Classifying Satellite Images Using Convolutional Neural Networks

Ahmed Ahres, Ali El Abridi and Khalil Elleuch
School of Computer Science and Communication Systems, EPFL

***Abstract*— This paper presents an approach to build a classifier used to segment satellite images. The segmentation is applied to separate regions representing roads from the rest. Several image segmentation approaches exist and have been developed in the past years, from which convolutional neural networks (CNN) have performed very well in general with extremely satisfying results in our particular problem. The model presented in this paper uses CNN optimized with the Leaky ReLU activation function, the use of Dropout layers, max-pooling as well as appropriate image augmentation. These optimizations have shown efficient improvements. Our final model has yielded efficient results with an F1 prediction score of 89.5%.**

I. INTRODUCTION

Image segmentation refers to the partition of an image into a set of regions that covers it. The goal is to represent meaningful areas of the image such as face detection or detection of roads in a satellite image. Segmentation has two main objectives. The first one is to decompose the image into parts for further analysis. In simple cases, we extract only the parts that need to be analyzed. The second objective of segmentation is to perform a change of representation. The pixels of the image must be organized to be more meaningful or more efficient for further analysis. The increase in popularity of machine learning algorithms has led to process images in a reasonable time with fascinating results. One of the efficient methods in deep learning is convolutional neural networks (CNN) which is a category of neural networks. It has been proven that CNN are very effective in image recognition and classification [1]. The aim of this project is to build a model capable of performing the segmentation of a variety of satellites images. The classification consists in detecting which part of the satellite image is a road and which one represents background (anything other than a road).

This report details methods used in order to solve this problem and explains how convolutional neural networks can be efficiently used to classify images. Section II discusses the data we used to train and test our model, while Section III details the process of our model selection. Afterwards, Section IV presents the implementation details, before showing our prediction results in Section V. Finally, Section VI discusses and compares the obtained results.

II. DATA ANALYSIS

The provided dataset consists of 100 satellite images of dimension 400x400 pixels with their respective ground-truth images for training, and 50 satellite images of dimension 608x608 pixels for testing. A ground-truth image is a binary representation image in which the white color (i.e. pixel value 1) represents a road and black color (i.e. pixel value 0) represents background. Figure 1 shows an example of a satellite image and its corresponding ground-truth image from the dataset.
Concretely, our goal is to classify blocks of 16x16 pixels as 1 (road) if the final value of the model is above a certain threshold (0.25 in our case) and 0 (background) otherwise. An exploration of the training set has led to following division:

Type	Proportion
Background	74.09%
Road	25.91%

TABLE I
PROPORTION OF BACKGROUND AND ROAD (16X16 BLOCKS OF PIXELS) IN THE TRAINING SET.

There are two main challenges with this dataset. First, we noticed that some roads are covered by trees making it complicated to detect those roads in certain situations. Secondly, areas such as parking lots can be confused with roads because of their similarity (in terms of color and presence of cars). As a result, our model should be able to solve these challenges by using the information of neighboring blocks to classify the current block. This is achieved in the training process and its implementation details are described in Section IV.

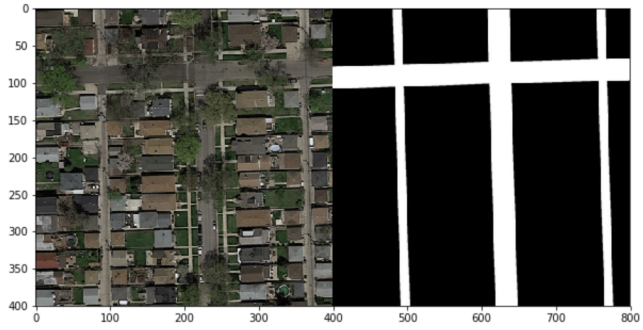

Fig. 1. An example of a satellite image and its corresponding ground-truth image

III. MODEL SELECTION

In order to assess the quality of the model, it is important to have a baseline model with which we can compare our result. An example of such a baseline model can be classifying all pixels as background (i.e. value 0) or all pixels as roads (i.e. value 1). Based on the fact that the satellite images of the dataset present more background pixels, a more plausible baseline model would to classify all pixels as background. Nevertheless, we decided to test both for more comparison bases.

Research has shown that an advanced and efficient approach is the use of neural networks. Since this dataset is dealing with images, convolutional neural netoworks have proved to be very efficient with image classification and have provided excellent results in our endeavors [1]. They have also been chosen as a final model to solve the task.

Numerous techniques exist in order to optimize the neural network and improve the accuracy of the model. These techniques are described in the following sub-sections, and have proved to be efficient for our task, with comparison results presented in Section V.

A. Data Augmentation

Data augmentation refers to the term of generating new data based on the existing data that we possess. Since our dataset is small (only 100 images), we can use image augmentation to generate new images for every training sample provided to the neural network. We can then effectively improve the learning process since it results in more training samples for the neural network model. By feeding more training images, the model will be able to improve its performance.

More specifically, every window supplied to the model is randomly rotated by steps of $\Delta \in$ [45, 90, 135, 180, 215, 260, 305, 350] degrees and flipped vertically/horizontally in a random manner as well. As a result, every provided image results in several images for the training model, which has improved the performance.

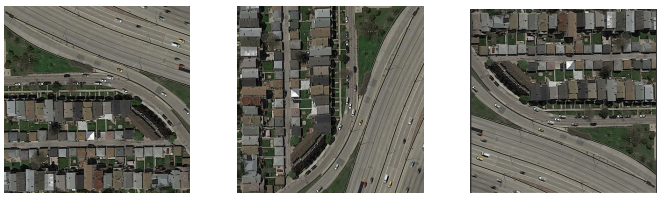

Fig. 2. Example of data augmentation on one of the images. From left: original image. Middle: original image rotated by 90 degrees. Right: original image rotated by 180 degrees

B. Max Pooling

After a convolution layer, it is common to add a pooling layer. The main objective of pooling is to continuously reduce the dimensionality, thus reducing the number of parameters and computations in the neural network. This does not only reduce the training time, it also controls overfitting. Several pooling methods exist such average pooling, max pooling and mean pooling. Research has shown that max pooling usually gives the best accuracy, and has been chosen for our model with window size 2x2 [2].

C. Regularization Technique

Dropout is a technique used to reduce overfitting, which has proved to be very efficient for convolutional neural networks when used after the pooling layer [3]. The goal of the dropout layer is to temporarily remove random neurons during the training process in order to force neighboring neurons to compensate and predict for the missing neurons. This technique has proved to work very efficiently in our dataset. Thus, a dropout layer has been added after every max-pooling layer as well as in the fully connected layer.

D. Activation Function

Finally, the *Rectified Linear Unit (ReLU)* is the most common activation function used in convolutional neural networks. *ReLU* has the advantage of being non-saturated, thus solving the so called "exploding/vanishing gradient" problem. Moreover, this activation function has effectively proved to speed up computation according Xu et al. [4]. However, when using a high learning rate, it suffers from the so-called "dying ReLU" problem. More specifically, some units may get stuck and cause incorrect results. Therefore, a variant of *ReLU* called *Leaky ReLU* was used, which has shown to produce better results with $\alpha = 0.1$ in our case [4].

With these research findings, the architecture of our final model has been designed and implemented in the following way:

Layers	**Characteristics**
Convolution	72 x 72 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Convolution	36 x 36 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Convolution	18 x 18 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Convolution	9 x 9 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Convolution	5 x 5 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Convolution	3 x 3 x 3
Leaky ReLU	Activation $\alpha = 0.1$
Max-Pooling	2 x 2
Dropout	p = 0.25
Fully Connected	128 neurons
Leaky ReLU	Activation $\alpha = 0.1$
Dropout	p = 0.5
Output	2 neurons

TABLE II

FULL ARCHITECTURE OF THE NEURAL NETWORK

IV. IMPLEMENTATION & TRAINING

The model has been implemented and tested using the *Keras* library with TensorFlow as backend. *Keras* is a deep learning library that has gained popularity in the past few years, and can use either TensorFlow or Theano as backend. The algorithm was designed and implemented in the following way: First a mini-batch of images (shuffled randomly) is generated from the training set. The images are of dimensions 72x72. The image augmentation technique described in Section III.A is then used on those generated images. Finally, they are fed to the learning algorithm.
The choice of dimensions 72x72 for the input images was decided due to the fact that a road in the dataset usually contains approximately 40 pixels of width or height. We added 16 pixels on each side of the street in order to take into account the context when training (as discussed in Section II), making up a total of 72. Furthermore, this value has shown to outperform several other window sizes in our model.

Fig. 3. A mini-batch consisting of 72x72 images fed to the learning model

Furthermore, the loss function used is the binary cross entropy function, which is a special case of the categorical cross entropy and is well suited for classification tasks according to research [5]. Finally, the model uses *EarlyStopping* callback which stops the learning when the model stops improving after 10 steps and *ReduceLROnPlateau* that reduces the learning rate when the model stops improving after 4 steps. This ensures that the model tries to reduce the learning rate at least twice before permanently stopping. In general, models show to often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates, which has also being effective for our resulting model.
The number of epoch was empirically chosen to be 150, with 1500 steps per epoch. The training took 10 hours and was achieved on a private server created in order to speed up the training process (instead of running the training on a local computer with less computing power). The server is equipped with 4 CPUs, 16GB of memory and one NVIDIA Tesla P100 as GPU.

Fig. 4. Example of 16x16 images that will be classified

V. RESULTS

With the above findings, different models have been tested on our dataset, with baseline having the lowest results and used for comparison purposes. Then, several architectures of CNN have also been implemented to see how they differ. The difference is adding and removing one or more of the optimization techniques discussed in Section III. Table III shows the results for each architecture.

Model	**F1 Prediction**
All Road	0%
All Background	42.4%
CNN	85.1%
CNN + Max-P	86.4%
CNN + Max-P + Leaky ReLU	87%
CNN + Leaky ReLU + DA + Max-P	88.6%
CNN + Leaky ReLU + DA + Dropout + Max-P	89.5%

TABLE III

PREDICTION RESULTS WITH THE DIFFERENT MODELS. ABBREVIATIONS USED: DA: DATA AUGMENTATION, MAX-P: MAX-POOLING

We notice that the use of *Leaky ReLU*, data augmentation, max-pooling and dropout greatly improved the result. We especially notice a difference when adding the data augmentation, resulting in an increase of 1.6% from the previously used model. Even though the difference is not very large between the different CNN models, the final result with our architecture is extremely satisfying for this problem.

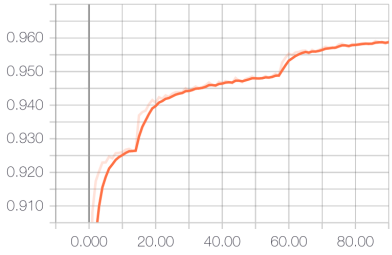

Fig. 5. Graph showing the evolution of the accuracy of the model. Y axis: Accuracy, X axis: epoch value

In order to show a concrete result of how the model performs on real images, a prediction result on arbitrarily chosen images from the dataset is shown below:

Fig. 6. Prediction visualization of background with our final model. Red means classified as background.

Fig. 7. Prediction visualization of road with our final model. Red means classified as road.

VI. CONCLUSIONS & DISCUSSIONS

Our solution using CNN has shown very efficient prediction results. We have also seen how techniques such image augmentation, dropout, max-pooling and the use of a more advanced activation function (Leaky ReLu) greatly improved the prediction result. On the other hand, we notice from the figures that the model is still confused between parking lots and roads in some situations. As discussed in Section II, this is a challenging task when looking at the images. This may due to the choice of 72x72 pixels or that 16x16 pixels are not well suited for these situations.
Furthermore, the model could still be improved by the use of advanced image augmentation (such as angle shearing or image translation) and different hyper-parameters for the techniques used as they were chosen empirically according to several heuristics.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems (pp. 1097-1105).

[2] Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). *A theoretical analysis of feature pooling in visual recognition*. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 111-118).

[3] Wu, H., & Gu, X. (2015, November). *Max-pooling dropout for regularization of convolutional neural networks*. In International Conference on Neural Information Processing (pp. 46-54). Springer, Cham.

[4] Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical evaluation of rectified activations in convolutional network.* arXiv preprint arXiv:1505.00853.

[5] Mannor, S., Peleg, D., & Rubinstein, R. (2005, August). *The cross entropy method for classification*. In Proceedings of the 22nd international conference on Machine learning (pp. 561-568). ACM.