Chapter 3
Ensemble Learning for Semantic Segmentation of Ancient Maya Architectures

Thorben Hellweg¹, Stefan Oehmcke², Ankit Kariryaa², Fabian Gieseke¹,², and Christian Igel²

Abstract  Deep learning methods hold great promise for the automatic analysis of large-scale remote sensing data in archaeological research. Here, we present a robust approach to locating ancient Maya architectures (buildings, aguadas, and platforms) based on integrated segmentation of satellite imagery and aerial laser scanning data. Deep learning models with different architectures and loss functions were trained and combined to form an ensemble for pixel-wise classification. We applied both training data augmentation as well as test-time augmentation and performed morphological cleaning in the post-processing phase. Our approach was evaluated in the context of the “Discover the mysteries of the Maya: An Integrated Image Segmentation Challenge” at ECML PKDD 2021 and achieved one of the best results with an average IoU of 0.8183.

Key words: Deep Learning, U-Net, DeepLabv3, Remote Sensing, Satellite Data, Lidar Data, Maya Archaeology

3.1 Introduction

Machine learning-based segmentation of aerial and satellite imagery is a promising tool for archaeological research. As areas of interest are often difficult to access or hidden by vegetation, satellite or aerial imagery offer the possibility to survey large territories at low cost. However, manual segmentation by experts is time-consuming and expensive in terms of human labor costs.

Recently, deep learning-based approaches such as the U-Net ¹ have successfully been employed for detecting objects in remote sensing. In special domains such as archaeology, however, only a small number of annotated training samples are usually available. Additionally, ground truth masks may exhibit labelling noise reflecting annotator-related biases. Against this background, the “Discover the mysteries of the Maya: An Integrated Image Segmentation Challenge”¹ at ECML PKDD 2021 sought for contributions on the localisation of three classes of ancient Maya architectures (buildings, aguadas² and platforms) by

¹ https://biasvariancelabs.github.io/maya_challenge/comp/
² artificial rainwater reservoirs [13]
performing integrated image segmentation of different types of satellite imagery and aerial laser scanning data.

In this work, we present our submission to the challenge, which is based on a heterogeneous ensemble of deep learning models. In an effort to find a suitable architecture for the segmentation task, models based on the U-Net and DeepLabv3 architectures were trained and extended. As not all data provided were equally suitable for the segmentation tasks, training inputs were selected from airborne laser scanning (ALS) and Sentinel-2 data. To increase robustness and to improve accuracy, we applied training data augmentation as well as test-time augmentation and used morphological cleaning as a final post-processing step. Since we eventually opted not to use a validation set in favour of larger training data (and thus supposedly better models), we combined the fitted models into an ensemble to mitigate the effect of overfitting.

Our approach generated a robust segmentation of all Maya structures and achieved an average IoU that was 5 percentage points higher compared to the provided DeepLabv3 challenge baseline.

3.2 Ancient Maya Settlements Dataset

We applied our approach to the challenge dataset covering the Chactún archaeological site [2, 7, 13]. The dataset comprises tiles derived from Sentinel-1, Sentinel-2, and ALS data, along with associated annotation masks. For Sentinel-1 and Sentinel-2, the dataset includes statistics and imagery for the years 2017 – 2020. Each Sentinel-1 TIFF file consists of 120 bands (24×24 pixels) and contain several temporal statistics, such as mean, median, standard deviation, coefficient of variance, for each year separately and for the entire period. The Sentinel-2 data groups 12 spectral bands (B01 – B12), each resampled to a 10 m resolution. For the period 2017 – 2020, there are 17 recordings with cloud masks and cloud cover below 6%. In total, each TIFF file therefore consists of 221 bands (17×13 bands; 24×24 pixels). Contrary to the low-resolution Sentinel tiles, the ALS data is provided with a resolution of 0.5 m per pixel in form of a visualization composite consisting of sky-view factor (band 1), positive openness (band 2), and slope (band 3) in separate bands (1 × 3 bands; 480 × 480 pixels) [13].

The data is split into training and testing data. For the training set, annotation masks are provided, separately for buildings, platforms, and aguadas (480×480 pixels, 8-bit, 0 corresponds to the target being present, 255 to not present). The distribution of Maya structures in the training set is highly imbalanced. While 1211 of the 1765 tiles contain one of the three archaeological structures, only 64 of these contain the structure aguada. In comparison, there are 952 tiles with platforms and 1129 tiles with buildings. Buildings and platforms are also often found together within a tile. Furthermore, buildings are typically placed on platforms, but can also be found outside of platforms.

3.3 Learning Segmentation Models

3.3.1 Pre-Processing

As the resolution of the Sentinel data is at least twenty times lower than that of the ALS data, much of the provided Sentinel data is potentially of little use for training the models. For this reason, we decided to only use the bands 2, 3, 4 (BGR), and band 8 (NIR) of the Sentinel-2 data for training. To reduce the Sentinel-2 images, we created median composites of cloud-free pixels. Both ALS and Sentinel-2 data were min-max normalised to a range between 0 and 1. For Sentinel-2, we used the 90% quantile value instead of the
Table 3.1 Considered augmentation schemes

<table>
<thead>
<tr>
<th>method</th>
<th>data</th>
<th>parameters</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random cropping</td>
<td>all</td>
<td>between 256 and 400 pixel of ALS size</td>
<td>100%</td>
</tr>
<tr>
<td>vertical flip</td>
<td>all</td>
<td></td>
<td>50%</td>
</tr>
<tr>
<td>horizontal flip</td>
<td>all</td>
<td></td>
<td>50%</td>
</tr>
<tr>
<td>advanced:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random rotation</td>
<td>all</td>
<td>between 0 and 359 degrees</td>
<td>25%</td>
</tr>
<tr>
<td>Gaussian blur</td>
<td>ALS</td>
<td>with kernel size 11 and randomly between 0.1 and 2</td>
<td>25%</td>
</tr>
<tr>
<td>additive noise</td>
<td>ALS</td>
<td>on each input individually</td>
<td>25%</td>
</tr>
<tr>
<td>uniform</td>
<td>ALS</td>
<td>between 0 and 0.1</td>
<td>50%</td>
</tr>
<tr>
<td>normal</td>
<td>standard deviation of 0.03</td>
<td></td>
<td>50%</td>
</tr>
</tbody>
</table>

maximum value to create a more robust scaling. Due to the low resolution, no data from Sentinel-1 were used to train the models. We initially split the available training set into two folds (training/validation), containing 80% and 20% of the samples, respectively. After testing the validity of our approach, we finally trained different models on the entire training set, based on which the ensemble is formed.

We also applied data augmentations to further mitigate the problem of the imbalanced and (relatively) small dataset. We divided our augmentations into normal and “advanced” augmentations. In contrast to the advanced augmentations, the normal augmentations – such as rotation and flipping – do not change the statistics of the images or drastically disturb the images, and are therefore considered safe to be applied. Depending on the model trained, only normal or both normal and advanced augmentations were used. Each augmentation has an individual probability of being applied, see Table 3.1.

3.3.2 Architectures

We experimented with different model architectures and combined them to obtain an ensemble of heterogeneous segmentation models. In particular, we made use of modified versions of the U-Net [11] architecture, which has already been successfully used for similar segmentation tasks in archaeological research [2]. In addition, we considered variants of the DeepLabv3 [3] architecture.

We first investigated several encoder networks to replace the downsampling part of the original U-Net. Although successfully employed in other research applications, using ResNet [5] as our encoder did not improve the model performance and was eventually omitted in favour of M NasNet, a comparatively small network optimized for mobile devices [15], and Swin-B, a self-attention network with pre-trained weights [9]. We used different activation functions for each: ReLU in ResNet, ELU [16] in M NasNet, and GELU [6] in Swin-B. For the Deeplabv3 architecture [3], we considered pre-trained weights for the ResNet101 encoder [5]. The pre-trained encoders were not optimized for the initial epochs to give the decoder time to adjust to the pre-trained weights.

For the upsampling path in the decoder part of the U-Net, instead of using bi-linear interpolation to initialise added pixels with the weighted average surrounding pixels, sub-pixel convolution with ICNR initialisation (pixel shuffle) was used [11, 12]. Contrary to most encoders that halve the resolution at each block, Swin-B downsamples to a quarter of the resolution per block. Consequently, we upsample two times in a row, which showed better results than directly upsampling to four times the resolution. As many super-resolution methods are known to generate artifacts, we applied 2 × 2 averaging filters after each upsampling operation to reduce artifacts [13]. Inspired by Zhang et al. [17], we introduced a self-attention layer in the second last upsampling layer, which allows to better model long-range dependencies and which
Fig. 3.1 An overview of our U-Net architecture

Table 3.2 Overview of different parameter assignments for: learning rate (lr), use of warm-restarts and cycling learning rates (cycle), batch size (BS), effective batch size (EBS), excluding aguada (no ag.), number of model checkpoints (# cp), usage of self-attention layer, and random crop size (rcs).

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Encoder</th>
<th>Loss</th>
<th>lr</th>
<th>cycle</th>
<th>BS</th>
<th>EBS</th>
<th>no ag.</th>
<th># cp</th>
<th>SA</th>
<th>rcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net</td>
<td>Swin-B</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>3</td>
<td>Yes</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>Yes</td>
<td>4</td>
<td>Yes</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>1</td>
<td>Yes</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VGG</td>
<td>BCE</td>
<td>$3 \times 10^{-4}$</td>
<td>No</td>
<td>2</td>
<td>Yes</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNasNet</td>
<td>VGG</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>6</td>
<td>Yes</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>4</td>
<td>Yes</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>4</td>
<td>Yes</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepLabv3</td>
<td>ResNet101</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>Yes</td>
<td>1</td>
<td>No</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>BCE</td>
<td>$6 \times 10^{-5}$</td>
<td>No</td>
<td>4</td>
<td>Yes</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

also increases the receptive field. As batch normalization in the decoder reduced the model performance, we replaced it with spectral normalization [17].

We conducted several experiments to incorporate the Sentinel-2 data at different stages of the U-Net encoder. Eventually, we decided to add a final cross-over ResNet block layer after the encoder, which received the concatenated encoder output and Sentinel-2 data as input. The Sentinel-2 data was upscaled to the same resolution as the encoder feature map.

3.3.3 Training

Various loss functions were used to train the networks. We found that good results could be consistently achieved with binary cross-entropy (BCE). (Focal) Tversky loss, which initially seemed promising considering the imbalance of the data, did not converge and was not used in the final ensemble for any of the models. In an effort to add a spatial loss function, we used the VGG loss [8] together with BCE. The VGG loss is calculated by inputting the predicted and true segmentation into an ImageNet pre-trained VGG16 model and by backpropagating the smooth L1 difference of the intermediate layers (blocks 2, 3, and 4). We used the Adam optimizer and considered various learning rates in the range between $1 \times 10^{-5}$ and $1 \times 10^{-6}$. To further smooth the learning curve during training, we accumulated the gradient over multiple
Instead of training individual models for each of the structures to be predicted, we used multi-task learning to exploit information shared among two or more connected tasks. In doing so, we assumed that predictions of platforms and buildings in particular are closely linked. With one exception, all models were trained to jointly predict aguadas, platforms, and buildings. For one model, we excluded aguada, which means that we did not consider the aguada mask during training as well as prediction. While oversampling of aguadas increased our aguada IoU at the beginning, its influence later became negligible and was, therefore, discontinued. When we stopped validating our models using a holdout set, we could no longer reliably determine whether models were already overfitting during training. In addition, since different models previously evaluated exhibited different strengths, the models trained on the full dataset were combined into an ensemble. Our final ensemble consisted of ten model configurations with 18 model instances (note that some ensemble members were represented at multiple points via checkpoints, where the validation error was lowest).

### 3.3.4 Post-Processing

To acquire the predictions of a single model, we applied test-time augmentation, where we flipped and rotated (in 90° steps) each test image. Hence, due to these operations and their combinations, several predictions were obtained per test image. The final output was the average of these predictions after reversing the applied operations. This was done to make our predictions more robust towards rotations and partly translations.

The final ensemble was obtained via soft majority voting of all chosen members to counter overfitting and to smooth the prediction borders. We also tried hard voting, where, instead of using the probabilities, only the discrete class decisions are used. The soft voting strategy produced better results and was, hence, preferred. Due to the time constraints, we could not investigate the effect of stacking \[10, 16\], which can potentially improve results and which can be used to identify models that perform better at predicting certain classes.

We noticed that, at times, our models predicted buildings, platforms, or aguadas, which were only a couple of pixels wide. Therefore, based upon the size distribution of the buildings, platforms, and aguada in the training masks, we removed the predicted objects whose size was below a certain threshold value. We used a different threshold value when the objects were partially or completely on the boundary of an
image. However, the thresholds were chosen ad-hoc and could be improved in the future. Likewise, very small objects (at the edge) could already be removed from the training masks during pre-processing.

The predicted masks had, at times, holes inside of predicted objects. In the training masks, we observed that the aguada and platform masks were usually convex without any gaps, and while buildings could take arbitrary shapes, they did not have holes either. Therefore, we decided to fill the holes in the predicted polygons during post-processing. Figure 3.2 demonstrates the effect of the morphological cleaning on the platforms predicted in tile 1832.

3.4 Results

The key components of our approach (i.e., ensembling heterogeneous models, augmentation in both training and testing, and morphological cleaning) turned out to be important drivers to improve the overall quality. Test-time augmentation helped to eliminate overconfident incorrect predictions. While it mainly affected the edges of the predicted masks, entire object masks could be corrected as well. By combining the individual models into an ensemble, this effect was further enforced. As the number of ensemble members increased, so did our average IoU. At the same time, however, the effect of test-time augmentation decreased, as overconfident predictions were corrected by ensembling many model instances.

Although only Sentinel-2 bands with the highest native resolution were selected for training, our models did not make use of the Sentinel-2 inputs (providing non-matching Sentinel-2 inputs after training did not lead to changes in the predictions).

An iterative evaluation of intermediate results on the test set via the challenge competition platform allowed for a best-breed approach, in which the best masks for each Maya structure were combined for the final contribution. We have refrained from improving our final submission in this way.

Figure 3.3 exemplifies the performance of our final model. It shows the predicted buildings, platforms and, aguadas for two sample tiles along with the corresponding ALS and Sentinel images. The organisers of the challenge provided the following results on the test set. Our ensemble approach was among the best submissions and achieved an average IoU of 0.8183 (aguadas: 0.9854, platforms: 0.7300, buildings: 0.7394).
Acknowledgements

SO, AK, FG, and CI acknowledge support by the Villum Foundation through the project Deep Learning and Remote Sensing for Unlocking Global Ecosystem Resource Dynamics (DeReEco).

References