Road segmentation challenge

Mateo Lopez
EPFL
Lausanne, Switzerland
mateo.lopez@epfl.ch

Francesco Ballestrazzi
EPFL
Lausanne, Switzerland
francesco.ballestrazzi@epfl.ch

Xingjian Christophe Huang
EPFL
Lausanne, Switzerland
xingjian.huang@epfl.ch

***Abstract*—Accurately extracting roads from aerial images is an efficient way to create reliable maps. This is important for several impactful tasks, such as urban planning and effective resources allocation. In this project, we use Convolutional Neural Networks (CNNs) to detect roads in a satellite image. We focus on the U-net architecture for our deep learning models, and apply various techniques to enhance the quality of our predictions of road pixels in a satellite image.**

I. Introduction

Semantic segmentation consists in automatically extracting from images the areas that describe particular objects depending on the specified task. In this project we want to do semantic segmentation of roads from satellite images. This task is not trivial as satellite images not only contains the roads but also various obstacle that can partially hide them. Trees, cars, buildings and others are omnipresent and enforce a road segmentation model to know where the roads are even when those objects obstruct a part of it.

In addition, we are working on a limited data set of training images (100 images). It is important that our model can learn from it without over fitting. To do so we will explore ways to augment our data to artificially increase the number of images on which our model will have access to train.

In this report, we will present two different models designed to fulfill this task.

II. Dataset

The dataset contains satellite images of roads, each one with the relative segmentation mask, where 1 identifies a road pixel and 0 other elements. The training set is composed of 100 images, while the test set is composed of 50 images.

The goal of this challenge is the semantic segmentation of roads in satellite images, i.e given a new image, the algorithm must be able to tell at each specific position whether there is a road or not. The main difficulties are represented by the presence of objects that could hinder the road recognition, e.g trees, shadows, buildings etc.

In the next paragraph we present the different models and methods used to reach this goal and to try to solve these problems.

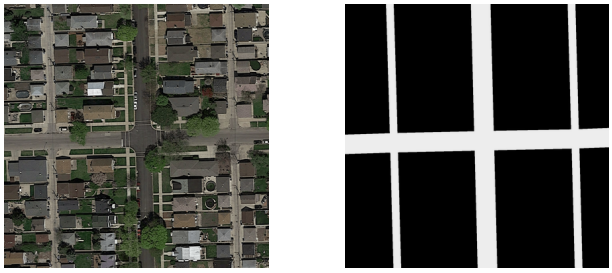

Fig. 1: Training set image and corresponding semantic segmentation ground truth.

III. Models and methods

In this project, we have been confronted to two major challenges. First of all, our training set is really small (100 images) so we need to find a way to pre-process our existing data in order to generate more training data, i.e. we need to augment our data. The second challenge was to find a model that is able to process a feature space with high dimensionality (160,000 features for each image). Here we will present our approach to overcome these issues.

A. Data augmentation

To augment the training data, we used different combinations of image transformations. Especially, we tried to flip horizontally and vertically, to apply different rotations, to zoom, to shift and to shear our images. This largely augmented our training set and allowed us to train our model with a reasonably sufficient amount of data. In particular we used the Keras augmentation dedicated function "ImageDataGenerator" to apply the transformations. We keep in mind that augmenting the data helps, but only to a certain extent because it

transforms only slightly the dataset, and it is incapable of creating completely new data. We must also be careful that the changes applied to our images do not affect the properties of the objects we are trying to detect. As an example, if we distort the image with an unnatural transformation, the roads will no longer keep their distinguishing features, and this will just be confusing for the model.

B. Baseline model

The baseline model is a simple neural network model with two convolution layers [2], each with relu activation and each followed by a Maxpooling layer. A fully-connected layer is then added at the end to predict an output image. No data augmentation nor regularization are performed in this model. The model takes as input a batch of images in the format (batch size, height, width, pixel channels). For simplicity, we use batch size = 16, width = 256, height = 256, pixel channels = 3.

The baseline CNN model has the following structure:

Baseline Model Structure

Layers	output shape
Input layer	(16, 256, 256, 3)
Convolution layer	(16, 256, 256, 32)
MaxPooling layer	(16, 128, 128, 32)
Convolution layer	(16, 128, 128, 64)
MaxPooling layer	(16, 64, 64, 64)
Fully-connected layer (Output)	(16, 256, 256, 64)

C. U-Net

For our model architecture, we decided to use a deep learning approach as they have proven their ability to process high dimensional features spaces. In particular, since we are working with images, we wanted to use convolution layers as they work particularly well on this type of data.

We decided to use models architectures that allow us to do semantic segmentation on images, all of them will be based on the same key principle described in the U-net architecture [1]. It consists in two parts, an encoder and a decoder block. The encoder consists in successive convolution layers that will extract from the images meaningful features and pooling layers which reduce the dimensionality space and keep only the most relevant local features learned from previous layers. The decoder will then use these features to reconstruct the final segmentation.

As the encoder erase a lot of information about the original images, it can be hard for the decoder to reconstruct a segmentation with the same “shape” as the input image. More precisely, since the CNN is translation invariant (i.e. the features of a road does not depend on its absolute position in the image), the initial position of the road pixels in the image need to be retrieved and fed to the final output. To address this problem we give to the decoder additional information. This is the skip-connection method, the decoder has access to the intermediate outputs of the encoder, thus they still have information about the original structure of the image.

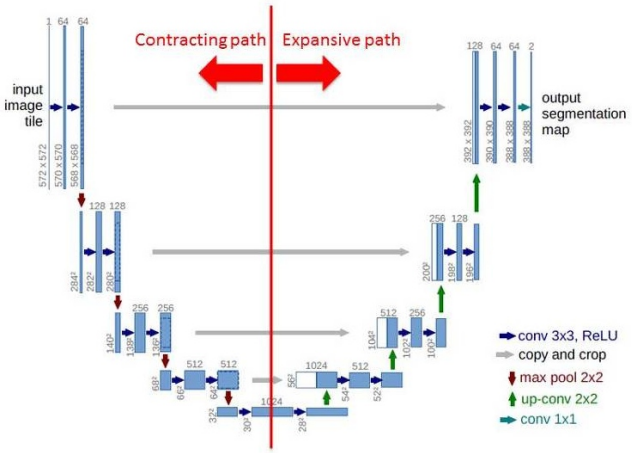

Fig. 2: Network architecture.

In this project, we used three different models based on this approach. Two classical U-net architecture with different structures for the encoder and decoder blocks and a U-net architecture that uses transfer learning.

The classical u-net model structure is shown in the table below. The encoder contains three blocks, each with a ”Convolution-Dropout-Convolution” layout, and a MaxPooling layer is added between two consecutive blocks. The decoder contains two blocks, each with ”Upsampling-Convolution-Dropout-Convolution” layout, and the output of each upsampling layer is combined with the output of the last convolution layer in the corresponding block in the encoder (skip-connection).

Classical U-net Model Structure

Layers	output shape
Input layer	(None, 400, 400, 3)
Encoding layers (3 blocks)	(None, 100, 100, 128)
Decoding layers (2 blocks)	(None, 400, 400, 32)
Final convolution layer	(None, 400, 400, 1)

1) Transfer learning: In the last U-net model, we use, as encoder, a previously trained network on a large data set of images. This technique is called transfer learning

Transfer Model Structure

Layers	output shape
Input layer	(None, None, None, 3)
Encoding layers (7 blocks)	(None, None, None, 1)
Decoding layers (4 blocks)	(None, None, None, 1)
Final convolution layer (Output)	(None, None, None, 1)

[2]. The idea here is to use a really generic network that was trained on very different images, we can then suppose that this network is able to extract the relevant features which we need to construct a road segmentation of our satellite images.

Such a network would obviously take a long time to be trained from scratch, but the pre-trained weights can be downloaded and loaded quickly into our model.

We use this network as a feature extractor. Then we just have to train a decoder for our specific task. There are two different ways to train this model. One is to use the encoder model as it is, the weights are fixed and we only train the decoder block. The other one is to allow the encoder to train his weights in order to adapt them to the new problem. In this project we tried both methods.

The structure of the transfer learning model is summarized in the above table. To allow input data flexibility, we set the input shape to be (None, None, None, 3). The model will automatically detect the size of the input images and change the "None" values accordingly. This model is rather complex and has many layers. The encoder part consists of 7 consecutive blocks, each with convolution, normalization and AveragePooling layers. The decoder uses upsampling, convolution and normalization layers.

2) Regularization: To avoid over-fitting of our model, we used a popular technique in deep learning architectures, called "dropout" [3]. During the training, some features are randomly not transmitted to the next layer. Using dropout layers force the model to find a good prediction even when not all the features are available, so that it cannot over-fit and put too much emphasis on a few features only. That way, when the model will predict unseen data, it should be able to give good results even if the features extracted from the image are not completely similar to the ones extracted during the training.

3) Loss: As loss function we used in the first place the binary cross-entropy (BCE) loss function as the problem can be seen as a binary classification problem. Each pixel is classified as being road (1) or not road(0).

$$L(p) = -(y \cdot log(p) + (1-y) \cdot log(1-p))$$

We then tried to use a combination (sum) of BCE with intersection over union (IOU) loss function. IOU is defined as follows:

$$L(T,P) = 1 - \frac{T \cap P}{T \cup P}$$

where T stands for the true label image and P for the prediction of the output image.

4) Optimizer: The optimization algorithm used in our model is Adam used with the default learning rate 0.001.

IV. Framework

The model has been implemented using the Keras library, with TensorFlow backend.

We used also another open source library "Segmentation models" to test the transfer learning model in a faster way.

We train our models using Google Colaboratory that offers free GPUs, with daily usage limits.

V. Results

After training the models, we can predict road segmentations on the given test set. The following example prediction is from our U-net model using transfer learning based on EfficientNet [4].

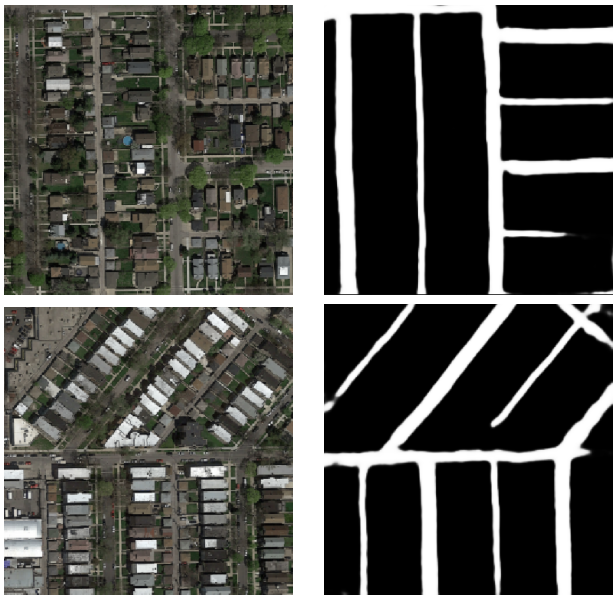

Fig. 3: Test set images and corresponding semantic segmentation predictions.

We tried to apply different combinations of data transformation in order to augment the data, such as image rotation, flipping, and so on. Because not all transformations are relevant to get a good road segmentation, we must find out which ones are most effective for this task.

The following table summarizes the prediction results of using different data augmentations with a transfer learning model based on EfficientNet with frozen weight. Since data augmentation is task-specific and not so much model-specific, we argue that the best combination should be reasonable for all of our U-net models.

Transfer learning using EfficientNetb0 with frozen weights. 100 epochs with 128 steps per epoch.		
Augmentation	train f1	test f1
No augmentation	0.9428	0.628
flp	0.9835	0.685
rot	0.9589	0.779
flp rot	0.9566	0.783
flp rot zoom	0.9505	0.858
flp rot zoom shear	0.9519	0.85
flp rot zoom shift	0.9453	0.857

We can see from the results that zooming on the images and applying rotations on them improved the model performance on the test set. This makes sense as this transformations generate images where the road have different angles and thickness, this helps the model to predict roads of any size and orientation.

From this data augmentation parameters tuning, we decide to keep the combination which gives the highest test score. This combination is given by (flip, rotation, zoom, reflect). Then, using this most effective data augmentations, we compare the prediction results on the test set after training on different base U-net models.

Model used	encoder freeze	test f1
baseline	-	0,82
our U-net	-	0,827
TL with Efficient-Netb0	yes	0.861
TL with vgg16 [5]	yes	0.811
TL with inceptionv3 [6]	yes	0.842
TL with resnext101 [7]	yes	0.871
TL with Efficient-Netb0	no	0,884
TL with vgg16 [5]	no	0,882
TL with inceptionv3 [6]	no	0.872

From the results we got here, the best combination of parameter that we retained for our model is the transfer learning model with the EfficientNetb0 model as encoder where we allowed the weights to keep training. The data augmentation that worked better consisted in applying to our data vertical and horizontal flips, rotations and zooming.

VI. Improvements and final thoughts

It is quite impressive that deep learning model can be so effective at finding objects in a picture with so few available training data. However, better recall is still needed to draw reliable maps. As mentioned previously, there are several ways in which we could improve our predictions. One such way involves the use of better resources: simply training our current models longer and on larger datasets can definitely boost the performance of our predictions. Another way could be to improve the model's output via post-processing techniques, which can sometimes compensate the weaknesses of the model in particular situations, like disconnected or diagonal roads.

Due mainly to computational constraints, we couldn't implement all these ideas, but it was still really enriching to go deeper into the semantic segmentation topic.

References

[1] Olaf Ronneberger and Philipp Fischer and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation" CoRR, 2015.

[2] Yann LeCun, Yoshua Bengio e Geoffrey Hinton, "Deep learning", in Nature, vol. 521, nº 7553, 2015.

[3] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", J. Mach. Learn. Res., 2014.

[4] Mingxing Tan and Quoc V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", CoRR, 2019.

[5] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision", 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015.