Preprocessing of High-Resolution Satellite Imagery for subsequent Segmentation using Convolutional Neural Networks

Balz Guenat & Lukas Bischofberger
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Image segmentation, specifically road detection from satellite imagery, is a computer vision problem that can be solved using convolutional neural networks. This work enhances a basic network architecture by preprocessing the images and postprocessing the predictions produced by the CNN. Two different preprocessing methods which compute texture features are evaluated and compared. Using preprocessing alone, the mean prediction error can be improved from 27.1% to 23.8%. Furthermore, our postprocessing method achieves up to half the mean error on validation data compared to any unprocessed model.**

I. Introduction

Image segmentation is a classic computer vision problem. We tackle the problem of road detection in satellite images. More specifically, the goal is to label each pixel of a given high resolution color image either as foreground (road) or background (not road). The reappearance of neural networks made this problem interesting in the dicipline of convolutional neural networks (CNN). We developed a pipeline using a CNN coupled with pre- and postprocessing to solve the challenging task. Our main contributions are an assessment and comparison of different preprocessing methods, their impact on the prediction accuracy and the use of postprocessing to improve accuracy.

II. Models and Methods

We preprocess the images to obtain more additional information which is not readily available in RGB image data. Then we divided the image in patches of different sizes. We tried different parameters (2,4,8,16,32) for the patch size resulting in different coarseness levels. The patches are packed into batches and fed into the CNN. Finally we have developed a postprocessing method to combine predictions from different coarseness levels and remove false positives from the final prediction. The following subsections describe these individual steps in more detail.

A. *Preprocessing*

We use different preprocessing methods to extend the input data with additional channels. In a first step we compute the saturation S and lightness L for each pixel $(R, G, B) \in [0, 1]^3$ according to the HSL[1] system as follows.

$$Cmax := max(R, G, B)$$

$$Cmin := min(R, G, B)$$

$$d := Cmax - Cmin$$

$$L := \frac{Cmax + Cmin}{2}$$

$$S := \begin{cases} 0, & \text{if } d = 0 \\ \frac{d}{1-|2L-1|}, & \text{otherwise} \end{cases}$$

We then extend the images with these newly computed channels, such that every pixel is now comprised of 5 values $(R, G, B, S, L) \in [0, 1]^5$. Figure 2 shows the resulting channels for the image shown in figure 1.

In a second step we compute texture features. We present two different methods which we shall call the *GLCM method* and the *MaxDiff method*, only one of which is used in each model. The *GLCM method* utilizes the Gray-Level Co-occurrence Matrix (GLCM) [1] as computed on the lightness channel. Kirthika et al. [2] achieved good results using a very similar method. For each pixel we compute the GLCM for an offset of 1, angles of 0° and 90° and 8 levels. We then use this GLCM to compute the contrast, correlation, energy and homogeneity features of the pixel, yielding 4 additional channels which we append to the pixel. Using this method, the preprocessed images are comprised of 9 channels $(R, G, B, S, L, T1, T2, T3, T4)$. Figure 3 shows the resulting channels.

The main disadvantage of the *GLCM method* is its complexity. Depending on implementation and hardware, the preprocessing of one image with this method can take up to several minutes, heavily restricting the applicability of this method to real-time classification. For this reason, we also present the *MaxDiff method* which is a more lightweight alternative to *GLCM*. It computes a contrast channel, also on the basis of the lightness channel. For each pixel p and its right and lower neighbors p_r and p_d, we compute the following contrast channel and append it to the pixel.

$$T := max(|p - p_r|, |p - p_d|) \in [0, 1]$$

Using this method, the preprocessed images are comprised of 6 channels (R, G, B, S, L, T). Figure 4 shows the resulting channel T.

A visual inspection of the channels produced by these two methods reveals a close similarity between all of them.

[1] Hue, Saturation, Lightness coordinate representation

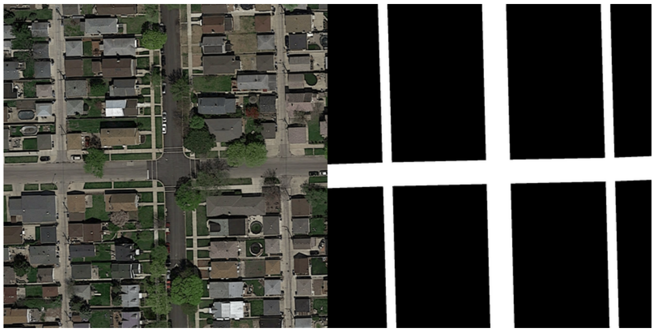

Figure 1. Original image and manually set labels

Figure 2. Saturation and lightness channels, computed in preprocessing step 1.

Because of this similarity, we expect both methods to have similar effects on the final prediction.

B. CNN model

For reasons explained below in the postprocessing sections we rely on three different CNN architectures and models, each working on different patch sizes cut from the original images. They are quite similar and only differ in their number of convolutional layers and filter sizes.

1) Patch size 2: For patches with size 2 we use one convolutional layer with filter size 2x2 and 80 channels followed by a RELU and a maxpooling layer of size 2x2.

2) Patch size 8: For patches of size 8 we use two convolutional layers followed by a RELU and a maxpooling layer each. The first convolutional layer has filter size 4x4. The second filter has size 2x2, they have 200 and 400 channels respectively.

3) Patch size 16: For the largest patches we use three convolutional layers all followed by a RELU and a max-pooling layer. The first two convolutional layers have filter size 4x4. The last filter has size 2x2, they have 200, 300 and 400 channels respectively.

The convolutional layers are followed by two fully connected layers with depth 1200, there we also included dropout to avoid overfitting. To train the network we used the AdamOptimizer [3] with an exponentially decaying learning rate.

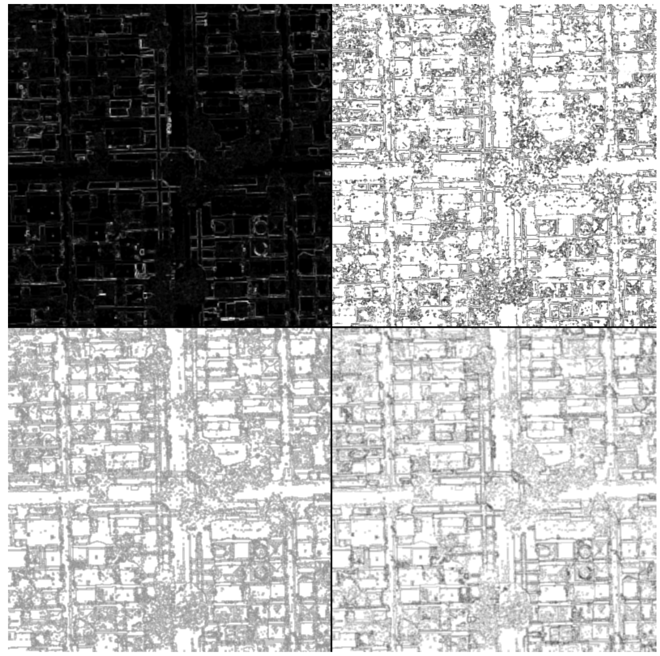

Figure 3. Contrast, correlation, energy and homogeneity channels, computed in preprocessing step 2A.

Figure 4. Contrast channel, computed in preprocessing step 2B.

C. Postprocessing

Each of the described CNNs have certain advantages and disadvantages. For example, a CNN with patch sizes 2 can accurately predict small areas. This is especially useful for removing false positives. Green areas of tress for example should not be predicted as roads if they are not covering the road. On the other hand, with small patch sizes the model is not able to distinguish between a rooftop and a road which are both gray. Therefore we also need the bigger context to avoid these kind of false positives. In postprocessing the predictions of models with patch sizes of [2,8,16] pixels are combined. A pixel is classified as road if the sum of the predictions over the different patch sizes is above a certain threshold. We calculate the predictions of all patches with size 16 independently for the different predictions, then we sum them up and classify only the submission patch sizes.

See the following equation where x and y are the pixel coordinated in an image and i and j are the pixel coordinates in a patch.

$$pr(x,y) := \begin{cases} 1 & \text{if} \sum_{s=2,8,16} \overline{pr(i,j,s)} > threshold \\ 0 & \text{otherwise} \end{cases}$$

D. *Expanding Training Data*

By default, images are divided into patches by laying a grid over them. We tried to expand the number of training patches by sampling patches at random positions without requiring them to be aligned by a grid. Using this method, the number of different training patches produced from one training image grows to the number of pixels in the image. In addition, patches can also be rotated and/or flipped, enhancing the size of training data even further. Unfortunately, for a reason unknown to us, our model failed to converge using these methods and we ended up not using them.

III. Evaluation

First we show how our preprocessed data performs in comparison to non augmented data and how different new features affect the prediction accuracy. Then we compare our pipeline to the baseline algorithm. Further we evaluate our postprocessing method, combining the models described above.

A. *Preprocessing*

We compare the prediction error of the two different preprocessing methods, the first preprocessing step only and no preprocessing at all. We train each model for 4495 steps at a batch size of 64 image patches and compute the mean prediction error on an validation set of 10 images.

With no preprocessing, the model achieves a mean error of 27.1%. After adding saturation and lightness channels, the mean error is 31.7%. Using the *GLCM method*, the mean error drops to 23.1%. Finally, using the *MaxDiff method* a mean error of 23.8% is achieved.

B. *Baseline comparison*

The baseline algorithm based on CNN was trained on 60 images and evaluated on a validation set of 10 images. We trained our pipeline on the same data and show the resulting errors in figure 5.

For the second baseline algorithm we trained it on 60 images again and predicted the one in figure 6 for comparison to our pipeline.

C. *Postprocessing*

We trained the three architectures described in section II with the following configuration. No preprocessing was used.

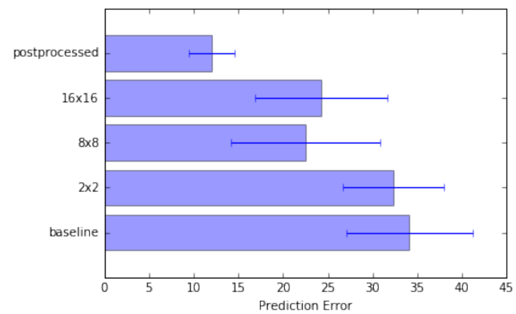

Figure 5. Mean errors of different patch sizes, postprocessed and baseline

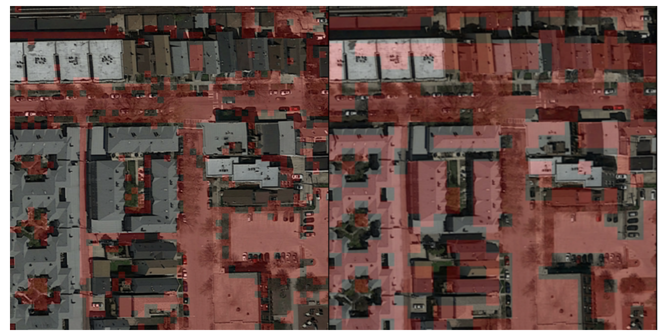

Figure 6. Our model (left) and prediction of baseline algorithm using logistic regression (right)

EPOCHS = 10
TRAINING_SIZE = 60
VALIDATION_SIZE = 10
BATCH_SIZE = 64

For all models we calculated the error on the validation set. Then we ran the postprocessing on the predictions with varying thresholds and calculated the prediction error on the newly produced predictions. See the results in figure 7 where we used thresholds between 2 and 3.2. The error does not decrease further by increasing the threshold beyond 3.2.

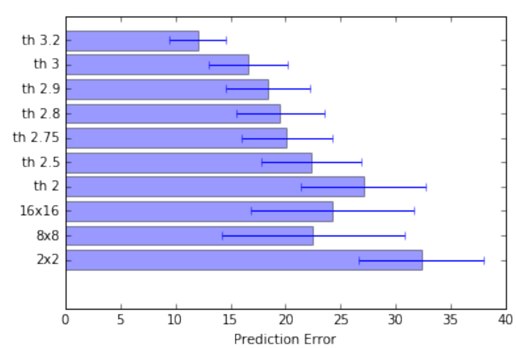

Figure 7. Mean errors of different patch sizes and postprocessed predictions with variying threshold

IV. Discussion

A. Preprocessing

As expected, preprocessing enables the model to make more accurate predictions. We are surprised that the inclusion of only the saturation and lightness channels actually diminishes the accuracy of the prediction. We expect this effect to disappear with a longer training time. We also see that the more involved *GLCM method* does not produce significantly better results than the *MaxDiff method*. Considering the large performance advantage of *MaxDiff* over *GLCM*, we see *MaxDiff* clearly as the better choice.

B. Postprocessing

Our evaluation of the postprocessing shows that we can reliably combine different models which are ran independently to gain better predictions. Unfortunately this approach relies on the assumption that the different models have different strengths. If no model is able to detect a certain structure, the postprocessing will not be able to improve the prediction.

V. Summary and Future Work

Our paper makes two main contributions. We describe different data augmentation schemes which easily compute features that would take a CNN additional time to learn. Furthermore, we describe a postprocessing method to combine different results to produce better predictions. We have shown that using several simple and thus fast models is a viable alternative to using large models that are hard to learn.

We believe it is worth investigating the impact of our postprocessing method on CNNs solving different classification problems. Furthermore, we are interested in whether other fast and simple preprocessing methods such as *MaxDiff* can improve prediction accuracy even more.

References

[1] M. Hall-Beyer. (2007) The glcm tutorial home page v2.10. [Online]. Available: http://www.fp.ucalgary.ca/mhallbey/tutorial.htm

[2] A. Kirthika and A. Mookambiga, "Automated road network extraction using artificial neural network," in *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*. IEEE, 2011, pp. 1061–1065.

[3] D. P. K. at al., "Adam: A method for stochastic optimization." ICLR, 2015.