Recurrent Neural Networks to Enhance Satellite Image Classification Maps

Emmanuel Maggiori
Inria - TITANE
emmanuel.maggiori@inria.fr

Yuliya Tarabalka
Inria - TITANE

Guillaume Charpiat
Inria - TAO

Pierre Alliez
Inria - TITANE

Abstract

The automatic pixelwise labeling of satellite images is of paramount importance in remote sensing. Convolutional neural networks represent a competitive means to learn the contextual features required to distinguish an object class from the rest. However, spatial precision is usually lost in the process, leading to coarse classification maps that do not accurately outline the objects.

While specific enhancement algorithms have been designed in the literature to improve such coarse neural network outputs, it requires a decision-making process to choose and tune the right enhancement algorithm. Instead, we aim at learning the enhancement algorithm itself. We consider the class of partial differential equations, see them as iterative enhancement processes, and observe that they can be expressed as recurrent neural networks. Consequently, we train a recurrent neural network from manually labeled data for our enhancement task. In a series of experiments we show that our network effectively learns an iterative process that significantly improves the quality of satellite image classification maps.

1. Introduction

One of the most explored problems in remote sensing is the pixelwise labeling of satellite imagery. Such a labeling is used in a wide range of practical applications such as precision agriculture and urban planning. Recent technological developments have substantially increased the availability and resolution of satellite data. Besides the computational complexity issues that arise, these advances are posing new challenges in the processing of the images. Notably, the fact that large surfaces are covered introduces a significant variability in the appearance of the objects. In addition, the fine details in high resolution images make it difficult to classify the pixels from elementary cues. For example, the different parts of an object often contrast more with each other than with other objects [1]. Using high-level contextual features thus plays a crucial role at distinguishing object classes.

Convolutional neural networks (CNNs) [15] are receiving an increasing attention, due to their capability to automatically discover relevant contextual features in image categorization problems. CNNs have already been used in the context of remote sensing [22, 26], featuring powerful detection capabilities. However, when the goal is to label images at the pixel level, the output classification maps are too coarse. For example, buildings are successfully identified but their boundaries in the classification map rarely coincide with the real object boundaries. We can identify two main reasons for this coarseness in the classification:

a) There is a structural limitation of CNNs to carry out fine-grained classification. If we wish to keep a low number of learnable parameters, the capability of learning long-range contextual features comes at the cost of losing spatial accuracy, i.e., a trade-off between detection and localization. This is a well-known issue and still a scientific challenge [5, 19].

b) In the specific context of remote sensing imagery, there is a significant lack of spatially accurate reference data for training. For example, the OpenStreetMap collaborative database provides large amounts of free-access maps over the earth, but irregular misregistrations and omissions are frequent all over the dataset. In such circumstances, CNNs cannot do better than learning rough estimates of the objects' locations, given that the boundaries are hardly located on real edges in the training set.

Let us remark that in the particular context of high-resolution satellite imagery, the spatial precision of the classification maps is of paramount importance. Objects are small and a boundary misplaced by a few pixels significantly hampers the overall classification quality. In other application domains, such as semantic segmentation, while there have been recent efforts to better shape the output objects, a high resolution output seems to be less of a priority. For example, in the popular Pascal VOC semantic segmentation dataset, there is a band of several unlabeled pixels around the objects, where accuracy is not computed to assess the performance of the methods.

To overcome the structural issues that lead to coarse classification maps, a recent tendency is to correct the CNN predictions in a second step, in order to better align them

(a) (b)

Figure 1: Samples of reference data for buildings. (a) Imprecise OpenStreetMap data. (b) Manually labeled data.

to image edges. A pioneering work in this direction was presented by Chen et al. [5], in the context of semantic segmentation of natural scenes, in which the coarse classification maps are iteratively refined taking into account the input color image. In our problem setting we can have access to large amounts of imprecisely labeled data (see Fig. 1a) but only a very small dataset of carefully labeled imagery done by hand (see Fig. 1b). We therefore adopt a two-step scheme: the imperfect data is used to train a system to learn the generalities of the object classes and produce coarse classification maps, and the small dataset is used to train a system to enhance these coarse maps. The goal of the enhancement step is to better align the object boundaries to relevant image features, producing a high-resolution output classification map. Different iterative algorithms have been presented in the literature to enhance classification maps, such as conditional random fields [5, 30] and domain transform filtering [4].

Instead of designing and tuning a particular enhancement algorithm, our goal is to let the system learn the appropriate process itself. We aim at providing a system that not only automatically learns relevant features but also the rationale behind the enhancement technique, thus reducing algorithm design efforts and leveraging the power of machine learning. For this, we first formulate a generic partial differential equation governing a family of iterative refinement algorithms. This equation conveys the idea of progressively refining a classification map based on local cues. We then observe that such an equation can be expressed as a combination of common neural network layers. This leads us to defining a recurrent neural network (RNN) that discovers the appropriate iterative algorithm from training data. Our resulting RNN requires little human intervention: we provide the network with a small piece of manually labeled image, and train it to learn how to improve the classification maps. The network automatically discovers relevant data-dependent features to enhance the classification as well as the equations that govern every enhancement iteration.

2. Related work

A common way to tackle the aerial image labeling problem is to use classifiers such as support vector machines [2] or neural networks [21] on the individual pixel spectral signatures (i.e., a pixel's "color" but not limited to RGB bands). In some cases, a few neighboring pixels are analyzed jointly to enhance the prediction and enforce the spatial smoothness of the output classification maps [9]. Hand-designed features such as textural features have also been used [18]. The use of an iterative classification enhancement process on top of hand-designed features has also been explored in the context of image labeling [25].

Following the recent advent of deep learning and to address the new challenges posed by large-scale aerial imagery, Mnih [22] used CNNs to learn long-range spatial contextual features in order to produce classification maps. Penatti et al. [23] used CNNs to assign aerial image patches to categories (e.g., 'residential', 'harbor') and Vakalopoulou et al. [26] addressed building detection using CNNs. These networks require some degree of downsampling in order to take a large context with a reduced number of parameters. They perform well at detecting the presence of objects but do not outline them accurately.

Our work can also be related to the area of natural image semantic segmentation. Notably, fully convolutional networks (FCN) [19] are becoming increasingly popular to conduct pixelwise image labeling. FCN networks are made up of a stack of convolutional and pooling layers followed by so-called deconvolutional layers that upsample the resolution of the classification maps, possibly combining features at different scales. The output classification maps being too coarse, the authors of the Deeplab network [5] added a fully connected conditional random field (CRF) on top of both the FCN and the input color image, in order to enhance the classification maps.

Zheng et al. [30] recently reformulated the fully connected CRF of Deeplab as an RNN, and Chen et al. [4] designed an RNN that emulates the domain transform filter [10]. Such a filter is used to sharpen the classification maps around image edges, which are themselves detected with a CNN. In these methods the refinement algorithm is designed beforehand and only few parameters that rule the algorithm are learned as part of the network's parameters. The relevance of these approaches is that both steps (coarse classification and enhancement) can be seen as a single end-to-end neural network and optimized simultaneously.

Instead of predefining the algorithmic details as in previous work, we formulate a general iterative refinement algorithm through an RNN and let the network learn the specific algorithm. To our knowledge, little work has explored the idea of learning an iterative algorithm. A family of PDEs has been expressed as a linear combination of fundamental differential invariants, the linear coefficients being learned for different problems such as image denoising and inpainting [16, 17]. Morever, an RNN has been used to learn a reaction diffusion process for image restoration [6].

3. Partial differential equations (PDEs)

Let us assume we are given a set of score (or "heat") maps u_k associated to every possible class $k \in \mathcal{L}$ in a pixelwise labeling problem. The score of a pixel reflects the likelihood to belong to a class, according to the classifier's predictions. The final class assigned to every pixel is the one with maximal value u_k. Alternatively, a softmax function can be used to interpret the results as probability scores: $P(k) = e^{u_k} / \sum_{j \in \mathcal{L}} e^{u_j}$.

Our goal is to combine the score maps u_k with information derived from the input image (e.g., edge features) to sharpen the scores near the real objects in order to enhance the classification. We now analyze different enhancement approaches before generalizing them.

We can formulate a variety of diffusion processes applied to the maps u_k as partial differential equations (PDEs). For example, the heat flow is described as:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(\nabla u_k(x)), \tag{1}$$

where $\mathrm{div}(\cdot)$ denotes the divergence operator in the spatial domain of x. Applying such a diffusion process in our context would smooth out the heat maps. Instead, our goal is to design an image-dependent smoothing process that aligns the heat maps to the image features. A natural way of doing this is to modulate the gradient in Eq. 1 by a scalar function $g(x, I)$ that depends on the input image I:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(g(I, x)\nabla u_k(x)). \tag{2}$$

Eq. 2 is similar to the Perona-Malik diffusion [24] with the exception that Perona-Malik uses the smoothed function itself to guide the diffusion. $g(I, x)$ denotes an edge-stopping function that takes low values near borders of $I(x)$ in order to slow down the smoothing process there. Several choices about $g(I, x)$ must be made to apply this approach. For example, one must choose the type of function such as exponential or rational [24], which in turn requires to set an edge-sensitivity parameter. Several extensions of this approach have also been proposed, such as a popular regularized variant that computes the gradient on a Gaussian-smoothed version of the input image [28].

Another possibility is to consider a more general variant in which $g(I, x)$ is replaced by a matrix $D(I, x)$. This matrix acts as a diffusion tensor that redirects the flow based on image properties instead of just slowing it down near edges:

$$\frac{\partial u_k(x)}{\partial t} = \mathrm{div}(D(I, x)\nabla u_k(x)). \tag{3}$$

This formulation corresponds to the anisotropic diffusion process [28], which requires to design $D(I, x)$.

Alternatively, one can draw some inspiration from the level set framework. For example, the geodesic active contours technique formulated as level sets translates into:

$$\frac{\partial u_k(x)}{\partial t} = |\nabla u_k(x)| \mathrm{div}\left(g(x, I) \frac{\nabla u_k(x)}{|\nabla u_k(x)|} \right). \tag{4}$$

Such formulation favors the zero level set to align with minima of $g(x, I)$ [3]. When scaling maps u_k so that segmentation boundaries match zero levels, formulations inspired by Eq. 4 may be used to enhance the classification maps.

As shown above, many different PDEs can be formulated to enhance classification maps. However, several choices must be made to select the appropriate PDE and tailor it to our problem. Instead of using trial-and-error processes to perform such design, our goal is to let a machine learning approach discover the appropriate PDE.

PDEs are usually discretized in space by using finite differences, which represent derivatives as discrete convolution filters. We then write a generic discrete formulation as follows:

$$\begin{aligned} \frac{\partial u_k(x)}{\partial t} = f_k\big((M_1 * u_k)(x), (M_2 * u_k)(x), ..., \\ (N_1 * I)(x), (N_2 * I)(x), ...\big), \end{aligned} \tag{5}$$

where $\{M_1, M_2, ...\}$ and $\{N_1, N_2, ...\}$ denote convolution kernels applied to the heat maps u_k and image I respectively. These convolution kernels may look like gradient filters in different directions (such as Sobel filters), second-derivative filters (such as Laplacians), etc., depending on the PDE. The response to the filters at a specific location is then inputted to a function f_k that computes the rest of the PDE. The convolutions convey all the spatial reasoning, such as image derivatives, while f_k conveys the calculations done on top, such as the products and function $g(x, I)$ in Eqs. 2 and 4. The family of PDEs discretized by finite differences are covered by this formulation, including (1)-(4).

PDEs are commonly discretized in time and solved by gradient descent as follows:

$$u_{k,t+1}(x) = u_{k,t}(x) + \tau \frac{\partial u_{k,t}(x)}{\partial t} = u_{k,t}(x) + \delta u_{k,t}(x), \tag{6}$$

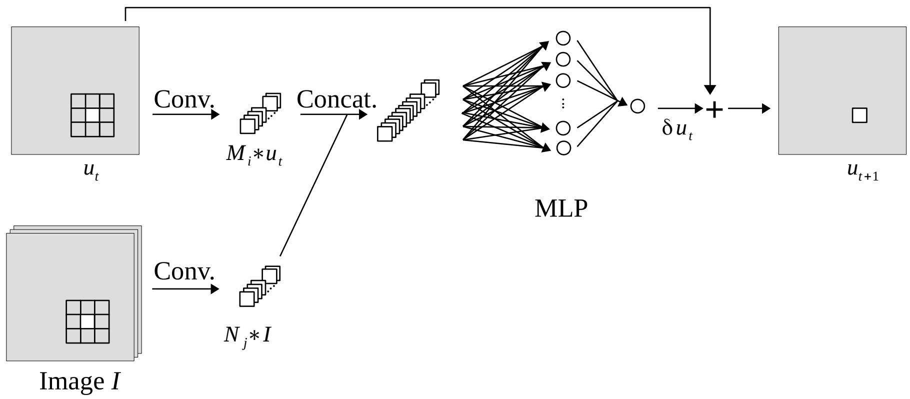


Figure 2: One PDE iteration represented as common neural network layers.

where τ denotes a time-step parameter and $\delta u_{k,t}(x)$ the overall update of $u_{k,t}$ at time t.

Note that the spatial features obtained by convolving the different M_i and N_j are shared across the classes, f_k determining the contribution of each feature to the PDE. This covers the case in which different evolution equations might be optimal for different classes, even if just in terms of a different time-step τ. In the next section we detail a way to learn the update functions $\delta u_{k,t}(x)$ from training data.

4. PDEs as recurrent neural networks

We now show that one PDE iteration, as defined in Eqs. 5-6, can be expressed in terms of common neural network layers. Let us focus on a single pixel for a specific class, simplifying the notation from $u_{k,t}(x)$ to u_t. Fig. 2 illustrates the proposed network architecture. Each iteration takes as input the image I and u_t, a given heat map to enhance at time t. In the first iteration, u_t is the initial coarse heat map to be improved, outputted by another pre-trained neural network in our case. From the heat map u_t we derive a series of filter responses, which correspond to $M_i * u_t$ in Eq. 5. These responses are found by computing the dot product between a set of filters M_i and the values of u_t in a spatial neighborhood of a given point x. Analogously, a set of filter responses are computed on the input image, corresponding to the different $N_j * I$ of Eq. 5. These operations are convolutions when performed densely in space, $N_j * I$ and $M_i * u_t$ being feature maps of the filter responses.

These filters are then "concatenated", forming a pool of features coming from both the input image and the heat map (i.e., the variables in Eq. 5). We must now learn the function δu_t that describes how the heat map u_t is updated at iteration t, based on these features.

Function δu_t can be potentially anything, since no constraints on f_k are introduced in (5). In (1)-(4), for example, it includes products between different terms and inverse functions (through $g(x, I)$). We therefore model δu_t through a multilayer perceptron (MLP), because it can approximate any function within a bounded error. We include one hidden layer with nonlinear activation functions followed by an output neuron with a linear activation (a typical configuration for regression problems), although other MLP architectures could be used as well. Applying this MLP densely is equivalent to performing convolutions with 1×1 kernels at every layer. The implementation to densely label entire images is then straightforward.

The value of δu_t is then added to u_t in order to generate the updated map u_{t+1}. This addition is performed pixel by pixel in the case of a dense input. Note that although we could have removed this addition and let the MLP directly output the updated map u_{t+1}, we opted for this architecture since it is more closely related to the equations and better conveys the intention of a progressive refinement of the classification map.

The overall iterative process is implemented by unrolling a finite number of iterations, as illustrated in Fig. 3, under the constraint that the parameters are shared among all iterations. Such sharing is enforced at training time by a simple modification to the back-propagation training algorithm where the derivatives of every instance of a weight at different iterations are averaged [29]. Note that issues with vanishing or exploding gradients may arise when too many iterations are unrolled, an issue inherent to deep network architectures. Note also that the spatial features are shared across the classes, while a different MLP is learned for each of them, following Eq. 5. As depicted by Fig. 3 and conveyed in the equations, the features extracted from the input image are independent of the iteration.

The RNN of Fig. 3 represents then a dynamical system that iteratively refines the class heat maps. Training such an RNN amounts to finding the optimal dynamical system for our refinement task.

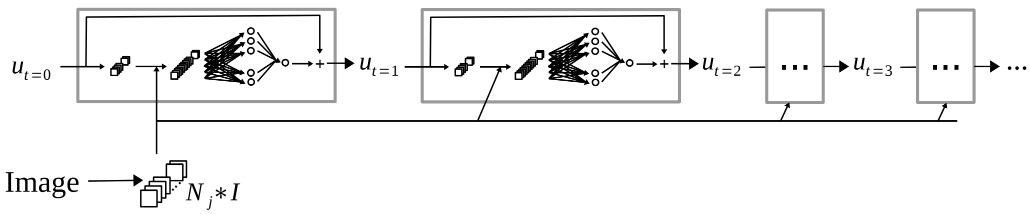


Figure 3: Modules of Fig. 2 are stacked (while sharing parameters) to implement an RNN.

5. Implementation details

We first describe the CNN used to produce the coarse predictions, then detail our RNN. The network architecture was implemented using Caffe deep learning library [13].

Our coarse prediction CNN is based on a previous remote sensing network presented by Mnih [22]. We create a fully convolutional [19] version of Mnih's network, since recent remote sensing work has shown the theoretical and practical advantages of this type of architecture [14, 20]. The CNN takes 3-band color image patches at 1m^2 resolution and produces as many heat maps as classes considered. The resulting four-layer architecture is as follows: 64 conv. filters (12×12, stride 4) $\rightarrow$ 128 conv. filters (3×3) $\rightarrow$ 128 conv. filters (3×3) $\rightarrow$ 3 conv. filters (9×9). Since the first convolution is performed with a stride of 4, the resulting feature maps have a quarter of the input resolution. Therefore, a deconvolutional layer [19] is added on top to upsample the classification maps to the original resolution. The activation functions used in the hidden layers are rectified linear units. This network is trained on patches randomly selected from the training dataset. We group 64 patches with classification maps of size 64×64 into minibatches (following [22]) to estimate the gradient of the network's parameters and back-propagate them. Our loss function is the cross-entropy between the target and predicted class probabilities. Stochastic gradient descent is used for optimization, with learning rate 0.01, momentum 0.9 and an L2 weight regularization of 0.0002.

Let us now detail the implementation of the RNN described in Sec. 4. We unroll five RNN iterations and learn 32 M_i and 32 N_j filters, both of spatial dimensions 5×5. As explained in Sec. 4, an independent MLP is learned for every class, using 32 hidden neurons each and with rectified linear activations. Training is performed on random patches and with the cross-entropy loss function, as done with the coarse CNN. The gradient descent algorithm used is AdaGrad [8], which exhibits a faster convergence in our case, using a base learning rate of 0.01 (higher values make the loss diverge). All weights are initialized randomly by sampling from a distribution that depends on the number of neuron inputs, as suggested in [11]. We trained the RNN for 50,000 iterations, until observing convergence of the training loss, which took around four hours with a single GPU.

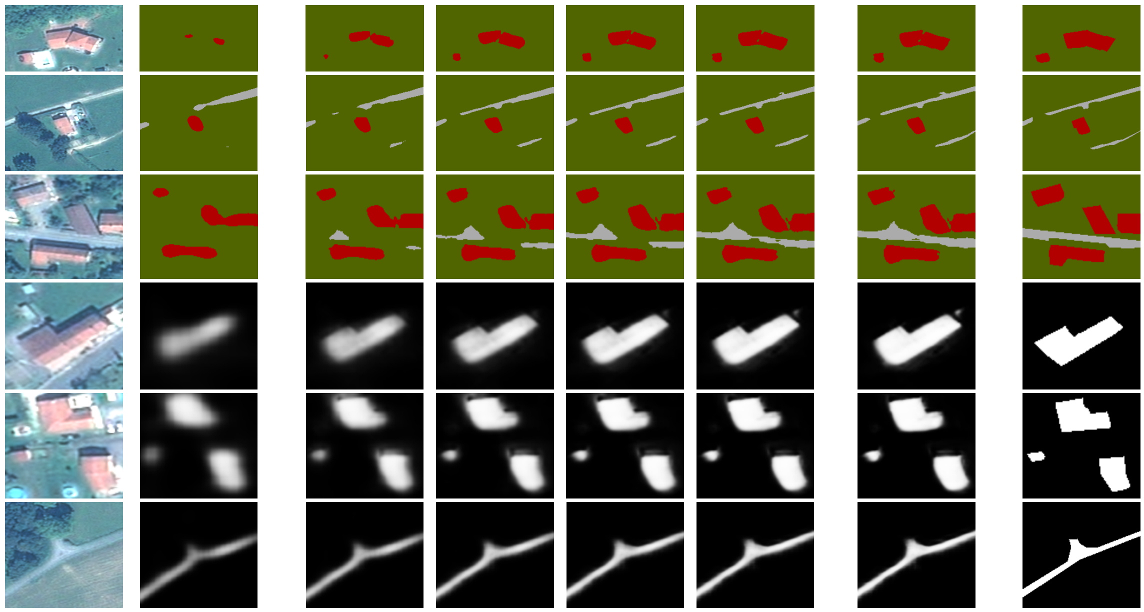
6. Experiments

We perform our experiments on images acquired by a Pléiades satellite over the area of Forez, France. A color image is used, obtained by pansharpening [27] the satellite data, which provides a spatial resolution of 0.5 m^2. Since the networks described in Sec. 5 are specifically designed for images with a 1m^2 resolution, we downsample the Pléiades images before feeding them to our networks and bilinearly upsample the outputs.

From this image we selected an area with OpenStreetMap (OSM) [12] coverage to create a 22.5 km^2 training dataset for the classes *building*, *road* and *background*. The reference data was obtained by rasterizing the raw OSM maps. Misregistrations and omissions are present all over the dataset (see Fig. 1a and further examples in suppl. material). Buildings tend to be misaligned or omitted, while many roads in the ground truth are not visible in the image (or the other way around). This is the dataset used to train the initial coarse CNNs.

We manually labeled two 2.25 km^2 tiles to train and test the RNN at enhancing the predictions of the coarse network. We denote them by enhancement and test sets, respectively. Note that our system must discover an algorithm to refine an existing classification map, and not to conduct the classification itself, hence a smaller training set should be sufficient for this stage.

In the following, we report the results obtained by using the proposed RNN on the Pléiades dataset. Fig. 4 provides closeups of results on different fragments of the test dataset. The initial and final maps (before and after the RNN enhancement) are depicted, as well as the intermediate results through the RNN iterations. We show both a set of final classification maps and some single-class fuzzy probability maps. We can observe that as the RNN iterations go by, the classification maps are refined and the objects better align to image edges. The fuzzy probabilities become more confident and sharpen their boundaries. To quantitatively assess this improvement we compute two measures on the test set: the overall accuracy (proportion of correctly classified pixels) and the intersection over union (IoU) [19]. Mean IoU has become the standard in semantic segmentation since it is more reliable in the presence of imbalanced classes (such as *background* class, which is included to compute the

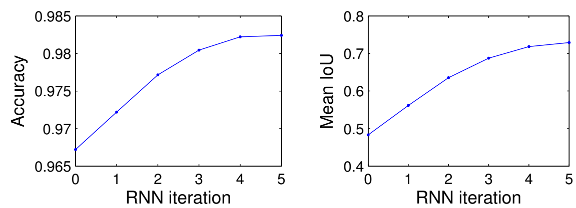


Color CNN map — Intermediate RNN iterations — RNN output Ground truth
(RNN input)

Figure 4: Evolution of fragments of classification maps (top rows) and single-class fuzzy scores (bottom rows) through RNN iterations.

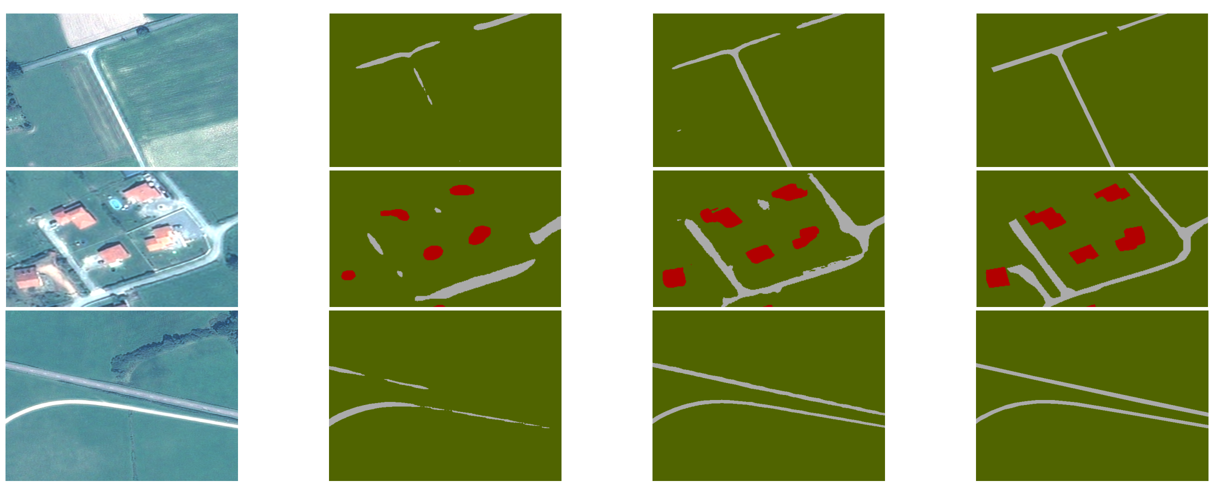
Method	Overall accuracy	Mean IoU	Class-specific IoU Build.	Road	Backg.
CNN	96.72	48.32	38.92	9.34	96.69
CNN+CRF	96.96	44.15	29.05	6.62	96.78
CNN+RNN$^{=}$	97.78	65.30	59.12	39.03	97.74
CNN+RNN	**98.24**	**72.90**	**69.12**	**51.32**	**98.20**

(a) Numerical comparison (in %)



(b) Evolution through RNN iterations

Figure 5: Quantitative evaluation on Pléaiades images test set over Forez, France.



Color image Coarse CNN classif. RNN output Ground truth

Figure 6: Initial coarse classifications and the enhanced maps by using RNNs.

mean) [7]. As summarized in the table of Fig. 5(a), the performance of the original coarse CNN (denoted by CNN) is significantly improved by attaching our RNN (CNN+RNN). Both measures increase monotonously along the intermediate RNN iterations, as depicted in Fig. 5(b).

The initial classification of roads has an overlap of less than 10% with the roads in the ground truth, as shown by its individual IoU. The RNN makes them emerge from the background class, now overlapping the ground truth roads by over 50%. Buildings also become better aligned to the real boundaries, going from less than 40% to over 70% overlap with the ground truth buildings. This constitutes a multiplication of the IoU by a factor of 5 for roads and 2 for buildings, which indicates a significant improvement at outlining and not just detecting objects.

Additional visual fragments before and after the RNN refinement are shown in Fig. 6. We can observe in the last row how the iterative process learned by the RNN both thickens and narrows the roads depending on the location.

We also compare our RNN to the approach in [5] (here denoted by CNN+CRF), where a fully-connected CRF is coupled both to the input image and the coarse CNN output, in order to refine the predictions. This is the idea behind the so-called Deeplab network, which constitutes one of the most important current baselines in the semantic segmentation community. While the CRF itself could also be implemented as an RNN [30], we here stick to the original formulation because the RNN is only interesting if we want to train the system end to end (i.e., together with the coarse prediction network). In our case we wish to leave the coarse network as is, otherwise we risk overfitting it to this much smaller set. We thus simply use the CRF as in [5] and tune the energy parameters by performing a grid search using the enhancement set as a reference. Five iterations of inference on the fully-connected CRF were preformed in every case.

To further analyze our method, we also consider an alternative enhancement RNN in which the weights of the MLP are shared across the different classes (which we denote by CNN+RNN$^=$). This forces the system to learn the same function to update all the classes, instead of a class-specific function.

Numerical results are included in the table of Fig. 5(a) and visual fragments are compared in Fig. 7. The CNN+CRF approach does sharpen the maps but this often occurs around the wrong edges. It also makes small objects disappear in favor of larger objects (usually the background class) when edges are not well marked, which explains the mild increase in overall accuracy but the decrease in mean IoU. While the CNN+RNN$^=$ outperforms the CRF, both quantitative and visual results are beaten by the CNN+RNN, supporting the importance of learning a class-specific enhancement function. In Fig. 8 we illustrate results for a large surface. We can observe a consistent improvement of the initial map by using the CNN+RNN method. The suppl. material includes the visualization of different filters learned by the RNN.

To validate the importance of using a recurrent architecture, and following Zheng et al. [30], we retrained our system considering every iteration of the RNN as an independent step with its own parameters. After training for the same number of iterations, it yields a lower performance on the test set compared to the RNN and a higher performance on the training set. If we keep on training, the non-recurrent network still enhances its training accuracy while performing poorly on the test set, implying a significant degree of overfitting with this variant of the architecture. This provides evidence that constraining our network to learn an iterative refinement process is crucial for its success.

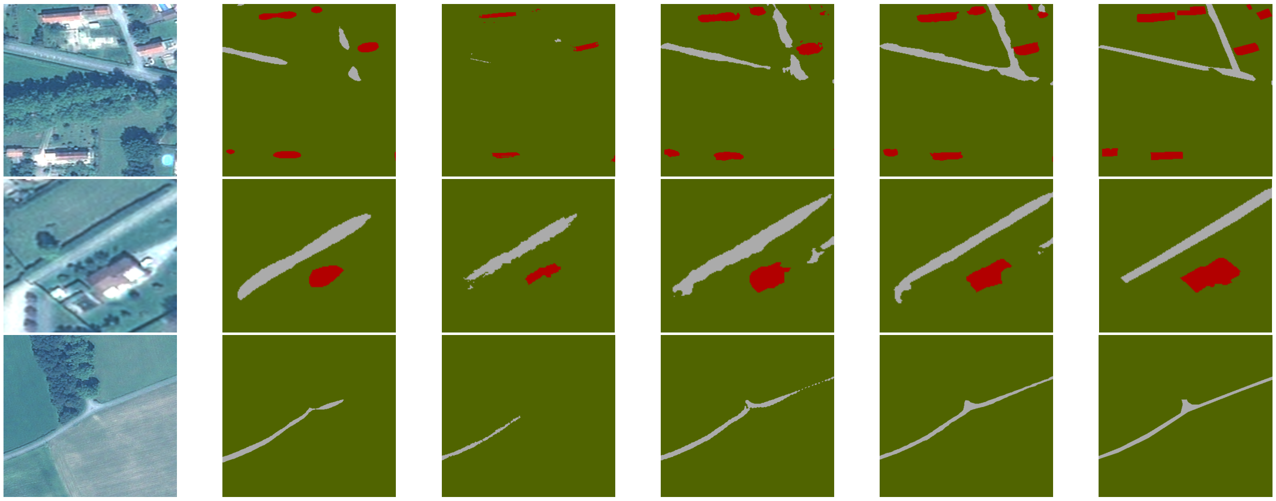
7. Concluding remarks

We presented a recurrent neural network (RNN) that learns how to refine the coarse output of another neural network. Little human intervention is required in the process since the specifics of the refinement algorithm are not provided by the user but learned by the recurrent network itself. After having analyzed different iterative alternatives we devised a general formulation that can be interpreted as a stack of common neuron layers. The inputs to the our RNN are both the coarse classification maps to be refined and the input color image. The output at every RNN iteration is an update to the classification map of the previous iteration. At training time, the RNN discovers the relevant features to be taken both from the classification map and from the input image, as well as the function that combines them.

The experiments confirm that our system learns an iterative process that progressively enhances satellite image classification maps. These maps are improved significantly, multiplying the overlap of the foreground classes with the ground truth by several times and outperforming other approaches. Thus, the method not only detects but also outlines the objects. Such improvement is obtained without specific instructions about the enhancement algorithm other than providing the RNN architecture and the training data. This is an innovative way of using neural networks, seeing them as machines that learn iterative algorithms. In the future we plan to investigate the use of this framework to solve other problems.

Acknowledgment

All Pléiades images are ©CNES (2012 and 2013), distribution Airbus DS / SpotImage. The authors would like to thank CNES for initializing and funding the study, and providing Pléiades data.



Color image | Coarse CNN | CNN+CRF | CNN+RNN$^{=}$ | CNN+RNN | Ground truth

Figure 7: Visual comparison on closeups of the Pléiades dataset.

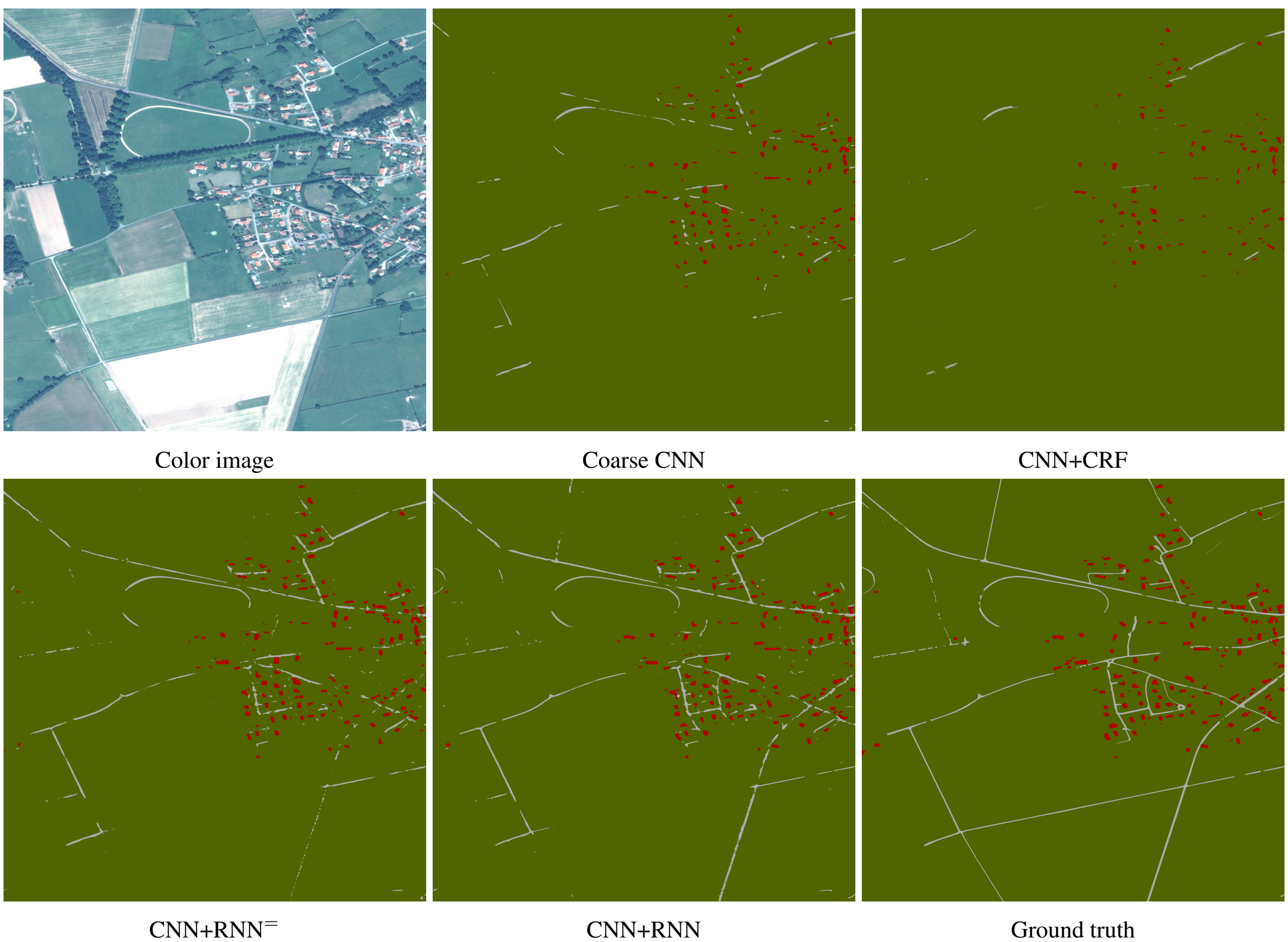


Figure 8: Visual comparison on a Pléiades satellite image tile of size 3000×3000 covering 2.25 km^2.

References

[1] H. G. Akçay and S. Aksoy. Building detection using directional spatial constraints. In *IEEE IGARSS*, pages 1932–1935, 2010.

[2] G. Camps-Valls and L. Bruzzone. Kernel-based methods for hyperspectral image classification. *IEEE TGRS*, 43(6):1351–1362, 2005.

[3] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[4] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, May 2015.

[6] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *IEEE CVPR*, pages 5261–5269, 2015.

[7] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan. What is a good evaluation measure for semantic segmentation?. In *BMVC*, 2013.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[9] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton. Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675, 2013.

[10] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, 2011.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[12] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *IEEE CVPR Workshops*, pages 1–9, 2016.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] R. Liu, Z. Lin, W. Zhang, and Z. Su. Learning PDEs for image restoration via optimal control. In *ECCV*, pages 115–128. Springer, 2010.

[17] R. Liu, Z. Lin, W. Zhang, K. Tang, and Z. Su. Toward designing intelligent pdes for computer vision: An optimal control approach. *Image and vision computing*, 31(1):43–56, 2013.

[18] C. Lloyd, S. Berberoglu, P. J. Curran, and P. Atkinson. A comparison of texture measures for the per-field classification of mediterranean land cover. *International Journal of Remote Sensing*, 25(19):3943–3965, 2004.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.

[20] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Fully convolutional neural networks for remote sensing image classification. In *IEEE IGARSS*, 2016.

[21] J. Mas and J. Flores. The application of artificial neural networks to the analysis of remotely sensed data. *International Journal of Remote Sensing*, 29(3):617–663, 2008.

[22] V. Mnih. *Machine learning for aerial image labeling*. PhD thesis, University of Toronto, 2013.

[23] O. Penatti, K. Nogueira, and J. Santos. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *IEEE CVPR Workshops*, pages 44–51, 2015.

[24] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE TPAMI*, 12(7):629–639, 1990.

[25] Z. Tu. Auto-context and its application to high-level vision tasks. In *IEEE CVPR*, pages 1–8. IEEE, 2008.

[26] M. Vakalopoulou, K. Karantzalos, N. Komodakis, and N. Paragios. Building detection in very high resolution multispectral data with deep learning features. In *IEEE IGARSS*, pages 1873–1876, 2015.

[27] Z. Wang, D. Ziou, C. Armenakis, D. Li, and Q. Li. A comparative analysis of image fusion methods. *IEEE TGRS*, 43(6):1391–1402, 2005.

[28] J. Weickert. *Anisotropic diffusion in image processing*. Teubner Stuttgart, 1998.

[29] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[30] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *IEEE CVPR*, pages 1529–1537, 2015.