U-Net based Network Ensemble for Satellite Road Segmentation

Daniele Chiappalupi, Elena Iannucci, Gianluca Lain, Samuele Giuliano Piazzetta
Group: PochiMaPochi
Department of Computer Science, ETH Zurich, Switzerland
{*dchiappal,eiannucci,gilain,spiazzetta*}*@student.ethz.ch*

***Abstract*—With the spread of Convolutional Neural Networks (CNNs), researchers achieved new state-of-the-art in previously challenging problems. One popular challenge in this field is image segmentation, which tackles the problem of detecting and marking objects inside an image. In this work, we describe our approach to the Kaggle competition "ETHZ CIL Road Segmentation 2020", that addresses the problem of road segmentation from satellite images. Our proposed method is an ensemble of deep neural networks based on the U-Net architecture and leveraging ResNet and Xception pre-trained backbones. Our approach allowed us to achieve a final accuracy of 0.92427 in the Kaggle challenge.**

I. Introduction

Remote sensing is a technique employed by several science fields characterized by satellite or aircraft-based sensor technologies to detect and classify objects on Earth. One domain-specific applicability deriving from this data collection is road segmentation. Road information extracted by this process is necessary for applications such as automotive navigation systems, autonomous driving, and geographic information update. Road segmentation is a challenging task since thousands of kilometers of new roads are newly constructed every day. Thus, a considerable volume of high-resolution satellite aerial images must be analyzed to capture the new urban changes. The initial approach consisted mainly of manual annotation, but this process is not scalable due to the unsustainable workload. Therefore, researchers extensively tried to find new techniques to automatize this analysis process. An initial approach dates back to 1994 by [1] and consists of road object-segmentation via texture-based filters. Although they can achieve a decent edge detection, this manual feature design method lacks generalization, since it is vulnerable to environmental changes. Afterward, new techniques in the field of machine learning were applied to overcome these limitations. The first attempts of road segmentation using supervised machine learning are investigated by Civco [2] and Das [3], which both leverage SVM to detect road areas.

In recent years, the development of deep learning brought substantial improvements in computer vision tasks. The adoption of deep neural networks in fields such as scene recognition and object detection [4] outperformed all previous methods, defining a new baseline of this area. In the field of road extraction, Mnih [5] proposed a method that combines Boltzmann machines with tailored preprocessing and postprocessing steps to further refine the segmentation. Saito et al. achieve more accurate and promising results in [6], where the authors proposed the use of Convolutional Neural Networks (CNNs) to extract roads directly from raw image data. After the ImageNet Challenge [7, 8] proposed initially in 2012, researchers explored new and usually deeper networks that resulted in [9–11]. These architectures respectively achieved an increasingly innovative and more powerful state-of-the-art.

Examples of remarkable and iconic networks architecture families are Inception [10], one of its successor, Xception [12], and ResNet [11]. The last one tackled the vanishing gradient problem, a common issue in very deep networks, by introducing the deep residual learning framework. The CNNs architectures were also applied to solve image segmentation tasks. Two significant implementations are the Fully Convolutional Networks (FCNs) [13], which used upsampling (a.k.a. "skip connections"), and Ronnenberg U-Net [14], that concatenates features maps from different layers to increase the final segmentation accuracy. They both achieve promising results via a combination of low-level details and high-level semantic information. Several works focused on road segmentation, such as [15], which extends FCNs to detect both roads and buildings, and Zhang [16] that uses the U-Net encoder/decoder schema, where the encoder is an instance of a ResNet.

In this work, we propose an extension of the idea presented in [16], consisting of an ensemble of multiple deep U-Nets. In each of the networks, we implement the encoder as either a ResNet or an Xception, leveraging transfer learning.

II. Models and Methods

A. Dataset

The dataset provided for this task is composed of 100 400×400 pixels RGB satellite images acquired from Google Maps. These images are used as the training set (we will refer to them as *competition data*), while the test set consists of 94 608×608 pixels RGB images. We are also provided with 400×400 pixels ground-truth masks, where each pixel gets assigned a value $v \in [0, 1]$ representing the probability of belonging to road area ($v = 1$) or background ($v = 0$).

B. General Network Architecture

In this project, we implemented a modified version of the CNN named U-Net, proposed in [14]. U-Net is composed by a downsampling step (encoder), and an upsampling step (decoder). The encoder architecture follows the standard idea of a neural network: the input is processed via ordered transformations performed by subsequent layers, that process the input pixels of an image, gradually transforming them into a class probability measure. The most common layers apply learned convolution filters, that leverage the idea of spatial localization of the input. Each neuron in a layer l processes a fixed size patch of the previous layer $(l-1)$, producing an output value that is the weighted sum of the values contained in the patch. In particular, to achieve the shift-invariance inside an image, the weights are shared across all neurons of a layer. The output of the layer is then non-linearly transformed by an activation function. Our implementations adopt the de-facto standard *ReLU* and one of its variants, i.e., *Leaky ReLU*. To allow the network to learn more high-level features, it is necessary to enlarge the receptive fields. We introduced Max-Pooling layers that allow downsampling the initial image, by reducing $2{\times}2$ output blocks into the corresponding maximum value. While the just-described process aims to learn the best-representative context of an image, semantic segmentation requires a final output mask obtained by the subsequent upsampling step. U-Net concatenates intermediate decoder representations with the corresponding equal-sized encoder layer output. These so-called "skip connections" allow the final segmentation to be more detailed. Finally, in the last layer, we apply the sigmoid function that maps each output to the respective probability measure in the range $[0, 1]$.

C. Implementation details

Inspired by [16], we modified the basic structure of the U-Net architecture, by using as encoding blocks the ones from either the Xception architecture proposed in [12] or one of the recent Resnet's implementation proposed in [17]. We adapted these networks to enable learning from 400x400 pixels images. Furthermore, we initialized the networks with pre-trained weights on the ImageNet dataset. On the decoder side, we modified the decoding blocks turning the simpler convolutional blocks into residual blocks, which reduces the networks' training time. The overall shape of the networks is symmetric and presents 6 levels of encoding and decoding blocks. Every block presents convolutional layers, characterized by a kernel size of $3{\times}3$ and an increasing number of filters. Starting from the input/output blocks down to the deepest layer, the number of filters is 16, 32, 64, 128, 256, and 512. Finally, each encoding and decoding block implements batch normalization and dropout (with $p = 0.5$ for Xception based networks and $p = 0.2$ for ResNet based ones) regularization mechanisms.

D. Data Augmentation and Pre-processing

Random transformations: Training a deep neural network is highly data-demanding, and the training dataset we have is rather small. One way to solve this problem is by using extensive data augmentation to increase the available samples significantly. We applied this technique by performing random transformations on every image of the training set and the corresponding ground-truth mask. More precisely, at the beginning of every training batch, we perform the random transformations on each image with a probability $p = 0.75$. By augmenting the data, the network will have a lower probability of incurring the same image more than once, thus reducing the risk of overfitting. The transformations we chose are the following: horizontal and vertical flips, $90° * k$ rotations with $k \in \{1, 2, 3\}$, and elastic transformations. Moreover, we applied some filters to the RGB values of the images, such as random alterations in contrast, saturation, hue, brightness[1].

Automatic data retrieval: To further improve the available samples on the training set, we gathered roughly 1000 additional images using the Google Maps API. We collected the images from different USA cities, such as Los Angeles, Chicago, Houston, Phoenix, San Francisco, and Boston. To be consistent with the given images, we tried to keep the same zoom-level and to select cities that present similar urban structures as in the images provided from the competition. These images are subject to the same process of random transformations described above. Furthermore, we checked that every newly introduced sample was not already part of the test set. We will refer to these images as *Google data*.

Image pre-processing: We followed the conventional approach of normalizing the input images before feeding them to our models, by dividing each pixel value by 255.

E. Training

Given the imbalanced nature of the problem, we learned all the best model weights by minimizing the dice loss function [19]:

$$DL = 1 - \frac{\sum_{n=1}^{N} p_n r_n}{\sum_{n=1}^{N} p_n + r_n}$$

where $0 \leq p_n \leq 1$ is the positive class predicted probability and $r_n \in [0, 1]$ the real label corresponding to each input unit $n \in N$ (each single pixel composing an image).

The training process and the involved tuning parameters are the same for all the models. We arranged the training process into two different phases. Firstly, we train the network on Google data, starting from a learning rate of $1 \cdot 10^{-4}$. It follows a fine-tuning phase on the competition data, where the learning rate starting value is lowered to $1 \cdot 10^{-5}$. All models are trained with Adam optimizer [20], configured with the default parameters, apart from the learning rates

[1]All the transformations we reported are implemented by the library Albumentations [18]

that are initialized as just described. We halved the learning rates every time the loss stopped decreasing.

It is a standard approach to monitor the training process on a validation set, to avoid incurring into overfitting. However, in our case, the limited size of the training set introduces two main problems: a) reserving images for a validation set implies a substantial reduction of the training set; b) a small validation set would lead to fluctuating performance evaluation, thus being poorly informative. Therefore, we decided not to adopt a validation set, but instead, we trained our model with a fixed number of epochs, varying it as a tuning parameter. To counteract this choice, we added strong regularization in the models, as described before.

F. Predictions

Submission format: All the network architectures we adopted accept 400×400 pixels input images. Conversely, the competition's test images have a different size of 608×608. To overcome the incompatible sizes, we adopted the following strategy: we subdivide each test image into 4 overlapping 400×400 images, each of them anchored to a different corner. Thus, we predict 4 different output masks, later combined to obtain the final prediction mask by averaging the intersecting areas. The Kaggle competition requires to output a $\{0, 1\}$ label for every 16×16 patch of an image. Since our models output pixel-wise predictions, we derive per-patch labels by averaging the pixel probabilities inside one patch, as illustrated in Figure 1. We assign a predictive label 1 if the resulting average is higher or equal than a threshold T.

Ensemble: High flexible models as deep CNNs are usually characterized by high variance in the resulting predictions. Thus, the best model selection approach would lead to a loss of information. In our case, selecting the best-model leads to losing portions of streets exclusively detected by less performing ones. To tackle this problem, we adopted the *ensemble* method, which consists of combining the results of different models. The resulting combination is characterized by lower variance and an averaged bias when compared to the single models. In our experiments, to produce the ensembles, we store the class probabilities prediction related to each pixel of the image, for each model. Afterward, we average these values to obtain the final probability.

III. Experiments

In this section, we illustrate the details of our experiments, reporting and commenting their results. We present each model along with its Kaggle's score, which measures the predicted patch labels accuracy on the 49% of the test set. In each experiment, the final patch labels are predicted as in II-F, with a threshold T empirically setted to 0.4.

A. Baseline models

As a first attempt and as a baseline to compare subsequent results, we decided to adopt the following models:

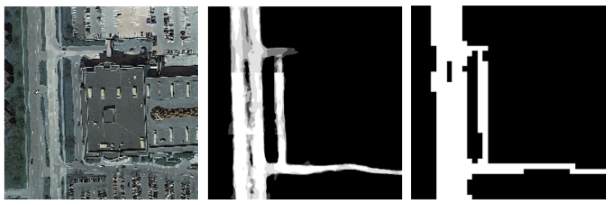
(a) Test Image (b) Prediction (c) Patched Image

Figure 1. Difference between a pixel-wise prediction and the corresponding patched submission format.

1) Logistic Regression: We trained a Logistic Regression binary classifier on image patches of size 16×16. Firstly, we implemented the naive Logistic Regression, setting the l2-regularization strength parameter to 10^{-5}, we obtained a poor result of 0.40917. On the other hand, by introducing different class weights, namely 0.3 for the negative class and 0.7 for the positive class, we achieved a better result of 0.554. Despite these results helped us confirming the unbalanced nature of the problem, they are not satisfactory at all, as it is just above random guessing.

2) U-Net: The second attempt was the vanilla U-Net model, as described in II-B. We first used the model as a binary classifier, attaching a dense layer to the output of the network and feeding the latter with image patches of size 16×16 as input. In doing so, we reached a score of 0.80041. We then tried to feed the U-Net with full-sized images as input, detaching the final dense layer and using the network to build full-sized road masks as outputs. This expedient enlarges the context of the predictions, increasing the accuracy to 0.83989. For this reason, we continued our experiments by analyzing full images instead of patches.

B. Data retrieval

After adding the *Google data* as explained in section II-D, we experienced a considerable improvement in terms of predictions accuracy. This was to be expected as deep neural networks are notoriously data-hungry, and training them on more data gathered from many different locations allows them to generalize better. Table I shows that training the U-Net using the *Google data* leads to a gain of roughly three percentage points of classification patch-accuracy, achieving a score of 0.86973. Hereby, all of our experiments will be done using the gathered *Google data*.

C. Pre-trained Backbones

We obtained even more promising results by using pre-trained networks in the encoding part of the U-Net as explained in II-C. Hereby, we will call the U-Net with the ResNet-backbone *U_ResNet*, and on the same note the U-Net with the Xception-backbone *U_Xception*. The former achieved a mean score of 0.90875, whereas the latter performed even better, reaching a mean score of 0.91661. These results should not be surprising: the Xception outperforms

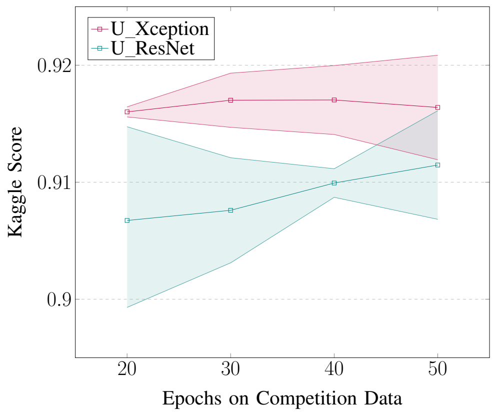

Figure 2. Public score trends of the two networks. The shaded areas represent the recorded standard deviations of the networks scores.

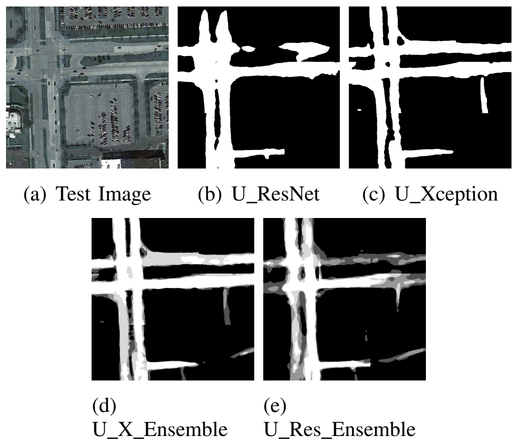

Figure 3. Different outputs of different models on the same image.

the ResNet also in the ImageNet challenge mentioned in Section II-C, proving to be better at finding latent representations of images. Figure 3 shows a comparison between the outputs of the two different networks. Since the training of these networks is divided between *Google data* and competition data, we had to tune the number of epochs of training on both the datasets. Hereby, we will call such parameters *Google data epochs* (GdE) and *Competiton data epochs* (CdE), respectively. We firstly tried varying the GdE while keeping the CdE fixed. This didn't lead to much variation in the results: the fine-tuning on the competition data lowers the difference between models trained with different GdE. We therefore decided to keep this parameter fixed at 40 epochs, because such value resulted to be slightly better than the other values we tried. We then proceeded to vary the CdE, and unsurprisingly we experienced much more variation in the results than before. Figure 2 shows the scores trends of the two types of networks when varying the CdE. We varied such parameter linearly between 20 and 50: we chose 20 as lower bound because we noticed it was the minimum number of epochs necessary for the networks to start understanding the hidden representations of the competition data; likewise, we chose 50 as upper bound because we observed that the networks started overfitting if trained for more epochs.

D. Ensembles

We finally report the results of our major contribution, that turned out to be the best-performing model. In each ensemble, we average the predictions of four different networks with the same architecture, each trained with a different number of epochs on the competition data. Such number of epochs are the ones described in Section III-C. Since we developed two different networks, we also built two different ensembles: one that averages the predictions of four different *U_ResNet* models and one that averages the predictions of four different *U_Xception* models. We named them *U_Res_Ensemble* and *U_X_Ensemble*, respectively. Both the ensembles improved the performances of the networks they are built from: the former reaches a mean score of 0.91615, whereas the latter achieves a mean patch-accuracy of 0.92359, resulting to be the best model among the ones we tried. Moreover, both the ensembles proved to have a considerably small variance between different runs. Figure 3 shows differences between ensemble's predictions.

TABLE I
FINAL RESULTS OF OUR CONSIDERED MODELS. FOR EACH MODEL, WE REPORT ITS BEST SCORE, MEAN SCORE AND STANDARD DEVIATION.

Model	Public Score	Mean Score	Std. Dev.
Logistic Regression			
- w/o Class Weights	0.40917	-	-
- w/ Class Weights	0.55400	-	-
U_Net on patches	0.80041	-	-
U_Net on full images			
- w/o Google data	0.83989	0.83493	0.00593
- w/ Google data	**0.86973**	0.86124	0.00751
U_Xception	**0.91965**	0.91661	0.00253
U_ResNet	0.91608	0.90875	0.00616
U_X_Ensemble	**0.92427**	0.92359	0.00073
U_Res_Ensemble	0.91659	0.91615	0.00061

IV. SUMMARY

In this report, we presented subsequent improvements to the base structure of the U-Net to successfully handle the road segmentation task. Initially, we verified how proper augmentation and extension of the data allow us to achieve better performance. Then, we leveraged transfer learning and residual blocks to obtain more precise segmentation and a speedup in the training process. More precisely, we replaced the encoding blocks of the U-Net with either ResNet or Xception blocks. Finally, we obtained our best result by adopting an ensemble of our best performing networks, that achieved the best score of 0.92427 on Kaggle and produced visually-satisfactory results.

References

[1] J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," in *Proceedings of the Intelligent Vehicles' 94 Symposium*. IEEE, 1994, pp. 260–265.

[2] M. Song and D. Civco, "Road extraction using svm and image segmentation," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 12, pp. 1365–1371, 2004.

[3] S. Das, T. Mirnalinee, and K. Varghese, "Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images," *IEEE transactions on Geoscience and Remote sensing*, vol. 49, no. 10, pp. 3906–3931, 2011.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[5] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in *European Conference on Computer Vision*. Springer, 2010, pp. 210–223.

[6] S. Saito, T. Yamashita, and Y. Aoki, "Multiple object extraction from aerial imagery with convolutional neural networks," *Electronic Imaging*, vol. 2016, no. 10, pp. 1–9, 2016.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[14] O. Ronneberger, P. Fischer, and T. 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.

[15] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, pp. 6054–6068, 2017.

[16] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[18] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *ArXiv e-prints*, 2018.

[19] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.