Machine Learning Project II - Road Segmentation

Adrian Villarroel, Andres Montero, Erick Maraz
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—The goal of the project is to develop a machine learning model capable of road segmentation on satellite images. Several techniques using convolutional neural networks (CNN), including two modified versions of U-net [1], are tested. The training data-set consists of 100 (400, 400) images and their ground truth masks of the same shape. The model is tested by classifying (16, 16) patches in 50 test images of shape (608, 608), i.e. every patch is classified resulting in whether 0 or 1. Data augmentation was performed and different activation functions were tested. Dropout, Batch Normalization, and Regularization were applied to the convolutional layers as well. In the end, the model that obtained the best score was a simple CNN, which gave us a score of 0.883 in the set of test images.**

I. Introduction

In the last years, there has been a rise in data in almost all fields and also the rise of computing power, thanks to the use of GPU. In particular, the need to process images to extract valuable information has motivated the research and development of Machine Learning (ML) models for a broad range of different tasks, giving extraordinary results with the use of CNN. One of the main tasks is image segmentation which has the purpose of simplifying the representation of an image into something that is more meaningful and easier to analyze. In practice, it is the process of assigning a label to each pixel in an image, such that pixels with the same label share certain characteristics [2].

The aim of this project is to develop a model capable of performing segmentation on satellite images in order to classify patches belonging to roads. This report explains the process of how we tackle this problem, from the data exploration and data augmentation to the model selection, using a basic CNN as a baseline and then using more complex architectures, detailing and comparing different techniques such as different activation functions, transfer learning, dropout, and batch normalization to avoid over-fitting. We present the importance of each decision comparing the improvement of the models and the end-results with the architecture that presented the best score in the test set.

II. Data Exploration

The training data-set consist of 100 satellite images, their ground-truth masks (figure 1), both of shape (400, 400) pixels. A test-set of 50 satellite images of size (608, 608) is provided to test our model in AIcrowd. The objective is to classify whether a patch of 16 by 16 pixels is part of a road or not, i.e. a single (608, 608) image is cut into 1444 (16,16) patches.

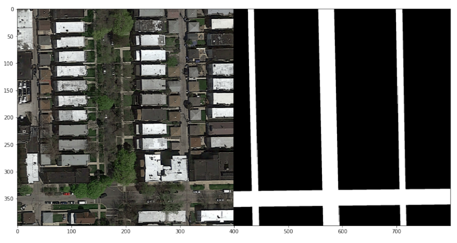

Figure 1. Satellite image and its ground truth mask

A. Data Augmentation

In order to increase the number of images in the training set. We proposed to rotate the images θ random degrees n times, i.e. we get n + 1 images including the original one. This results in *extended* images depending on the algorithm used to fill the unknown space, i.e. *reflect, wrap, mirror*. It was proposed to get the biggest square image possible from this rotated images. Consequently, a new center and length are calculated for the new images. They are resized to their original size and finally, they are mirrored vertically and horizontally randomly.

Figure 2 was created from figure 1. It can be noticed that this image was not only rotated but also mirrored. Moreover, it was scaled and some pixels were changed due to the final resize where *cubic-interpolation* was used. The augmented training data set has 900 images in total, meaning that $n = 8$ rotations were performed (the performance did not improve with greater number of rotations).

The output images of the current model are not rotated or mirrored under a uniform distribution. This means that the quality of the augmented data can be improved.

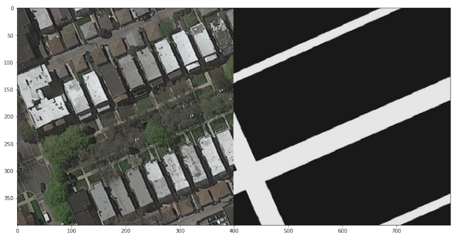

Figure 2. Image and its ground truth mask rotated

B. Batch generation

A batch consisting of windows-sized images (64, 64) is fed to the models (this size was chosen due to hardware, the greater the size of the patch, the fewer patches used for the batch). These images are patches cut of the (400, 400) training-set images. New patches cut off from different images and different space regions are computed every new batch. The label is however calculated only from the centered (16, 16) patch of the ground truth image under the following condition:
`if mean(ground-truth-patch) > 0.25, then road = 1, else road = 0`
This is how we aimed to provide the model extra information about its neighbors while still focusing on the original patch. Recall, that this label will be the same for the whole windows-sized patch.

In order to fully use the training data, we added padding of 24 to the original image. This helped our model to be robust when dealing with borders where the pixels can sometimes be only padding.

III. Model Architectures

A. Baseline

The baseline architecture is the CNN provided in the repository of the course. See figure 3.

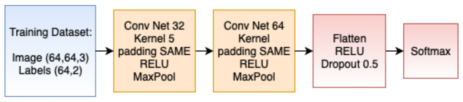

Figure 3. Architecture of the baseline

B. U-Net

While doing research we found that for image segmentation U-Net is among the favorites architectures for this task, it performed extremely well in medical images and is also a good approach for road segmentation. U-NET is a fully convolutional network, meaning that the output size is the same as the input size, to achieve this first a downsampling step (encoder) is applied, i.e. convolutional layers with max-pooling, then an up-sampling step (decoder) also with convolutional layers, no max-pooling and a concatenation with the previous layer of downsampling step, for a detail explanation please refer to the paper of U-net [1].

Given the previous research, we decided to implement this architecture in the project, but we also added a convolutional layer and a dense layer of two neurons with sigmoid activation at the end of the architecture as our goal is to classify each patch as either road or not. This is described in figure 4.

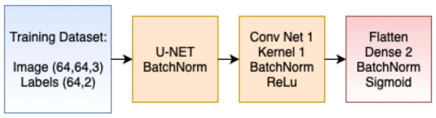

Figure 4. U-NET Architecture.

C. ResNet-U-Net

An interesting approach that we found uses a pre-trained network as the encoder part of a U-Net architecture. This technique is a particular application of transfer learning which is the use of neural network layers trained for a specific task in another similar one [3]. Our implementation has a residual network as the encoder. We are using specifically ResNet50 trained on the ImageNet data-set which contains around 14 million images of objects [4]. Hence, ResNet50 must have learned to recognize edges, colors, shapes and other basic features of images that are useful for our task.

In the following subsections, we give more details about our implementation. First, we give a brief introduction to residual networks and second we give a concise description of the architecture.

1) Introduction to Residual Networks (ResNets): One of the main problems of very deep architectures is the degradation of training accuracy, not caused by over-fitting, but by the complexity of the neural network (determined by the number of layers) [5]. This problem has been verified by thorough experimentation concluding that the training and test error increase if we add more layers to a suitable model for a given dataset [5].

To solve this problem the residual block is proposed. As can be seen in figure 5, this block has the stacked non-linear layers fitting a residual mapping $\mathcal{F}(x) = \mathcal{H}(x) - x$, where $\mathcal{H}(x)$ is the original mapping. It is expected that the residual mapping is easier to optimize [5]. A ResNet is a configuration that uses these blocks.

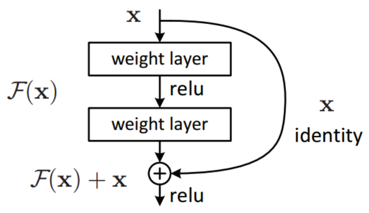
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 5. Residual block [5]

2) Architecture: This configuration (see figure 6) uses a ResNet50, pre-trained on the ImageNet dataset, as the encoder part of the U-Net. It consists of residual blocks made of convolutional, batch normalization, max pooling, and ReLU layers. The decoder is formed by blocks of convolutional and convolutional transpose layers. Then, the output of each block of the encoder is concatenated to the output of the corresponding block in the decoder. Finally, the output of the decoder is connected to a block of layers that flatten and perform the classification of patches using a sigmoid activation.

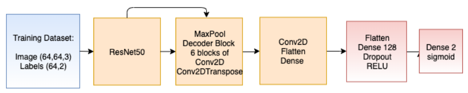

Figure 6. ResNet-UNet Architecture

D. *Proposed CNN architecture*

This approach is a modification of the architecture shown in figure 3. Each layer increases the number of filters by 2 in relation to the previous one. The number of layers depends on the size of the patches fed to the CNN. For an input of shape (64, 64), 5 convolutional layers performed the best.

The first model that was tested was a CNN of 3 convolutional layers. The model trained correctly and there was no over-fitting. However, the training and validation loss where greater than 0.15 and the F-1 score achieved was around 0.85.

Therefore, a fourth layer model was proposed and over-fitting occurred when the validation loss achieved was less than 0.1. Nevertheless, it was not evident since the validation and training losses were close. It was during the submission to AIcrowd, when there was no validation set, that this behavior was noticed.

Finally, with 5 layers, there was no over-fitting and the model was robust enough when dealing with rotated and/or scaled versions of the validation set images. It was concluded that over-fitting would not happen if the callbacks were correctly stated and the full training data set was used.

Dropout and Maxpool were applied to the layers. Moreover, the activation function was changed to Leaky ReLU. All these changes caused the model to train faster but apparently without any change in the performance.

The final architecture is displayed in figure 7.

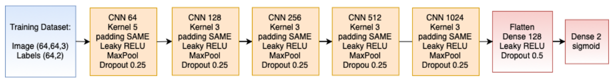

Figure 7. Proposed CNN architecture

IV. Implementations, Performance

In order to shorten the time needed to train the models, we used either colab [6] or our laptops with the CUDA environment enabled (review readme). The framework we used to build the architecture is TensorFlow 2.0 [7], in particular, KERAS library [8]. It is also important to notice that 3 callbacks were used during the training of our model, EarlyStop, to avoid overt-fitting of the model, stops the training if there was no improvement for 10 epochs in our case. Learning Rate decrease, if the model did not improve for 5 epochs the learning rate was reduced to ensure that a local minimum was found and also ModelCheckpoint, to save the model each time there was an improvement.

A. *U-Net*

The base-line of this model gave an F-1 score of 79% on aicrowd, but in the training set we achieved an F-1 score of 95%, clearly, the model was over-fitting. For this reason, we decided to divide the training data in two, 80% for training and 20% for validating. This was to have an idea at every epoch if the model was over-fitting or not. We also explore the possibilities of adding dropout layers, batch normalization layers, different activation functions and regularization at the dense layer. However, the best hyper-parameters were: ReLu as activation function, no dropout, batch normalization and small regularization, 1e-5. This agrees with the paper [9] that states that if the network is using batch normalization, dropout is not necessary. The result of the evolution of the model with the best hyper-parameters can be observed in figure 8. This model trained

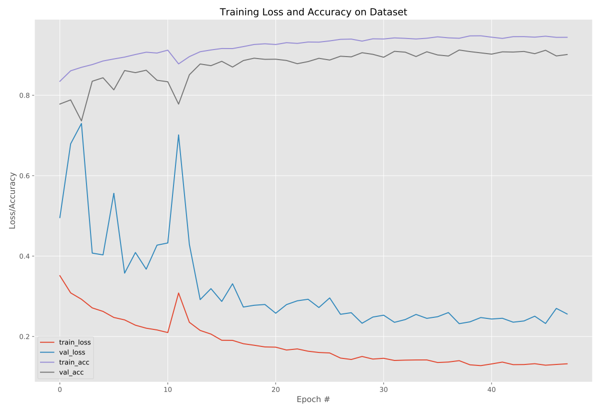

Figure 8. U-NET Training Accuracy-Loss.

for around 50 epochs with the EarlyStop parameter to avoid over-fitting, each epoch took around 160 seconds. With this configuration the score obtained in aicrowd is 0.835 F-1.

B. *ResNet-U-Net*

Our initial approach gave us a test F-1 score of 0.837 using an L2 regularization of 3e-6 in the final layers, but it was a very strong regularization which led to under-fitting because the training loss was higher than the validation. Then we tried with dropout and batch normalization layers but results didn't improve. Finally, we changed the l2 regularization of the final layers to 1e-6 achieving the best test F-1 score of 0.851. Nevertheless, the validation accuracy was still higher than the training accuracy as can be seen in figure 9.

C. *CNN*

The baseline of this architecture, without data augmentation, yield an F-1 score of 84.0. We then implemented data augmentation and it gave a result of 88.3 with the best

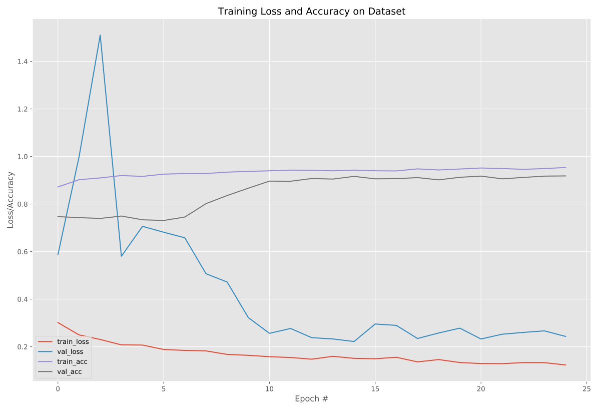

Figure 9. ResNet UNet Training Accuracy-Loss.

parameters. It is important to mention that we tried different values for dropout, regularization values on the dense layers and activation functions. In the end, the best hyper-parameters were: LeakyRelu, dropout 0.25, regularization of 1e-6. The results of the training of the CNN with this configuration can be seen in figure 10. Using Colab [6] this

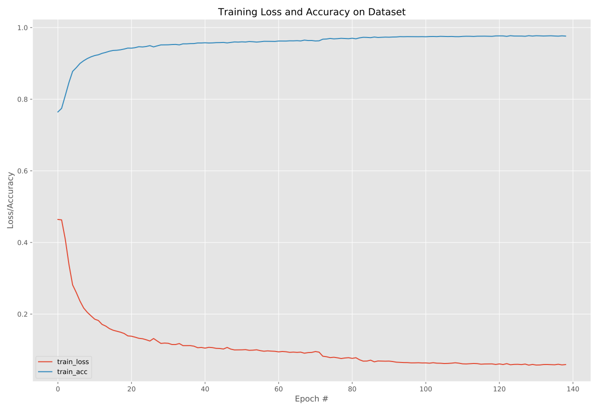

Figure 10. CNN Training Accuracy-Loss.

model trains around 4 hours and 110 epochs with 100 steps per epoch before the callback stops it. The batch size is 1200 (64, 64) and every new epoch, a new patch is randomly taken from the shuffled training images.

V. Results

The overall results, comparing the evolution of the models with data augmentation are summarized on the table I. We can observe that the best performance was achieved with the CNN architecture with data augmentation. The use of dropout, LeakyRelu as activation function and L2 regularization at the dense layers made the model robust

Table I
MODEL SCORES

Model	**F-1**	**Accuracy**
Baseline	65.6	76.9
ResNet-U-Net	84.0	91.5
ResNet-U-Net + Data Aug	84.7	92.3
U-Net	80.2	89.7
U-Net + Data Aug	83.5	91.6
ConvNet	84.0	92.0
ConvNet + Data Aug	**88.3**	**93.8**

enough to avoid over-fitting, so that a validation set was not needed. The ID of the submission in aicrowd is: 31018. As an example of a prediction mask that this model gives, we generated figure 11. As expected, this mask has a resolution of 16 by 16 and correctly classifies most of the road section. However, some roads are cut and there is even a miss-classified road point in the middle of a house.

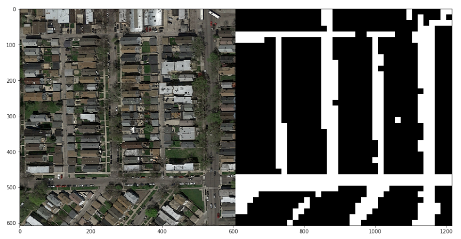

Figure 11. Test image and mask generated by the proposed CNN architecture

VI. Conclusion

We presented three different architectures that performed well for road segmentation. We explore the use of transfer learning, CNN, dropout, regularization techniques, and different activation functions. Although the best model presented an F-1 score of 88.3, we can observe that for example in image 11 the model is not able to classify correctly areas with shade or that are too dense. We suggest that to try to overcome these issues a different patch size for classification should be explored instead of the (16, 16). We also used partial images of (64, 64, 3), another approach could be to explore different sizes to feed the network and of course try other architectures and methods to train ML models (Q-learning, GAN, etc).

Although over-fitting seemed very unlikely due to the nature of our batch generator, it appeared during the last epochs of the previous version of the best model. Our interpretation is that over-fitting may occur suddenly and that a validation set is required to measure the robustness of the model.

References

[1] O. Ronneberger, P. Fishcer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[2] Wikipedia. Image segmentation. [Online]. Available: https://en.wikipedia.org/wiki/Image_segmentation

[3] S. J. Pan and Q. Y. Fellow, "A survey on transer learning," 2009.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," 2009.

[5] K. He, X. Zhang, S. Ren, J. Sun, and M. Research, "Deep residual learning for image recognition," 2015.

[6] "Google colab." [Online]. Available: https://colab.research.google.com/

[7] "tensorflow." [Online]. Available: https://www.tensorflow.org/

[8] "keras." [Online]. Available: https://keras.io/

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.