Project 2 report - Road Segmentation
CS-433 Machine Learning

Deniz Ira, Stanislas Jouven, Jonathan Labhard
Department of Data Science, EPFL, Switzerland

Abstract—**The following paper is our report for project 2 of class CS-433 Machine Learning. In it we will present the task we were given, what were the problems we needed to solve and the different steps we took to solve them. The task is to perform road segmentation given satellite/aerial images acquired from Google Maps. We want to determine which pixel of an image corresponds to the road and which pixel corresponds to the background. For that task, we found that U-Net architecture outperformed basic models and basic convolutionnal neural networks.**

I. Introduction

Image segmentation and classification is a problem that has seen a large increase in attention over the past decades, from technologies such as the GPS to more current ones such as self driving cars using camera images to predict roads. However, outdoor image classification is a complex task as our learning algorithm needs to take into account a lot of different parameters. That being said, with the rise of calculating power, more and more parameters can be taken into account making this kind of problem a new challenging but interesting task.

In this report, we analyze road segmentation classification using different methods: from logistic regression to the more complex convolutional neural network with different architectures.

II. Exploratory data analysis

The data we are given is composed of a training and a test dataset. The training set is composed of 100 aerial images of size 400 x 400 pixels given with their groundtruth representation. These are grayscale images where white pixel appear when there is a road and black pixel when there is not. The test set is composed of 50 images of size 608x608 images, and the goal is to find their groundtruth representation with the best performance.

Exploring the images, we notice that the task is more complicated than just locating paths which may look like roads: the parts indicated as roads in groundtruth images are only roads where the traffic actually circulates.

The images seem to be taken from a similar altitude, however one can observe differences in the time of the day, the general location as well as the weather conditions. All these variations may be beneficial for a better generalization of the model but they have to be considered in the training of our model.

Furthermore, the training images we are given contain roads which are often perpendicular or parallel to the orientation of the image: this may result in a poor generalization of the model, data augmentation techniques to counter that effect are considered. Data augmentation techniques will also be useful in order to have more data to train our model with.

III. Performance metrics

The metric that we use to evaluate and compare the performance of our models is the F1-Score of its prediction with respect to the groundtruth images.

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

In our internal analysis this metric is computed pixel-wise. However, for the final submission it is computed patch-wise. In other words, the final goal is to classify each patch of 16x16 pixels as either road or background.

IV. Models

When given the training data, the features we start with are the different RGB values of each pixel of an image. From these, multiple models can be built to use the features in the most efficient way. We will first look into the two given models to have a better understanding of what can be useful and what is not. Based on these observations and other researches, we will implement a new model and finally compare the result obtained with the different models.

A. Logistic Regression

In the project description, a model implementing logistic regression for this classification problem is given. Logistic regression is the simplest way to tackle a classification problem as complex as this one.

The idea behind the model is to split our images into a list of patches of size 16 x 16 pixels, and then classify each patch as corresponding to road (value 1) or background (value 0) using logistic regression. To help with such a task, the model uses features for each patch corresponding to the mean and the variance of the pixel values inside the patch.

However, as we can observe from the prediction visualization in Fig. 1, the model has trouble differentiating between road and roofs. That is because they both have sort of a gray color, and since our model only takes into account the mean and variance of the colors in a patch, the roads and the roofs will have very similar features.

At this point, we figure we need more features for the model to be able to make the difference between patches with very similar colors. What seems to be the best option is to take into account the context of a patch i.e. its neighbours. However, for such a task it seems more natural to use more advanced approached such as Convolutional Neural Networks.

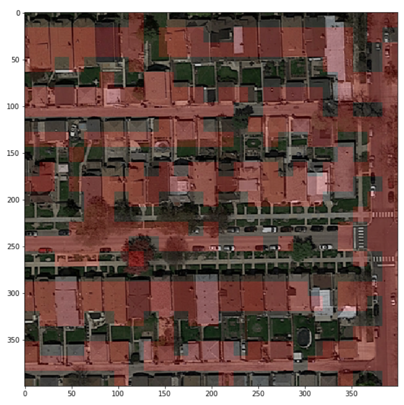

Fig. 1. Overlay visualization of logistic regression prediction

B. *Convolutional Neural Network (CNN)*

As explained previously, it is very hard to tune parameters for such a task using Logistic Regression. Neural networks are more adapted for this problem since it will automatically try to build the best features.

In particular, Convolutional neural networks are very efficient at processing images since the convolution operation will create large combinations of elements in the neighbourhood of a point of the image.

The model given here is a CNN that takes a patch as input. It is composed of two 2D convolutional layers with filter size 5x5 and respective depth 32 and 64, each followed by a ReLU activation function and a max pooling of size 2x2. It is then fed to two fully connected layers to finally output two values, corresponding to the two possible labels 0 or 1.

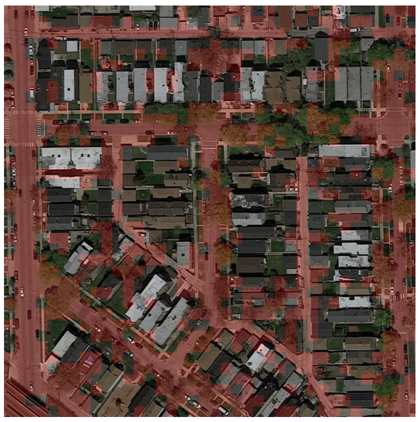

Fig. 2. Overlay visualization of CNN prediction

We can observe clear improvements from the previous model. However, we notice the model still has some trouble finding the borders of the roads and often still classifies roofs as road.

This CNN model is a good start but it has one big flaw: it only takes one patch of the image as input. This means that the features of the patch will not be affected by the neighbouring patches. The convolutions are made within the patch, and thus only take into consideration the neighbourhoods of pixels within the patch.

To improve on this model, we want to be able to take the whole image and classify each pixel, with the context of each pixel affecting the features. Luckily, this is a recurring problem in image segmentation researches and we can make use of some very insightful work that has already been done.

C. *U-Net Model*

Knowing the number of layers and convolution operations to put into our neural network in order to have the best results is a hard task. It would require training a model with every single combination of parameters and picking the parameters that give the best performance.

However, it has been shown that the architecture called U-Net, found at the Computer Science Department of the University of Freiburg in Germany [1] for biologic image segmentation outperforms other neural network architectures while maintaining computational efficiency and yields more precise segmentation on images.

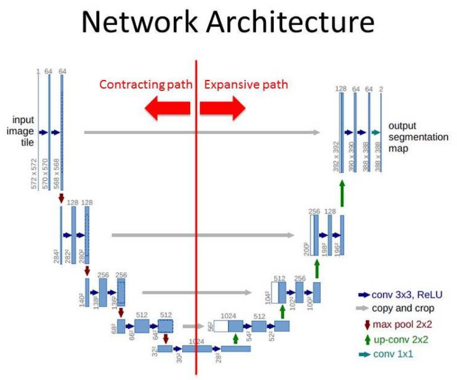

Fig. 3. Representation of the U-Net architecture

The U-Net model as shown in Fig. 2 is symmetrical and can be decomposed in two parts: a contracting path and an expansive path.

The contracting path consists of repeating a block of convolutions four times. A block is the operation of applying two convolutions with kernel size (3x3) to our input data, with increasing depth generating large number of channels. The series of convolutions in each block has the objective to capture features such as the edges, gradient orientation or color from a neighbourhood in the image, then the multiple layers of these blocks combine them and achieve a global understanding of the image. Each convolution is followed by a rectified linear unit (ReLU) activation function — which has the purpose of introducing non-linear properties to the network — which itself is followed by a max pooling operation with a stride of (2x2) contracting the size of the data while capturing its essence and reducing noise.

The expansive path, in contrary, consists of upsampling the data from the contracting part by applying up-convolutions with kernel size (2x2) expanding the size of the data, with the goal to end up with a same sized image at the end. After repeating this process four times, another convolution, this time with kernel size (1x1) is applied giving the output segmentation map in Fig. 2.

In our project, we make the choice to output only one channel because the road segmentation task takes into account only two labels. Thus, in order to make the distinction between road and background the output of the last convolution is fed with a sigmoid activation function.

V. U-Net Parameters Tuning

The U-Net model presented above has a lot of parameters that will affect its performance. In this part, we aim to optimize those parameters in order to obtain the best performance. We will analyze the effect of changing each one of the parameters in a vacuum in order to find the best value for each parameter. The only parameters that remains unchanged for our experiments are the loss and optimization functions. We use binary cross-entropy for loss and Adam for optimization.

A. Preprocessing

The given training data is composed of 100 images and is not enough to train our model properly. The Keras library built in tensorflow offers tools to automatically generate transformed images.

When looking into the training images we realized that most of the roads where vertical , horizontal or diagonal. The natural way to train the model with diagonal roads is to add rotations to the training images. For more diversity, we also flipped the images horizontally.

Furthermore, since brightness of satellite images may differ from one to another, we decided to generate images with different levels of brightness. However, this solution was not viable since adding this parameter made our loss be negative. We also considered shifting the channel values of pixels randomly, with the idea of better generalization with respect to the time of the day or weather conditions of the day where the image was taken, but to our surprise we obtained worse results with this method.

The following table gives all the possibilities along with their performances. Each model was built by generating 2000 images; 1600 of them are used for training and 400 for validation. We do not use more than 2000 images due to memory limits.

Rotation	Horizontal Flip	Training F1	Validation F1
90	No	0.880	0.873
90	Yes	0.878	0.875
180	Yes	0.883	0.879

TABLE I
Influence of preprocessing transformations on results

B. Input Load

When entering input to our model, we have to decide its size. We know that we input images of 400x400 pixels, however the number of images we input at each iteration will affect the performances. Furthermore, the input load is directly related to the depth of our convolutional layers, since the tools have their limits. In the following table we looked at the different possibilities and their performances. Each model was built with 300 epochs, activation function ReLU and dropout 0,25.

Batch Size	Depth	Training F1	Validation F1
16	16	0.872	0.867
16	32	0,879	0.873
32	16	0.872	0.870
16	64	0.885	0.875
32	32	0.885	0.876
64	16	0.875	0.868

TABLE II
Influence of input load on results

C. Activation Function

The U-Net architecture uses the rectified linear unit (ReLU) activation function after each convolution. However, the ReLU function is 0 for negative entries which leads to zero gradients and could end up with non optimal results. Therefore, we also looked into the Leaky ReLU activation function which avoids 0 values and compared the results.

The table below shows their performances for a comparative study. One thing to note is that Leaky ReLU is more demanding on the memory and thus forces us to reduce our input load if we want to use it. This is taken into account as a cost of this activation function.

Activation Function	Training F1	Validation F1
ReLU	0.884	0.879
Leaky ReLU	0.883	0.872

TABLE III
Influence of different activation functions on results

D. Regularization

One of the major problems of neural network model is to avoid overfitting. Applying the U-Net architecture into our road segmentation problem may result in up to 35 million parameters.

One way to avoid overfitting is to add a dropout value after convolutional layers. This will decrease the number of connections between nodes and thus allow our model to have different types of connections at each layers.

The following table shows the influence of the quantity of dropout use on the performance of the model.

Dropout	Training F1	Validation F1
0	0,884	0,867
0.25	0,885	0,876
0.5	0,884	0,866

TABLE IV
INFLUENCE OF DROPOUT ON RESULTS

Layer operation	Parameters
Input	(32x400x400x3) RGB images batch
Contracting path	
2D Convolution	Kernel (3x3); Depth 32
Activation function	ReLU
Batch Normalization	-
2D Convolution	Kernel (3x3); Depth 32
Activation function	ReLU
Batch Normalization	-
Max Pooling	Stride (2x2)
Dropout	0.25
Repeat 4x	Convolution depth x2 at each repeat
Expansive path	
2D Convolution	Kernel (3x3); Depth 512
Activation function	ReLU
Batch Normalization	-
2D Convolution	Kernel (3x3); Depth 512
Activation function	ReLU
Batch Normalization	-
2D Transpose Convolution	Kernel (3x3); Depth 256
Layers concatenation	-
Batch Normalization	-
Repeat 4x	Convolution depth /2 at each repeat
Output generation	
2D Convolution	Kernel (3x3); Depth 32
Activation function	ReLU
Batch Normalization	-
2D Convolution	Kernel (3x3); Depth 32
Activation function	ReLU
Batch Normalization	-
2D Convolution	Kernel (3x3); Depth 1
Activation function	sigmoid
Output	(32x400x400x1) Grayscale images batch

TABLE V
FINAL MODEL ARCHITECTURE

E. *Final U-Net with optimized parameters*

Taking the parameters that yielded the best performance, we obtain the following architecture (Table V) and prediction (Fig. 4) for our U-Net model

Fig. 4 represents the same overlay image as CNN, this time using U-Net model with the parameters presented in Table V. We notice a really big improvement from logistic and CNN. As we can see the U-Net model can almost perfectly extract horizontal, vertical and curve roads without mistaking roofs for roads.

VI. RESULTS

Model	F1 Score
Logistic Regression	0.468
CNN	0.601
U-Net	0.906

TABLE VI
F1 SCORE ON SUBMISSION FILE FOR EACH MODEL

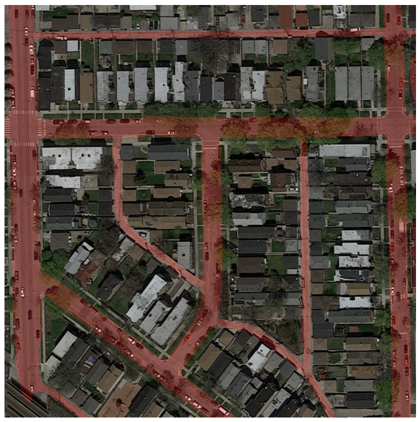

Fig. 4. Overlay visualization of U-Net Model prediction

VII. DISCUSSION

We can see that the U-Net model clearly has a higher performance than the other models. This is very much what we expected, not only because the model is a lot more complex than the other two, but also because we made the decision of focusing on improving this model rather than the two others. It is possible to improve the performance of the other models, but seeing the size of the gap in performance, it is very unlikely that they can outperform the U-Net architecture.

One interesting point that we can note is that we have a higher F1 score on the final submission than we had with the training set. We think that is due to the final F1 score being calculated patch-wise rather than pixel-wise.

Finally, it is important to add some details about the implementation of the U-Net. We trained the model using an Nvidia Tesla V100 GPU provided by Google Cloud services to be able to try several variations of parameters. Even then, the training took time and the GPU had limits. For the final run we used continuous generation of images instead of an initial set of 2000 images; this improved the performances but forced us to reduce that size of the batch entered as input, which is why it is smaller than the optimal parameter. We also used many tools provided by the keras library built in tensorflow to help building neural networks.

VIII. CONCLUSION

The goal of our project was to create the best model to predict roads classification on a test set of satellite images. We therefore started with a basic logistic regression, then moved on to a simple convolutional neural network to finally dive into the U-Net architecture and find the most optimized parameters. Looking at the performances of these models emphasize on the power of deep neural network, especially on the U-Net model.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597