Road Segmentation Challenge

Mohamed Saad Eddine El Moutaouakil, Hafsa Aoutir, Emna Maghrebi
CS-433: Machine Learning, EPFL, Switzerland

***Abstract*—In recent years, Convolutional Neural Networks have seen a lot of development, which has enabled the achievement of highly performing segmentation models. In this paper, we will present our work on the different approaches and models we implemented to efficiently tackle the Road Segmentation challenge on AICrowd. All models are compared based on a given test set, and their accuracies are tabulated.We have obtained a 0.920 F1-Score using a DeepLabV3-ResNet101 architecture and data augmentation techniques implemented specifically for data segmentation problems.**

I. Introduction

Data segmentation is highly sought after thanks to its capability to automate and facilitate delineation of regions of interest, in our case city roads. Our purpose for this challenge is to identify the roads apart from the other objects present in the images, such as houses and trees. We were given a dataset of satellite images taken from Google Maps , consisting of 100 training images of size 400x400 and 50 test images of size 608x608 . We were also provided with ground-truth images for the training set consisting of masks, where each pixel is either white, indicating a ”road” pixel or black otherwise.

Our approach consists of pixel-wise predictions, followed by an empirical threshold for classifying each pixel as ”Road” or ”Background”. In Section II , we will explain the different methods we implemented for data augmentation. In Section III, we present the different models we chose to train our data on, the loss function to assess the models' error, the optimizers used and the metric for evaluating their performances. Section IV will present the results we achieved and discuss them.

II. Data Augmentation

Data Augmentation is a technique used for increasing the size of data used for training a model, when the dataset is very small. Such transformations allow the model to generalize better, and thus perform well when predictions should be made for the test set. For this step, standard options from the PyTorch library : torchvision were used, with Functional transforms allowing a fine-grained control of the transformation pipeline. The transformations we opted for are described below.

A. Rotations

Rotations are one of the most common data augmentation techniques. They allow the model to overcome the variance introduced by the angle at which objects are present within the picture. In our case, rotations will help the model identify not only parallel and perpendicular roads, but also diagonal ones. We used the following angles for rotations : [-90, -60, -45, -15, 15, 45, 60, 90].

B. Flips

We decided to use the technique of image flipping to complement the rotations. We flipped both vertically and horizontally the images to cover the edge cases where a symmetry pattern had been present in some original roads' structure but was not entirely captured in one picture. Roundabouts are typical examples illustrating this effect. If a roundabout happens to be in the edge of the fixed image size frame, at the phase where maps where transformed to pictures, it will be symmetrically distributed over four corners of four images (One corner per image). Information will thus be lost if the four images are not all included in the training set.

C. Crop and resize

We used cropping and resizing to enable the model learning more features about specific regions. We decided to crop our images starting from position (40,40) and of size 240x240. To keep the input images constant we resized our cropped images to the original size of 400x400.

D. Other transformations

We implemented different other transformations for the sake of increasing its performance. We applied normalization, mirroring, transposing and translation.

After a thorough comparison of the results, we decided to only use rotations and flips since they gave us the best accuracy.

III. Models and Methods

A. Baseline

Any statistical comparison can only become significant after defining a baseline. For this experiment, we chose the model given with the documentation of the project as our baseline. It is relying on a very simple convolutional neural

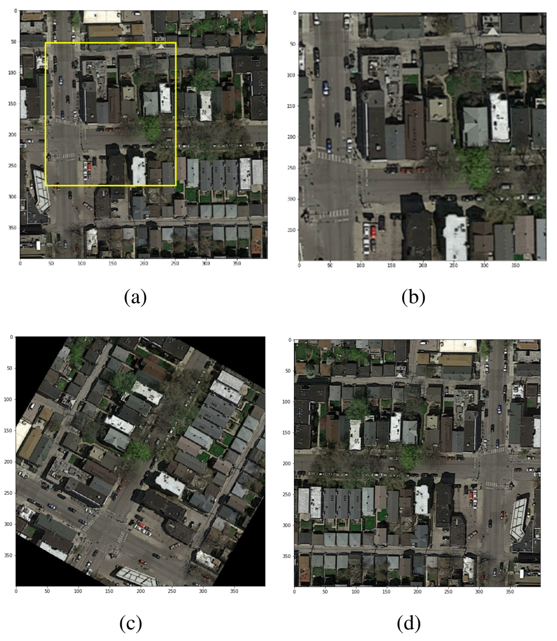


Figure 1. (a) original image (b) original image after crop and resize (c) original image after rotation with 60° (d) original image after a flip

network. The latter takes as input a square patch of the training image of size 16, that flows into two convolution layers before reaching two fully connected layers performing the binary classification task based on the outputted probabilities. The model uses simple momentum optimizer, a softmax cross entropy with logits as a loss function and introduces an exponential decay for the learning rate.

B. *U-Net*

Segmentation tasks find many applications in various fields. One of the relevant examples is the medical industry characterized by the scarcity of the data bounded by the physical limits of data acquisition on one hand, and the need for high performance metrics on the other hand. Typically, models can only be trained on small datasets and are expected to yield high precision. In this context, U-Net was introduced.

U-Net, is a convolutional neural network that concatenates the result of each step of its hierarchy. The latter consists of two paths: Contracting path and expansive path. As Introduced by Ronneberger et al. in their paper [6], the contracting path consists of a series of convolutions, separated by ReLu and max pooling operations. Each step thus doubles the number of feature channels. The result is cropped due to the border pixel loss introduced by every convolution and stored to be concatenated with its corresponding upsampling step in the expansive path.

Once the base is reached, the expansive path starts. Each step consists of an up-convolution halving the number of features, a concatenation with the corresponding down-

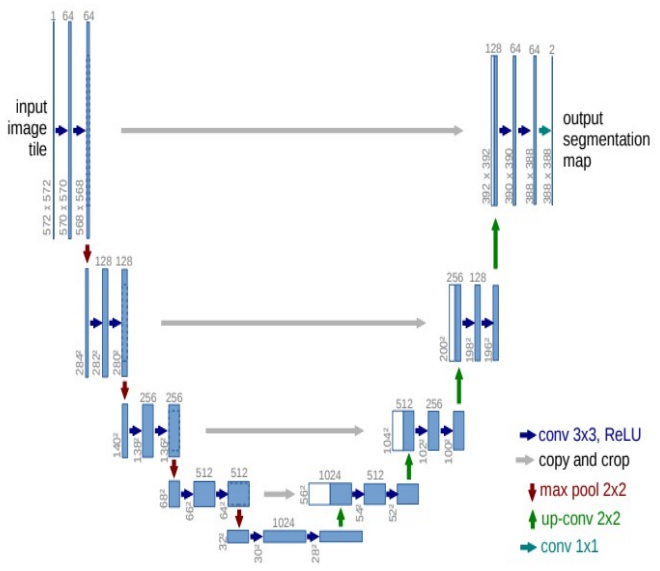


Figure 2. U-Net architecture from Ronneberger et al. [6]

convolution from the contracting path and two convolutions each followed by ReLu. Figure 2 summarizes the architecture in a simplistic way.

The usage of U-Net for road segmentation can be justified by its similar characteristics to biomedical images. Mainly, the dataset used is very small and the structure of the roads requires contextual classification, since many surrounding pixels may encode features relevant to the surrounded pixel itself.

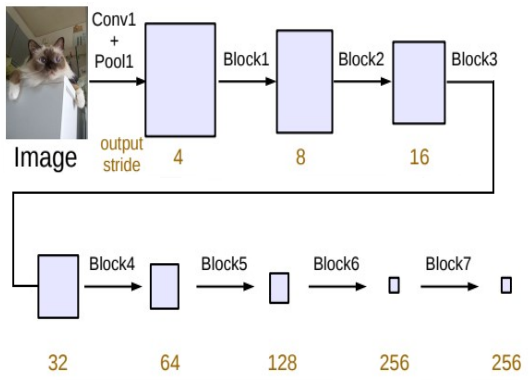
While the original U-Net model can be used to perform road segmentation, we fine-tuned its structure to better match our requirements. In fact, multiple depth levels have been tried and compared. We kept the same blocks explained above while varying the number of steps to simulate different levels of deepness for the same model. The original model used taken from Milesial's Github repository [7], modified as described above and trained on our dataset.

C. *DeepLab V3*

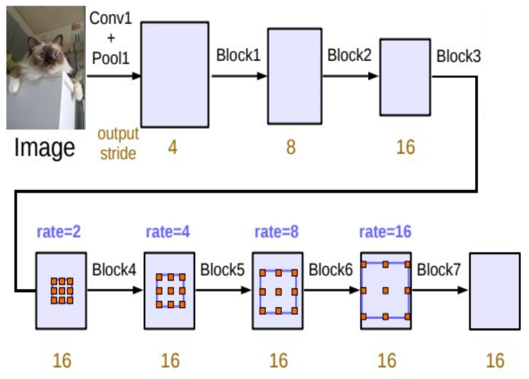
Deep convolutional neural networks progressively learn abstract feature representations by iteratively applying pooling and convolution striding. While the image transformation's invariance insured by the latter technique is desirable for many aspects of the segmentation task, it reduces the spatial resolution of the resulting feature map, thus hindering the areas where spatial information is precious, like dense prediction tasks.

To tackle this challenge, Chen et al [8] proposed DeepLabV3, a state-of-art model relying on atrous convolution. It allows the explicit adjustment of filter's field of view and the control of resolution density of feature responses.

We believe this model is appropriate to our task for two reasons. Firstly, segmenting roads is a dense prediction



(a)



(b)

Figure 3. Chen's et al.[8] illustration of Atrous Convolution process (a) Going deeper without Atrous Convolution (b) Going deeper with Atrous Convolution.

task since the prediction happens at the pixel level. As a result, we can leverage our model abilities through atrous convolution as explained above. Secondly, not only DeepLab by design allows various depths, mainly using Resnet-50 or Resnet-101 backbones, but is also compatible with the usage of other neural networks, such as MobileNets which are far more efficient with respect to size and speed [9].

As we are performance driven, we discarded the use of the pre-trained model, and preferred to train the models on our training set. The implementations used, are the ones provided by Pytorch on TorchVision.models.segmentation

D. LRASPP MobileNet V3 Large

Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP) model uses the same principles of Atrous convolution described in the DeepLab model. The difference that this model introduces is mainly in the design of the segmentation head. Howard et al [10] employed large pooling kernel with a large stride and only one convolution in the latter. The model is thus computationally more efficient than other ASPP models.

We decided to try LR-ASPP with a MobileNetV3-Large backbone instead of MobileNetV3-Small, because both models are designed for mobile use, with the former being reserved for high performance devices, which in term of computational power meet largely our equipments. We trained the model, provided by Pytorch on TorchVision.models.segmentation, on our own dataset.

E. Loss functions and Optimizers

The choice of loss functions and optimizers can make significant difference in the model's performance.

Our segmentation road task encompasses a binary classification task, where each pixel belongs to the "Road" class or "Background" class. Consequently, we used the binary cross entropy with logits loss as a loss function.

Concerning optimizers, we tried both Stochastic Gradient Descent (SGD) and ADAM optimizers. The latter consistently performed better than the former, as will be shown in the results section.

F. Performance metrics

To evaluate the performance of our model, we opted for F1-score instead of accuracy as a performance metric.

The reason behind this choice resides in the fact that F1-score captures more details about the classification results, through the precision and recall and thus is more robust. Moreover, the distribution of the binary classes – Background and Road – in the dataset is unbalanced and largely skewed towards backgrounds.

G. Hyperparameters tuning

Machine learning models rely on parameters that are expected to be empirically set by the user. The process of determining the optimal hyperparameters is called hyperparameters tuning. In order to have meaningful comparisons between our models, we only compare them after setting the optimal hyperparameters.

In addition to the number of epochs, the learning rate, we tuned the prediction threshold variable THRESHOLD_ROAD_LABEL_PREDICTION which sets the boundary between the road and the background in the output of the sigmoid function at the end of the inference running's process.

We used two ways to hypertune our parameters, Grid-Search and Random-Search. The later is more time efficient but less performant than the former.We proceeded through cross validation after splitting the training dataset into effective training (70%) and validation set (30%). However for the sake of maximising the accuracy for AICrowd, we used the full training set when we compared the models.

IV. Results and Discussion

Comparison of Results

Model	**Nb of epochs**	**learning rate**	**batch size**	**Prediction threshold**	**Optimizer**	**F1-Score AICrowd**
Baseline	50	Exp Decay	16	0.25	SGD	0.653
U-Net Depth 3	120	0.00025	6	0.35	ADAM	0.896
U-Net Depth 4	120	0.00025	6	0.4	ADAM	0.904
U-Net Depth 5	120	0.00025	6	0.4	ADAM	0.908
DeepLab V3 Resnet-101	75	0.0002	6	0.4	ADAM	0.920
DeeplabV3 MobileNet V3-Large	75	0.00025	6	0.45	ADAM	0.902
LR-ASPP MobileNet V3-Large	20	0.0003	3	0.4	ADAM	0.909

The best model is DeepLabV3 Resnet-101, with a F1-score of 0.920, one of the highest scores on AICrowd leaderboard. Except the baseline, all the models performed very well for the task of road segmentation.

The baseline performs poorly (0.653) because on one hand the architecture is too simple to learn appropriately, and on the other hand, by design the model labels patches of the image of size 16x16 instead of pixel-wise resulting in false predictions.

In U-Net, we see a light improvement of F1-score when the model deepness increases. This behaviour was expected as deepness allow the model to intercept more relevant features.

The DeepLabV3 Resnet-101 performs excellently thanks to the combination of the Atrous convolution scheme combined with the deepness of Resnet-101 backbone. The drawback of this model is its runtime latency, which is significantly higher in both training and prediction compared to the other models. The DeepLabV3 MobileNetV3-Large or the LR-ASPP MobileNetV3-Large seem to be good compromises between performance and latency, if the latter is a requirement.

V. Summary

After augmenting the provided dataset, tuning hyper-parameters, we made sure to use a well-suited loss function with respect to the characteristics of the given classification problem. Additionally, we chose the appropriate optimizer ADAM. As a result, using DeepLabV3 Resnet-101 we achieved a F1-score of 0.920 on the test set.

Traditionally, semantic segmentation of aerial images in deep learning relies on the use of a large data set. The emergence of powerful, yet data-hungry deep learning methods aggravates the situation. Therefore, in future work it may be useful to determine the effect of the dataset's size on the models' performances.

References

[1] M. Sörsäter, "Active learning for road segmentation using convolutional neural networks," https://liu.diva-portal.org/smash/get/diva2:1259079/FULLTEXT01.pdf.

[2] S. Wang, S.-Y. Hu, E. Cheah, X. Wang, J. Wang, L. Chen, M. Baikpour, A. Ozturk, Q. Li, S.-H. Chou, C. D. Lehman, V. Kumar, and A. Samir, "U-net using stacked dilated convolutions for medical image segmentation," 2020.

[3] Transforming and augmenting images. [Online]. Available: https://pytorch.org/vision/master/transforms.html#functional-transforms

[4] Road surface semantic segmentation. [Online]. Available: https://towardsdatascience.com/road-surface-semantic-segmentation-4d65b045245

[5] Image processing and data augmentation techniques for computer vision. [Online]. Available: https://towardsdatascience.com/image-processing-techniques-for-computer-vision-11f92f511e21

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[7] Milesial. Pytorch implementation of the u-net for image semantic segmentation with high quality images. [Online]. Available: https://github.com/milesial/Pytorch-UNet

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[10] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.