Road Segmentation

Pauline Theimer-Lienhard, Yann Ennassih and Timo Achard
PYTyeee team
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—The development of satellite imaging technology has led to an increase in the number of applications using huge amounts of data for various purposes. Among these applications, road segmentation from satellite images plays a pivotal role in urban planning, transportation management, and environmental monitoring. In this study, we present our efforts toward achieving accurate and efficient road segmentation using deep learning techniques. We show that by using an UNet architecture and a resnet34 encoder with good hyperparameter tuning and some basic data augmentation, we can achieve an F1 score of 0.893.**

I. Introduction

This project aims to identify roads in a set of aerial images from Google Maps. The task boils down to a binary classification of image regions into background or road labels, which can be done by training convolutional neural network (CNN) based models to classify each pixel or patch of the input images. We will see in this report all the steps we took to best prepare the data for training our Neural Network (NN), how we chose our NN model, and how we tuned it to have the best predictions. In the end, we will also see what post-processing steps we tried in order to improve our prediction and we will discuss the different amelioration we could implement if we had to continue this project.

II. Data preprocessing

A. *Data augmentation*

Since we are working with images, we knew from the beginning that we would use some big Neural Networks (NN) to do our prediction and big NN are data hungry. Indeed, since there are a lot of weights to train, the NN needs a lot of data to adjust all those weights.

Yet, the original dataset contains only 100 aerial images which is far from what we needed. So we decided to increase the number of training images by doing some data augmentation. The library Imgaug [1] and Albumentations [2] were tested and both offered a large number of possible modifications. At first, we thought that a linearly changed dataset (with some mildly changed images up to some heavily changed images) would be the best option since it would be more robust [Fig 1]. But through cross-validation, we discovered that it was not the case and our best results came from an "easy" dataset where only one or two augmentations were applied per image and where there was no added noise nor big distortions. To be more precise our best result came from a dataset that was created in the following way:

- 100 images rotated by 90 degrees
- 100 images rotated by 45 degrees
- 100 images 30% cropped
- 100 images horizontally flipped
- 100 images vertically flipped
- 100 images 20% sheared

For rotation and shear augmentation, we decided to fill the empty corner through the symmetric mode.

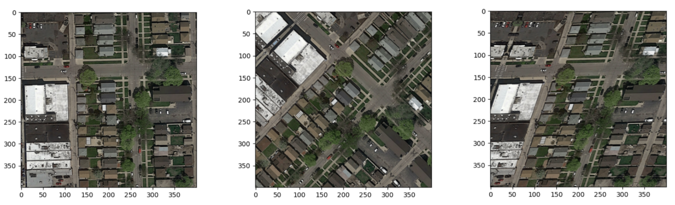
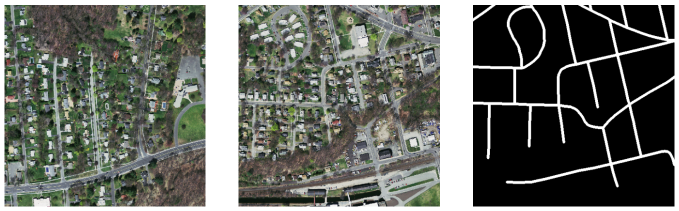

(a) Original (b) 45° Rotated (c) 20% Sheared

Fig. 1: Examples for data augmentation

We still had a problem with our data, we went from 100 images to 700, and this is still not enough to train big NN. Therefore, we decided to find a new dataset that we could add to the current one and the Massachusetts Roads dataset caught our eyes [3]. This dataset contains 1500x1500 aerial images of the Massachusetts and its cities. The roads were also highlighted and served as groundtruth exactly like our original dataset, we therefore thought that this new data was perfect to incorporate with our other images to better train our model. We took 20 pictures that seemed interesting with a lot of city parts and we each cropped them into 9 different sub-images to form 180 new images (with groundtruth) for our dataset. Unfortunately, this did not improve our F1 score it even reduced it a bit, the images are maybe too far away from the original dataset and they acted more like a burden for the NN learning than a real improvement. Also they were taken further away and the scale of the roads and the houses were not the same than in our original images, it surely didn't help our model to learn. Maybe a bigger NN model could resolve the issue by taking more advantage of this diversity in the data, but we couldn't try as we didn't have enough memory on the GPU.

III. Models and Methods

Now that we had a convenient dataset it was time to find a convenient NN capable of doing a good segmentation. After some research on the segmentation problem by looking at blog posts and papers, it was pretty clear that we would need a kind of UNet to tackle the task since everybody was using this kind of architecture to do similar predictions and even



(a) New Image 1 (b) New Image 2 (c) GT of image 2

Fig. 2: Images we added to our dataset, these images where taken from the Massachusetts Roads dataset [3]

more complicated projects than us. We didn't want to code a UNet ourselves using Pytorch since we wanted to be able to easily change the architecture to find the best model possible. Moreover, having to change the model in PyTorch every time would have been time-consuming and prone to errors. So we decided to look for a good library, and we found the following one: Segmentation Models Pytorch (SMP) [4]. This library has 5 different architectures: UNet, Linknet, FPN, PSPNet and PAN. PAN is not usable since it only takes images of size 256x256 and the hustle of making this work was probably not worth so we didn't try. For the rest, we tried all 4 at least once but we didn't have the time to really dig into them, the best results came from the UNet architecture but this should be retried with more tests to ensure that this is truly the best architecture. Our choice was also made by the fact that most examples we saw on other road segmentation dataset were made with a UNet and this paper [5] showed that UNet tends to be the most reliable architecture. UNet architecture, as shown in Fig 3., follows a U-shaped structure with a contracting path (encoder) and an expansive path (decoder), as well as a bottleneck layer in between.

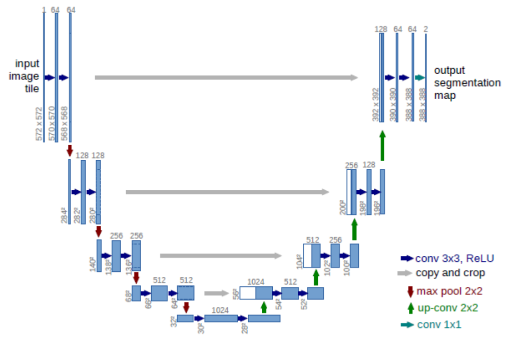


Fig. 3: UNet architecture

The 4 architectures from the SMP library can then be modulated by choosing the encoder architecture from a list of 46 different possibilities [6]. Again, due to a lack of time or computing power (see later for comments on that) we were not able to test everything. We mostly tried the resnet34 or resnet101 architecture since it is what we mostly saw in papers and on the web.

The library is also advantageous as all the models come with pre-trained weights.

IV. Hyperparameters

A. Cross validation

Like always we choose our hyperparameters by doing some cross validation. However, the fact that there are only 100 images is a real problem. Indeed, we only do simple data augmentation and if we choose at random 10% of the dataset to use as validation set, then the overlap between the dataset and the validation set is too big and our validation metrics are not accurate at all. In a way, we train on the validation set because some images in the validation set are just rotations of the training images.

To resolve this issue we picked at random 5 images and did the augmentations on them and that was our validation set. The problem with this method is that we lose 5 base images and that some images are very specific, we want them both in our training set as they are important for training on edge cases and on our testing set to test for those edge cases.

In the end, we decided to let randomness choose, but then we had to keep in mind that the accuracy scores we were getting weren't perfect.

Note that to do our AIcrowd submission we used no validation set to train on all the data possible to maximise our training.

B. Learning rate

We tried 3 different types of learning rates:

- Fixed learning rate
- Exponentially decreasing learning rate
- Cosine Annealing Warm Restarts (CAWR) learning rate

They gave pretty much the same results with minimal differences but in the end, the CAWR was slightly better. Maybe because the restart made that we were less likely to be stuck in a local minima.

C. Loss function

We also tried 3 different loss functions:

- Dice loss $s = 1 - \frac{2|X \cap Y|}{|X| + |Y|}$
  It measures the overlapping between the prediction and the ground truth. And we used it as it is less affected by the imbalance of a dataset than a regular cross entropy loss.
- Jacquard loss $1 - \frac{|X \cap Y|}{|X \cup Y|}$
  It is pretty much the same as a dice loss, just the way of calculating the total area of the predictions and the groundtruth changes.
- F1 loss
  It is a changed F1 score so that the gradient is computable. We used it since our primary metric was the F1 score we thought that by training on getting the lowest F1 score possible we would in the end get a bigger F1 score.

The cross validation showed that the Dice loss and the Jacquard loss had pretty much the same results as we anticipated. However, the F1 loss did significantly worse results and the idea was rapidly dropped.

In the end, we choose to go with the dice loss, used in most road segmentation pipelines.

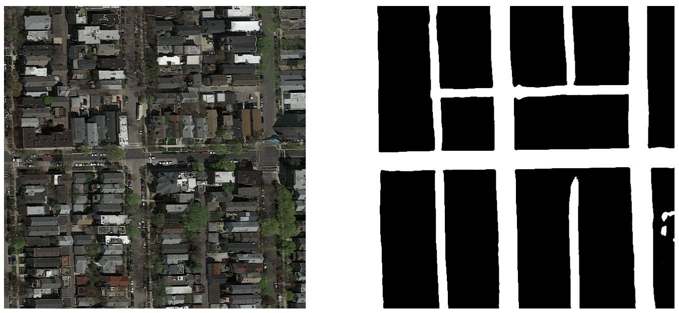
V. Post Processing

Our model gave us a pixel-wise prediction and it was easy (for us humans) to see that in some places it was not logical to have a prediction that looked like that [Fig.4]. Some roads suddenly got thinner and then larger again, or stopped for no reason.

A. Post processing on groundtruth

We thought that maybe some post-processing could resolve the issue and we tried for a bit with some image processing libraries like skimage we looked at different post-processing we could do but none of it was satisfactory. The main problem was that it didn't really resolve our road-shrinking problem and as it was not an "intelligent" algorithm it made sometimes things better and sometimes things worse. It was also hard to do testing and we would often overfit the testing data, so we decided to drop the idea even though it could have been interesting.

Techniques such as filling holes, connected components as well as removal of small isolated areas were performed but did not improve the results.



(a) Original image (b) Prediction

Fig. 4: We see in the prediction that sometimes the roads stops or it gets bigger/thinner before an intersection

B. Patching

As the models were trained on pixels, the images needed to be patched for submission. A simple threshold of 25% pixels was used but it became clear that it was not ideal. However, methods such as majority spatial voting did not improve. While it could slightly improve the results in some specific cases, this approach is not strong enough.

C. Ensembles

We also looked at ensembles, the idea is to train 5-10 different NN with the same or different architecture and the same or different loss function and then aggregate their predictions. We tried with the mean of all the predictions, it mildly ameliorated our result for a lot of work since we had to train at least 5 models. And so by lack of time, we abandoned the idea. It could also be interesting and it seemed to have more potential than the post-processing we talked about earlier.

D. Conditional random fields

Finally, we tried to do some conditional random fields to make our prediction better. It is a bit like a post-processing but where neighboring predictions were taken into consideration. However, we didn't have the time to understand the math behind and we didn't find any good clear library with CRF already implemented. Moreover, the fact that we used a library for our base model made it harder to add a CRF layer afterward, so the idea was dropped.

VI. Results and Discussion

A. Result

By applying everything that was said before, using a resnet34 encoder architecture, the pretrained weights given by the SMP library [4], the sigmoid function as the activation function, a batch size of 32 and by training for 60 epochs we managed to achieve an F1 score of 0.893 on AIcrowd (Submission ID : 247564).

TABLE I: Model Training Parameters

Model	Batch Size	# Epochs	Encoder	Score
Unet	32	60	resnet34	0.893
Unet++	2	30	resnet34	0.866
Unet	10	70	desnet-03	0.865
Unet	16	60	resnet101	0.861
Unet++	16	80	resnet34	0.858
DeepLabV3	16	30	resnet34	0.855
ResUnet	16	30	resnet34	0.844
Unet	10	100	resnet50	0.734

Table 1 shows the performance of some of the different image segmentation models we tested and their corresponding training configurations. We used different batching sizes and different numbers of training epochs but it shows that the UNet model is still the best one.

B. Discussion

This result is satisfactory but disappointing, we were able to do as good or better than most examples we saw on the internet in terms of F1 score, but it seems that our dataset is cleaner and easier than the Massachusetts roads dataset [3] for example, and those improved performances were to be expected. On the other hand, we are far away from the best submissions on AIcrowd, and we don't really understand where it comes from. We did a lot of hyperparameter tuning, tried a lot of different models, tried a lot of different dataset and did a lot of post-processing to try augmenting this score but still, in the end nothing worked. For all we did, our best model is a simple UNet with trivial data augmentation and a basic loss function, and with a bit of tunning, we obtained an

F1 score of 0.893.

If we could continue to work on this, our next steps would be to try an even bigger UNet with more complicated data augmentations and maybe add images from the Massachusetts roads dataset [3]. The idea is that a bigger NN would benefit from this bigger and changed dataset and not be affected like the UNet we used. Or we could dig into Conditional random fields to try to improve our predictions. This has some potential and could resolve the imprecision of the UNet without needing a bigger architecture or a lot more of data.

VII. Ethics

EPFL's provided digital ethics canvas [7] was used to evaluate the ethics and fairness on this project, considering two different contexts.

A. Education

The main aim of this project consists of a machine learning competition. The pipelines are kept private, so the solutions created only benefit the knowledge of the participants. The models only access the original dataset and its transformations.

The images hold no sensitive data according to law definitions [8]. Personal data can be contained in the images but can be hardly identified. The data is similar to a lot of massive datasets available online and does not particularly increase the risk of private information disclosure. Although the data is publicly available on AIcrowd without any protection walls, there is no transparency about the creation process.

In terms of environmental impact and energy consumption, the main factors are the training process and basic network manipulation to upload and download several thousands images and 100 MB models. The architectures were trained on Nvidia T4 15 GB ram GPUs on Google Colab clusters and each session lasted less than 2 hours. Exclusive human labor in this project concerns tons of image prediction visualization.

In short, given this context, the scope of identifiable persons impacted by this work, that is mainly participants and teaching staff, is limited and it is therefore difficult to derive from the models any harmful use a human could not directly apply on the original images.

B. Out of educational context

More risks and issues rise if the pipeline is taken out of the educational competition context. Pre-trained models can be used on other datasets, which will emphasize data and model biases. Indeed, if the models are used for example on a larger scale for common navigation task, training data can enforce a model to perform well on crowded urban areas while it would struggle on any other poorly represented regions. This creates discrimination based on the geographical situation. In this example, data errors are also prone to lead to false routing.

In this enlarged context, considerations to take into account are:

- Make the model public for total reproducibility and transparency
- Publish the data creation process
- Enrich and balance the data with other datasets.

These steps would help to fight against the mentioned discrimination risk, but were not implemented given the highly competitive aspect of the original task.

VIII. Summary

To achieve accurate road segmentation on satellite imagery we experimented a lot and found some surprising results, in general we found that bigger and more complicated was not better. First, we found that doing simple augmentations like rotations and flips gave better results than more complicated transformations or combinations of different transformations. We also found that a UNet architecture with a basic resnet34 encoder gave us the best results rather than a more complicated resnet101 architecture. Our attempts to do post-processing on the prediction made by our model were promising but all failed for different reasons. The ensemble methods were too time-consuming and didn't give great enough improvement to justify trying them more. The post-processing of the prediction was too random and as it was not "intelligent" we were too worried to just overfit the testing set and not improve the prediction. Finally, the conditional random field technique, even though it looked promising with the result it had in other contexts, was too complicated to implement and had to be drooped. We therefore archived our best score using a UNet architecture and a resnet34 encoder. The results are satisfactory but not perfect and maybe could be ameliorated with a combination of bigger neural networks and bigger dataset. We sadly couldn't test it as we were limited by the GPU's RAM on Google Colab. Or maybe the result could be perfected by looking more in detail at post processing steps and ways of more cleverly patching the result that we got.

References

[1] imgaug — imgaug 0.4.0 documentation. [Online]. Available: https://imgaug.readthedocs.io/en/latest/

[2] Albumentations. [Online]. Available: https://albumentations.ai/

[3] Massachusetts roads dataset. [Online]. Available: https://www.kaggle.com/datasets/balraj98/massachusetts-roads-dataset

[4] "Segmentation models pytorch," https://segmentation-models-pytorch.readthedocs.io/en/latest/.

[5] P. D. Paschalis Bizopoulosa, Nicholas Vretosa, "Comprehensive comparison of deep learning models for lung and covid-19 lesion segmentation in ct scans," 2023, https://arxiv.org/pdf/2009.06412.pdf.

[6] "Segmentation models pytorch," https://segmentation-modelspytorch.readthedocs.io/en/latest/encoders.

[7] Epfl digital ethic canvas. [Online]. Available: https://go.epfl.ch/canvas

[8] Loi fédérale sur la protection des données (lpd). [Online]. Available: https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/fga/2020/1998/fr/pdf-x/fedlex-data-admin-ch-eli-fga-2020-1998-fr-pdf-x.pdf