Machine Learning - Project 2

Amine Lamouchi, Charbel Chucri, Quinten Dewulf
Department of Computer Science, EPFL, Switzerland

I. Introduction

In computer vision, image segmentation consists of dividing an image into distinct regions or segments, and giving a label to each region. This task has many practical applications, including medical imaging, video surveillance and computer vision for autonomous vehicles.

In this project, we work with satellite imagery, and our goal is to distinguish between roads and background. This can be used for improving navigation tools for instance.

II. Augmenting the Data

We initially have access to 100 images of 400x400 pixels along with their ground-truth. However, this dataset was not large enough to yield good results when training an Encoder-Decoder or a U-Net network. To increase the size of the training dataset, we applied various identical transformations to the images and their corresponding masks. This included:

- Flipping and Cropping: Through the convenient Torchvision method $TenCrop$, we cropped the images into four corners and a central crop, as well as horizontally flipped versions of these crops. All the crops are of size 256x256 pixels so that they are actually different from each other.
  Our motivation to use this function was to make the model invariant to "zooming-in" since most streets in the initial training set were of similar size.
- Rotating: We rotated each image by all multiples of 45° (8 in total if we include 0°). For angles 45°, 135°, 225°, 315°, we kept a central crop of 256x256 pixels to only keep parts of the initial image. For the other angles, we just resized to 256x256 pixels for consistency. Note that we use the initial images as they correspond to 0° rotations.
  Our motivation was to make our model invariant to road orientation.

Overall, these transformations allowed us to multiply the size of our initial dataset by 18, resulting in a dataset of 1800 images (1000 coming from flipping and cropping and 800 coming from rotations). Our model without data augmentation would give us a validation F1 score that is at most 0.81. Using data augmentation, the improvement was clear as we reached a max F1 score of 0.92 on our validation data using U-Net.

III. Baseline Convolutional Network

The provided baseline network won't directly produce segmented images, instead, it takes a small patch of an image and tells whether or not it contains a road. The final segmented image is made by compositing the labels of the various patches that make up the total image. Using the provided 100 training images and reserving 20 for validation we achieved a maximum F1 score of 0.642 on our validation data. This however improved to 0.705 with the augmented dataset.

IV. Encoder Only

When searching for popular image-segmentation architectures we quickly ran into convolutional encoder-decoder networks. [1] The informal explanation of these networks often mentions that the encoder part of the network learns to construct a lower-dimensional representation of the high-dimensional input image that still captures all essential information. This encoding is then fed to the decoder who learned to construct a segmented image with the same dimensions as the input image.[2] We can think about the wanted 16x16 pixels segmentation image as an encoding of a larger 256x256 input satellite picture that still holds all essential information. If we could train a convolutional encoder to make 16x16 encodings that hold all of the road data then there is no more need for a decoder.

input→output	opertion	parameters
(256,256,3)→(256,256,16)	Conv2d, BatchNorm, ReLu	kernel=5
(256,256,16)→(128,128,16)	MaxPooling2D	
(128,128,16)→(128,128,32)	Conv2d, BatchNorm, ReLu	kernel=5
(128,128,32)→(64,64,32)	MaxPooling2D	
(64,64,32)→(64,64,64)	Conv2d, BatchNorm, ReLu	kernel=5
(64,64,64)→(32,32,64)	MaxPooling2D	
(32,32,64)→(32,32,128)	Conv2d, BatchNorm, ReLu	kernel=5
(32,32,128)→(16,16,128)	MaxPooling2D	
(16,16,128)→(16,16,2)	Conv2d, Bias, ReLu	kernel=5
(16,16,2)→(256,2)	Reshape	

Table I
Architecture of the encoder network

The architecture of the proposed network was inspired by the SegNet encoder. [3] There's no Softmax layer at the end because the used loss function, categorical cross entropy, takes logits as input and the final segmentation image at inference time is computed using argmax over the tuples representing the 256 pixels.

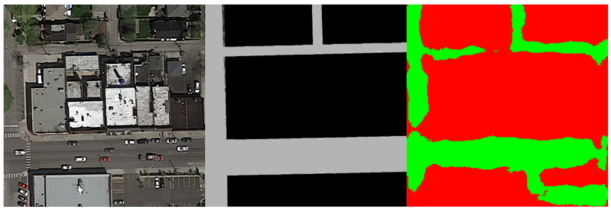

Figure 1. A training image, its ground truth and its output of the simple encoder-decoder network

The best results were achieved by training on 1450 images for 137 epochs, resulting in an F1 score of 0.782 on our validation set of 350 different satellite pictures.

V. Simple Encoder-Decoder

We've implemented a simple convolutional encoder-decoder network that is inspired by SegNet but lacks its pool indices transfer mechanism. The Input image is of size (256,256,3), the encoder creates an encoding of size (16,16,128) and the decoder makes a segmentation image of size (256,256,2) through convolutions and upsampling layers. There are no fully connected layers in the network. The highest F1 score obtained on the validation data was 0.716.

VI. Custom Ragged Loss

A. Motivation

Upon inspection of the generated output of our custom simple encoder-decoder network, we noticed that (apart from its difficulties with parking lots) it consistently made segmentation images where the streets had very irregular, ragged, edges. It's often the case that streets have "big bites" taken out of them and their borders are not straight. In an attempt to reduce this problem by using a smarter loss function instead of a more complex network, we came up with ragged loss.

B. New Loss Function

Ragged loss is a part of the total loss that is high when streets have ragged edges and low when street edges are regular.

$$total\,loss = CategoricalCrossEntropy + \gamma \times ragged\,loss$$

The goal of introducing ragged loss is to minimise the number of non-street pixels that have two, three or four street pixel neighbours in the segmented image. This is because when a correctly trained network is convinced that three or four neighbours of a certain pixel are street pixels then it should also be certain that that pixel itself is a street pixel. Non-street pixels that have two street pixels are very rare in ground truth segmentations, occurring only at intersections of roads and at edges of roads with a slope. However, they occur very often when roads are ragged, this is why we'll also try to minimise their occurrence.

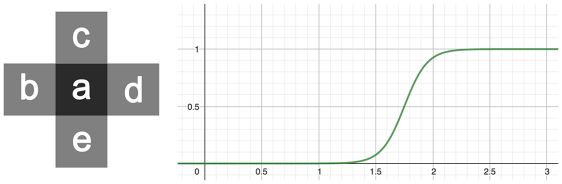

Figure 2. Pixel a with its four neighbours and the function s.

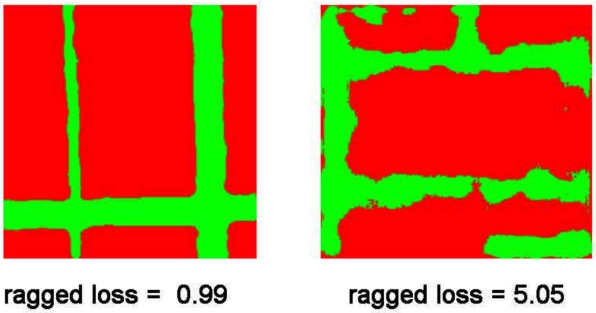

Figure 3. Ragged loss values of two image segmentations. Because these are produced by the network and are not ground truths the ragged_pixel_value of any pixel is never truly 1 or 0.

The ragged loss of a segmented image is defined as the sum of the ragged_pixel_value of all its pixels. Each pixel is represented by a tuple (α, β) with α being the probability that this pixel is a non-street pixel and β the probability that it is a street pixel. The ragged_pixel_value of pixel a with neighbours b, c, d and e is defined as:

$$ragged\_pixel\_value(a) =$$

$$s(a[0] \times [(1 - b[0]) + (1 - c[0]) + (1 - d[0]) + (1 - e[0])])$$

With the function s being:

$$s(x) = sigmoid(10 \times (x - 1.75))$$

So when $a[0] \approx 0$ or $a[0] \approx 1$ and zero or one neighbours ≈ 0 and the others ≈ 1 (max one street neighbour) then $ragged\_pixel\_value(a) \approx 0$. However, if $a[0] \approx 1$ and two or more neighbours ≈ 0 then a is a non-street pixel with two or more street pixel neighbours and $ragged\_pixel\_value(a) \approx 1$.

C. Experiment

The custom loss function was implemented using Tensorflow and we've trained the simple encoder-decoder network with multiple values for the ragged loss hyperparameter γ.

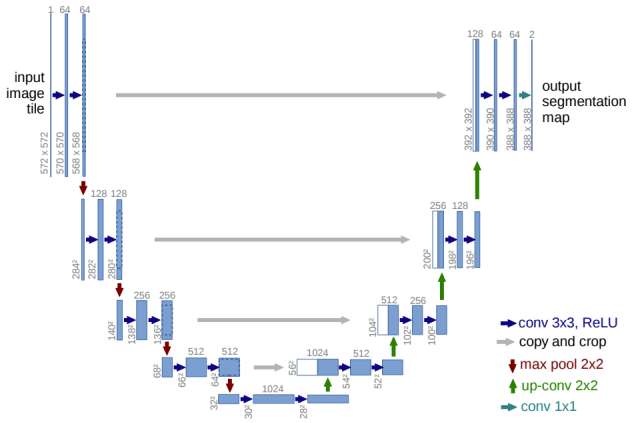

Figure 4. U-Net Architecture [4]

D. Results

When the influence of ragged loss on the total loss is high (e.g. $\gamma = 1$ or 0.5) the network converged to a point where it would only make segmentation images without any roads. It learned that it was better to not have any roads in order to fully minimise ragged loss. One way to reduce ragged loss is indeed to have less street pixels, since if there are no street pixel neighbours, no pixel can have more than one of them. For very small values of γ (e.g. 1e-8) the influence of ragged loss was not noticible. For less extreme values of γ, like 0.001 or 0.0001, the network tended to produce segmentation images that had overall less street pixels than identical networks without ragged loss. These segmentations still had ragged edges, but the smaller amount of street pixels did resolve in a smaller ragged loss value.

E. Future Work

Applying ragged loss from the very beginning of the training process did not deliver an increase in the regularity of road edges in produced image segmentations. It would be interesting to see what the result are of applying ragged loss after the network has already been trained for a couple of epochs with the traditional loss function. This way the network could first "learn to create ragged roads" and perhaps would then clean up the edges under the influence of ragged loss.

VII. U-Net

A. Architecture Overview

U-Net was originally developed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox for biomedical image segmentation [4]. As seen in Figure 4, it is based on the idea of a "U-shaped" network architecture, which consists of a contracting path (left part), an expansive path (right part), and skip connections between the two paths. The contracting path is a series of convolutional and pooling layers that progressively downsample the input image, learning detailed features in the process. The expansive path is a series of upsampling and convolutional layers that upsample the features learned in the contracting path and combine them with features from the skip connections. The skip connections allow the network to preserve pixel-level information from the contracting path, while also using global context information from the expansive path. Note that to make the network more stable while training, we added a batch normalization layer in between the convolution layer and the ReLU activation function (dark blue arrow).

Overall, the U-Net architecture is able to learn detailed, pixel-level features while also preserving global context information, making it well-suited for image segmentation tasks and thus a good candidate for our project.

B. Training

We implemented U-Net using PyTorch. We trained our model for 50 epochs, while saving the weights of the model that achieved the highest validation F1 score (or the highest F1 score if there is no validation data). The final model was trained on the full training data, without any splitting.

We used the the Sørensen–Dice coefficient (also called Dice coefficient) for our loss function. It is a measure of similarity between two sets that is commonly used in the field of image segmentation to evaluate the performance of a model, particularly in the context of binary classification tasks. The Sørensen–Dice coefficient is defined as:

$$DiceCoefficient = \frac{2 \times |A \cap B|}{|A| + |B|}$$

This coefficient has a range of 0 to 1, with a higher value indicating a greater degree of similarity between the two sets.

We used Adam [5] for stochastic optimization with a learning rate of 2×10^4. We picked Adam because it performs quite well with problems that have large data and parameters.

Finally, to regularize the weights of the neural network, we used L_2-regularization. In particular, we added a penalty of $10^{-4} \times \sum_i w_i^2$ to our loss function.

C. Results

U-Net was our best performing model. When trained using the original 100 images, it achieved a maximum F1 score of 0.811 on our validation data. Moreover, by using the augmented data, it did extremely well, achieving a maximum F1 score of 0.927.

VIII. Results

We tested multiple models on our data. For each model, we optimized the hyper-parameters when applicable and then we quantified how good a model is using cross-validation locally. The results of the different models are summarized in Table II.

Without Augmented Data	
Method	Validation F1 Score
Baseline	0.642
Encoder Only	0.693
Encoder-Decoder	0.672
U-Net	0.811
With Augmented Data	
Method	Validation F1 Score
Baseline	0.705
Encoder Only	0.782
Encoder-Decoder	0.716
U-Net	0.927

Table II
F1 SCORES OF DIFFERENT MODELS TESTED

IX. SUMMARY

In summary, the U-Net architecture was the most effective for identifying roads in satellite images. Moreover, augmenting the initial data further improved the performance of our model. Finally, by applying various techniques to improve the learning process of our model, we were able to achieve a score of 0.904 on AIcrowd.

REFERENCES

[1] wiki.tum.de, "Image semantic segmentation," *https://wiki.tum.de/display/lfdv/Image+Semantic+Segmentation*.

[2] v7labs.com, "autoencoders-guide," *https://www.v7labs.com/blog/autoencoders-guide*.

[3] A. K. Vijay Badrinarayanan and R. Cipolla, "Segnet," *https://paperswithcode.com/method/segnet*.

[4] 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.

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