CS-433 Project 2: Road segmentation

Yann Vonlanthen, Tiago Kieliger, Benno Schneeberger
Ecole Polytechnique Federale de Lausanne, Switzerland

***Abstract*—This report outlines our efforts to tackle the problem of classifying satellite images into road and background segments. This task is non-trivial as the land depicted by satellite images can be very diverse, and the bird's-eye view on roads is oftentimes obstructed. We show that Very Deep Convolutional Networks are well suited for this task and perform a detailed analysis of our proposed model and hyperparameters. We test different methods to perform data augmentation on the small data set and compare our model to various different approaches such as Transfer learning. Our proposed model works well even with very little hand-annotated ground truth and is thus highly practical.**

I. Introduction

Image segmentation is a task in computer vision that aims to partition a digital image into sets of pixels, where each set shares some common characteristics. This change in representation can facilitate further processing of the image and enable many interesting applications. The goal of this project is the segmentation of satellite images into two classes, road and background. The data set consists of images taken from Google Maps and contains mostly urban areas.

The difficulty lies in the fact that roads are not always easily distinguishable when seen from above, as buildings, trees, shadows and cars often hinder a clear view of the street. Additionally some other structures like parking lots, rooftops, railroads and even hydroelectric dams look like roads but really shouldn't be classified as such. The task at hand is particularly interesting in a time of growing interest in autonomous vehicles and can for example also be of great help in underdeveloped countries or after natural catastrophes, where humanitarian help needs to be delivered.

To tackle this challenge we start by establishing a simple baseline upon which we incrementally build up a model by testing the impact of different architectures on the performance. We state which approaches turned out to be successful, running our model on a wide choice of hyperparameters. We then continue by comparing this model to other possible approaches, such as transfer learning or multiple input convolutional neural networks. Our final contribution is a simple open-source pipeline which allows to reproduce all our results and can serve as a test-bench for further model testing.

II. Exploratory Data Analysis

The training data set consists of 100 satellite images, each 400x400 pixels in size with three RGB channels. To each training image a human annotated ground truth image is associated, represented by a black and white mask with white pixels corresponding to roads and black pixels to background. While the ground truth has a precise pixel wise segmentation, the task we are asked to perform is the binary classification of 16x16 patches, where a patch is considered as road if more than 25% of its pixels are labeled as such.

Figure 1 below shows an example of a satellite picture along with the corresponding ground truth.

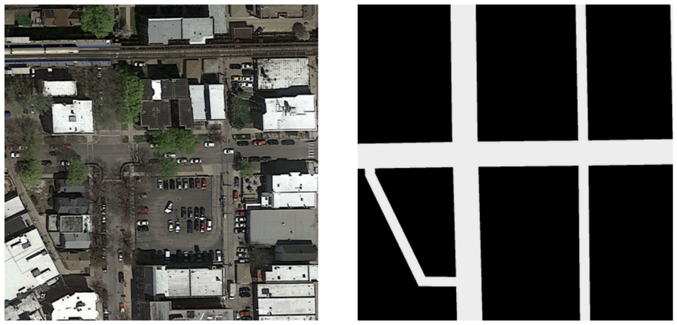
Fig. 1: Example of a picture and the corresponding ground truth

While the aforementioned difficulties created by the satellite view being obstructed seem to be the most important, other challenges lie in the fact that the color of a road can vary significantly, due to wear and the shadows cast on it. Finally quite a large amount of ambiguities can arise, namely when walkways, railroads and parking lots are present. In the given data set we additionally realized that most roads run either vertically or horizontally to the frame of reference, which represents a bias. We will address these issues and propose solutions in section III.

A. Score metric

Since approximately 75% of the data set consists of background, a trivial constant model that always predicts background will have an accuracy of around 75%. This means that a model with high accuracy does not necessarily perform well for the task. Therefore, to mitigate this issue we also considered the $F1$ score metric throughout this report, which is the metric used by the competition framework to evaluate submissions.

The $F1$ score metric is defined as the harmonic mean of precision and recall and ranges from 0 (worst) to 1 (best), where the precision is the ratio between true positives and false positives and the recall is the ratio between true positives and false negatives. In this case, a true positive is when the model correctly classifies a patch of the image as a road. Under the $F1$ score metric, the trivial constant model would have zero precision and zero recall, which is defined as an 0.0 $F1$ score. In most models, the $F1$ score and the accuracy are highly correlated, hence it is reasonable to try to maximize the accuracy during the tuning of the model and use the $F1$ score as a sanity check once the model is tuned.

III. Models and Methods

While today the task of semantic image segmentation usually describes the assignment of a class to each pixel, here a more coarse-grained classification of patches is asked. This has many implications on possible data augmentation as well as on the choice of the model. Most of the state-of-the art-solutions apply the "Fully Convolutional Network paradigm" presented by Long et al. in 2014. [1][2] However we concluded after some research that in order to perform binary classification on entire patches it is more convenient and powerful to directly use image classification models. In image classification Deep Neural Net's such as VGG, GoogleNet or ResNet are currently the most popular choices. [3] After testing some more traditional methods such as linear regression, we quickly realized that they are at a significant disadvantage, as feature extraction needs to be done by human engineering. We thus decided to build our own Neural Network by using state-of-the-art primitives, coupled with hyperparameter selection based on experimental results.

Our model follows the idea of very deep convolutional networks such as VGG, where the input is passed through multiple stacks of convolutional layers with a max-pooling layer between each stack of convolutional layers. On top of these stacks are added one or more fully connected layers with a ReLU activation function. We chose to use this architecture since it has been shown to perform well [4]. Below, a representation of the first few layers of our model:

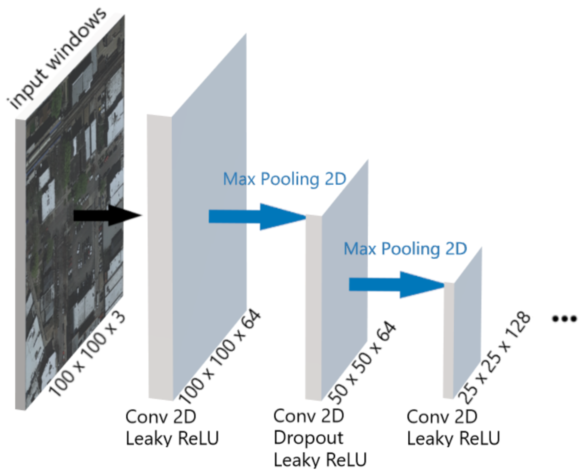

Fig. 2: First layers of the model

The results of our best performing model can be found in Table I and its complete description can be found in Table II.

A. Data augmentation

1) Considering context: Considering only patches of 16x16 pixels for training and prediction is far from optimal, as many patches lack enough context to discern between road and background. Consider for example a patch containing a car: is that car simply driving on a road or in a parking lot?

To tackle that problem, we consider our model's input to be a larger region of the image that we call window. The model then has to learn to predict the center part of it. This method considerably improved the model's accuracy and $F1$ score, but it is important to tune the window size to provide sufficient but not too much context, while computational complexity also comes into play. Special care has to be taken when training on a patch for which the corresponding window would exceed the image boundaries. To solve this problem, we pad the image by mirroring the pixels adjacent to the border.

2) Transformations: Deep learning models perform better the more training data they have at their disposal. Hence, as a first step to augment our data set we decided to add horizontal and vertical symmetries as well as rotations by increments of 45 degrees of the training images. Having 45 degrees rotations allows for better classifications of roads running diagonally through the image. In a second step we also take advantage of the preciseness of the ground truth, by generating patches from a random location inside the training image.

3) Imbalanced data set: As expected we have observed that when sampling patches we are much more likely to have data points from the background class, rather than depicting a road. It has been shown that such skewing is likely to have a negative impact on performance. [5] There are various ways of tackling that problem, including up sampling the minority class, down sampling the majority class, giving higher weights to the minority class when training the model or adjusting the decision threshold. Experimentally, we have seen best results using up sampling, i.e. by forcing our data generator to output batches containing equal amounts of data points per label. It can be noted that this data generator runs parallel to our model and will thus not affect performance. Figure 3 shows how up-sampling reduces the convergence time by more than 50 epochs, as well as having a beneficial impact on the training accuracy.

B. Regularization

It has been shown that regularization can be used to avoid the over-fitting of our model. Being one of the most popular and commonly used methods, we chose to add Dropout layers to our model. The idea of such layers is to set a fraction σ of randomly selected inputs to 0, which is equivalent to deleting them. We decided to add one such layer after performing two convolutions respectively. In order to determine the value which leads to the best result, we tried different rates σ with the exact same model. Considering the results we achieved for each value, we decided to use $\sigma = 0.1$.

C. Activation function

The most popular activation function for convolutional neural networks is the Rectified Linear Unit (ReLU) function. It performs well and converges around 6 times faster than other activation functions such as tanh or sigmoid. [6] The issue with ReLU, called "dying ReLU", is that it maps negative values

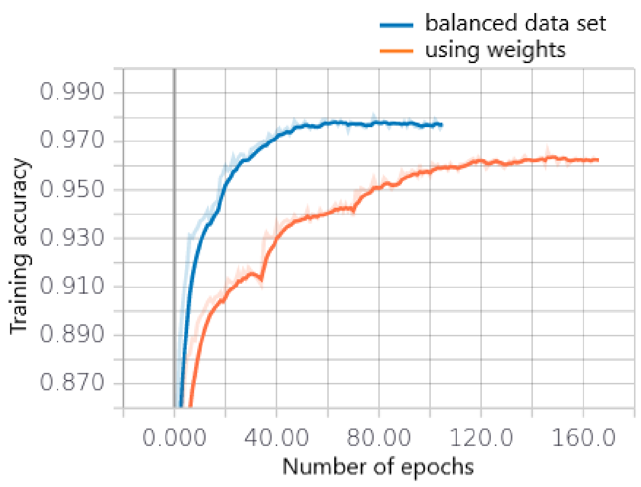

Fig. 3: Using a balanced data set (blue) versus simply giving more weights to under represented class (orange).

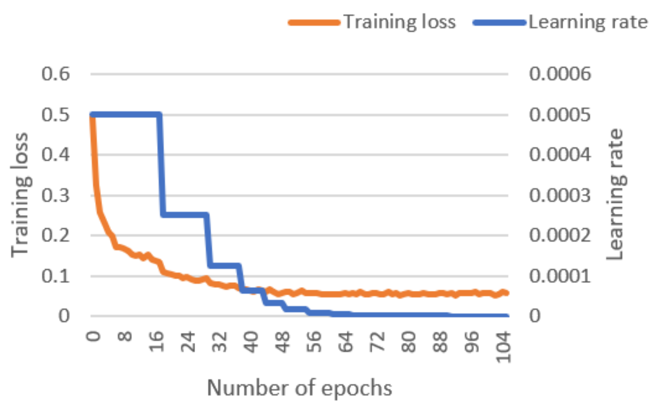

Fig. 4: Training loss and learning rate.

immediately to 0, which may affect the ability to train or fit the data properly. [7] Therefore, we decided to use the Leaky ReLU function, which is a modified version of the ReLU function with an additional parameter α that addresses that issue, by having a small negative slope of $-\alpha$ when $x < 0$. It is equivalent to the ReLU function when $\alpha = 0$. We tried different values of α's with the same model and decided to use $\alpha = 0.1$.

D. *Training details*

We have implemented the aforementioned model on Keras, running on top of a Tensorflow backend. To tune hyperparameters we ran our models in parallel on multiple AWS servers (8 core CPU with 32 GB of RAM) where convergence takes around six hours as well as in Kaggle notebooks, where the K80 GPU with 14 GB of VRAM allow a running time below two hours. To assure fast convergence of our models we started out with a relatively high learning rate, that was automatically reduced each time we reached a plateau (see Figure 4). The number of epochs was set to a artificially high number, but adding an Early stop condition to terminate training if no improvements in the validation accuracy have been achieved avoided unnecessary computations. The stochastic gradient descent was performed by Adam, a state of the art optimization algorithm.

IV. Further approaches

While we have built up a model through careful tuning of hyperparameters, we wanted to explore further ideas and compare performances to other possible approaches that we deemed interesting enough. They are briefly explained here and their results can be seen in Table I.

A. *Transfer Learning*

Very deep networks such as the one we used are very expensive to train, and typically require a large data set to work well [8]. In practice these issues are often overcome by using Transfer Learning. Testing this approach also allows us to verify our choice of hyperparameters. We imported the VGG 16 model and its weights, removing the top layers and replacing them with three randomly initialized layers that fit our binary classification task. Since our data set is small and we want the model to benefit from the previously learned features, we freeze the weights of the bottom convolutional layers that are responsible for the problem independent general feature extraction. The results can be seen in Table I and are briefly discussed in section VI.

B. *Using predictions to augment feature space*

When analyzing the predictions of our best model, we noticed a pattern in some of the classification errors: the predictions for roads often showed ”unprobable” roads, which were interrupted or had a small region of background inside them. Clearly in most cases, road do not stop and restart abruptly but the model may think it does when trees or bridges obstruct the view. To mitigate this, we tried to devise a multiple input neural network, which would use as input the usual window of pixels around the patch to predict but would have as additional input the prediction for neighboring patches made by our pretrained best model.

The assumption behind this new model is that the convolutional network would benefit from knowing the predictions of neighboring patches to avoid cases of ”unprobable” roads when possible. However, this did not yield interesting results, either because the above assumption is wrong or because our tuning of the proposed model was not adequate. The results of this model can be seen in section table I.

C. *Creating more hand annotated training data*

Finally when analyzing the produced masks, we found our model to have limitations in border cases that it hadn't seen much before, such as around water and highways. Therefore, we annotated by hand the ground truth of 25 new satellite images to augment the given data set of 100 images but it did not yield any improvement on the performance of our model. Possible reasons for seeing no improvement are that this augmentation is too small and that the additional images and corresponding ground truths are too different from the provided train set because they were generated separately and annotated by a different person.

V. Results

While we used only a single validation set during hyper-parameter selection, we validated our results through 4-fold Monte-Carlo cross-validation. The baseline is a model that classifies every patch to the background class. In the following, models called **CNN** are based on the architecture from Table II, where the difference between them is the window size and whether rotations and symmetries were used to augment the data set.

Model	Training acc	Validation acc	Validation $F1$
Baseline	0.741	0.741	0.0
CNN 16 x 16 window	0.890 ±0.006	0.873 ±0.022	0.743 ±0.047
CNN 100 x 100 window	0.962 ±0.009	0.953 ±0.035	0.906 ±0.071
CNN 100 x 100 window with rotations **best model**	0.972 ±0.013	0.958 ±0.043	0.919 ±0.056
Transfer Learning (VGG 16)	0.937	0.894	0.774
Predictions to augment feature space	0.970	0.944	0.876

TABLE I: Comparison of various models

VI. Discussion

We were able to observe a steady increase in the $F1$ score throughout the construction of our model. The largest improvement was made through enlarging the size of the window, i.e. giving our model a lot more context to work with.

Figure 5 displays the predicted road patches on a test set image in green, and shows that our model indeed provides satisfying results. We however saw some limitations when shadows were cast onto the road or when trees obstructed the view for example.

The further approaches that we tried were less successful, as we were not able to rival the previously achieved score. However this also shows that our model is sound and rather well tuned.

Concretely we think that our Transfer Learning approach suffers from the fact that we only trained the top layers, freezing the convolutional layers. Though preserving the weights was recommended for small data sets, we think that satellite images are quite different to images from the original ImageNet data set. Finally road detection in the center of a big image is probably a significantly different task to the image classification the model was trained on.

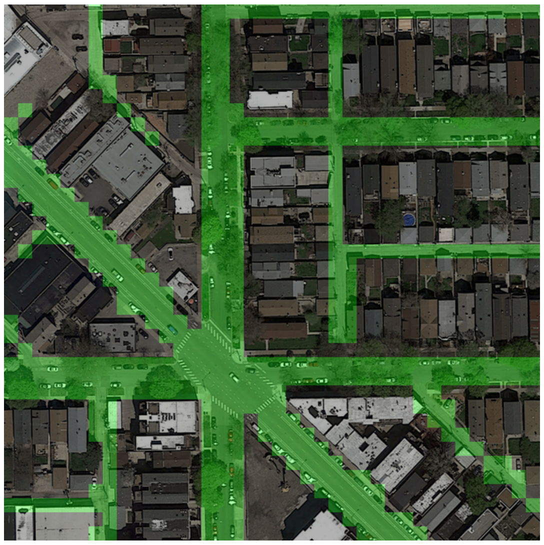
Fig. 5: Example of a prediction on the test set

VII. Summary

Through the tuning of the structure of the Convolution Neural Network and its hyperparameters, as well as some data augmentation, we are able to obtain a $F1$ score of 0.884 on the test set. We have also shown that considering context as well as using a balanced data set for training yields the most significant performance increase.

Type of layer	**Parameters used**
Convolution 2D LeakyReLU Max Pooling 2D	64 filters, (3,3) kernelsize $\alpha = 0.1$ (2,2) window size
Convolution 2D Dropout LeakyReLU Max Pooling 2D	64 filters, (3,3) kernelsize $\sigma = 0.1$ $\alpha = 0.1$ (2,2) window size
Convolution 2D LeakyReLU Max Pooling 2D	128 filters, (3,3) kernelsize $\alpha = 0.1$ (2,2) window size
Convolution 2D Dropout LeakyReLU Max Pooling 2D	128 filters, (3,3) kernelsize $\sigma = 0.1$ $\alpha = 0.1$ (2,2) window size
Convolution 2D LeakyReLU Max Pooling 2D	256 filters, (3,3) kernelsize $\alpha = 0.1$ (2,2) window size
Convolution 2D LeakyReLU Max Pooling 2D	256 filters, (3,3) kernelsize $\alpha = 0.1$ (2,2) window size
Fully connected Softmax	2 neurons

TABLE II: Full description of our CNN architecture

REFERENCES

[1] S. Chilamkurthy, "A 2017 guide to semantic segmentation with deep learning," http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review, accessed: 2018-12-17.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[3] P. Jay, "Image classification architectures review," https://medium.com/@14prakash/image-classification-architectures-review-d8b95075998f, accessed: 2018-12-18.

[4] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition," https://arxiv.org/abs/1409.1556, accessed: 2018-12-17.

[5] F. Provost, "Machine learning from imbalanced data sets 101," in *Proceedings of the AAAI2000 workshop on imbalanced data sets*, 2000, pp. 1–3.

[6] U. Udofia, "Basic overview of convolutional neural network (cnn)," https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17, accessed: 2018-12-17.

[7] S. Sharma, "Activation functions: Neural networks," https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, accessed: 2018-12-17.

[8] P. Jay, "Transfer learning using keras," https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8, accessed: 2018-12-18.