Machine Learning Project 2
Road Segmentation Challenge

Luca Bracone Omid Karimi Gianni Lodetti

***Abstract*—Given a satellite image of an urban landscape, we train a deep convolutional neural net to classify each pixel as being part of a road or not. We experiment with data augmentation, post-process gaussian filtering, different loss functions, and topological data analysis. Our best model, selected by five-fold cross-validation, obtains an F1-score of 86.7% on the AICrowd challenge.**

I. Introduction

In recent years, image analysis has been one of the most important subject of research in the domain of machine learning. There are many tasks in computer vision that can be examined and in this project we will focus on image segmentation, more specifically road segmentation. First the goal is to detect and differentiate road and background pixels from aerial images using various machine learning methods for classification, and then assess and discuss these methods with different metrics involving efficiency and effectiveness. This problem comes from the AIcrowd website and we are tasked to obtain the best accuracy and F1-score on the provided test dataset. For our implementations, we use PyTorch [1].

II. Data

For the problem at hand, we are given a set of satellite images acquired from Google Maps and a set of ground truth images of their respective road segmentation. In the ground truth image, the pixels are labeled as road in white and as background in black. We have 100 such images used for training our classifiers and we have an additional 50 aerial images (without the ground truth) for testing and submitting to the evaluation system.

A. Preliminary exploration

The images provided for training are squares with side length of 400 pixels, but the set of images for the final test have sides of length 608 pixels. We will have to pay attention to this later, as we will train our model with smaller images than the ones used for testing and this could lead in differences in performance.

(a) A provided satellite image **(b)** corresponding ground truth

Figure 1

B. Data augmentation

One hundred images for the training set, without even taking a validation set into account, might be a little bit poor to build a good classifier. There are many ways to update our images and augment the size of our dataset and improve our results. One idea is to add the rotation of the images with various angles. We used 3 different angles: 90° , 180° and 270° and added the rotated

images to our training set. In a similar way we also add a mirrored version of each image to complete our dataset. After further investigation of the test set, we noticed that some images have roads in diagonal that our model would not detect, giving us poor results for the submission. That is why we also augmented the data with a rotation of 45°. Note that the training is quite computationally heavy the more we add images. Also we will see later that we did not always used them all, as it does not necessarily improve the score, probably because of overfitting.

III. Model selection

We started by looking at the provided examples from the AIcrowd resources, and implemented a simple fully connected neural network that takes patches of size 16x16 binary images as input and outputs a label between 0 and 1 to classify this patch. We did not get very good results with this as our fully connected model was rudimentary, consisting of only a single layer. We assume better results could have been achieved with more layers however we decided to move directly from this basic model to a more advanced model with convolutions.

When building neural networks with images as input, it makes more sense to use a convolutional neural networks. With some research on the topic of road segmentation, we came to notice that U-Net usually provide good performances, hence we decided to implement the more widely known U-Net model.

IV. The U-Net Model

U-Net, first introduced in [2], is a neural network designed for image segmentation. The main idea is to supplement a usual contracting network by successive layers, where pooling operations are replaced by upsampling operators. Hence these layers increase the resolution of the output, and then a final convolutional layer can learn to assemble a precise output based on this information.

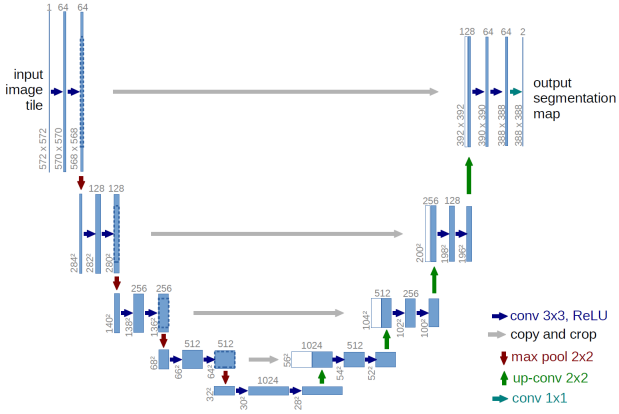

Figure 2: Architecture of original U-Net model

The U-Net contains convolutional layers that allows the model to come up with its own filters, and thus better understand the relationship between pixels by producing its own features, as opposed to our fully connected model. Finally a U-Net can process the entire image at a time and create an output image labelling every pixel, which is what we would like to achieve, as we felt this would achieve better results than only labelling patches.

We also considered building a more classic convolutional neural network that would use a cropped patch of the neighborhood of every pixel, and classify every pixel of an image based on its neighborhood, by running this neighborhood patch through the convolutional neural network, however we ultimately decided to stick with using the U-Net as our baseline model as represented in Figure 2.

V. Experimenting with persistence diagrams

We decided to make use of topological features of the image i.e. how many connected components and closed loops it has. We wanted to solve a problem in which our predictions would not keep roads connected at the intersections. For an explanation of this process see [3], we will not give an explanation because four pages is too short. We make use of the library described in [4] for

the Pytorch implementation of this computation. The new loss would be

$$L_{new} = L_{old} + \lambda L_{topo}$$

for some scalar λ. We tried two loss functions. The first involves matching points on the persistence diagram of the prediction and the ground truth, and computing the mean squared error. The second, inspired by [5], involves computing how many connected components and loops the ground truth has, and penalizing the prediction if it had more than that. The main drawback of this technique is that computing it is $O(n^3)$, and training takes too long. Furthermore, the lack of tooling meant that we were limited in what we were able to do, and the results were not great. Therefore we did not use it for the final submission. See the following for successful applications of this method: [6], [5], [7], [8], [9], [10]. The most notable of which is [6]. Finally, see the following for introductory textbooks on the matter: [11], [12].

VI. Post-Processing

We notice that some of the output images had spots in areas where they were no roads, to try to mitigate this we applied a Gaussian filter to attempt to smoothen the edges, remove spots and ultimately improve the prediction, however this was unsuccessful, as this slightly decreased our best F1 score by 0.05. After observing our post processed images we noticed that some of the spots were correctly predicted on top of roads but they shrunk aswell. We believe that this is ultimately why our results suffered due to Gaussian filtering.

VII. Results

To test our models locally we simply split the data into a validation and train set with 20% and 80% of the samples respectively. Here are the computer specifications used to obtain these time performances:

- CPU: Intel(R) Core(TM) i7-10700 @ 2.90GHz
- GPU: Nvidia RTX 3070, 8GiB

With this setup, training a U-Net model takes around 60 seconds for 80 images, using the GPU, otherwise it takes up to an hour. Depending on the data augmentation used, this can go up to 5 minutes of training. We first went with the baseline model as presented in Figure 2, with ReLU as activation, Adam as optimizer and Binary Cross-entropy (BCE) loss. We performed cross-validation for different numbers of epochs and learning rate and used the best values for future runs.

Figure 3: Cross-validation

Even though the neural network outputs a prediction for every pixel, we evaluate our models by cropping the prediction image into 16x16 patches and assigning a label a to every patch based on the number of 1's predicted within the cropped patch. Our goal is to maximize the F1 score over our prediction of the patches, since this is the way the submission website also evaluates our predicted images.

Table I: F1 score of baseline model

Model	**F1 score[%]**
SGD + Binary Cross-entropy	45.29
Adam + Binary Cross-entropy	85.36
Adam + Jaccard loss	85.11

We quickly noticed that Stochastic Gradient Descent (SGD) did not give good results and kept

the Adam optimizer for the rest of the models. For the loss, Binary cross-entropy and Jaccard loss are usually very close. Next we tried adding a dropout layer, with probability 0.2, right after the activation function and also tried leaky ReLU as activation.

Table II: F1 score of updated model

Model	**F1 score[%]**
Leaky + BCE	85.94
Leaky + Jaccard	85.23
Dropout + BCE	84.34
Dropout + Jaccard	84.37
Leaky + Dropout + BCE	83.98
Leaky + Dropout + Jaccard	83.76

Note that for these runs, we did not used all the augmented images, namely only the base images with their rotations of 90° , 180° and 45°. Using all the augmentations gave results a little bit worse. The best outcomes we got were with the U-Net using leaky ReLU as activation and binary cross-entropy loss and achieved a F1 score of 86.7% and an accuracy of 92.9% on the submission website.

VIII. Discussion

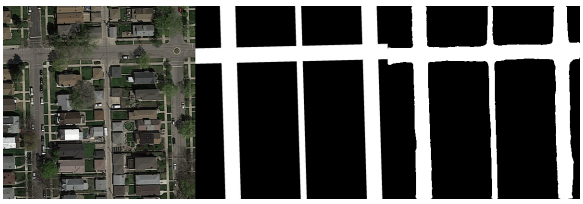

Figure 4: Our prediction on a residential area

Looking at the above figures we observe that our model performs very well for standard residential areas, however it has more difficulty for other types of areas. Noticeably we observe in Figure 5 that our model struggles sometimes to separate parking from roads, and does not always output straight lines as predictions.

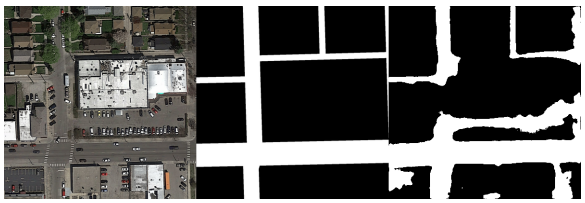

Figure 5: Our prediction on a industrial area with parking lot

We tried making our project more original by using topological data analysis and it was quite disappointing that it did not produce better results. The training times were too high and we tried to mitigate this problem by selecting small random chunks of the prediction and computing relative homology on it, but unfortunately it only confused the neural network and it never converged to a good local minimum. We think this may be due to the fact that the losses calculated on the small images were not representative of the image at large, and so the model was not learning the correct thing at all.

IX. Conclusion

Given a set of 100 satellite images and their ground truth road segmentation's, we augmented this data with rotations and mirroring operations. We then implemented a U-Net model and trained it with different optimizer, activation and loss function parameters, and retained the best results. The best results was attained using Binary Cross Entropy as optimizer, Leaky ReLU as the activation function and data augmentation.

We attempted using Gaussian filtering to post process the images however this did not improve our F1 score. It would be interesting to try other post processing ideas such as attempting to straighten our road predictions by correcting them perhaps by using and edge map and Hough transforms.

REFERENCES

[1] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[3] Gary Koplik. *Persistent Homology: A Non-Mathy Introduction with Examples*. 2019. URL: https://towardsdatascience.com/persistent-homology-with-examples-1974d4b9c3d0 (visited on 12/23/2021).

[4] Rickard Brüel-Gabrielsson et al. *A Topology Layer for Machine Learning*. 2020. arXiv: 1905.12200 [cs.LG].

[5] James R. Clough et al. *Explicit topological priors for deep-learning based image segmentation using persistent homology*. 2019. arXiv: 1901.10244 [cs.CV].

[6] Xiaoling Hu et al. *Topology-Preserving Deep Image Segmentation*. 2019. arXiv: 1906.05404 [cs.CV].

[7] James Clough et al. "A Topological Loss Function for Deep-Learning based Image Segmentation using Persistent Homology". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.3013679. URL: http://dx.doi.org/10.1109/TPAMI.2020.3013679.

[8] Christoph Hofer et al. *Deep Learning with Topological Signatures*. 2018. arXiv: 1707.04041 [cs.CV].

[9] Ephy R. Love et al. *Topological Deep Learning*. 2021. arXiv: 2101.05778 [cs.LG].

[10] Aıcha BenTaieb and Ghassan Hamarneh. "Topology aware fully convolutional networks for histology gland segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 460–468.

[11] Allen Hatcher. *Algebraic topology*. Cambridge: Cambridge Univ. Press, 2000. URL: https://cds.cern.ch/record/478079.

[12] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010, pp. I–XII, 1–241. ISBN: 978-0-8218-4925-5.