Scanner Neural Network for On-board Segmentation of Satellite Images

Florent Lafarge, Gaétan Bahl

SCANNER NEURAL NETWORK FOR ON-BOARD SEGMENTATION OF SATELLITE IMAGES

Gaétan Bahl[1,2] *and Florent Lafarge*[1]

[1]Université Côte d'Azur - Inria
[2]IRT Saint Exupéry
email: firstname.lastname@inria.fr

ABSTRACT

Traditional Convolutional Neural Networks (CNN) for semantic segmentation of images use 2D convolution operations. While the spatial inductive bias of 2D convolutions allow CNNs to build hierarchical feature representations, they require that the whole feature maps are kept in memory until the end of the inference. This is not ideal for memory and latency-critical applications such as real-time on-board satellite image segmentation. In this paper, we propose a new neural network architecture for semantic segmentation, "ScannerNet", based on a Recurrent 1D Convolutional architecture. Our network performs a segmentation of the input image line-by-line, and thus reduces the memory footprint and output latency. These characteristics make it ideal for on-the-fly segmentation of images on-board satellites equipped with push broom sensors such as Landsat 8, or satellites with limited compute capabilities, such as Cubesats. We perform cloud segmentation experiments on embedded hardware and show that our method offers a good compromise between accuracy, memory usage and latency.

***Index Terms*—** Image Segmentation, Satellite Imagery, On-board processing, Recurrent Convolutional Network, ConvLSTM, Cloud Segmentation

1. INTRODUCTION

Over the past decade, thanks to the increase in computing power and wide availability of data, Convolutional Neural Networks [1] have gradually imposed themselves as the *de facto* solution for computer vision tasks such as image classification [2], object detection [3] and image segmentation [4]. Image segmentation is the task of assigning a label to every single pixel of an input image. Thus, the output segmentation maps have the same resolution as the input images. This task is not only crucial in remote sensing applications such as cloud segmentation [5, 6], land cover estimation [7], or road mapping [8], but also in other domains, such as medical imagery [4] and self-driving [9].

As a consequence of the increasing demand for edge processing, techniques such as distillation [10] and quantization [11] have been used as a way to port deep learning models onto low-power devices like UAVs, smartphones or satellites. However, semantic segmentation models remain very compute intensive, and dedicated architectures with a low number of convolution filters had to be developed in conjunction with these techniques in order to be used on low-power satellites [5, 6]. Moreover, many earth observation satellites feature push broom sensors. These sensors acquire images of varying height line-by-line (i.e. by "scanning" the ground). In cases where the acquisition runs for a long time, the images created in this way can become too large to be efficiently processed in memory by traditional 2D convolutional neural networks. Finally, since CNNs traditionally process the whole image at once, there is a significant latency between the acquisition and the output of the result, since we have to wait until the whole image is acquired and processed before viewing the result.

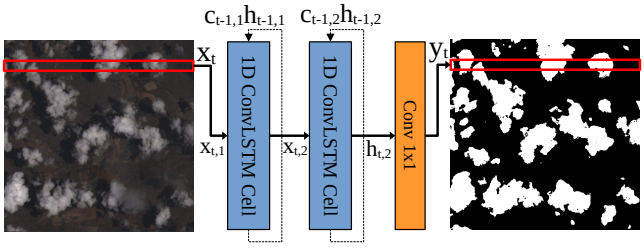

Fig. 1. Our 1D Convolutional LSTM Neural Network, ScannerNet, segments the input images line-by-line to save memory and reduce latency. We use two 1D ConvLSTM [12] layers and a single 1x1 convolution to obtain a binary output. Our architecture allows a 280x reduction in memory usage compared to previous works on cloud segmentation at a similar number of parameters.

In order to alleviate these issues, we propose a semantic segmentation method based on a one-dimensional Convolutional LSTM [13, 12] (ConvLSTM), neural network. Our network parses the input image line-by-line, and outputs the result at the same time, thus only keeping the necessary information in memory and achieving a very low latency.

2. METHOD

In this section, we provide a description of our architecture, shown on Figure 1. Our architecture is based on a two-layer 1D Convolutional LSTM (ConvLSTM), inspired by the work of Shi et al. [12] on 2D ConvLSTM. We implement the ConvLSTM cells without self-loops as in [13]:

$$\begin{aligned} i_t &= \sigma(\text{Conv1D}([x_t; h_{t-1}], W_i) + b_i) \\ f_t &= \sigma(\text{Conv1D}([x_t; h_{t-1}], W_f) + b_f) \\ o_t &= \sigma(\text{Conv1D}([x_t; h_{t-1}], W_o) + b_o) \\ g_t &= \tanh(\text{Conv1D}([x_t; h_{t-1}], W_g) + b_g) \\ c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\ h_t &= \tanh(c_t) \circ o_t \end{aligned} \tag{1}$$

where $\circ$ is the Hadamard product, σ is the sigmoid function, $[x_t; h_{t-1}]$ is the concatenation of the input line x_t at step t and previous hidden state of the cell h_{t-1}. We notice that all the convolution operations in Equation 1 can be optimized by combining them into a single 1D convolution with weights $W = [W_i; W_f; W_o; W_g]$, $b = [b_i; b_f; b_o; b_g]$ and output $y = [i_t; f_t; o_t; g_t]$. We denote as F_1 and F_2 the number of feature maps of the hidden states in layer 1 and 2, respectively. The final output line y_t is generated by a 1x1 convolution that maps the F_2 feature maps of the hidden state of layer 2 $h_{t,2}$ to a single line of the segmentation map, as shown on Figure 1.

In order to obtain results for two different total numbers of parameters and thus easily compare our networks to previous works, we create two networks with varying F_1, F_2 and filter size. Table 1 shows the details of these two networks.

Network	F_1	F_2	Filter size	Params.
ScannerNet	4	8	9	4 521
ScannerNet Small	4	4	7	1 717

Table 1. Number of filters, filter size and number of parameters for each of our two ConvLSTM networks. F_1 and F_2 are the number of filters in the first and second 1D ConvLSTM layer, respectively.

3. EXPERIMENTS

In order to validate our design, we perform a cloud segmentation experiment. On-board cloud segmentation is an important topic as it can be the first step in the on-board compression of satellite images. Indeed, a large portion of the Earth is covered by clouds at any given time, thus the removal of cloud pixels translates to large savings in both bandwidth and storage space. We compare our results to the ones from [5], who also worked on lightweight networks for on-board cloud segmentation.

For this experiment, we use the 38-Cloud dataset [14] and implement our architecture in the PyTorch 1.10 deep learning framework. We re-implement the C-FCN and C-UNet++ networks shown in [5] in this framework, in order to ensure a fair comparison. As done in [5], we only use the RGB bands of the input images. We also implement a non-recurrent 1D convolutional baseline network with a similar number of parameters as our architecture. The 38-Cloud dataset is composed of 384x384 pixel crops from Landsat 8 scenes. We use 15% of the training set as a validation set. All networks are trained until convergence for a maximum of 1000 epochs. We use a batch size of 64 and the Adam optimizer for all networks. Training was done on a machine with an AMD Ryzen 9 3900X CPU, Nvidia GeForce RTX 3090 GPU and 64GB of DDR4 RAM.

Network	Param.	Memory usage (MB)
Baseline	1 409	0.08
C-FCN [5]	1 471	28.62
C-UNet++[5]	9 129	47.19
ScannerNet Small (ours)	1 717	**0.10**
ScannerNet (ours)	4 521	**0.15**

Table 2. Number of parameters and memory usage (in Megabytes) for a forward pass of our architecture, compared to networks from [5]. Our networks allow significant memory savings during inference.

In Table 2, we report the number of parameters used by our networks, and the memory required for a single forward pass of each network, as reported by the `torchinfo` module. We observe that our architecture enables some significant memory savings compared to the already memory efficient architectures presented in [5]. Indeed, at a similar number of parameters as C-FCN [5], we obtain a 280x reduction in memory usage.

Network	IoU	F1	Acc.	Prec.	Rec.
Baseline	68.14	72.89	83.58	72.92	72.85
C-FCN [5]	82.04	85.77	91.74	89.68	82.19
C-UNet++[5]	**86.18**	**89.75**	93.52	86.19	**93.62**
ScannerNet Small (ours)	81.86	85.51	91.71	90.86	80.75
ScannerNet (ours)	85.80	88.98	**93.65**	**93.79**	84.65

Table 3. Quantitative results of our architecture on the 38-Cloud dataset [14] at a detection threshold of 0.5, compared to networks from [5]. Our method obtains similar metrics at similar numbers of parameters, while using less memory and having less latency.

Table 3 and Figure 2 show quantitative results using commonly used metrics (see definitions in [14]). We observe that our ScannerNet Small network obtains results similar to the ones of C-FCN, even though its memory usage is much lower. Our bigger ScannerNet obtains slightly lower scores than C-UNet++, also while consuming less memory for its forward pass. The baseline 1D convolutional network (without LSTM) obtains much worse results, even though it has a similar number of parameters as ScannerNet Small, which

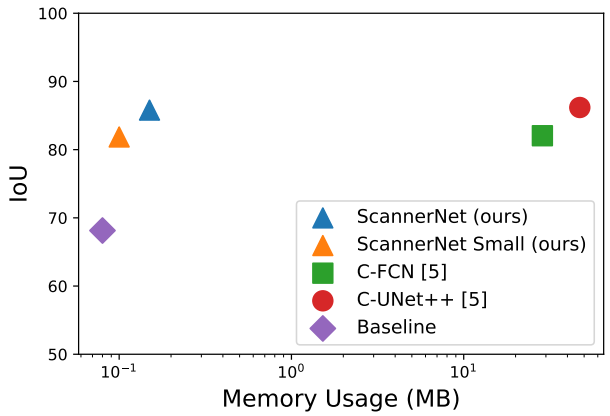

Fig. 2. IoU (Intersection over Union) of segmentation output at a threshold of 0.5 with respect to memory usage (in Megabytes, log scale), compared to previous works. Our networks achieve IoU results similar to previous works, while having a comparatively very low memory usage.

shows that the recurrent part of our architecture is necessary to obtain a decent segmentation.

Figure 3 shows a selection of qualitative results taken from the 38-Cloud test set. We observe that even though it can only use one line of input at any step to generate the output segmentation result, our network manages to generate coherent results that match the output of C-UNet++[5]. Moreoever, we can observe some block artifacts in the output of C-UNet++ that do not exist in the output of ScannerNet. Our intuition is that since ScannerNet performs all processing at the same resolution as the input image, it can output a more refined segmentation than C-UNet++.

Next, we use a low-power edge device to measure the latency of our neural networks. In our context, we define the latency as the delay between the start of the processing and the output of the first pixel of the segmentation map, without including data loading time. For our ScannerNet architecture, since the processing is done line-by-line, the latency is thus measured on the computation of a single line of segmentation. For other neural networks, since the processing is done on the whole image at once, the latency is measured on the computation of the whole segmentation.

As our test system, we use an Nvidia Jetson Nano development kit with 2GB of memory, since it is a very popular platform for embedded systems. It features an SoC equipped with a quad-core A57 ARM CPU and a 128-core Nvidia Maxwell GPU, consuming up to 10W. For benchmarking, we lock the clock of the SoC to its maximum using the `jetson_clocks` utility, and cool the board with a Noctua NF-A4x20 fan running at maximum speed. We run each test 500 times in a row and discard the first 10 runs. We use the same software configuration as the one used for training. We test each network on both the CPU and the GPU to take more usage scenarios into account (e.g. systems without GPUs).

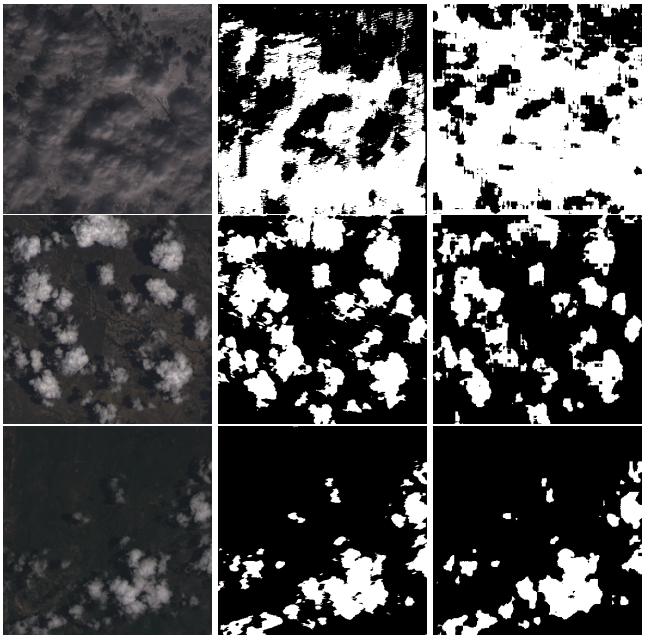

Fig. 3. Qualitative results of **ScannerNet (ours, middle)** against C-UNet++ [5] (right) on a selection of images from the 38-Cloud test set. Our architecture performs well under a variety of scenarios, including images with snow backgrounds, even though it only processes one line of the input image at a time.

The results of this latency testing are shown in Table 4. We observe a significant reduction in latency compared to previous works. In particular, our small ScannerNet offers a 3.7x reduction in latency compared to C-FCN [5], with a similar number of parameters. The improvement is even larger when comparing to C-UNet++. We notice a bigger improvement when running on the CPU than on the GPU. Our intuition is that since GPU architectures are more efficient when asked to process a large amount of data at once, our architecture does not offer as much benefit as on the CPU. Nevertheless, the reduction in latency is still significant.

Network	Latency (CPU)	Latency (GPU)
C-UNet++[5]	638.6	27.9
C-FCN [5]	296.1	11.1
ScannerNet (ours)	91.5	4.6
ScannerNet Small (ours)	**80.3**	**4.5**

Table 4. Average latency (milliseconds, lower is better) of our networks, measured on the CPU and GPU of a Jetson Nano development kit, compared to networks from [5], on 384x384 pixel images of the 38-Cloud dataset. Our networks provide a 3.7x reduction in latency at the same number of parameters (i.e. when comparing ScannerNet Small to C-FCN).

4. CONCLUSION

In this paper, we have presented ScannerNet, a 1D Recurrent Convolutional neural network architecture with very low memory usage and latency for on-board segmentation of satellite images. We have shown that our networks obtain results comparable to previous works in a cloud segmentation task while occupying 286 times less memory and having a lower latency at a similar number of parameters. We have observed real-world gains on a low-power device, with a reduction of the latency by a factor of 2.4 in the worst case.

Our approach is orthogonal to previous works on compact architectures and quantization of neural network weights. In particular, our networks could be further optimized by using integer or binary weights. We have also shown that 1D convolutions can work just as well as 2D convolutions for image segmentation, a technique that is seldom explored in the literature. We hope to pave the way for more varied neural network architecture designs adapted to different kinds of on-board sensors. Future work includes testing our architecture on FPGA chips and images from push broom sensors.

5. ACKNOWLEDGMENTS

This work is funded by the CIAR project at IRT Saint Exupéry. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

6. REFERENCES

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[5] Gaétan Bahl, Lionel Daniel, Matthieu Moretti, and Florent Lafarge, "Low-power neural networks for semantic segmentation of satellite images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[6] Frédéric Férésin, Erwann Kervennic, Yves Bobichon, Edgar Lemaire, Nassim Abderrahmane, Gaétan Bahl, Ingrid Grenet, Matthieu Moretti, and Michael Benguigui, "In space image processing using ai embedded on system on module: example of ops-sat cloud segmentation," in *2nd European Workshop on On-Board Data Processing*, 2021.

[7] Yonghao Xu, Bo Du, Liangpei Zhang, Daniele Cerra, Miguel Pato, Emiliano Carmona, Saurabh Prasad, Naoto Yokoya, Ronny Hänsch, and Bertrand Le Saux, "Advanced multi-sensor optical remote sensing for urban land use and land cover classification: Outcome of the 2018 ieee grss data fusion contest," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 6, pp. 1709–1724, 2019.

[8] Gaetan Bahl, Mehdi Bahri, and Florent Lafarge, "Road extraction from overhead images with graph neural networks," *arXiv preprint arXiv:2112.05215*, 2021.

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[11] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[12] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] S. Mohajerani and P. Saeedi, "Cloud-net: An end-to-end cloud detection algorithm for landsat 8 imagery," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, July 2019, pp. 1029–1032.