Two Deep Neural Network architectures to find roads in satellite images: a comparison of Convolutional versus Fully Convulational Networks in a semantic segmentation task.

Albert Buchard, Remy Joseph

Abstract— **Neural networks have been studied extensively and are subject to active research in numerous fields including the dynamic field of image recognition. Using a recently released framework for machine learning called Tensor Flow, and the Keras library, this work compares the performance in semantic image segmentation of two Deep Neural Network architectures trained to discriminate roads from non-roads in satellite images. A Convolutional Neural Network using Shift-And-Stich method on 8*8 patches performed largely better (F1 = 0.86) than our Fully Convolutional Neural Network architecture (F1 = 0.56) adapted from FCN-8s and training on full sized images. We expanded our training set using random and non-random transformations. During testing, we augmented the test set in a similar fashion (8x for the CNN and 32x for the FCN) and for each image, an element-wise product of prediction maps from the extended set was used to infer predictions for each pixel. We report a final classification score of 0.862 on new data. Contrary to our initial expectations and previous work in the field, this short study is in favor of a CNN approach. However, those two models complexity being orders of magnitude apart further study is necessary to tease out the effect of fine parameter tuning from a real superiority of the network architecture. Our code is openly available on GitHub (http://www.github.com/albertbuchard/semantic-segmentation).**

I. INTRODUCTION

With the collection of cross-modal datasets, the fast growth of computation power, and the acquisition of satellite images of increasing precision, recent years have seen the development of free and robust navigation systems used on a daily basis by millions of users. Research on algorithms able to use this data and make informed inference has been booming since the development of convolutional neural networks. Those networks take advantage of the translational invariance of images to perform classification tasks. Here we compare two approaches, one using a classic CNN performing classification on small patches of images, and a fully convolutional neural network (FCN) that takes as input a full image, and outputs a full image of classified pixels.

II. MATERIAL AND METHOD

A. Architectures overview

In this work, we compare semantic classification by a Convolutional Neural Network (CNN, or convNet) using the so-called Shift-and-Stich technique, and a Fully Convolutional Network (FCN) adapted from [Shelhamer et al., 2016], [Long et al., 2014].

B. Keras library

Keras is an open source library that wraps around Theano and Tensor flow, two lower level machine learning libraries able to assist in the construction of neural networks and optimize their training. In particular, those frameworks allow optimizing computation time by using the GPU instead of the CPU. Keras greatly simplify the model construction by reducing the number of parameters, abstracting out variable creation and size management.

C. Data augmentation

For the CNN we augmented the dataset by a factor 8 with four 90 degrees rotation on the base image and four 90 degree rotations on the transpose of the base image (800 unique images). For the FCN we used a random generator function from Keras called ImageDataGenerator that generates for each training epoch a new set of images with random shifts, rotation, and mirroring (1000 unique images).

D. FCN-8s

Fully connected networks architectures being relatively new, few architectures have been studied in detail. From previous work, at the Berkeley vision lab ([Shelhamer et al., 2016], [Long et al., 2014]) we adapted the FCN-8s architecture to our needs using only one merge layer and two skips (Third and Fourth pool layer). The model is developed using Keras and is fast to train. We used non-trainable bilinear upsampling on the deep deconvolution layers and cropped the largest upsampled layer to match size during the fusing of layers. We fused layers using concatenation. The final layer is a convolutional layer with a kernel of size 1, with a sigmoid activation function. We

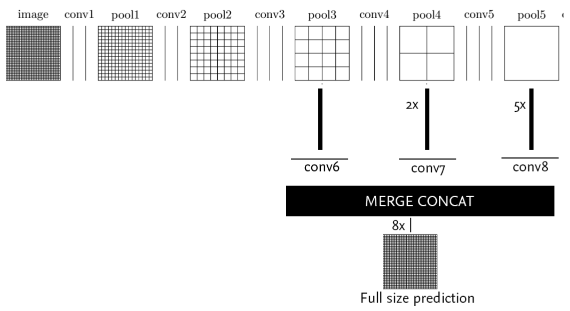

Fig. 1. FCN architecture

used a custom-made loss function called Pixel-Wise Loss

(PWL) from the 400*400 predicted and true labels matrices as follows:

$$PWL = \frac{\sum |y_{true} - y_{pred}|}{400^2} \tag{1}$$

Other FCN models have been tried: FCN-8s adapted from the Berkeley Vision Lab, as well as the U-NET architecture.

Despite our best efforts none of those architectures converged with the parameters we tried and our best results peaked around chance-level with F1 = 0.56.

III. CNN

We used the code provided by Aurelien Lucchi to make our final predictions. Our best submission was performed with the base architecture provided by the author of this code, which consists of two fully connected convolutional and pooling layers with a soft-max loss with an ℓ_2 normalization on the weights and biases. We used our enhanced training set with rotations and transposed to train the network. We reduced the patch sizes to 8 by 8 in order to increase the resolution on our predictions.

The code makes use of adaptive learning rate that decreases exponentially at each epoch. In our best run, we used 50 epochs.

IV. Post-processing of the predictions

We choose to apply a post processing regularization on the predictions that take the problem at hand into account. Given we are asked to find roads, that are mostly connected to one another, we can remove isolated islands (small blocks of pixels) identified as "road" or "non-road" in our predictions. Given the city we have been asked to map here has very rectilinear streets and that the cuts of the images in both training and testing sets are mostly aligned with the streets (meaning the edges of the images are parallel or perpendicular to most of the streets), we also tried to enhance straight lines of predicted roads.

To do so, we applied a normalized convolution kernel that favors groups of pixels and straight lines, to the predictions resulting from the CNN run. The un-normalized kernel is shown in equation 2

$$h = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{2}$$

In addition to this, we use the fact that CNNs are not rotationally invariants, meaning that an image and its rotated version by a given angle θ do not display the same elements. Therefore, a given location in an image can be classified as a road by the network, but the same location on its rotated counterpart can be classified as non-road. In order to take this information into account, for each image, we perform a prediction on the image, on the rotations of this images by an angle $90n$ degrees with n an integer and on the transpose of this image along with its 90 degrees rotations. The resulting predictions are rotated back to match the orientation of the original image and multiplied with each other (equivalent to a logical `or`) to obtain a final prediction.

A summary of these post-processing steps is given in figure 2

A. Aborted attempts

1) MCA feature enhancement: As an attempt to enhance interesting features for classification, we investigated the use of Morphological Component Analysis (MCA [Starck et al., 2004]) to separate straight lines from more isotropic features. Streets being, for most of them, straight lines (at least in the American-like city) and houses being slightly more isotropic, we considered this to be a sensible option.

The principle behind MCA is to consider that when two (or more) signals are linearly mixed if it exists two (or more) transformed domains over each of which one signal is sparsely represented and not the other(s), they can be easily separated by linear regularization under a sparse regularization. Since a signal is considered sparse in a given dictionary when it is possible to represent it with a small number of coefficients, sparse regularization boils down to minimizing the number of non-zero coefficients of the signals to separate in their respective transformed domains. In mathematical terms, MCA aims at solving:

$$\underset{S_1,S_2}{argmin} ||Y - S_1 - S_2||_2^2 + \lambda_1 ||\Phi_1 S_1||_1 + \lambda_2 ||\Phi_2 S_2||_1 \tag{3}$$

Where Y is the input signal, S_1 and S_2 are the signals to extract, Φ_1 and Φ_2 are their respective associated dictionaries and λ_1 and λ_2 are two regularisation parameters.

In practice, MCA works by iteratively projecting the input signal over each dictionary and thresholding each of these decompositions in order to keep only the most significant coefficients. Here, we perform MCA on each slice of each image. Future work could test a decomposition that considers the color variation of each component [Joseph et al., 2016].

The two dictionaries used here are ridgelets ([Do and Vetterli, 2003]) and starlets ([Starck et al., 2007]). Ridgelets represent very sparsely straight lines, while starlets are very efficient at representing isotropic features. We aimed at capturing the trees and the roofs of houses with starlets. The results show that these features were rather well separated, however, on this specific cities, the sideways are very bright on the images while roads can be very bright or very dark, therefore this might not have made things easier for the network to identify features. We fed our networks with the concatenation of the original images bands, the ridgelet bands, and the starlet bands.

An example of the results of MCA decomposition is provided in figure 3

Although the idea seamed appealing, we only achieved a score of 0.81 on the kaggle challenge. Codes for the

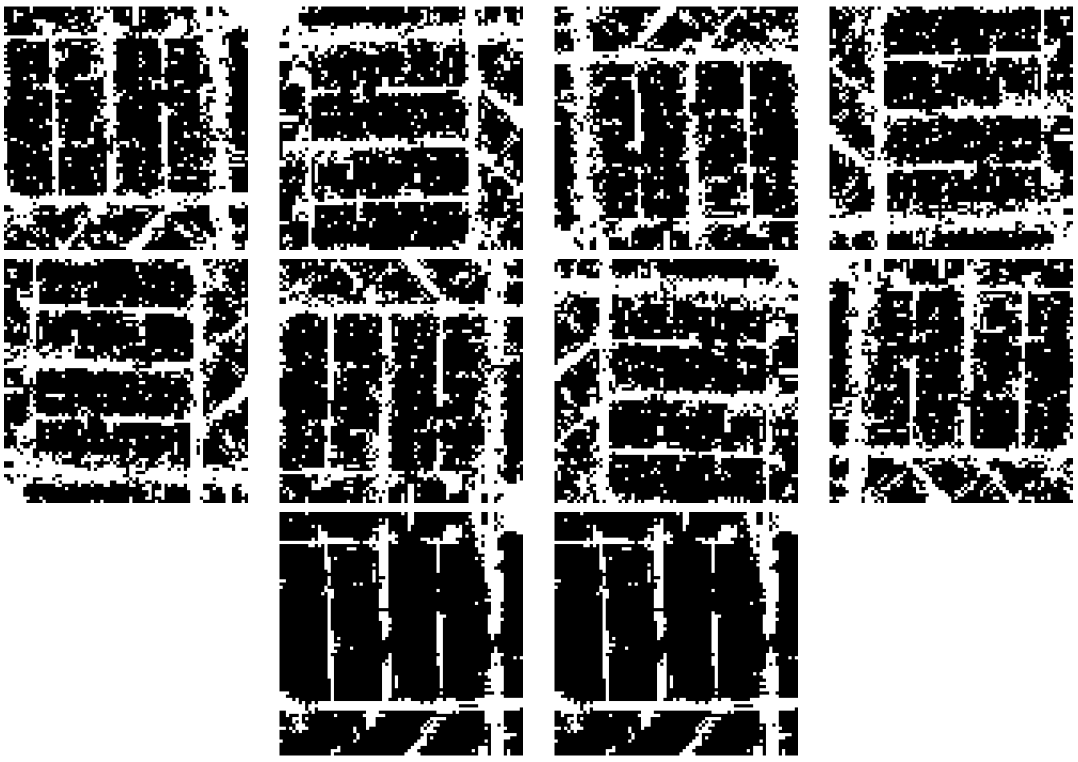
Fig. 2. Illustration of the post-processing steps. The first two rows show the predictions for each rotation/transpose of a given image. The first image of the last row shows the combination of all de-rotated prediction. The second image shows the final prediction after convolution by the kernel.

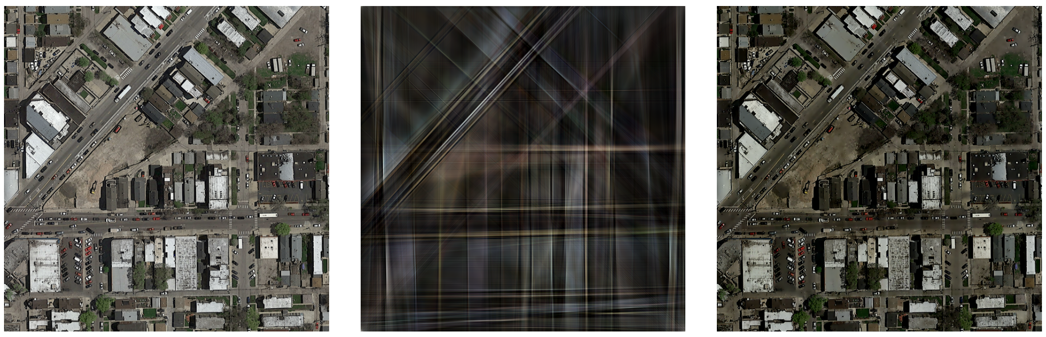
Fig. 3. MCA decomposition. *From right to left:* the original image, the ridgelet component and the starlet component

MCA process, the ridgelet and starlet forward and backward transform can be found in the `mw_transform.py` file of the submission.

V. CONCLUSIONS

The FCN approach did not work with the parameters we tested, although further exploration would be necessary to understand the underlying cause. The CNN approach allowed for a classification rate of 0.862 using a simple convolution kernel on the output that favors groups of pixels and straight lines. Furthermore, we realized that most of the contaminants come from false detection due to parking or roofs that look highly similar to roads even sometimes for a human eye. We designed the MCA algorithm with the aim to circumvent this issue and tried to compare two different approaches but this approach peak at 0.81.

References

[Do and Vetterli, 2003] Do, M. N. and Vetterli, M. (2003). The finite ridgelet transform for image representation. *IEEE Transactions on Image Processing*, 12(1):16–28.

[Joseph et al., 2016] Joseph, R., Courbin, F., and Starck, J.-L. (2016). Multi-band morpho-Spectral Component Analysis Deblending Tool (MuSCADeT): Deblending colourful objects. *aap*, 589:A2.

[Long et al., 2014] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *ArXiv e-prints*.

[Shelhamer et al., 2016] Shelhamer, E., Long, J., and Darrell, T. (2016). Fully Convolutional Networks for Semantic Segmentation. *ArXiv e-prints*.

[Starck et al., 2004] Starck, J., Elad, M., and Donoho, D. (2004). Redundant multiscale transforms and their application for morphological component separation. *Advances in Imaging and Electron Physics Series*.

[Starck et al., 2007] Starck, J.-L., Fadili, J., and Murtagh, F. (2007). The undecimated wavelet decomposition and its reconstruction. *Image Processing, IEEE Transactions on*, 16(2):297–309.