Machine Learning Project 2: Road Segmentation

Jonas Blanc (287508), Nolan Chappuis (287883), Antoine Masanet (288366)
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this report, we introduce our implementations of a classical convolutional network as well as multiple versions of U-Net to classify satellite road images. A significant data augmentation through images rotation was added to improve the model's accuracy. The best achieved result is a 0.896 F1 score and 0.945 accuracy on the AIcrowd test set.**

I. Introduction

The goal of this project is to train a model that can differentiate the road pixels from the background pixels of a satellite image. To achieve this goal, we are provided with a training set of a 100 Google Maps images of size 400×400 and their associated ground truths where a 1 indicates a road and a 0 a background. To evaluate the performance of our model, we are also provided 50 similar unlabelled images of size 608x608. The predictions are to be performed at a 16x16 patch size level where each patch must be labeled road or not. The AIcrowd evaluation system then outputs the F1-score and accuracy of the 38x38 pixels predictions against the ground truths. In section II, III, IV and V we will talk about the different models we trained to solve the problem. In section VI we will talk about the data augmentation we performed as well as how it improved the model. Finally, section VII contains all of our results and their respective explanations.

II. Baseline

For this project, our baseline will be the convolutional neural network provided with the project's data. This network takes as input a 16x16 pixels patch and output a single label indicating whether this patch is a road or not. To do so, the network consists of 2 consecutive convolution-max pooling layers to try and extract local features of the data. Those convolutions outputs 64 channels of 4x4 patches on which two fully connected layers are applied, thereby reducing the data from 1024 values to 1 value. A sigmoid function is finally applied to obtain the final prediction for the patch's label. This method works reasonably well with a f1-score of 0.644 obtained on AIcrowd using the default parameters provided. This model has the advantage of directly producing the desired AIcrowd's output as well as trying to optimize the AIcrowd's loss (i.e. correct classification of 16x16 patches). However, a major limitation of this method is its inability to use the global context of the image and only relying on the pixels of the 16x16 square to make its prediction. In the provided images, typical roads widths are between 20 and 40 pixels, hence, a patch located in the middle of a road will not provide any insight of the surrounding sharp road/background boundaries to the network. Furthermore, if an object such as a tree or a car obstructs the road, the patch located on that object will most likely classify it as non-road even thought it is located in the middle of the road. To mitigate those issues, we implemented a convolutional neural network on the whole image.

III. Classical CNN

To better benefit from the context of the patch in the images, we created a classical convolutional neural network that operates on the whole image (see fig 1).

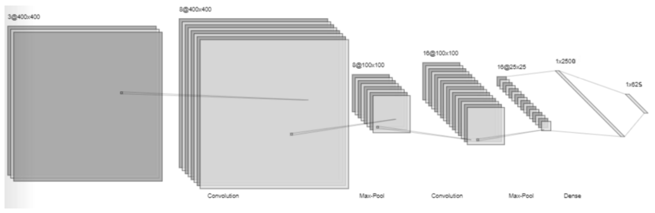

Figure 1. CNN architecture

This network combines two convolution-max pooling layers then finally two fully connected layers to predict a 25x25 pixel segmentation image at the end. To be able to train this network, we reduced the ground truth images to 25x25 as can be seen on figure 2. The high number of parameters quickly lead to overfitting and a bad generalization of the model.

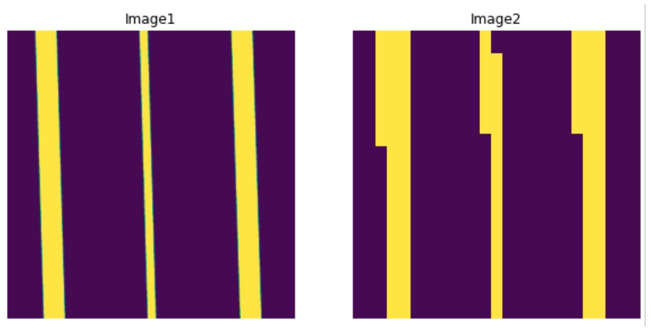

Figure 2. Patch ground truth

To reduce the number of parameters, we implemented a second CNN with the same architecture as the one mentioned above that instead of taking the full image takes 100x100 pixel images as an input. Splitting each 400x400 image into 16 100x100 images lead to a 16 times bigger training set. We estimated that a 100x100 context box was sufficient for the model and would help it reduce overfitting.

Despite those improvement, this custom CNN network is still overfitting thus performing rather poorly (best f1-score of 0.595) which lead us to use a U-Net, a well-known convolutional architecture specialized for image segmentation.

IV. U-Net

The U-Net[1] is an architecture developed in 2015 in the context of image segmentation for microscopic pictures of cells. This architecture has the main advantage of producing a pixel-wise classification of the original image. To do so, its architecture (fig 3) consists of two parts, the left part the *contracting path* and the right part the *expansive path*.The purpose of the *contracting path* is to capture context by compressing the image. This effectively losses the precise pixel localization which is regained using the copy and crop in the *expansive path*.

We can see that the original U-Net expects the input size to be of dimension 572x572 pixels. As a consequence, we extended the training set images from 400x400 to 572x572 by mirroring the edges. The neural network then outputs a segmentation map of dimension 388x388. To be able to train our model, we then made the choice to crop the 400x400 ground truth to match the networks output size.

The *AIcrowd* submission test set consists of 608x608 images. As a consequence, we first extend the input to 792x792 by mirroring the edges, then it is split with overlap into 4 patches of (572x572). We then run the network on those patches and merge the 4 output patches of (388x388) into one prediction of (608x608) with a weighted sum.

Finally, to predict a given pixel, we threshold the probability of being a road obtained after the sigmoid at 0.5. We then threshold the probability of a 16x16 patch being a road at 0.25 to take into account that roads occurs less often than background as well as to be consistent with the *AIcrowd system*.

A. Parameters

We use 3x3 convolution filters, a ReLU activation function at each hidden layer and a sigmoid activation function at the last layer. Furthermore, we used an Adam optimizer with default parameters to update the gradient. Finally, we implemented an approximation of the f1-score that computes the f1-score on the probabilities obtained after the sigmoid before the tresholding. As a consequence, it is differentiable with respect to the models weights and can be used as a loss function in our model. However, this loss lead to a high variance of the model as well as worse results than the BCE loss that we used thereafter.

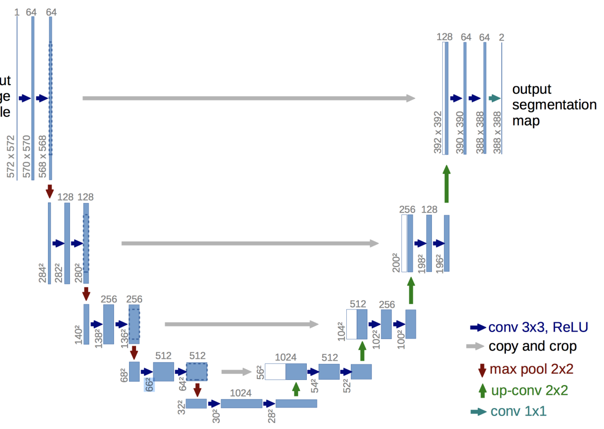

Figure 3. U-Net architecture

V. Adapted U-Net

In the previous section, in order to match the standard U-Net input size, we added fake data to the image using padding. This mirror padding creates an unrealistic transition between the image and the margin.

The U-Net architecture is based on convolutions which are invariant to the input size, thus we adapted the U-Net architecture to take as input the 400x400 training images and to output a prediction of the same size (using 1 pixel padding during convolutions). To predict the submission images, we directly feed the U-Net with the 608x608 images. The U-Net is able to predict different sizes images thanks to the invariant-size convolution property and to the fact that 400 and 608 are both divisible by 2^4 (each of the 4 max pooling operation divide each dimension of the input by 2). We have verified through testing that the model gives better predictions on 608x608 images than on four patches of 400x400 merged back on a 608x608 images after prediction, even thought the model was trained on 400x400 images. We explain it by the fact that the model has more context information when provided with the full image.

We implemented dropout to limit overfitting. By zeroing channels at random with a given probability, we force the model to predict images using less parameters. The first version implement dropout after(before) each convolution block during the compressing(decompressing) phase. The second version only makes use of dropout after the low dimension central block.

VI. Data augmentation

The original U-Net paper suggested elastic deformations (good in context of cell segmentation) but we did not because most road are straight and we would not want our model to train on roads that would not appear in the test set. Furthermore, the images from the training set and the test set seem to originate from the same camera, as a consequence, we did not add any artificial noise such as

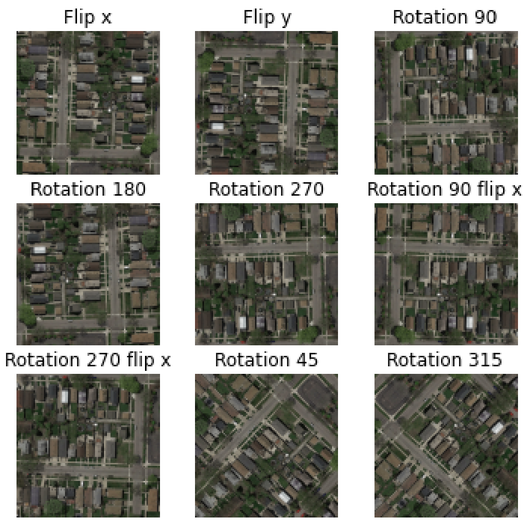

Figure 4. Transformation applied to the dataset

Gaussian noise to our training data. To augment our data, for each of our 100 training images, we added all possible combination of 90° rotation and symmetry. Looking at the predicted ground truths (see fig 5), we realised that the U-Net was performing better on the vertical and horizontal roads than on the diagonal ones. As a consequence, we further augmented our training set with 45° and 315° rotation of the images (see fig 4).

To avoid any correlation between the training and testing dataset we selected at random 10 images out of the 100 originals to be used for the testing set. After the augmentation this gives us 900 training and 100 testing images.

VII. U-Net Results

Using the same parameters, the adapted U-Net performs on average **2.6%** better than the original U-Net with respect to the f1-score (it could be explained by the absence of a large padding and the submission prediction taking the whole image). Consequently, the following results will be computed using our adapted U-Net. In the following comparison, for each settings we will keep the model giving the best test f1-score during a training session of 80 epochs.

To mitigate overfitting, we decided to add dropout layers in the model. As we can see from table I, applying dropout at every layers results in a small f1-score increase. Furthermore, only applying dropout at the middle layer gave us the best results with an increase of **1.8%** of the f1-score.

Dropout type	No dropout	Dropout 0.2	Middle dropout 0.2
Submission F1-score	0.871	0.875	0.887

Table I
Dropout improvements

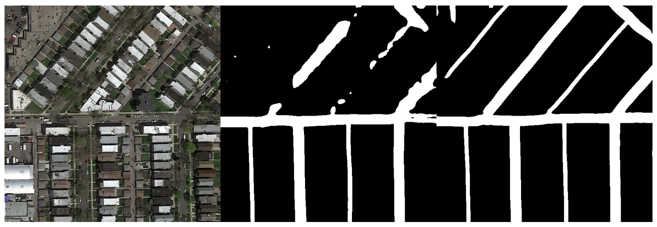

Figure 5. Original, prediction without 45° augmentations and prediction with 45° augmentations

Adding 45° rotations to the dataset drastically increased the accuracy of diagonal roads without impacting vertical or horizontal roads (see figure 5). Adding the rotations ended up increasing the f1-score by only **1%** due to the low proportion of diagonal roads in the test set.

Augmentation type	No Augmentation	90° augmentations	45° augmentations
Submission F1-score	0.851	0.878	0.889

Table II
Augmentation improvements

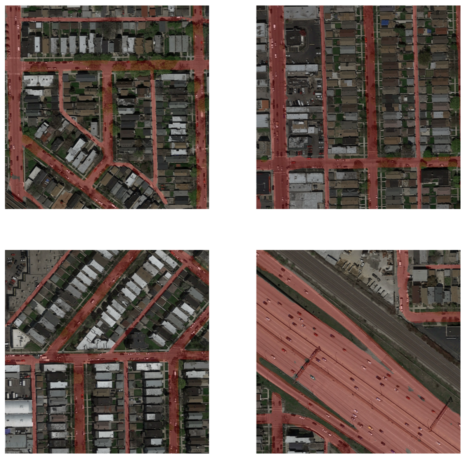

Figure 6. Selection of predicted images with overlay

In the end our best model was trained with a batch size of 4, a middle dropout of 0.2 and the fully augmented training set for 136 epochs for an accuracy of 0.945 and an f1-score of 0.896. Figure 6 illustrates some of the predictions made by this model on the test set.

VIII. Discussion and improvements

Our best results were obtained using a slightly modified state of the art model that is specialized for image segmentation. Data augmentation had a significant impact on our model's performance. To further improve our model, we could have identified training images where the model was performing most poorly and give them more importance during the training of the model (ie iterate more often over them). As we have seen with diagonals this method is effective. We could also have further augmented our dataset by adding channel/feature to the images such as extracted edges or color blobs. We were able to train the U-Net for a high number of epoch thanks to a Nvidia GeForce GTX 1080 graphics card. Before, the recent CPUs of our computers would take 3h to train the model over a single epoch, which lead to insufficiant training. To obtain our best model (136 epochs) it would have taken over 400h of training.

REFERENCES

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