Road Classification on Satellite Pictures using Machine Learning

Alleon Antoine
Computational Science & Engineering
Eail: antoine.alleon@epfl.ch

Cleres David
Computational Science & Engineering
Email: david.cleres@epfl.ch

Lesimple Nicolas
Computational Science & Engineering
Email: nicolas.lesimple@epfl.ch

***Abstract*—This second Machine Learning project aimed to create a program that was able to detect segments, corresponding to roads, on high-resolution satellite images. Among various techniques, *Convolutional Neural Networks* (CNN) seemed to solve this challenge of separating the areas that represented roads from the rest with the best accuracy. CNN proved to be powerful visual models yielding hierarchies of features that outperforms all the predictions of other tested techniques. Data Augmentation during the preprocessing phase and taking into account that roads are continuous elements in a landscape during the post-processing enabled My Little Poly Revenge to get acceptable results with the trained network.**

I. Introduction

Semantic segmentation presents a major step towards understanding different objects and their arrangements observed in a scene. This powerful tool has a wide range of applications in various trending but also topical fields, like autonomously driving cars. Deep learning and Machine Learning methods raised the public attention many times in the past years through moonshot projects like IBM's Watson or the AlphaGo project from the IT-giant Google. However, researchers worldwide celebrated huge breakthroughs in handwritten digit and speech recognition. More, categorizing whole images and detecting objects in images also seen growing interest in semantic pixelwise labeling problems. Semantic segmentation of images consists in labeling each part of an image by a trained algorithm, is this specific project the aim was to find segments on satellite images that corresponded to roads. The sliding window approach was used to predict is a given "window" showed a road fragment or something else. This report starts with a description of the dataset. Then, the basic principles of CNN will be presented and CNN will be compared to other techniques such as logistic regressions. Section IV will be dedicated to present the results of the different neural net architectures and regressions that were used to perform a reliable prediction. Finally, discussion and auto-critic on the results will be made in this final part of the report.

II. Dataset

The dataset consisted of a training set which was divided into two parts. The first part was made of 100 high-resolution RGB satellite images acquired from Google Maps. The other part of the dataset consisted of the 100 corresponding gray level images (*Ground-truth images (GT)*) where each pixel was labeled in a binary way, meaning either as road or background element.

Fig. 1: High Resolution Satellite Image

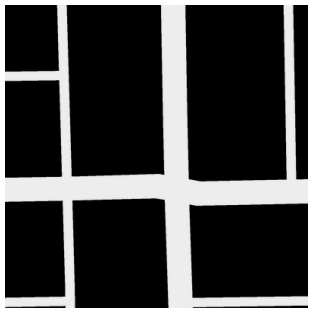

Fig. 2: Corresponding Gray level image to Fig.1

The models used in this report were trained on these training pairs of satellite and GT images. This training enabled the models to recognize and label the road parts on an off-set provided test images. The task was to classify blocks of 16 x 16 pixels, considering that the label associated with each block corresponds to 1 if the average value of the ground truth pixels in that block is greater than a threshold (0.25), 0 otherwise. While discussing with people who did not take the class it seemed that finding roads on a picture was a straightforward task. However, for a machine detecting roads, roads were not easy tasks since some of the roads were covered by trees, diagonal, had different colors,... For these reasons, the model had to take into account not only the pixel itself but also the neighborhood pixels. This was even more useful, since not all concrete areas were labeled as roads (e.g. parking lots and walkways) even if on a single pixel they look like roads. All these things that the human eye could easily distinguish could potentially confuse the model.

III. Methods

A. Preprocessing by data augmentation:

By a more depth analysis of the train and test images provided, it was observable that the images in the test set contained more diagonal roads than in the training dataset. To tackle this issue, the team implemented a data augmentation script which was able to perform rotations on the training and GT images. It was a great issue to be able to rotate the images without creating distortions and to keep the 400x400 size of

Fig. 3: rotated image

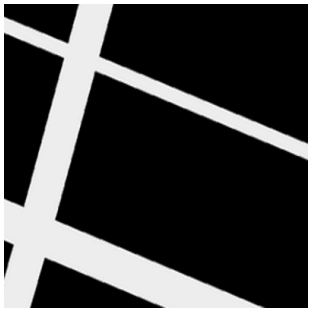

Fig. 4: Rotated labels

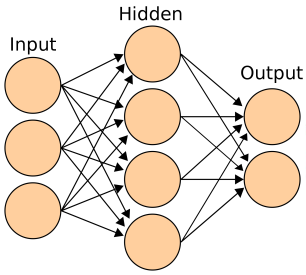

Fig. 5: Neural Network

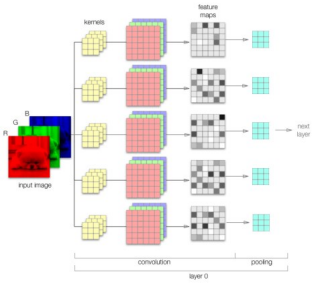

Fig. 6: Convolution neural network diagram

the training images. In order to rotate the images properly as intended, the *Augmentor* library was used. However, this library was able to rotate randomly single images but not to rotate simultaneously two images (satellite and GT picture). For this reason, the source code of the library was modified to enable this kind of processing. These increased the number of training images that were available for the different model and allows to include in the training the diagonalized images. This process should allow the model to be trained better with diagonal roads than it would be without data augmentation. Furthermore, increasing the amount of training data was a good fact because the MNIST dataset was so small and made it much more complicated for the model to memorize the data. Finally, the data was also augmented many other ways, like rotation, translation, flipping, and others.

B. Penalized Logistic Regression

The penalized or regularized logistic regression was already implemented from the previous project. Recall the regularized logistic regression using the same cost function as the logistic regression, plus a regularizing term, penalizing high norms for the solution vector:

$$\sum_{n=1}^{N} ln[p(y_n|x_n^T, w)] + \frac{\lambda}{2}||w||^2 \tag{1}$$

C. Convolutional Neural Networks (CNN)

In the world where visual recognition meets machine learning, CNN are one of the most powerful tools we have access to. CNN is a type of learning algorithm using a score function based on neurons (nodes) and synapses (connections) where weights are assigned to synapses as the model trains until the cost function has been minimized. Typically, the input and output are defined as one layer, and the layers in between are called hidden layers and allow for a more or less complex model (Fig. 5). Because simple neural networks do not scale well images, the use of convolutional layers before allows a better optimization of the cost function. 2d convolutions allow to identify certain patterns in the image, by superimposing convolutions, it is possible to identify more and more complex patterns in the image. Because convolutions increase the dimension of the data, we use pooling after each convolution to reduce the dimension (Fig. 6).

The chosen model is presented in fig. 14. The model uses four 2d convolutions each with a max pooling on a 2x2 kernel, using LeakyReLU as activation function with an $\alpha = 0.01$. Following these convolutions, the matrix was flattened into a vector for the input layer of the neural network. Finally, the data went through two hidden layers of size 128 and 64 to end in the two-dimensional output layer, assigning a binary value which was either 0 or 1. The loss function was the *softmax* categorical cross-entropy, which has been minimized using the Adam optimizer (a variant of SGD). Various hyper parameters for the different layers have been used to train the model, the effect of varying the chosen hyper-parameters will be discussed in the results section.

D. Regularization & Windows size

Although the dataset augmentation allowed to reduce overfitting, the use of Dropout layers (50% in the Dense layers and 25% in the convolution layers) showed more impact on behalf of prediction accuracy. They were added to each max-pooling layer (with p = 0.25), and also after the fully connected layer (with p = 0.5). Furthermore, L2 regularization has been used for the weights (and not biases) of the fully connected and output layers, with $\lambda = 10^{-6}$. The window size has been empirically chosen in order to take into account a context that is large enough, considering that large windows are computationally expensive. Therefore, a size of 50 x 50 has proved to be an acceptable trade-off.

E. Hardware Details & Specifications

The model was trained on an Apple MacBook Pro with 16GB of RAM and with an NVIDIA GeForce GTX 960 GPU (with 2 GB of VRAM), using 32-bit floating point precision and *TensorFlow* backend.

F. Morphological Post-Processing

After testing the model on the test dataset, post-processing could improve the prediction accuracy. As a matter of fact, the predictions from the trained model were obviously not perfect as shown in Fig. 7. In the output images, could observe single, disconnected pieces of road pixels. Given the prior knowledge about the structure of road networks, it would be safe to assume that the pieces of roads in Figure 7 represented false positives while the gaps are false negatives.

Luckily, image processing techniques exist to tackle this issue. Several transformations were tested, first on the input data, like for instance increasing the contrast of the training by using histogram equalization (which was also applied to the test images). However, this did not increase the accuracy of the predictions. Finally, mathematical morphological operations applied on the output of model was employed and lead to a better prediction accuracy. Mathematical morphology contributes a wide range of operators to image processing, all based on a few simple mathematical concepts from set theory. The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation. In the frame of this project, erosion, dilation, opening and closing were tested on the output images. These operations were also combined and superimposed but at the end, the most accurate prediction was performed using an opening by a 3x3 cross structuring element. Nevertheless, using this technique only increased the prediction accuracy by a decimal.

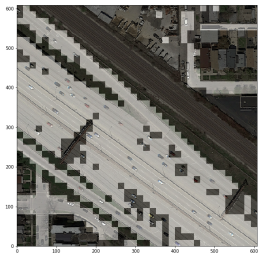

Fig. 7: Original prediction made by the trained model

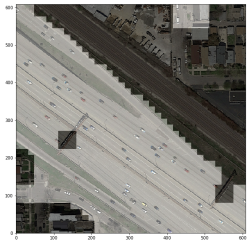

Fig. 8: Post-Processing made by opening with a 3x3 cross

G. Neighborhood based Post-Processing

To help increase the prediction score, another type of post-processing can be done. To remedy against road outliers or even holes in roads, another post-processing has been implemented based on the number of neighbors of one particular type in a 4 patches neighborhood. For example, if a road patch is only surrounded of non-road patches, it is updated to a non-road patch or, if a non-road patch is surrounded by at least three road patches it is changed to a road patch. Figure 9 showed the patches before the post processing while figure 10 showed them after post-processing.

IV. Results

In the first steps of the project, the penalized logistic regression of project 1 was used in order to assess the quality of this model for the kind of task imposed by the new project. Furthermore, using this already existing model helped to assess if the code used to make a submission on Kaggle matched the constraints. The accuracy of the penalized LR was of 0.79140 when the 100 initial images were taken as an input without any data augmentation or post-processing. Interestingly, as shown in figure 5, when the input data

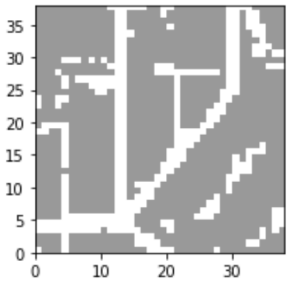

Fig. 9: prediction on test image 4 before neighborhood post-processing

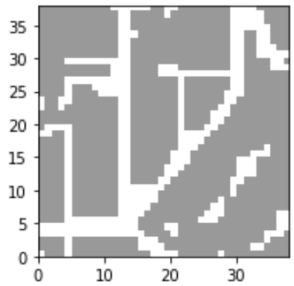

Fig. 10: prediction on test image 4 after neighborhood post-processing

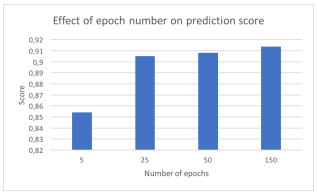

Fig. 11: Effect of epoch number on prediction score

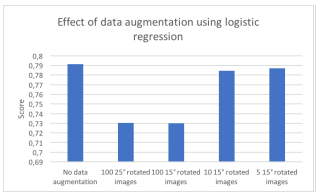

Fig. 12: Effect of data augmentation on prediction score

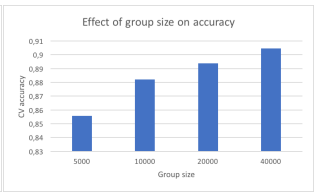

Fig. 13: Effect of group size on Cross Validation accuracy

was augmented by adding 5, 10, 25 or 100 rotated images to the input, the prediction accuracy dropped (figure 12).

In a second step of the project, the group decided to focus on CNN to perform a qualified prediction on behalf of the roads in the satellite images. Varying different parameters in the model (epoch, epochs per step, dropouts, batch-size, ...) was helpful to understand how to improve the model. In figure 11, one could see that as the number of epochs increased, the prediction score increased as well. There was, however, a trade-off as the training time also increases a lot as the number of epoch increased, while the score saturated. Figure 13 showed that increasing the group size (number of samples taken for each epoch), the cross-validation accuracy increased. As the model propagated more samples, the model learned better, which was similar to increasing the number of epochs. However, the computation time increased a lot with the number of samples. The best score on Kaggle was obtained using the model shown in figure 14. The model was fitted using 300 samples per epoch and for a total of 150 epochs. The prediction accuracy was 0.92030.

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 50, 50, 64)	4864
leaky_re_lu_9 (LeakyReLU)	(None, 50, 50, 64)	0
max_pooling2d_5 (MaxPooling2	(None, 25, 25, 64)	0
conv2d_6 (Conv2D)	(None, 25, 25, 128)	73856
leaky_re_lu_10 (LeakyReLU)	(None, 25, 25, 128)	0
max_pooling2d_6 (MaxPooling2	(None, 13, 13, 128)	0
conv2d_7 (Conv2D)	(None, 13, 13, 256)	295168
leaky_re_lu_11 (LeakyReLU)	(None, 13, 13, 256)	0
max_pooling2d_7 (MaxPooling2	(None, 7, 7, 256)	0
dropout_3 (Dropout)	(None, 7, 7, 256)	0
conv2d_8 (Conv2D)	(None, 7, 7, 512)	1180160
leaky_re_lu_12 (LeakyReLU)	(None, 7, 7, 512)	0
max_pooling2d_8 (MaxPooling2	(None, 4, 4, 512)	0
flatten_2 (Flatten)	(None, 8192)	0
dense_6 (Dense)	(None, 256)	2097408
leaky_re_lu_13 (LeakyReLU)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 128)	32896
leaky_re_lu_14 (LeakyReLU)	(None, 128)	0
dense_8 (Dense)	(None, 64)	8256
leaky_re_lu_15 (LeakyReLU)	(None, 64)	0
dense_9 (Dense)	(None, 32)	2080
leaky_re_lu_16 (LeakyReLU)	(None, 32)	0
dense_10 (Dense)	(None, 2)	66

Fig. 14: Model Summary for the final CNN

Model	Accuracy
LR	0.79140
CNN	0.78800
CNN + DA with 10 000 TS	0.88196
CNN + DA with 20 000 TS	0.89365
CNN + DA with 40 000 TS	0.90460
CNN + G + DA with 50 E	0.91343
CNN + G + DA with 150 E	0.91966
CNN + G + DA + NPP 150 E	0.92030

TABLE I: Prediction accuracy for different model architectures

- DA: Data Augmentation
- TS: Training samples
- G: Generator
- E: Epochs
- NPP: Neighborhood Post Processing
- LR:Logistic Regression

V. Discussion & Improvements Possibilities

In the first project, one of the major drawbacks of using regression was its high time consumption, especially as the number of data increased. This time, it was one of the major goals to decrease this amount time in order to have the time to try a large number of different approaches. However, since CNN was used, the computation time for the best prediction also took a long time.

In order to verify the robustness and the accuracy of the model, an overlay between the model's prediction and the real image was performed. By doing so one could assess where the model was making the major part of the mistakes so that the groups could orient its effort in a certain way. First, one was could see that diagonal roads were difficult to detect. Therefore, we augmented manually the data in addition to the preprocessing. Since CNN required a long time to be trained, it is not feasible to optimize their hyper-parameters through a grid search. Therefore, it was quite difficult to adjust the hyper-parameters, amount of epochs, batch-size, and number of steps per epoch apart from the "old-school" manually tunning according to various heuristics.

Furthermore, in the last two week of the project, our group saw that we got stuck on the 0.92% mark. This could be due to the fact that the model never considered an image pixel by pixel. For instance by doing so, the code forms batches of pixels at the very beginning and everything was fitted on these collections of pixels. It could be that training each pixel of an image and then form the patches based on these values is more accurate since it could give better approximations for pixels at the interface between road and non-road elements.

Finally, concerning the post-processing, simply by thinking about the nature of the data, one could remark that roads were continuous elements in the satellite picture. Nevertheless, as shown in figure 7, the predictions showed two obvious problems. First, there were gaps in the predicted roads and the post-processing methods used in this project were not efficient enough to increase reliably the accuracy of the model, with more time this would be a major part to improve.

VI. Conclusion

This project was a good way to consolidate the acquired knowledge given in class about CNN. The group had the occasion not only to test many different architectures but also many different pre- and post-processing approaches to increase the precision of the model. However, even if more time was invested in comparison to the first project which ranked at the ninth position, the group was not able to match this high standard a second time which was a bit disappointing. Nevertheless, we were very satisfied with the perseverance shown by each member of the group in other to carry the prediction a bit further to 100% of accuracy. My Little Poly did not succeed to take its revenge but it will come back even stronger next year. This being said, we wish a **Merry Christmas** and a happy new year. **Thank you** for everything!

VII. Appendix

References

[1] CS231n: Convolutional Neural Networks for Visual Recognition, Stanford. Fei-Fei Li, Justin Johnson, Serena Yeung. $http : //cs231n.stanford.edu/$

[2] Hormese, J., Saravanan, C. (2016). Automated Road Extraction From High Resolution Satellite Images. Procedia Technology, 24, 1460–1467. $https : //doi.org/10.1016/j.protcy.2016.05.180$

[3] Wang, W., Yang, N., Zhang, Y., Wang, F., Cao, T., Eklund, P. (2016). A review of road extraction from remote sensing images. Journal of Traffic and Transportation Engineering (English Edition), 3(3), 271–282. $https : //doi.org/10.1016/j.jtte.2016.05.005$

[4] Courtrai, L., Lefèvre, S. (2016). Morphological path filtering at the region scale for efficient and robust road network extraction from satellite imagery. Pattern Recognition Letters, 83, 195–204. $https : //doi.org/10.1016/j.patrec.2016.05.014$

[5] Mnih, V., Hinton, G. (2010). Learning to Detect Roads in High-Resolution Aerial Images. Proceedings of the 11th European Conference on Computer Vision (ECCV), 1–14. $https : //doi.org/10.1007/978 - 3 - 642 - 15567 - 3_16$

[6] Bakhtiari, H. R. R., Abdollahi, A., Rezaeian, H. (2017). Semi automatic road extraction from digital images. Egyptian Journal of Remote Sensing and Space Science, 20(1), 117–123. $https : //doi.org/10.1016/j.ejrs.2017.03.001$