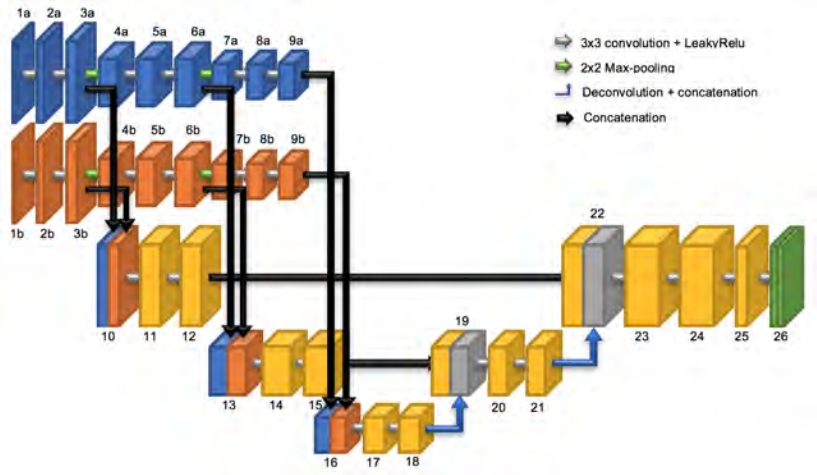
$$C(\mathrm{w}) = \mathop{\mathbb{E}}_{(I_1,I_2,\phi_{GT})\in\mathcal{D}} \left[\sum_{\mathrm{x}\in\Omega(I_2)} \left\| \widehat{\phi}_{(\mathrm{w})(I_1,I_2)}(\mathrm{x}) - \phi_{GT}(\mathrm{x}) \right\|_2^2 \right]$$

Task at scale s: Solve the alignment problem for the image pair (I_1, I_2), with a precision required of $\pm 2^s$ pixels, under the assumption that the amplitude of the registration to be found is not larger than 2^{s+1} pixels.

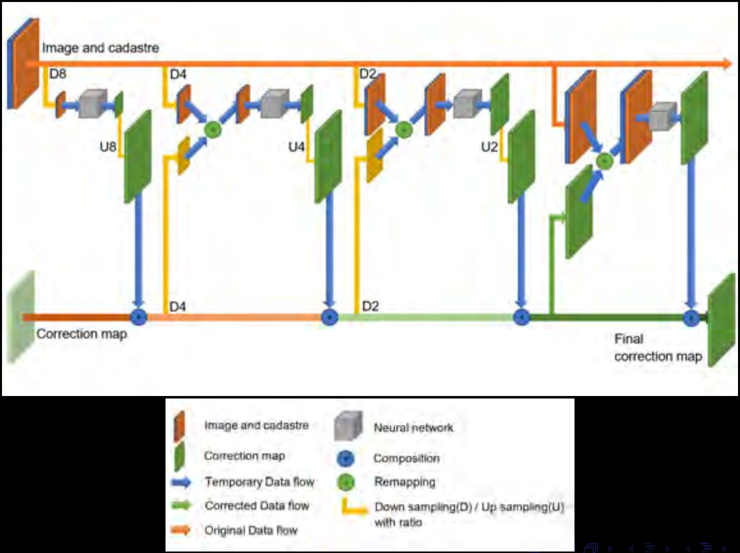
Solution for task at scale s: Downsample the images by a factor 2^s; solve the alignment task at scale 0 for these reduced images, and upsample the result with the same factor.

Full alignment algorithm: Given an image pair (I_1, I_2) of width w, iteratively solve the alignment task at scale s, from $s = \log_2 w$ until $s = 0$.

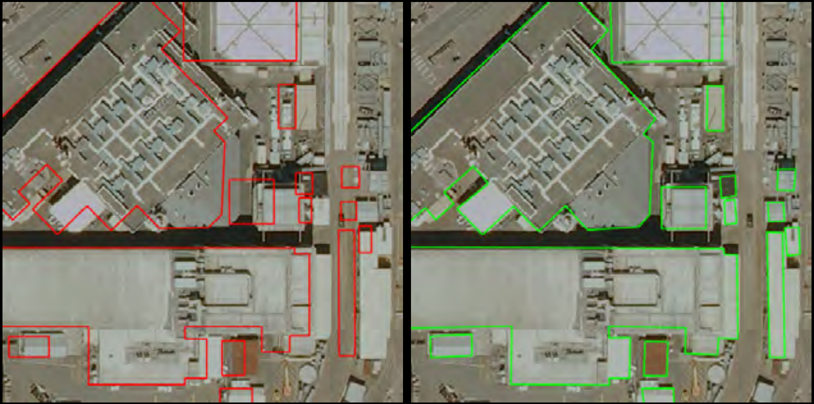
Network to process a specific scale



Global network: chain of scale-specific networks $\simeq$ compositional ResNet



Results



Example of image alignment. Original image and OpenStreetMap (OSM) map / Alignment result.

Part II

Dataset self-denoising

Training set for the alignment task

- pick locations where RGB image I and cadaster map M look not too badly aligned (or align manually)
  $\Longrightarrow$ training sample $\big((I, M), \mathrm{Id}\big)$
- generate random smooth deformations ϕ, and add $\big((I, M \circ \phi), \phi\big)$ to the training set

$\Longrightarrow$ sensitivity of the training w.r.t. alignment quality between original I and M?

Training set for the alignment task

- pick locations where RGB image I and cadaster map M look not too badly aligned (or align manually)
  $\Longrightarrow$ training sample $\big((I, M), \mathrm{Id}\big)$
- generate random smooth deformations ϕ, and add $\big((I, M \circ \phi), \phi\big)$ to the training set

$\Longrightarrow$ sensitivity of the training w.r.t. alignment quality between original I and M?

Dealing with noisy training data

- Dataset of examples $(x, y + \varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?

Dataset self-denoising

Red: given ground truth
Green: the real but unavailable one

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions
- replace the target labels by the ones predicted $\implies$ new dataset $\mathcal{D}_1$

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions
- replace the target labels by the ones predicted $\Longrightarrow$ new dataset $\mathcal{D}_1$
- train on $\mathcal{D}_1$

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions
- replace the target labels by the ones predicted $\Longrightarrow$ new dataset $\mathcal{D}_1$
- train on $\mathcal{D}_1$
- test on $\mathcal{D}_0$ $\Longrightarrow$ form new dataset $\mathcal{D}_2$

Nicolas's idea: update the dataset iteratively

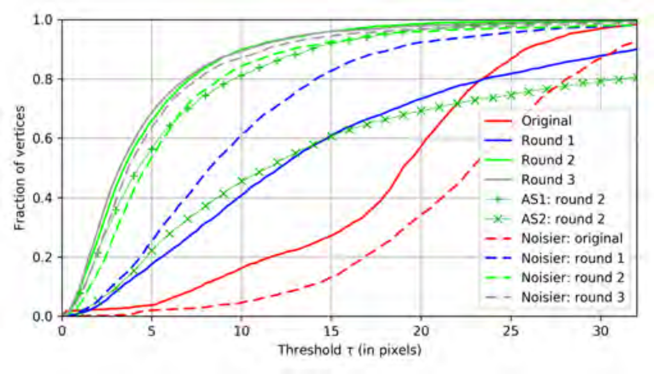
- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions
- replace the target labels by the ones predicted $\implies$ new dataset $\mathcal{D}_1$
- train on $\mathcal{D}_1$
- test on $\mathcal{D}_0$ $\implies$ form new dataset $\mathcal{D}_2$
- train on $\mathcal{D}_2$

Nicolas's idea: update the dataset iteratively

- given: dataset $\mathcal{D}_0$ with noisy labels
- train on $\mathcal{D}_0$
- test on $\mathcal{D}_0$: imprecise predictions
- replace the target labels by the ones predicted $\implies$ new dataset $\mathcal{D}_1$
- train on $\mathcal{D}_1$
- test on $\mathcal{D}_0$ $\implies$ form new dataset $\mathcal{D}_2$
- train on $\mathcal{D}_2$
- test on $\mathcal{D}_0$ $\implies$ form new dataset $\mathcal{D}_3$
- ...

Dataset self-denoising

Quantitative results:



Dashed lines: control experiment with added noise on ground truth

Dealing with noisy data

- Dataset of examples $(x, y + \varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?

Dealing with noisy data

- Dataset of examples $(x, y+\varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?
- Yes!

Dealing with noisy data

- Dataset of examples $(x, y + \varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?
- Yes!
  - one point x, with true label y
  - presented n times with noisy labels $y_i = y + \varepsilon_i$
  - assumption: i.i.d. noise ε, centered.
  - L^2 loss:

$$\inf_{\hat{y}} \sum_i \|\hat{y} - y_i\|^2$$

Dealing with noisy data

- Dataset of examples $(x, y+\varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?
- Yes!
  - one point x, with true label y
  - presented n times with noisy labels $y_i = y + \varepsilon_i$
  - assumption: i.i.d. noise ε, centered.
  - L^2 loss:

$$\inf_{\hat{y}} \sum_i \|\hat{y} - y_i\|^2$$

  - best fit: the average: $\hat{y} = \frac{1}{n} \sum_i y_i$

Dealing with noisy data

- Dataset of examples $(x, y + \varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?
- Yes!
  - one point x, with true label y
  - presented n times with noisy labels $y_i = y + \varepsilon_i$
  - assumption: i.i.d. noise ε, centered.
  - L^2 loss:

$$\inf_{\hat{y}} \sum_i \|\hat{y} - y_i\|^2$$

  - best fit: the average: $\hat{y} = \frac{1}{n} \sum_i y_i$
  -

$$\hat{y} \simeq y \pm \frac{1}{\sqrt{n}}$$

[Noise2Noise: Learning Image Restoration without Clean Data; Lehtinen et al., 2018]

Dealing with noisy data

- Dataset of examples $(x, y + \varepsilon)$ with noisy labels
- Is it possible to train and get accuracy higher than the noise variance?
- Yes!
  - one point x, with true label y
  - presented n times with noisy labels $y_i = y + \varepsilon_i$
  - assumption: i.i.d. noise ε, centered.
  - L^2 loss:
    $$\inf_{\hat{y}} \sum_i \|\hat{y} - y_i\|^2$$
  - best fit: the average: $\hat{y} = \frac{1}{n} \sum_i y_i$
  - $$\hat{y} \simeq y \pm \frac{1}{\sqrt{n}}$$

  [Noise2Noise: Learning Image Restoration without Clean Data; Lehtinen et al., 2018]
- Number of similar examples?
- Quantify: input similarity?

Part III

Input similarity
from the network's point of view

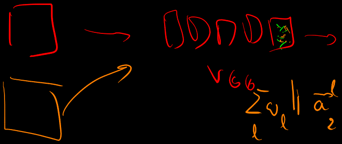
Notions of similarity

- Predefined metric (e.g., pixelwise L^2)
  - issue: small translations $\Longrightarrow$ large distances

Notions of similarity

- Predefined metric (e.g., pixelwise L^2)
  - issue: small translations $\Longrightarrow$ large distances
- Perceptual loss

  [Johnson, Alahi and Li Fei-Fei: Perceptual losses for real-time style transfer and super-resolution, ECCV 2016]
  - to evaluate auto-encoder reconstruction error
  - compare VGG activities $\Longrightarrow$ more semantic
  - in practice: arbitrary choices (pick one layer)



$$\sum_l \omega_l \| a^l_2 - a^l_1 \|$$

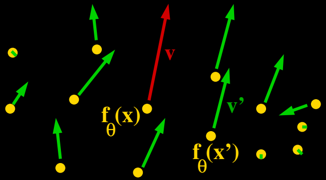
Notions of similarity

- Predefined metric (e.g., pixelwise L^2)
  - issue: small translations $\Longrightarrow$ large distances
- Perceptual loss
  [Johnson, Alahi and Li Fei-Fei: Perceptual losses for real-time style transfer and super-resolution, ECCV 2016]
  - to evaluate auto-encoder reconstruction error
  - compare VGG activities $\Longrightarrow$ more semantic
  - in practice: arbitrary choices (pick one layer)
- Principled way?

Defining similarity by undissociability

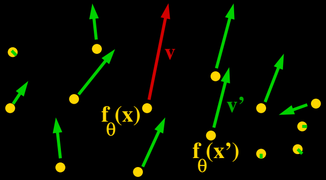
- Given a trained neural network f_θ
- and two input points x and x′
- how similar are x and x′ **for the network**?

Output space:



Quantify the influence of a data point x over another one x′ by how much the tuning of parameters θ to obtain a desired output change v for $f_\theta(x)$ will affect $f_\theta(x')$ as well.

Output space:



Influence of x over x′ = how much the tuning of parameters θ to obtain a desired output change v for $f_\theta(x)$ will affect $f_\theta(x')$ as well.

Derivation in 1-dim output case

- To change $f_\theta(x)$ by a small quantity ε, update θ by $\delta\theta = \varepsilon \frac{\nabla_\theta f_\theta(x)}{\|\nabla_\theta f_\theta(x)\|^2}$.
- Indeed, after parameter update, new value at x:

$$f_{\theta+\delta\theta}(x) \quad = \quad f_\theta(x) + \nabla_\theta f_\theta(x) \cdot \delta\theta + O(\|\delta\theta\|^2) \quad = \quad f_\theta(x) + \varepsilon + O(\varepsilon^2).$$

- This parameter change induces a value change at any other point x′ :

$$f_{\theta+\delta\theta}(x') \; = \; f_\theta(x') + \nabla_\theta f_\theta(x') \cdot \delta\theta + O(\|\delta\theta\|^2) \; = \; f_\theta(x') + \varepsilon \frac{\nabla_\theta f_\theta(x') \cdot \nabla_\theta f_\theta(x)}{\|\nabla_\theta f_\theta(x)\|^2} + O(\varepsilon^2).$$

Symmetric similarity

$$k_\theta(x, x') = \frac{\nabla_\theta f_\theta(x)}{\|\nabla_\theta f_\theta(x)\|} \cdot \frac{\nabla_\theta f_\theta(x')}{\|\nabla_\theta f_\theta(x')\|}$$

- kernel, valued in $[-1, 1]$
- Neural Tangent Kernel!

Symmetric similarity

$$k_\theta(x, x') = \frac{\nabla_\theta f_\theta(x)}{\|\nabla_\theta f_\theta(x)\|} \cdot \frac{\nabla_\theta f_\theta(x')}{\|\nabla_\theta f_\theta(x')\|}$$

- kernel, valued in $[-1, 1]$
- Neural Tangent Kernel!

Properties for vanilla neural networks:

Theorem 1

For any real-valued neural network f_θ whose last layer is a linear layer (without any parameter sharing) or a standard activation function thereof (sigmoid, tanh, ReLU...), and for any inputs x and x',

$$k_\theta(x, x') = 1 \implies \nabla_\theta f_\theta(x) = \nabla_\theta f_\theta(x') \implies f_\theta(x) = f_\theta(x')$$

Symmetric similarity

$$k_\theta(\mathrm{x}, \mathrm{x}') = \frac{\nabla_\theta f_\theta(\mathrm{x})}{\|\nabla_\theta f_\theta(\mathrm{x})\|} \cdot \frac{\nabla_\theta f_\theta(\mathrm{x}')}{\|\nabla_\theta f_\theta(\mathrm{x}')\|}$$

- kernel, valued in $[-1, 1]$
- Neural Tangent Kernel!

Properties for vanilla neural networks:

Theorem 2

For any real-valued neural network f_θ without parameter sharing, if $k_\theta(\mathrm{x}, \mathrm{x}') = 1$ for two inputs x, x', then all useful activities computed when processing x are equal to the ones obtained when processing x'.

Link with the *perceptual loss*

- Perceptual loss:

$$\sum_{\text{activities } i \neq 0} \lambda_{\text{layer}(i)} \; a_i(\mathrm{x}) \; a_i(\mathrm{x}')$$

- Our similarity measure for vanilla networks:

$$k_\theta(\mathrm{x}, \mathrm{x}') = \sum_{\text{activities } i} \lambda_i(\mathrm{x}, \mathrm{x}') \; a_i(\mathrm{x}) \; a_i(\mathrm{x}')$$

where $$\lambda_i(\mathrm{x}, \mathrm{x}') = \sum_{\text{neuron } j \text{ using } a_i} \frac{df_\theta(\mathrm{x})}{db_j} \frac{df_\theta(\mathrm{x}')}{db_j}$$

- For parameter-sharing networks:

$$k_\theta(\mathrm{x}, \mathrm{x}') = \sum_{\text{params } i} \left(\sum_{(j,k) \in \mathcal{S}_i} a_k(\mathrm{x}) \frac{df_\theta(\mathrm{x})}{db_j} \right) \left(\sum_{(j,k) \in \mathcal{S}_i} a_k(\mathrm{x}') \frac{df_\theta(\mathrm{x}')}{db_j} \right)$$

$\Longrightarrow$ reflects network invariances (e.g., translation-inv for convnets)

Counting neighbors

- Similarity measure $k_\theta \implies$ notion of neighborhood
- Number of neighbors of point x ?
- Hard-thresholding, for a given threshold $\tau \in [0,1]$:

$$N_\tau(\mathrm{x}) = \sum_{\mathrm{x}'} 1_{k_\theta(\mathrm{x},\mathrm{x}') \geqslant \tau}$$

- Soft estimate:

$$N_S(\mathrm{x}) = \sum_{\mathrm{x}'} k_\theta(\mathrm{x},\mathrm{x}')$$

- The two are linked:

$$\int_{\tau=0}^{1} N_\tau(\mathrm{x}) d\tau = \sum_{\mathrm{x}'} \int_{\tau=0}^{1} 1_{k_\theta(\mathrm{x},\mathrm{x}') \geqslant \tau}\, d\tau = \sum_{\mathrm{x}'} k_\theta(\mathrm{x},\mathrm{x}')\, 1_{k_\theta(\mathrm{x},\mathrm{x}') \geqslant 0} \simeq N_S(\mathrm{x})$$

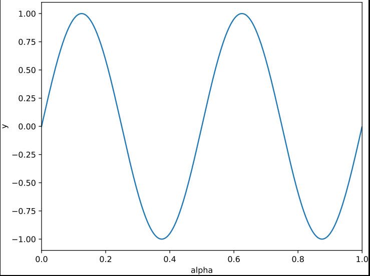
- Low complexity of the soft estimate:

$$N_S(\mathrm{x}) = \sum_{\mathrm{x}'} k_\theta(\mathrm{x},\mathrm{x}') = \sum_{\mathrm{x}'} \frac{\nabla_\theta f_\theta(\mathrm{x})}{\|\nabla_\theta f_\theta(\mathrm{x})\|} \cdot \frac{\nabla_\theta f_\theta(\mathrm{x}')}{\|\nabla_\theta f_\theta(\mathrm{x}')\|} = \frac{\nabla_\theta f_\theta(\mathrm{x})}{\|\nabla_\theta f_\theta(\mathrm{x})\|} \cdot \mathrm{g} \quad \text{with } \mathrm{g} = \sum_{\mathrm{x}'} \frac{\nabla_\theta f_\theta(\mathrm{x}')}{\|\nabla_\theta f_\theta(\mathrm{x}')\|}$$

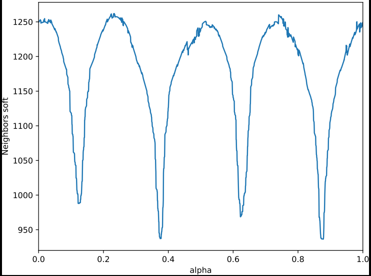
- Very fast to compute! Estimate density at every point in 2 passes over the dataset

Testing these density estimators

- Experiment design: train networks to imitate sinusoids of various frequencies



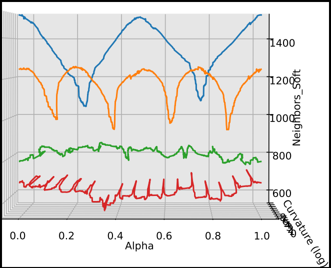
(a) Function to predict.

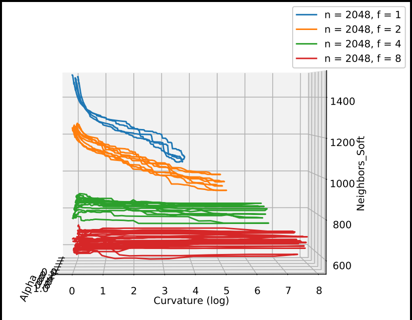


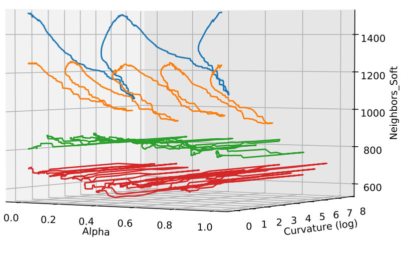
(b) Neighbors soft estimate.

Figure: Toy problem with the frequency $f = 2$.

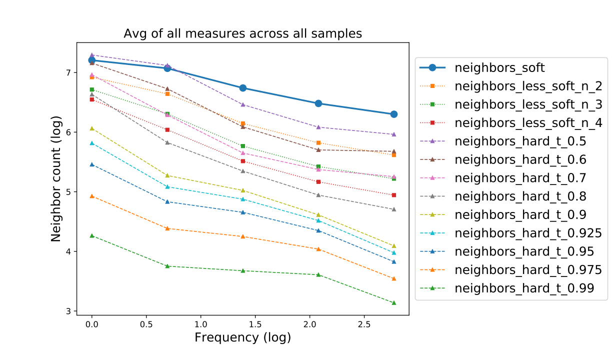
Better viewed in 3D: with curvature







Color: frequency
Depth: curvature
Height: density



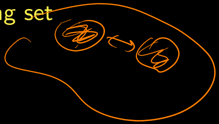
Density estimation using the various approaches (log scale). All approaches behave similarly and show good results, except the ones with extreme thresholds.

Density, so what?

- Test at point x: very low density? $\Longrightarrow$ not reliable prediction!
  - as no neighbor $\Longrightarrow$ independent of training set
  - quantify prediction uncertainty
- Very high density? Might underfit.
  - useful to know during training
- Differentiable quantities... $\Longrightarrow$ possible to optimize them while training!

Density, so what?

- Test at point x: very low density? $\Longrightarrow$ not reliable prediction!
  - as no neighbor $\Longrightarrow$ independent of training set
  - quantify prediction uncertainty
- Very high density? Might underfit.
  - useful to know during training
- Differentiable quantities... $\Longrightarrow$ possible to optimize them while training!



By the way...

- Differentiable similarity estimate $\Longrightarrow$ possible to enforce while training that some examples should be perceived as similar (or different) by the network
- Enforcing similarity on a classification task: small boosting effect (on MNIST...)

Part IV

Back to remote sensing image registration

What do neighbors look like?

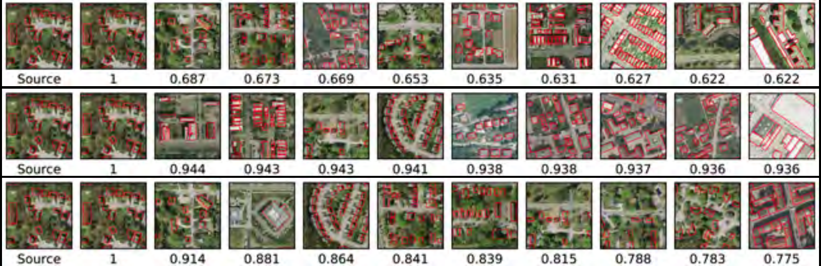


Figure: Example of nearest neighbors for a patch. Each line corresponds to a round. Each patch has its similarity written under it.

(a) Round 1 (b) Round 2 (c) Round 3

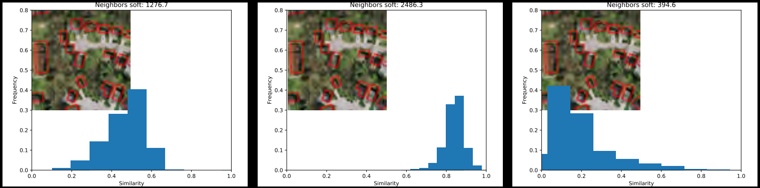


Figure: Histograms of similarities for one patch across rounds.

Back to remote sensing image registration

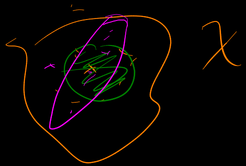
Figure: Closest neighbors to the leftmost patch, using the perceptual loss (first row) and our similarity definition (second row).

Part IV - bis

Back to dataset self-denoising

From similarity statistics to self-denoising effect estimation

$$
\begin{aligned}
\text{input} &: \mathrm{x}_i \\
\text{true (unknown) label} &: y_i \\
\text{(unknown) noise} &: \varepsilon_i \text{ (iid, centered)} \\
\text{noisy (available) label} &: \widetilde{y}_i = y_i + \varepsilon_i \\
\text{predicted label} &: \widehat{y}_i = f_\theta(\mathrm{x}_i) \\
\text{training loss} &: L(\theta) = \textstyle\sum_j ||\widehat{y}_j - \widetilde{y}_j||^2
\end{aligned}
$$



- at convergence $\nabla_\theta L = 0 \implies \mathbb{E}_k[\widehat{y}] = \mathbb{E}_k[\widetilde{y}]$
- $\mathbb{E}_k[a] := \sum_j a_j\, k_\theta(\mathrm{x}_i, \mathrm{x}_j)$: mean value around x_i

$$\widehat{y}_i - \mathbb{E}_k[y] = \mathbb{E}_k[\varepsilon] + (\,\widehat{y}_i - \mathbb{E}_k[\widehat{y}]\,)$$

- $\widehat{y}_i - \mathbb{E}_k[y]$: prediction error to smoothed true labels
- $\mathbb{E}_k[\varepsilon] \;\propto\; \sigma_\varepsilon \, ||k_\theta(\mathrm{x}_i, \cdot)||_{L2} \implies$ denoising factor: 0.02 ($\simeq$ constant)
- Shift: $(\,\widehat{y}_i - \mathbb{E}_k[\widehat{y}]\,)$: 4.4 px (varying) /50

Conclusion

Conclusion

- Defined input similarity as perceived by the neural network
- Skipped the maths for the higher-dim case
- Fast similarity / density estimation
  $\Longrightarrow$ opens the door to underfit/overfit/uncertainty analyses and control
- Similarity enforced during training: dataset-dependent boosting effect (cf supp.mat.)
- Extended Noise2Noise to non-identical inputs: self-denoising effect as a function of inputs similarities
- Links with Neural Tangent Kernel [4]: same concept! used differently
- Code available on GitHub:
  `http://github.com/Lydorn/netsimilarity`

Recent news:

- Our first citation! [Hanawa et al.]
- Comparison of several criteria for similar image retrieval $\Longrightarrow$ ranked first!
- It seems they did not compute the right quantity...

Papers

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka.
Input similarity from the neural network perspective.
In Thirty-third Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, December 2019.

Nicolas Girard, Guillaume Charpiat, and Yuliya Tarabalka.
Aligning and Updating Cadaster Maps with Aerial Images by Multi-Task, Multi-Resolution Deep Learning.
In Asian Conference on Computer Vision (ACCV), Perth, Australia, December 2018.

Emmanuel Maggiori, Guillaume Charpiat, Yuliya Tarabalka, and Pierre Alliez.
Recurrent neural networks to enhance satellite image classification maps.
TGRS, abs/1608.03440, 2016.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez.
Convolutional Neural Networks for Large-Scale Remote Sensing Image Classification.
IEEE Transactions on Geoscience and Remote Sensing, September 2016.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez.
Fully Convolutional Neural Networks For Remote Sensing Image Classification.
In IEEE International Geoscience and Remote Sensing Symposium, Beijing, China, July 2016. IEEE GRSS.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez.
Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark.
In IEEE International Symposium on Geoscience and Remote Sensing (IGARSS), Fort Worth, United States, July 2017.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez.
High-resolution aerial image labeling with convolutional neural networks.
IEEE Transactions on Geoscience and Remote Sensing, 55(12):7092–7103, Dec 2017.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez.
High-resolution image classification with convolutional networks.
In IEEE International Symposium on Geoscience and Remote Sensing (IGARSS), Fort Worth, United States, July 2017.

Armand Zampieri, Guillaume Charpiat, Nicolas Girard, and Yuliya Tarabalka.
Multimodal image alignment through a multiscale chain of neural networks with application to remote sensing.
In European Conference on Computer Vision (ECCV), Munich, Germany, September 2018.