

A DEEP NETWORK APPROACH TO MULTITEMPORAL CLOUD DETECTION

*Devis Tuia*¹, *Benjamin Kellenberger*¹, *Adrian Pérez-Suay*², *Gustau Camps-Valls*²

¹ Laboratory of GeoInformation Science and Remote Sensing, Wageningen University, The Netherlands

²Image Processing Laboratory, Universitat de València, iSpain

{devis.tuia,benjamin.kellenberger}@wur.nl, {gustau.camps,adrian.perez}@uv.es

ABSTRACT

We present a deep learning model with temporal memory to detect clouds in image time series acquired by the Seviri imager mounted on the Meteosat Second Generation (MSG) satellite. The model provides pixel-level cloud maps with related confidence and propagates information in time via a recurrent neural network structure. With a single model, we are able to outline clouds along all year and during day and night with high accuracy.

Index Terms— Cloud detection, Seviri, deep learning, convolutional neural networks, recurrent neural networks

1. INTRODUCTION

Cloud detection is paramount for a wide amount of tasks exploiting remote sensing optical data. For example, in the case of ground-based targets, clouds mask the Earth’s surface and provide incorrect reflectance values. In this case, they must be excluded (as other missing data) and replaced before the image can be further used: common strategies include temporal interpolation along a time series [1] or spatial geostatistics [2]. Another example is weather forecasting, where geostationary satellite missions like the Meteosat Second Generation (MSG) acquire wide swath images repetitively covering the same portions of the globe, making fine registration of the single acquisitions compulsory for image navigation and geometric calibration [3]: such registration is generally performed by using landmark points defined all over the globe (Fig. 1). In this case, it is compulsory to know when such landmarks are covered by clouds, since the registration may become inaccurate [4]. In this paper we tackle this second scenario.

We consider the problem of cloud detection over a time series covering a landmark repetitively imaged by the Seviri instrument. In [4], the task is approached at the single pixel level and by a classical ‘feature extraction, then classification’ approach, where a number of spectral and contextual features

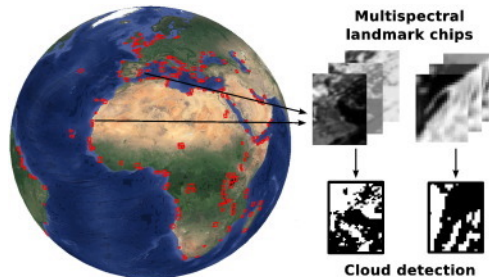


Fig. 1: Principle of cloud detection over landmark locations.

are first extracted at the image level and then each pixel is used in a committee of experts formed by dedicated Support Vector Machine classifiers for single pixel cloud classification. The system runs four separate models, specializing to four different periods of the day characterized defined by specific sun zenith angles.

In this paper, we approach the problem from a similar perspective, since we also design a system first extracting contextual features from the spectra and then deploying a pixel-classifier. However, we propose a *single, end-to-end* framework, where features and classifier are learned by a deep Convolutional Neural Network (CNN). CNNs have become a prominent framework of interest in remote sensing [5]: their characteristic to learn naturally contextual features at multiple scales [6] makes them a promising framework for pixel-level, dense semantic segmentation (in our case) of infrared images. We limited our inputs to the infrared bands of the Seviri instrument (we omitted the RGB bands) to have an input space that is informative, both during day and night, and thus design a single CNN classifier providing the cloud mask for the image, independently from the specific time of acquisition. Two alternatives could be envisioned: either training two models, one for the day and another for the night (similarly to the strategy in [4]), or still training a single model with all the bands, while using systematic dropout to the RGB channels during nighttime. If the first strategy doubles to computational load with minor accuracy improvements, the second led to uninformative features being learned for the RGB bands, (i.e. the model learned to ignore them), so we decided to work only with the IR channels and with a single model.

DT and BK acknowledge the Swiss National Science Foundation (no. PP00P2-150593). APS and GCV acknowledge the support by the Spanish Ministry of Economy and Competitiveness (MINECO-ERDF, TIN2015-64210-R and TEC2016-77741-R projects), the EUMETSAT Contract No. EUM/RSP/SOW/14/762293, and by the European Research Council (ERC) under the ERC-CoG-2014 SEDAL under grant agreement 647423.

Despite the good results provided by CNN approaches, proceeding this way ignores the temporal structure of the data. The Seviri images are acquired every 15 minutes, so temporal correlation about clouds presence should be informative to improve performance. We therefore expand the model to include temporal dependencies, by unwrapping a sequence of identical CNNs with shared weights and allowing the prediction obtained at the previous time step to influence the next one. By doing so, we inject temporal memory in the CNN, which becomes a Recurrent Neural Network (RNN) [7].

2. UNWRAPPED CNN WITH TEMPORAL MEMORY

This section introduces the CNN model and the strategy followed to inject temporal memory in the model.

2.1. Core model: the modified hypercolumn CNN

Figure 2 presents the core network architecture used in this study. It follows the hypercolumn architecture originally presented in [8] with the modifications proposed in [9, 10].

The main trunk of the architecture (top part of Fig. 2) consists of a standard classification network. Such a network learns a set of filters organized in three blocks, each one separated by nonlinearities (using the ReLU unit), batch normalization, and spatial downsampling (max-pooling) operations (see [6] for details about these operations).

Differently from a classical classification network, the following fully-connected layer is removed and the activations are used instead: each activation set (corresponding to the result of convolving the input with a set of learned filters) is upsampled at the original image resolution and concatenated to it. As a result of this upsampling-plus-concatenation phase, we retrieve a 3D tensor of the same size as the original image in the two first dimensions ($r \times c$), and with as many features

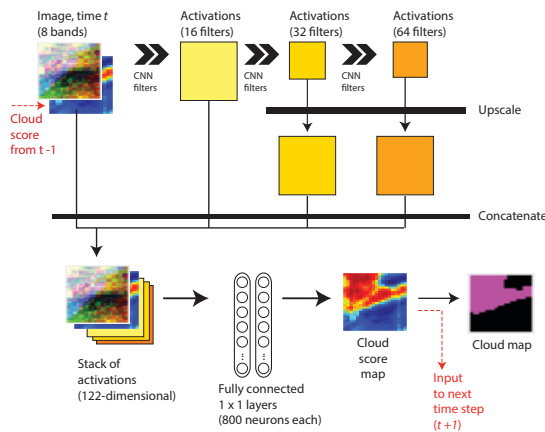


Fig. 2: Our modified hypercolumn CNN. Red arrows depict the temporal connections added to the base architecture.

as the sum of the original bands plus the number of filters learned by the CNN in the third dimension. Proceeding this way mimics a contextual feature extraction phase, since it results in a set of hierarchical features for each pixel.

This tensor is then used in a fully convolutional way [11], i.e. to perform dense segmentation providing the cloud score at every image location. To do so, we use 1×1 convolutions mapping into binary class scores. This is equivalent to learning a multilayer perceptron (MLP) model. Since the proposed model involves only standard convolutions operations, the weights can be learned end-to-end.

2.2. Injecting temporal memory

To account for short term memory, we use an intuitive approach (sketched by the red arrows in Fig. 2): we re-inject the class scores obtained at time step $t-1$ into the model learning the features using time step t . This type of recursion is typically used in standard recurrent neural networks [7], and we use it here in the following way. Since the problem of cloud detection is binary, the class score is two-dimensional (one score per class) and of the same size as the input ($r \times c$). For a model considering T time steps, we make T exact copies of the hypercolumn network presented in Section 2.1. Then, we modify the input of the network by adding two features to the original image by concatenation: such additional input corresponds to the cloud scores map (of size $r \times c \times 2$) obtained at the previous time step $t-1$ and previous to the softmax classifier (in other words, the scores do not sum to one). For the first time step ($t = 1$) we do not have such scores map: in this case the scores map is replaced by an array of zeros. These simple temporal connections make each time step aware of the prediction of the previous one and force temporal smoothness through time. We avoid forgetting of previous states by using a composite loss function, detailed below.

Losses. Since the model is now a concatenation of T identical CNNs, each prediction task has its own loss function, each one corresponding to a standard cross entropy loss for semantic segmentation:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_c \omega_c \mathbb{1}[y_i = c] \log(p(y_i = c | x_i)). \quad (1)$$

The losses are merged with a weighted linear combination:

$$\mathcal{L}(y, \hat{y}) = \sum_{t=1}^T \beta^t \mathcal{L}^t(y^t, \hat{y}^t), \quad (2)$$

where labels y^t correspond to the cloud map at step t , and β^t weights the losses by importance. In this case, the importance is increasing the closer we get to the time instance being predicted, since 1) we want to be accurate in the final prediction (where large temporal information is accumulated) and 2) we do not want the first loss (at $t = 1$) to become too important (since it uses an empty array of cloud scores as an input).

Parameters update. The T CNN models are initialized as identical models and correspond to a single model deployed over time. Since we want it to predict correct cloud maps rather than specializing in predicting the mask at a specific time step, the models must remain identical during training. This has several advantages: 1) the first timestep learns informative filters even though it is not receiving any input from the unavailable cloud scores; 2) all the time steps classifiers improve after each backpropagation step; 3) eased backpropagation through the sequence, since a single backpropagation through the whole chain of models will make reaching the first models hard, because of vanishing gradient problems for the last loss [12]. To do so, we simply average parameter values corresponding to the same filters of the hypercolumn, for all the convolutional (filter weights and bias) and batch normalization filters, at the end of each backpropagation pass. We do not use LSTM units.

3. EXPERIMENTS

3.1. Data and setup

We test our network on a dataset provided by EUMETSAT, containing Seviri/MSG Level 1.5 acquisitions acquired in 2010. We considered one landmark location in Dakhla (Western Sahara), which involves 35'040 MSG acquisitions (one every 15 minutes over the year) with a fixed resolution of 20×26 pixels. To be able to train a single model during day and night, we discarded the visible RGB bands and consider only the eight infrared bands of Seviri.

Level 2 cloud masks were provided [13] and used as the best available ‘ground truth’. Since the ground truth was sometimes incorrect around coastal areas or thermal exchange regions, we masked pixels in a buffer area around them during training, so that the network does not learn spurious patterns related to these inconsistencies. The entire cloud mask was nevertheless considered at test time.

The network setup is sketched in Fig. 2. We trained our networks for 500 epochs, with an initial learning rate of $3.5 \cdot 10^{-5}$, halved after 350, 400, 425 and 450 epochs.

CNN models use the whole images as training samples: in this sense, a random extraction of pixels (as in [4]) to validate the models is not possible. To avoid temporal positive biases, we use the following train / test strategy: we keep every third month as a separate set of test samples, i.e. the months of March, June, September and December, and use all the data issued from the other eight months as training samples. Each image, corresponding to a 15 minutes time span, is used in its entirety either to train or test the models. Temporal sequences of length $t = [2, \dots, 5]$ are considered in the RNN model.

We compare our proposed RNN with a hypercolumn CNN using only the image at time t to perform cloud detection (therefore with $t = 1$). We train several instances of the RNN with different timesteps, ranging from $t = 2$ to $t = 5$.

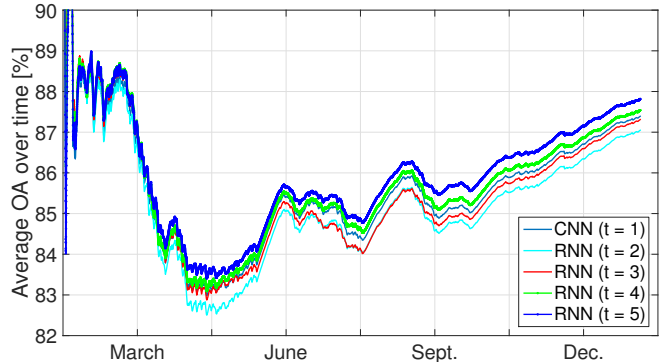


Fig. 3: Performance of the RNNs in the four test months for different temporal lags t .

3.2. Results and discussion

Figure 3 illustrates the numerical comparisons. We used weighted averaged overall accuracy

$$\overline{OA}(i) = \frac{1}{i} \sum_{j=1}^i OA(j),$$

where $OA(j)$ is the overall accuracy obtained at time instant j preceding the current one. We observe promising performances over all the different months left out for testing. Averaged accuracies span between 82% and 88% consistently over the image time series sequence.

The injection of temporal memory improves the model only when sufficient time steps are involved in learning. For instance, for $t = 2$ the result slightly worsens the original hypercolumn predictions. We suppose that this behavior is due to the fact that, in the case with $t = 1$ the network simply learns to ignore the extra inputs (the $(r \times c \times 2)$ matrix of zeros), while in $t = 2$ the weights are updated in an informative way after the first recursion, but the role of the empty inputs is still too strong. With $t = 4$ and $t = 5$, the RNN makes full use of the predictions obtained at the previous steps and provides better predictions, with a moderate +1% improvement of the prediction over the whole sequence.

Figure 4 illustrates three parts of the sequences predicted by the RNN with $t = 5$: the first, a morning sequence in March, shows that the RNN can follow the global structure of the cloud, but lacks slightly in geometrical accuracy, as it can be seen in the clouds scores obtained by the final classifier. In the second, a night sequence in July, the model predicts correctly the cloud over land, but fails predicting precisely the cloud over the sea. The last, a night sequence in December, show a very detailed prediction, precisely outlining the cloud. Note that both predictions are outputted by the exact same model, contrarily to the divide-and-conquer strategy in [4].

Day Time	March 3 rd				June 19 th				December 16 th			
	09 : 42	09 : 57	10 : 12	10 : 27	04 : 42	04 : 57	05 : 12	05 : 27	23 : 12	23 : 27	23 : 42	23 : 57
Seviri												
RNN Score Pred.												

Fig. 4: Examples of the RNN predictions for the model with $t = 5$ in three time instants of the test set. Yellow surfaces represent clouds.

4. CONCLUSIONS

In this work, we studied the suitability of a convolutional neural network with temporal memory to the prediction of pixel-wise cloud delineation from infrared imagers on meteorological satellites. The 15-minutes sampling rate allows to encode strong temporal information, which we injected by using a recurrent neural network structure, where the output of the network at time t becomes an input for the model predicting $t + 1$ and further timesteps.

We built a single model able to predict clouds accurately day and night, and tested it in a region in the Western Sahara over a 1 year sequence acquired in 2010. The results show that the proposed RNN can predict clouds with an accuracy close to 90% over the whole sequence, and has no persistent bias related to day/night or the validation month.

5. REFERENCES

- [1] C. Pelletier, S. Valero, J. Inglada, N. Champion, and G. Dedieu, "Assessing the robustness of random forests to map land cover with high resolution satellite image time series over large areas," *Remote Sens. Environ.*, vol. 187, pp. 156–168, 12 2016.
- [2] C. Zhang, W. Li, and D. J. Travis, "Restoration of clouded pixels in multispectral remotely sensed imagery with cokriging," *Int. J. Remote Sens.*, vol. 30, no. 9, pp. 2173–2195, 2009.
- [3] D. Just, "SEVIRI instrument level 1.5 data," in *MSG RAO Workshop*, Bologna, Italy, 2000.
- [4] A. Pérez-Suay, J. Amorós-López, L. Gómez-Chova, V. Laparra, J. Muñoz-Marí, and G. Camps-Valls, "Randomized kernels for large scale earth observation applications," *Remote Sens. Environ.*, vol. 202, no. Supplement C, pp. 54 – 63, 2017.
- [5] X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, 2017.
- [6] M. Volpi and D. Tuia, "Dense semantic labeling of subdecimeter resolution images with convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 881–893, 2017.
- [7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Computer Vision and Pattern Recognition CVPR*, 2015.
- [9] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "High-resolution aerial image labeling with convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. in press, 2017.
- [10] D. Marcos, M. Volpi, B. Kellenberger, and D. Tuia, "Land cover mapping at very high resolution with rotation equivariant CNNs: towards small yet accurate models," *ISPRS J. Int. Soc. Photo. Remote Sens.*, in press.
- [11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Computer Vision and Pattern Recognition CVPR*, 2015.
- [12] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. IEEE Press, 2001.
- [13] M. Derrien and H. L. Gléau, "MSG/SEVIRI cloud mask and type from SAFNWC," *Int. J. Remote Sens.*, vol. 26, no. 21, pp. 4707–4732, 2005.