Detecting Road Segment From Satellite Images With Machine Learning

David Schroeter, Luca Carroz & Paulo Ribeiro
CS-433 Project 2, December 2023, EPF Lausanne, Switzerland

***Abstract*—With the recent rise of Machine Learning (ML), new discoveries in the various fields of chemistry, physics, medicine, etc. have been made possible. Thanks to the never ending increase in computational power of our computers, ML allows us to better understand our world and to solve complex problems faster and faster. This paper shows how ML algorithms can help us identify road segment from satellite images comparing simple Logistic Regression and more advanced methods like Convolution Neural Network.**

I. Introduction

Road segmentation, the identification of roads in aerial or satellite imagery, is pivotal for diverse applications. This paper explores a machine learning approach to address the complexities of this task, leveraging advancements in computer vision and deep learning. Through rigorous experimentation and evaluation, we demonstrate the efficacy of our proposed method.

II. Model and Methods

Our dataset consists of 100 colored (RGB) satellite images of dimension 400 by 400 pixels (px). For each image, we have a corresponding black and white image of the same size, where a pixel is white if it corresponds to a road pixel in the original image and black if it corresponds to background pixel. Our task is, given a new set of images never seen by our model, to predict whether a box of 16px by 16px is of type road or background.

In order to accomplish this task, we began by implementing a simple Logistic Regression, then we moved to a U-Net (a famous Convolutional Neural Network (CNN) architecture) and finally we tried a DLink-Net (an other CNN architecture).

1) *Logistic Regression*: Since this type of model is typically not fit to work on images, we chose to split our 400px by 400px images into patches of size 16px by 16px. From each patch, we extracted relevant features to give to our model. We chose those features to be the mean and standard deviation of pixels intensity for each of the three channels (red, green and blue). The model will thus learn how to differentiate road and background patches from the average and the variance of colors of each patch.
Before fitting our model, we chose to ℓ_2-normalize our features, since the different features have quite different ranges.
In order to choose the best hyperparameters for our model we performed a k-fold on the following hyperparameters: the solver method (i.e. the algorithm used in the optimization problem), the penalty norm (by example the ℓ_1 or ℓ_2 norm), the regularization factor and the class weight.
After having found the best hyperparameters, we used them to fit the model one last time with the whole test set, and we analyzed the performances of our model on our validation set.

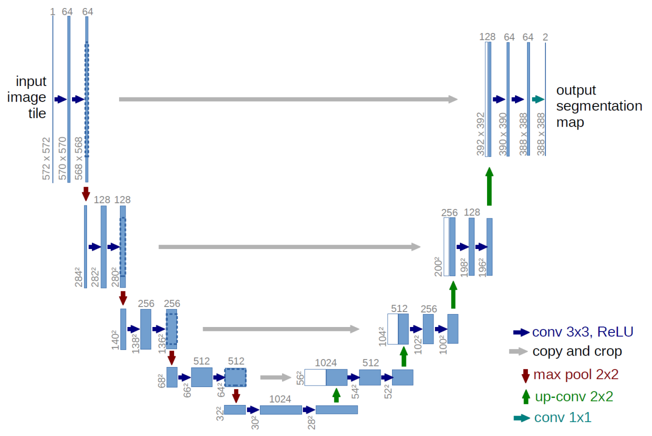

Figure 1. U-Net architecture (source: [1])

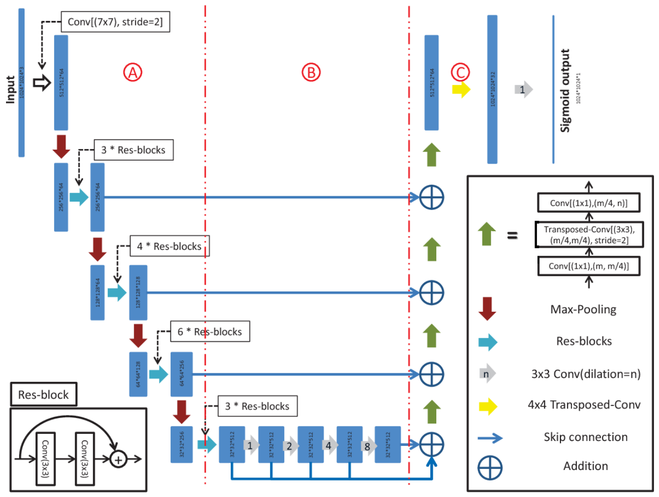

Figure 2. DLink-Net architecture (source: [2])

2) *U-Net* [1]: Since we are working with images, a natural choice was to use a CNN. We thus first chose one famous architecture, namely the U-Net. This architecture (Figure 1) is known to work well on medical image segmentation but also in other domains such as satellite image segmentation, which is precisely our task. We thus replicated this model using PyTorch, and trained it on an augmented dataset to make the model more robust to input variations.
3) *DLink-Net* [2]: The DLink-Net architecture (Figure 2) has shown good results across previous research in road extraction, and we thus implemented it in PyTorch. As the dimensions of our images deviated from the typical specifications employed by the DLink-Net architecture we encountered, some adjustments were necessary for compatibility.
   The training process was very similar to the U-Net: we generated an augmented dataset in the exact same manner but we had to decrease the number of epochs for the training, since this more complex model took more time to train for each epoch.

III. Data Augmentation

The augmented dataset was created by applying a series of transformations on our original images, by using PyTorch. Those transformations are:

i. *Random Crop*: Extract a random rectangle from the original image, in order to make the model more robust to dilation.
ii. *Horizontal/Vertical Flip*: Flip the image horizontally/vertically, in order to make the model more robust to symmetries.
iii. *Rotation*: Rotate the image by an random angle drawn uniformly in $[0°, 360°[$, in order to make the model more robust to rotations.
iv. *Gaussian Blur*: Blurs the image by applying a random kernel on each pixel that will change its original value by a weighted sum of its neighbours. This transformation should make the model more robust to different image resolutions.
v. *Color Jitter*: Randomly changes the brightness, contrast and saturation of the image, in order to make the model more robust to different image lightnings and imaging devices.

Note that we did not apply all those transformations to all images. The reason behind this choice is that most images in our dataset are actually very similar (in terms of lightning, resolution, etc.) and it did thus not make much sense to try to make the model extremely robust to those transformations. That is why we decided to apply transformations i. to iv. (as well as the brightness transformation from v.) not all the time, by sampling a Bernoulli random variable with probability of success $\frac{1}{2}$ for each transformation and applying the transformation only when the coin toss was a success. We did the same for the contrast and saturation transformations of v. but with a probability of success $\frac{1}{4}$, since those changes seemed to be even less present in our dataset compared to the others.

With the augmented dataset that contains both the original train images and transformed ones we trained our model for a certain number of epochs. By visual inspection, we noticed that the model had different strengths and weaknesses for each of the last epochs.

We thus decided to save the model state for the last few epochs and use those trained models together to generate our "combined" predictions. This was achieved by assigning the value 1 (i.e. road) to a pixel if at least a fraction of all the models predicted 1 for that pixel.

IV. Results

Prior to presenting the performance attained by each of our trained model, it is necessary to outline their key parameters.

1) *Baseline*: We opted for a baseline comparison to assess the statistical significance of our solutions compared to a dummy model. Following that, a random solution was "trained" to emulate the identical distribution of fraction numbers of roads pixels across its predictions.
2) *Logistic regression*: We used the Logistic Regression from sklearn, and we performed cross validation to find the best hyperparameters: .
   a) *Solver*: lbfgs, newton-cg, newton-cholesky, sag, saga
   b) *Penalty norm*: no penalty, l1, l2, elasticnet (mix between l1 and l2, depending on the l1 ratio)
   c) *Regularization factor*: a log space between 10^{-8} and 10^{8}
   d) *L1 ratio*: 10 equally spaced values between 0 and 1 (only used with elasticnet)
   e) *Class weight*: none, balanced, weight of 1 for class 0 and weight of 3 to 6 for class 0.
3) *U-Net (400x400)*: We employed the U-Net architecture and conducted training and prediction with inputs sized at 400px by 400px. The training environment included the following settings: .
   a) *Number of trained epochs*: 60
   b) *Augmentation factor*: 99
   c) *Validation size*: 15%
   d) *Learning rate*: 1×10^{-3}
   e) *Weight decay*: 1×10^{-2}
4) *U-Net (96x96)*: We employed the U-Net architecture and conducted training and prediction with inputs sized at 96px by 96px. The training environment included the following settings: .
   a) *Number of trained epochs*: 100
   b) *Augmentation factor*: 99
   c) *Validation size*: 15%

d) *Learning rate*: 1×10^{-3}
e) *Weight decay*: 1×10^{-2}

5) *DLink-Net (96x96)*: We employed the DLink-Net architecture and conducted training and prediction with inputs sized at 96px by 96px. The training environment included the following settings: .
   a) *Number of trained epochs*: 50
   b) *Augmentation factor*: 5
   c) *Validation size*: 15%
   d) *Learning rate*: 1×10^{-3}
   e) *Weight decay*: 1×10^{-2}

During training, we kept track of several metrics to observe the evolution of training for each model (c.f. Jupyter Notebooks). For clarity reasons, only the most pertinent metrics are used to compare models. We utilized multiple batches of a validation dataset to perform the metrics computations. This way, we can compute the average performance distribution per metric and per model (c.f Figure 3 and 4).

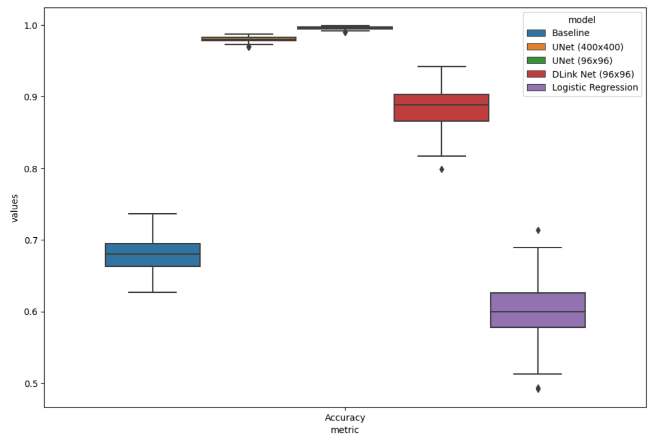

Figure 3. Accuracy comparison of all models

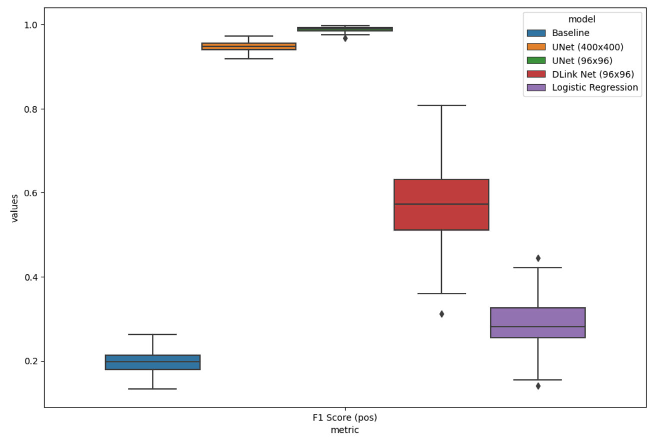

Figure 4. Accuracy comparison of all models

V. Discussion

Near the end of our project deadline, we noticed two flaws in our work on which we want to comment here.

1) *Training and Validation Split*: After implementing the data augmentation by applying transformations on images, we made an error and we split the augmented dataset in training and validation sets **after** applying the transformations. This means that both datasets potentially contained the same images, just with different transformations. Since we discovered this bug after having trained most of our models for a long time, we chose not to fully retrain all our models with a valid split between the two dataset (i.e. by splitting **before** applying the transformations) since this would have taken too much time. Instead, and in order to argue that our results are still valid, we trained a U-Net with a smaller input size and a smaller number of epochs on a valid train and validation sets. We then compared the old (and invalid, see 5) validation curve with the new (valid, see 6) curve, and indeed, we can see that the validation accuracy and loss have a very different shape.

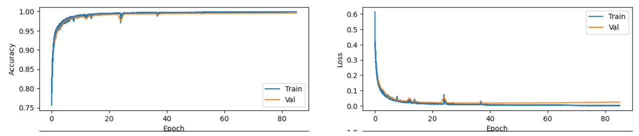

Figure 5. Accuracy and loss evolution with invalid validation set

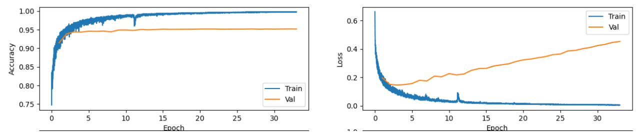

Figure 6. Accuracy and loss evolution with valid validation set

2) *DLink-Net*: Quite surprisingly, we observed that our DLink-Net performed substantially worse than our U-Net, even by training them over the same amount of time. This was an unexpected result, since from the documentation we read on those two architectures, we expected the DLink to outperform the UNet. We have two explanations of those results: first, it might be that we did not train the DLink for enough epochs and with too little number of images. Since the DLink was much slower to train than the UNet, we have to lower those parameters in order to train the model in a reasonable time, and the reason behind the ”bad” results of the DLink might just be due to under fitting. The second explanation is that we might have wrongly adapted the original architecture of UNet. Since the architectures we found online were all fit

for 1024x1024 px. images, we had to adapt our model to accept 400x400 px. images as input (and 96x96 px. images when we did the splitting). Changing the shape of the input implied a few changes on the kernel and padding sizes of some convolutional layers, which might have been the cause of our bad results. It was not very clear for us how to adapt the architecture, in particular if we had to lower all kernel sizes since our input was smaller than the intended one, and so the bad results of the DLink might be our mistake.

VI. Ethical Risk

Being able to cartography our roads by simply taking satellite pictures and passing them through a machine learning algorithm to identify roads might sound harmless and actually very helpful. We could think of using this technology with Google Maps by example, in order to update the map of the world as roads changes.

However, this technology could also be used in armed conflicts, by example by using autonomous armed flying devices (such as drones) that could now detect roads and destroy them. Using our model in those circumstances could greatly harm those infrastructures and even worse might mistakenly identify other locations such as civilian buildings as roads and destroy them, taking innocent lives.

Even though it is unlikely that a government would take our model as is and use it with this aim, the severity of the risk can be considered very high, since lives are at stake.

We unfortunately did not find any feasible solution to avoid such scenario, except controlling who gets access to our model.

VII. Summary

In summary, we compare Logistic Regression, U-Net, and DLink-Net models on a dataset of 100 RGB images. Logistic Regression extracts features from 16x16 pixel patches, while CNN architectures leverage data augmentation. Despite expectations, DLink-Net performs worse than U-Net, possibly due to underfitting or architectural adaptations. Ethical concerns are raised regarding potential misuse in armed conflicts. The study acknowledges a data split error, providing a comparison between invalid and valid validation curves for U-Net. In conclusion, the research explores model performance, potential flaws, and ethical considerations in satellite image analysis.

References

[1] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," vol. 9351, pp. 234–241, 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a

[2] M. W. Lichen Zhou, Chuang Zhang, "D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," 2018. [Online]. Available: https://ieeexplore.ieee.org/document/9324236