Computational Intelligence Lab — Satellite Road Segmentation

Neeraj Kumar, Lorenzo Laneve, Alessia Paccagnella, Tommaso Pegolotti
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In this project, we aim to create a classification model that segments aerial satellite images from Google Maps, separating roads from backgrounds. We present and compare two different methods to tackle this problem, both based on *Convolutional Neural Networks*: one based on a per-patch classification, and the other based on a fully convolutional network. Moreover, we show that bootstrapping, along with image augmentation techniques, including random rotation, zoom, brightness, and contrast, can induce a strong regularization on the models, improving the overall performance. The best model obtained in Kaggle's public test data achieved an F1 score of 0.91954.**

I. Introduction

Image segmentation is a fundamental topic in computer vision. It consists of partitioning a digital image into multiple segments, where every segment is a set of pixels, to understand what is the content of the image and allow easier analysis of each part. For example, it could involve separating foreground from background, or clustering regions of pixels based on similarities in color or shape.

Modern Computer Vision technology, based on AI and deep learning methods, has evolved strongly in the past decade, together with the increase of computing performances and the use of specialized graphics processing units (GPUs) to reduce computation time. Therefore today's image segmentation techniques use models of deep learning for computer vision to understand, at a level which was unimaginable only a decade ago, exactly which real-world object is represented by each pixel of an image. Machine learning can learn patterns in visual inputs and predict object classes that form an image, and this is mainly done with architectures called *Convolutional Neural Networks*.

In this project, image segmentation is applied to satellite images taken from *Google Maps*. Essentially, the task is to classify each pixel of the image with a probabilistic label $p \in [0, 1]$, in order to detect which parts of the images are made of roads ($p = 1$), and which are made of background ($p = 0$). Thus, in the end, we should have a model that outputs a grayscale mask, of the same size of the input image, containing the mentioned probabilities in each pixel. We are given a training set of 100 images of size 400×400, along with the corresponding ground-truth masks and an evaluation set containing 94 images of size 608×608. An example of training data is shown in Figure 1.

Figure 1: Example of training image (image 7) and its groundtruth mask. White means $p = 1$, while black means $p = 0$.

The paper is structured as follows. In Section II, we discuss two simple baseline models and analyze their weaknesses.
Section III shows a per-patch classification approach and how some major problems encountered on the baselines can be overcome. Section IV discusses a variant of the U-Net, a Fully Convolutional architecture introduced by Ronneberger et al. [1] for biomedical image segmentation, explaining why the architecture is particularly suitable for the road segmentation problem. Section VIII shows the F1 scores for all the models presented, while the rest of the report introduces data augmentation and ensemble predictions, which improved the performance of the models significantly.

II. Baseline Models

Before attempting to create a complex solution for our problem, we started from some very simple baselines, in order to have a meaningful reference point for the following results. In our classification task, the most common prediction is the background, as it occupies more pixels than roads in images. This can be seen using a zero classifier, i.e. a model that always returns zero (background) for all the pixels, which achieved a classification accuracy of about 0.73 in cross-validation, meaning that usually $\frac{3}{4}$ of the pixels in the images are made of background while only $\frac{1}{4}$ are linked to roads. This implies a class imbalance for our classification problem.
From this simple model, we started developing more significant baselines. We proceeded in the implementation of the following ones:

A. Per-Patch Logistic Regression

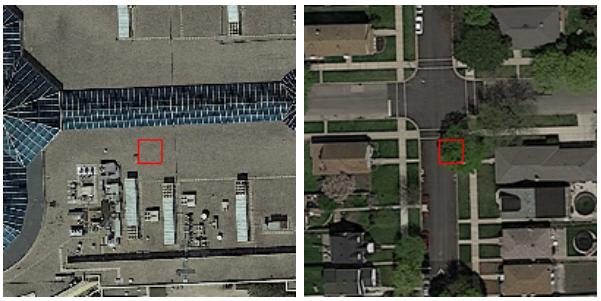
This model divides an image into patches of size 16×16 and then classifies each patch using a logistic regression

model. The features we consider are mean color and variance for each color channel and polynomials of them up to degree 5. Also, the training set is divided in patches of size 16×16, and the ground-truth patches are reduced to a single bit that is 1 if and only if the mean label of the patch is above 0.25. This model achieved an F1 score of 0.51 on the public test set.

B. Naive Convolutional Network Model

The most appropriate choice for this task is to use a Convolutional Neural Network. Indeed, a CNN is a feed-forward neural network that is very effective in reducing the number of parameters without losing the quality of the models. This is an advantage when dealing with images, as they have high dimensionality. For this reason, for the second baseline, we divided the image into patches like in the Logistic Regression model, but each patch is classified using a simple Convolutional Neural Network with 6 layers: two convolutional layers with, respectively, 32 and 64 filters of size 5×5, each followed by a 2×2 max pooling layer and ReLU activation. These two convolutional blocks are followed by one dense + ReLU layer with 64 units and a final single unit dense layer with sigmoid activation. This model achieved an F1 score of 0.73.

These two models present a weakness: a patch alone does not give enough information on whether it is on the road or on the background. Figure 2 gives concrete examples from data where the baseline models fail.



(a) Test Image 11 (b) Training Image 1

Figure 2: It is not possible to have a correct prediction by only looking at the patches (red squares).

III. Context-Sensitive Per-Patch Classification

Our first attempt to overcome the problem introduced in the previous section naturally extends the idea of the baseline models by considering, for every patch of the image, a **context window** of size 200×200 as input for classification, centered on the patch we want to classify (the **focus** of the window). Figure 3 illustrates an example of context window and its focus. With this approach, we created two different classifiers, all based on Convolutional Neural Networks:

Figure 3: We consider the context window (green square) to assign a class to the pixels of its focus (red square).

A. Context-Sensitive Convolutional Neural Network

This classifier is a Convolutional Neural Network consisting of 4 convolutional layers and 2 classification (fully connected) layers. Table I briefly describes the architecture. This model aimed to check if a context window could allow significant improvements to the per-patch classification approach.

Layer	**Parameters**
Convolution + ReLU + Max Pool	$32, 5 \times 5$
Convolution + ReLU + Max Pool	$64, 3 \times 3$
Convolution + ReLU + Max Pool	$128, 3 \times 3$
Convolution + ReLU + Max Pool	$256, 3 \times 3$
Dense + ReLU	64 units
Dense + ReLU	64 units
Dense + Sigmoid	1 unit

Table I: Architecture of the Naive CNN baseline model. All max pooling layers have 2×2 size. Before each max pooling layer a dropout with $p = 0.3$ is applied, while a dropout with $p = 0.5$ is applied to each dense layer.

B. Xception

Xception is a deep convolutional neural network that is a linear stack of depthwise separable convolution layers.

It is a modern architecture presented in a 2017 CVPR paper [2], and proven to outperform counterparts like *VGGNet*, *ResNet*, and *Inception-v3* when trained on the *ImageNet* dataset. An open-source implementation of Xception using Keras and TensorFlow is provided as part of the Keras Applications modules, under the MIT license. Therefore we decided to see if this big and sophisticated network, when trained with our small dataset, gave better results than a basic convolutional network.

One experiment we tried with this architecture was exploiting an open-source dataset [3] in addition to the one given on Kaggle and disjointed from it. We preprocessed them by cropping the new images to a dimension of 400×400 and by zooming them to have the same proportions of the

provided ones. We added 48 images, reaching a total of 148 training images and relative ground-truths. These led to a small improvement in the F1 score, as the table in section VIII shows. However, this approach revealed to have limitations due to the big dimensions of this network and the memory limits on our environment, which prevented us from using a higher amount of images from the new dataset.

IV. U-Net Architecture

The approach we have described in III, although scalable, was not accurate enough on test data. Because of this reason in this section we introduce a different architecture that lets us outperform the rest. This new model is a shrunken version of the architecture defined in [1], and it is depicted in Figure 4. It consists of a Fully Convolutional Neural Network, composed of two phases: a compressing phase of four max pooling layers and an expanding phase consisting of four upsampling layers.
From Figure 4 one can also notice that the most compressed layer still has shape 19×19, therefore not all the spatial information is compressed, and this enforces a local property on the segmentation.

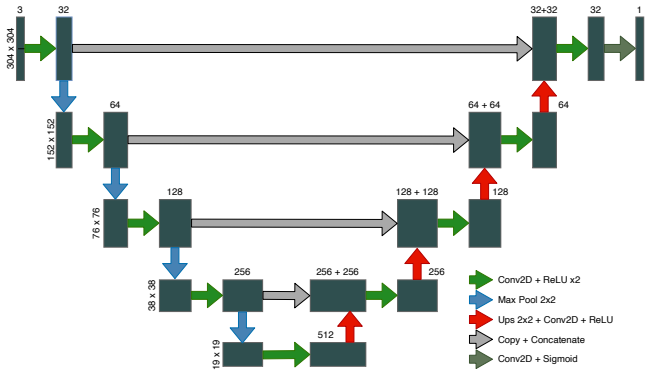


Figure 4: Overview of the U-Net based architecture.

The input and output images have size 304×304, so that the prediction covers exactly $\frac{1}{4}$ of the test images. Therefore, our first predictions are made by simply composing a 608×608 image with four, non-overlapping 304×304 masks. However, we observed evident discontinuity between parts, probably due to the fact that segmentation is less accurate on the borders. Our workaround to mitigate this discontinuity is to consider 9 different predictions instead of 4, and crop them like in Figure 5.

V. Bootstrapping and Data Augmentation

Since the training data we have is very small (100 images), data augmentation was crucial for the performance of all the models we created. For every model, we used a bootstrapping technique to extract batches that are provided to the neural networks during the training phase. Since our models take smaller images as inputs, this procedure chooses

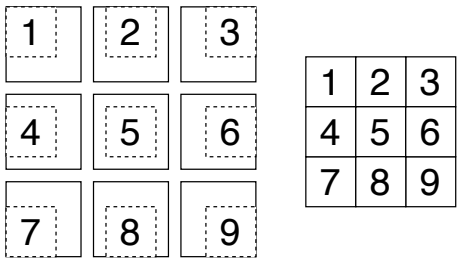


Figure 5: Using 9 predictions to compose the 608x608 image. Dashed images are the cropped windows of size ∼200x200 we take from each prediction.

Figure 6: Rotation/Flip/Zoom Augmentation for U-Net. The red lines are the borders of the original image.

an image from the set uniformly at random and then samples a sub-image of the desired size from it. This method allows the introduction of a simple augmentation: apply random transformations to the sampled sub-images.

A. U-Net

Since our U-Net architecture has input size 304×304, we sampled sub-images of this size. Data is then augmented using the following transformations:

- **Rotation/Flip/Zoom augmentation**: a rotation angle is chosen uniformly at random in $[0, 2\pi]$, a flip is done with probability $\frac{1}{2}$, and a zoom factor is chosen at random between $[0.8, 1.2]$. The image is filled with reflection on boundaries, as shown in Figure 7;
- **Color augmentation**: A brightness factor $b \in [1, 1.1]$ and a contrast factor $c \in [1, 1.1]$ are chosen independently and uniformly at random. These two factors are applied to an image X using the following transformation:

$$X' = \kappa\left(\frac{1}{2} + c\left(bX - \frac{1}{2}\right)\right) \tag{1}$$

where $\kappa(x) = \max(0, \min(1, x))$ and addition with scalars are applied component-wise.

Brightness augmentation revealed to be a strong regularizer for the U-Net model, allowing it to recognize asphalt with different shades of gray.

B. Per-Patch Classification Models

The sampled sub-image is a window, and the classification label for that window is taken by computing the mean label of its focus in the corresponding ground-truth: the label is assumed to be 1 for the entire patch if the mean is at least 0.25. Then, the following transformations are applied:

- **Rotate/Flip augmentation**: the sub-image is randomly rotated in steps of $90°$, and flipped with probability $\frac{1}{2}$;
- **Color augmentation**: the following transformation is applied to the sampled sub-image X:

$$X' = \kappa\left(bX\right) \tag{2}$$

where $\kappa(x) = \max(0, \min(1, x))$ and $b \in [1, 1.2]$ is chosen uniformly at random.

Notice that these transformations do not change the value of the ground-truth label.

VI. Ensemble

In order to get more robust predictions, we implemented a very simple ensemble we call **Rotate and Mean**: classify or segment an image for each of the 8 possible rotation/flip combinations, and compute the mean of the predictions. This not only decreased the number of false positives significantly, but it also induced invariance with respect to rotation and flip, a property that is reasonable for this kind of segmentation problem.

The second ensemble strategy we tried was a **bagging** of 9 different U-Net models, trained with slight changes to the data augmentation parameters. Since the evaluation set only considers 16×16 patches, we decided to compute a patch-wise majority. This significantly improved the score, at the expense of a higher (but still reasonable) prediction time. Section VIII quantifies the improvements obtained by the two ensemble strategies.

VII. Software Architecture and Training Details

Our models are built and implemented using *Keras 2.3.0* with backend *Tensorflow 2*. Other libraries that we used to open images and to perform data augmentation are *numpy 1.19.0* and *PIL 7.2.0*, while the Logistic Regression model is implemented using *scikit-learn 0.26*.

All the experiments were done on a Google Colaboratory environment, with GPU acceleration. The Neural Networks are trained with the Adam optimizer [4] with 0.001 as initial learning rate, minimizing a *binary cross-entropy* loss with sigmoid activation.

VIII. Results

We can now summarize and compare the results obtained, which are computed on Kaggle's public test set.

Other combinations are omitted because they did not manifest significant differences.

Model	F1 Score
Zero Classifier	0.00000
Logistic Regression	0.51071
Naive (Context-free) CNN	0.73277
Per-Patch Basic CNN	0.87808
Per-Patch Xception	0.89128
Per-Patch Xception + AD	0.90078
Per-Patch Xception + AD + RM	0.90250
U-Net	0.88115
U-Net + RA	0.88886
U-Net + RA + BA	0.90405
U-Net + RA + RM	0.90565
U-Net + RA + BA + RM	0.91652
Bagging of 9 U-Nets	**0.91954**

AD: Additional Data, **RM**: Rotate and Mean,
RA: Rotate+Flip+Zoom Augmentation,
BA: Brightness+Contrast Augmentation

(a) Xception prediction (b) Unet prediction

Figure 7: Final prediction on a sample image (Test X). The red overlay indicates the pixels classified as "background".

IX. Conclusions

In this project, we created a deep learning model able to segment roads from satellite images taken from *Google Maps*. The paper discussed several approaches to the problem; each of these can detect with a certain accuracy which part of an image contained a road.

One thing we observed from the data is that U-Net, compared to per-patch classifications, presented cleaner predictions that easily recognized even roads covered by trees and bridges, whereas per-patch classifiers yield false negatives more often. Both models, however, rarely misclassified background pixels.

One of our expectations was that an already known architecture like Xception could give better results than other simpler models. However, its complexity and its training time did not give us much room for improvements. On the contrary, the introduction of data augmentation (section V), the use of ensemble (section VI) contributed reaching higher performances with all the models. In general, the results we achieved are improvable but satisfying.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv e-prints*, p. arXiv:1505.04597, May 2015.

[2] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *arXiv e-prints*, p. arXiv:1610.02357, Oct. 2016.

[3] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[4] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.