Convolutional Encoder-Decoder
for Satellite Imagery Semantic Segmentation

loic.landrieu

May 2019

1 Dataset

The dataset considered in this exercise was acquired by the SPOT-6 high resolution satellite (1.5m per pixel). The annotations are from the BDTopo produced by IGN, the French Mapping Agency.

It is comprised of 300 tiles of 256×256 pixels (200 for training, 100 for validation).

The pixels are annotated within 6 semantic classes + 1 class for unannotated data:

Class Number	**Class Name**	**Color (in the notebook)**
99	not annotated	white
0	urban area	pink
1	water	blue
2	fields	yellow
3	roads	grey
4	vegetation	green
5	buildings	red

2 Convolutional Encoder-Decoder

2.1 Principle

We propose to implement a purely Convolutional Encoder-Decoder (CED) network inspired by SegNet and U-Net.

The first step is to implement a convolutional encoder of an input image which alternates between convolution layers to extract learned features and spatial reduction operations (maxpooling). Once the feature map is small enough, the decoder alternates between convolutions and upsampling (unpooling) to generate an output of the same size of the original input.

We differentiate between the size of a feature map, ie its resolution (eg: 128×128) and its depth, ie the size of the embeddings of each pixel (eg: 64)

To help with upsampling, two types of information can be passed from the encoder to the decoder:

- Unpooling layer of the decoder can use the indices of the pixels selected in the corresponding maxpooling layer of the encoder. This way, the unpooling layer can localize the original information more precisely. The feature maps must have the same dimension however.

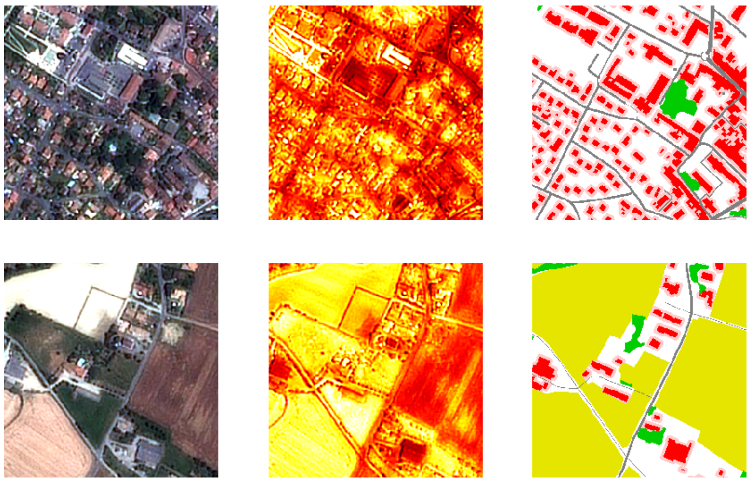

Figure 1: Tiles corresponding to urban (top) and rural (bottom) areas. Left: RGB; middle: infrared; right, ground truth.

- Some feature maps of the encoder are concatenated to the corresponding feature maps of the decoder as long as they have the same size (not necessarily the same depth), in order to bypass high resolution information.

2.2 Description

The CED takes an image of dimension $H \times W \times C$ with $H \times W$ the size of the image and C the number of channels (ie. the depth of the input). In this exercise $H = W[= 256$ and $C = 4$ (RGB + infrared). The channels are normalized in the $[0, 1]$ range.

The objective is to obtain a prediction map y of size $H \times W \times K$ where K is the number of semantic classes, and $y_{i,j,k}$ is the class score of pixel i, j for class k. Class probabilities can be obtained from class scores with the `softmax` function.

We write here the equation for a CED with 3 levels a, b and c, and comprised of 10 convolution layers (each comprised of a convolution, a batch normalization unit, and a ReLu non-linearity):

$$
\begin{aligned}
&\textbf{Encoder} && (1)\\
&x_1 = \mathrm{Conv}_2\left(\mathrm{Conv}_1\left(\mathrm{input}\right)\right) && (2)\\
&x_2, \mathrm{indices}_{a\to b} = \mathrm{Maxpool}\left(x_1\right)) && (3)\\
&x_3 = \mathrm{Conv}_4\left(\mathrm{Conv}_3\left(x_2\right)\right) && (4)\\
&x_4, \mathrm{indices}_{b\to c} = \mathrm{Maxpool}\left(x_3\right)) && (5)\\
&x_5 = \mathrm{Conv}_6\left(\mathrm{Conv}_5\left(x_4\right)\right) && (6)\\
&\textbf{Decoder} && (7)\\
&y_4 = \mathrm{Unpool}\left(x_5, \mathrm{indices}_{b\to c}\right)) && (8)\\
&y_3 = \mathrm{Conv}_8\left(\mathrm{Conv}_7\left([y_4, x_3]\right)\right) && (9)\\
&y_2 = \mathrm{Unpool}\left(y_3, \mathrm{indices}_{a\to b}\right)) && (10)\\
&y_1 = \mathrm{Conv}_{10}\left(\mathrm{Conv}_9\left([y_2, x_1]\right)\right) && (11)\\
&y = \mathrm{Conv}_D\left(y_1\right) && (12)
\end{aligned}
$$

All convolutions have a kernel of size 3×3 and use mirror padding. We denote by d_n the depth of convolution n (*ie.* size of the produced feature map). The size of the tensor above is given in the table below:

Tensor	**Size**
input	$H \times W \times 4$
x_1	$H \times W \times d_2$
x_2	$\lceil H/2 \rceil \times \lceil W/2 \rceil \times d_2$
x_3	$\lceil H/2 \rceil \times \lceil W/2 \rceil \times d_4$
x_4	$\lceil H/4 \rceil \times \lceil W/4 \rceil \times d_4$
x_5	$\lceil H/4 \rceil \times \lceil W/4 \rceil \times d_6$
y_4	$\lceil H/2 \rceil \times \lceil W/2 \rceil \times d_6$
y_3	$\lceil H/2 \rceil \times \lceil W/2 \rceil \times d_8$
y_2	$H \times W \times d_8$
y_1	$H \times W \times d_{10}$
y	$H \times W \times K$

2.3 Loss Function and Metrics

As we consider the task of semantic segmentation the loss used is the cross-entropy.

We consider the two following metrics, defined with respect to the confusion matrix CM of size $k \times k$:

- **Overall Accuracy**: a global metric defined as the ratio of correct prediction divided by the number of (annotated) points
$$OA = \frac{\mathrm{Trace}(CM)}{\mathrm{Sum}(CM)}.$$
- **Class IoU**: this per-class metric is defined as the ratio between true positives divided by the sum of false positives, false negatives and true positives.
$$IoU_i = \frac{CM_{i,i}}{CM_{i,i} + \sum_{j \neq i}\left(CM_{i,j} + CM_{j,i}\right)}.$$

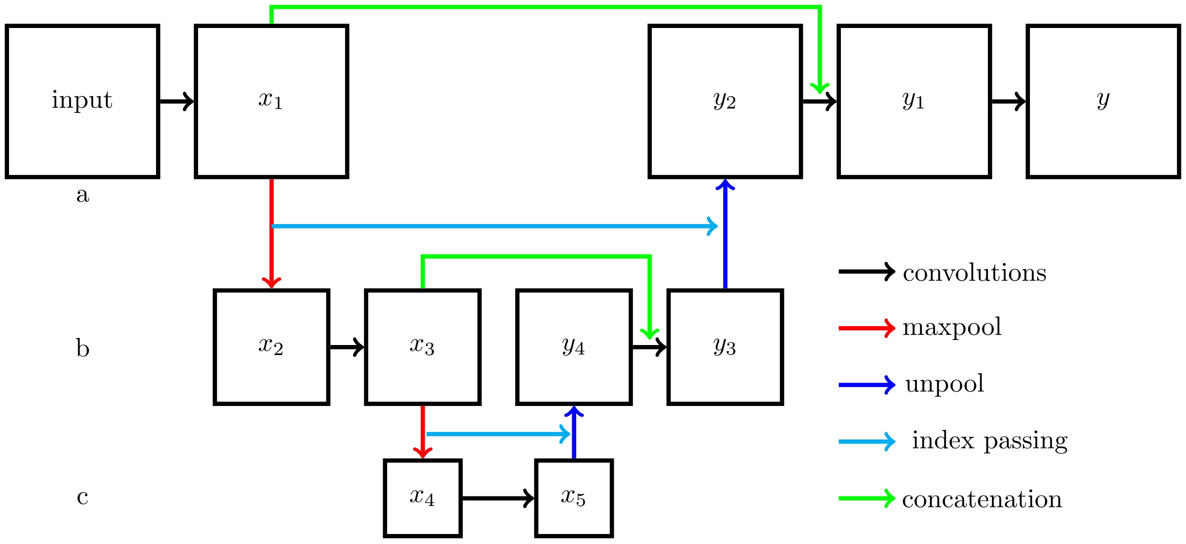

Figure 2: Illustration of the neural architecture of CED for semantic segmentation. The three levels are denoted by a, b, and c.

3 Exercise

All the code that needs to be completed is marked down with `TODO`. Make sure you complete them all before moving on to the next question. Do not hesitate to create new cells and copy some code to play around with the tensors and check their size, it can be difficult to debug a notebook otherwise. When encountering an error, a good idea is always to print the shape of the incriminated tensors just before the error.

You will be required to look up some documentations on `numpy` and `torch` functions online.

We have added tests and assertions at the end of crucial steps. Make sure that the asserts are satisfied (no error) before moving on to the next question.

Some parts of the following questions are **in bold**. These require a written response in your report and will be graded.

3.1 Installation and Data Download

Q1 Login to your google account and open the following collab file: `https://drive.google.com/open?id=1m4borLeZE1BvCPy4kQbrOYC6uLILHuL_` (copy and paste in the url bar, tested with firefox).

Q2 Select File→Save a copy in Drive. Select Runtime→Change runtime type and select GPU.

Q3 Execute the cell [1] and [2], click the authentication link, copy the key and paste it in the box. Make sure that you obtain the following message:
`200 tiles for training, 100 tiles for testing`.

Q4 Execute the data loader cell [4]. The function `tile_loader` loads a single tile and format it.

Do not change the augment function for now.

Q5 Execute the cell [5] containing the visualization functions. Read and understand the implemented functions. Using the documentation, complete cell [6] to represent 5 random tiles of the test set with RGB, infrared and ground truth with the function `viewer`.

Q6 Complete the cell [7] for the computation of OA and IoU. Run the cell [8] and make sure you obtain the following:
`OA = 85.00%`
Urban : 71.43% — Water : 57.14% — Fields : 100.00% — `Road :  100.00% | Vegetation : 100.00% | Buildings :  66.67%`

Q7 Read the SegNet class in cell [9]. Complete the functions `init` and `forward` implementing equations (1)-(10). You can use the function `torch.cat` to concatenate embeddings. Comment on the assert within the `init` function. **Why is it necessary that $d_4 = d_6$ and $d_2 = d_{10}$?** Make sure the `assert` in cell [10] passes without error.

Q8 Read in the cell [11] the function `train`. Complete the function `eval` which performs inference with the trained model. This function computes the loss but does not compute gradients nor make an optimizer step.

Q9 Train a model with the default parameter in cell [12]. **Comment on the evolution of the performance on the training set and test set.** In cell [13] Visualize the values RGB, Infrared, ground truth, prediction and error for 5 random tiles of the test set. Visualize with and without the masks, and **propose an explanation as to why the pink class surrounding the buildings was added**.

Q10 Complete the function `view_U` in cell [14] (copy and paste from cell [9]) and view the embeddings in cell [15].

Now that we have a working model, we can improve it. Questions with $\star$s are more difficult. Remember do save a copy of your notebook at this point before breaking the code too much! After coding your best model, run the training for 100 epochs to obtain your best mIoU (you should get around 75% mIoU).

Grade Score: response to questions in bold: 20%. Completion of Q1-10: 30%. First 3 stars completed correctly = 10% each. Each star after that= 5%. Send us a pdf report with a link to your collab (don't forget to activate link sharing).

$\star$**Q11** Study the effect of the batch size and learning rate. Try very small and large values, report the best configurations. In particular, interpret the observed relationship between batch size and learning rate, and propose an explanation.

$\star$**Q12** Add a drop out layer on the previous to last layer of the classifier layer. Comment on the effect, if any.

$\star$**Q13** The data set is imbalanced. Add class weights to the cross entropy loss to increase the importance of vegetation and buildings (see torch doc). Use the following weight vector: $[0.2, 0.1, 0.1, 0.3, 0.1, 0.2]$, and try other values and comment on their effect on the OA and mIoU.

$\star$**Q14** Remove the first `return obs, gt` in the function `augment` in cell [4] and complete the code to add a random rotation and a Gaussian perturbation. Explain why such an augmentation should be a good idea. Comment on the actual effect, and suggest an explanation.

$\star$**Q15** Change the depth of the convolutions to try to find a better parametrization. What happens when the network gets too large (too many parameters).

$\star$**Q16** Implement adaptive learning rates using the `MultiStepLR` scheduler (look up the online torch documentation for usage). Implement a decay of 0.7 and milestone at epochs 10 and 15 when training for 20 epochs, and 70 and 90 when using 100 epochs.

$\star\star$**Q17** Create the class `SegNet3` which operates on 4 levels instead of 3 (with the final layer of the encoder being of size $\lceil H/8 \rceil \times \lceil W/8 \rceil \times d$. Test this architecture and comment. Don't forget to add the necessary assertions.

$\star\star$**Q18** Modify the cell [4] and the function `train_full` in cell [11] to add a new set: the validation set. It is comprised of 20% of the train set, chosen randomly. Make `train_full` return the model at the epoch which performs best on the validation set wrt the mIoU. Note: you can copy a model using `copy.deep_copy`.

$\star\star$**Q19** Implement a 5-fold cross-validation (easier if you have already completed Q18) to select the best model configuration.

$\star\star\star$**Q20** We want to increase the spatial regularity of the output by adding a low total variation prior on the prediction. We define $\texttt{loss\_TV}(y) = \sum_{i,j \in E} \sum_{k \in \mathcal{K}} |y_{i,k} - y_{j,k}|$ where $\mathcal{K}$ is the set of classes, $y_{i,k}$ is the logit of class k for pixel i and E the 4-neighbors adjacency structure of the image (use the already implemented function `compute_adjacency_graph` in cell [16] and by completing the function `TV` in cell [17] and the learning functions [18]).

Define the new loss at epoch t as: $\texttt{loss}(y) = \texttt{cross\_entropy}(y, gt) + \alpha_t \texttt{loss\_TV}(y)$, with gt the ground truth, and $\alpha_t = \alpha_0 \times t/\texttt{max\_epochs}$ (this is called *loss annealing*).

Try different values of α_0 and comment on the effect on the accuracy and the spatial regularity of the output. Why is annealing appropriate here?

Q21 Open question: try to think of other ways to improve the model, and report your reasoning and results.

Reference

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham.