Multi-scale patch-wise semantic segmentation of satellite images using U-Net architecture

Eugenio Fella
eugenio.fella@gmail.com

Theivan Pasupathipillai
theivan.pasu@gmail.com

Carlo Sgorlon Gaiatto
carlo.sgorlon.gaiatto@gmail.com

Abstract

This paper presents a semantic segmentation technique for satellite images using deep convolutional neural networks. We utilize the DeepGlobe Land Cover Classification dataset and employ a patch-wise pre-processing pipeline, including image rescaling and data augmentation, to improve the representation of input image features and cope with computational challenges. The imbalance between classes is addressed by weighting the multi-class cross entropy loss function. We implement a U-Net architecture as the baseline and demonstrate improvements with our approach. We also compare it with the state-of-the-art model, DeepLabV3+, fine-tuned on the dataset. The results indicate that our method achieves competitive performance according to the chosen metrics, showing its effectiveness in accurately segmenting satellite images. It outperforms the DeepGlobe challenge baseline score and it is comparable to the competition winners.

1. Introduction

Semantic segmentation is a fundamental task in image analysis, which aims to assign a label to each pixel in an image. The application of semantic segmentation on satellite images has been of great interest in recent years, as it enables the automatic interpretation of large-scale geographic information. It can be used for several tasks such as road extraction, building detection and land cover classification. Our analysis focuses on the latter, which is important in the framework of sustainable development, agriculture, forestry and urban planning.

The state-of-the-art in semantic segmentation is driven by deep learning-based approaches, particularly architectures built on Convolutional Neural Networks (CNNs). Most successful models used in this field are usually trained to detect common objects in everyday life images, which differ from remote sensing imagery in image size, color imbalance, object rotation variation and scale variation. We propose a pre-processing and data augmentation pipeline consisting of extracting patches at different scales to handle both large size images and multi-scale features. Furthermore, this procedure helps to increase the size of the training set.

The paper is organized with a description of the architectures used, an overview of the dataset, a section with the pre-processing and data augmentation methods along with the training process, and finally a discussion of the results using different metrics.

2. Related Work

The are several architectures which enable good performance in semantic segmentation. We focus on the two following ones.

2.1. U-Net

U-Net [1] is a popular deep learning architecture for semantic segmentation introduced to deal with biomedical images. The architecture consists of two main parts: the encoder and the decoder. The encoder is a traditional CNN that is used to extract features from the input image and reduce its resolution. It consists of 4 down-sampling blocks, each one composed of 2 convolutional layers and a max pooling layer. This process produces feature maps with reduced spatial resolution and a high level of abstraction. The encoder output is further processed by a bottleneck made of two convolutional layers. The decoder, symmetrical to the encoder, increases the resolution of the feature maps and generates the final segmentation mask. It consists of 4 up-sampling blocks, each one composed of a transposed convolutional layer and 2 convolutional layers. The decoder also uses skip connections, which concatenate feature maps from the corresponding block of the encoder to the decoder. The skip connections allow the network to propagate fine details from the encoder to the decoder, thus helping to preserve the spatial resolution of the output. Finally, there is an additional convolutional layer used to reshape the number of output channels to match the number of classes.

2.2. DeepLabV3+

DeepLabV3+ [2] is an extension of the DeepLabV3 [3] architecture used for semantic segmentation. It consists of three main components: an encoder, an Atrous Spatial Pyramid Pooling (ASPP) module [4], and a decoder. The encoder is a pre-trained CNN, such as ResNet[5], that is used to extract features from the input image. The ASPP module increases the receptive field of the network allowing the model to efficiently segment objects at different scales. This is done by applying atrous convolutions with different dilation rates to the feature maps from the encoder that are then concatenated together. DeepLabV3+ also uses an image-level feature, which is a global average pooling of the feature maps from the encoder, to provide an additional context information. This image-level feature is concatenated with the feature maps from the ASPP module before being passed to the decoder. The decoder consists of a simple bilinear upsampling layer that is used to increase the spatial resolution of the feature maps generated by the ASPP module. The upsampled feature maps are then concatenated with those from the encoder backbone, passed through a convolutional layer and followed by an additional upsampling layer that increases the spatial resolution and generates the final segmentation mask. DeepLabV3+ also applies depthwise separable convolution to both the ASPP module and the decoder module, resulting in a faster and stronger encoder-decoder network.

3. Dataset

The dataset we analyze is the one from the DeepGlobe Land Cover Classification challenge[6] announced for the CVPR in 2018. It consists of 803 RGB satellite images focusing on rural areas, 2448x2448 pixels in size with a resolution of 0.5 meters per pixel. Each satellite image is paired with a mask image for land cover annotation with 7 classes encoded using RGB values. They are the following.

- Urban land: Man-made, built up areas with human artifacts (roads and bridges were not annotated).
- Agriculture land: Farms, any regular plantation, cropland, orchards, vineyards, nurseries and ornamental horticulture areas.
- Rangeland: any non-forest, non-farm, green area.
- Forest land: Any land with at least 20% tree crown density plus clear cuts.
- Water: Rivers, oceans, lakes, wetland, ponds.
- Barren land: Mountain, rock, dessert, beach, no vegetation areas.
- Unknown: Clouds and others.

Any istance of a class larger than a roughly 20mx20m was annotated by humans so some small errors are inevitable. Moreover, roads and bridges are not annotated. The class distributions are shown is in Figure 1.

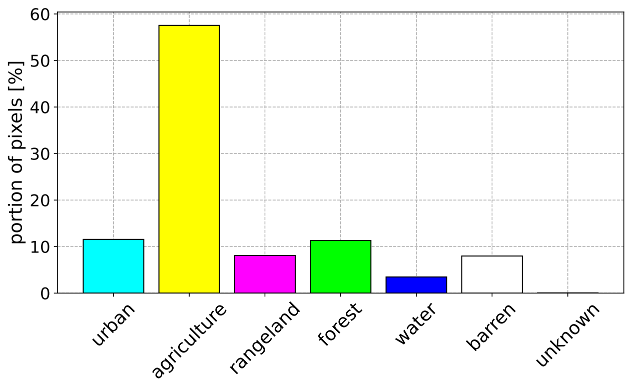

Figure 1. Class distributions in the DeepGlobe land cover classification dataset.

4. Method

We split the dataset into training, validation and test sets, each with 642, 80 and 81 images (corresponding to a split of about 80%/10%/10%). Given that the dataset is quite small, we implement a pre-processing and data augmentation pipeline that takes into account the peculiar features of satellite imagery.

First of all, there is no clear notion of top-bottom and left-right, therefore we randomly apply all the dihedral group D4 transformations, namely rotations of 0, 90, 180, 270 degrees, horizontal and vertical flips, and transpositions. Furthermore, the scale of objects in remote sensing images exhibits high variability, both intra-class and inter-class. One way to tackle this problem is to apply a scaling transfomation with a randomly selected factor, which we chose to be between 0.5 and 1.25. This should allow the model to improve its ability to handle multi-scale objects. Another characteristic is that the color of landscapes on the ground varies with the season. In addition, remote sensing images are synthesized based on light reflection, which depends on many factors, leading to a strong color imbalance. To account for this, we add a color jittering transformation to the pipeline that randomly changes brightness, contrast, and saturation.

In terms of computing resources, we use an NVIDIA Tesla T4 GPU with 16 GB of memory. Since the huge image size can lead to out of memory errors, we decide to use a random crop of 224x224 pixels. Finally, we apply Z-score normalization so that the mean and standard deviation along each channel are 0 and 1, respectively. The only transformations we apply to the validation and test sets are the random crop and the normalization using the values obtained in the training set.

4.1. Baseline: U-Net

The architecture we use as our baseline model is U-Net. We implement it from scratch using Pytorch [7], introducing some variations with respect to the original paper in which it was first proposed. In particular, we include a batch normalization layer after each convolutional layer, before the activation function, and we add zero padding to preserve the size of the feature maps. In the last layer we use the softmax activation function that converts the logits into probabilities of each pixel to belong to a certain class. For more details on the architecture, see the GitHub repository [8]. The loss we use is a multi-class cross entropy and the optimizer is Adam [9] with starting learning rate of 10^{-3} and weight decay parameter equal to 10^{-5}. Note that even if the model processes a small random patch of the image, it must load the full-size image into the GPU first, which forces us to choose a batch size of at most 4 to avoid out of memory errors.

4.2. Proposal: U-Net with patches

The training strategy of the baseline model brings up some challenges. A small batch size increases the amount of randomness in the training process and using random cropping limits the number of pixels the model sees per epoch. To address this issue, we divide the images into non-overlapping 224x224 pixel patches, discarding excess pixels. As suggested in previous research [10], before splitting the images into patches, we also rescale the entire training set using four different scales (0.5, 0.75, 1, 1.25) to account for scale variability. This procedure replaces the random scale and cropping transformations used in the baseline, allowing us to take advantage of a larger batch size of 32 and increase the number of pixels seen by the model at each epoch. In order to reduce the computational time per epoch, we randomly select 60000 out of the 229836 patches in the training set. Furthermore, we also divide the validation and test sets into patches without changing the original scale, and randomly select 8000 samples from each set. The optimizer is again Adam with the same starting learning rate and weight decay parameter, while the loss function is a multi-class cross entropy with the addition of class weights to address the highly unbalanced class distributions. The weights are calculated using median frequency balancing. The class frequency is determined by the number of pixels belonging to that class, divided by the total number of pixels in the images where that class is present. The weight for each class is then defined as the median of all class frequencies divided by its own frequency.

4.3. State-of-the-art: DeepLabV3+ with patches

We compare the performance of our proposed model with that of a state-of-the-art architecture for semantic segmentation, namely DeepLabV3+ which is described in the previous section. We import the DeepLabV3+ model from the SMP library [11] and we choose ResNet-101 [5] as its backbone encoder, pretrained on the ImageNet dataset [12]. To fine-tune the DeepLabV3+ model, we employ the same training procedure as the one used for our model with the patch-wise pre-processing pipeline.

5. Experiments

In order to accurately evaluate the performance of the models, we introduce a set of metrics that are commonly used in semantic segmentation tasks. Two important metrics used in this field are the Mean Intersection over Union (MIoU) and the F1-score. Both of these metrics provide important insights into the performance of a model, and are defined as follows.

- MIoU measures the average intersection of predicted and ground truth regions, divided by their union. It ranges between 0 and 1, with 1 indicating perfect overlap and 0 indicating no overlap:

$$\text{IoU}_i = \frac{tp_i}{tp_i + fp_i + fn_i}$$

$$\text{MIoU} = \frac{1}{k-1}\sum_{i=1}^{k-1} \text{IoU}_i$$

- F1-score is a measure of a test's accuracy which considers both precision and recall. It is the harmonic mean of precision and recall:

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

  where precision and recall are defined as

$$\text{precision} = \frac{1}{k-1}\sum_{i=1}^{k-1} \frac{tp_i}{tp_i + fp_i}$$

$$\text{recall} = \frac{1}{k-1}\sum_{i=1}^{k-1} \frac{tp_i}{tp_i + fn_i}$$

The true positive (tp_i), true negative (tn_i), false negative (fn_i) and false positive(fp_i) values for each class are obtained by summing over all pixels of all images in the test set. The total number of classes is represented by k. By calculating the metrics using these formulas, we are applying an average over the classes, which is known as macro-averaging [13]. Additionally, it's important to note that in our evaluation, we do not take into account the *unknown* class, as suggested by the DeepGlobe challenge itself.

The results, presented in Table 1, indicate that the multi-scale patch-wise pre-processing technique combined with a weighted multi-class cross entropy loss function significantly improves the performance of the baseline model.

Model	U-Net	U-Net with patches	DeepLabV3+ with patches
Loss	1.05	0.39	0.32
MIoU	0.31	0.65	0.66
F1-score	0.47	0.77	0.78
Epochs	175	30	20
Parameters	31M	31M	46M

Table 1. The table shows an overview of the results for the three considered models. The best models are chosen according to the best validation loss achieved during training.

While the state-of-the-art model, DeepLabV3+, achieves the best results, our proposed model reaches comparable performance in both MIoU and F1 metrics. In addition, it is worth noting that the pre-trained architecture of DeepLabV3+ achieves these high levels of performance with a lower number of training epochs.

In Figure 2, we present the trend of IoU per class for our proposed model evaluated in the validation set. It can be observed that all classes show a general improvement in performance over the epochs. However, it is worth noting that the *rangeland* class does not obtain the same performance as the other classes. This poor behavior is not solely due to the frequency of the class in the training set, as it is probably caused by the poor characterization of the class. In fact, the *water* class, which is less present in the dataset, has a better MIoU score. This highlights the importance of accurate class annotation in the training images to achieve better model performance.

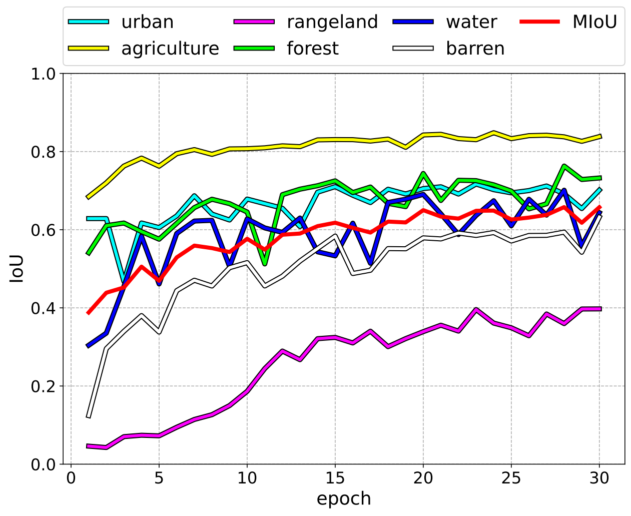

Figure 2. The figure presents the trend of Intersection over Union on the validation set for the different classes as the training goes on.

The confusion matrices for the test set for our U-Net and DeepLabV3+, both with patches, are presented in Figure 3 and Figure 4, respectively. It can be observed that in both cases, when the models make mistakes in classification, they tend to choose the *agriculture* class, which is also the most populated class in the dataset. The DeepLabV3+ model, however, demonstrates considerable improvement in the predictions of the *rangeland* and *unknown* classes.

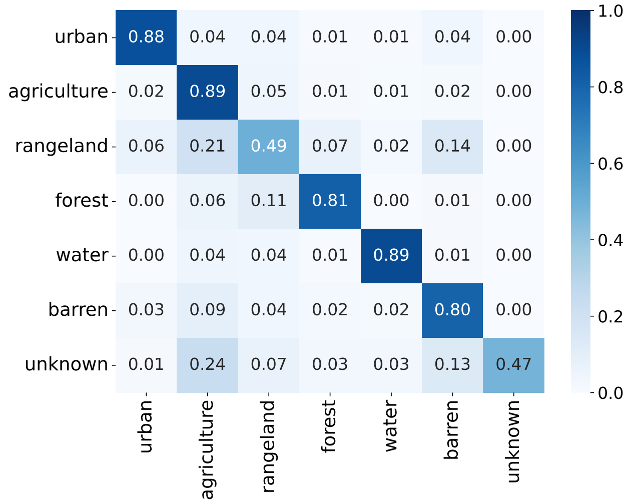

Figure 3. Confusion matrix of U-Net with patches.

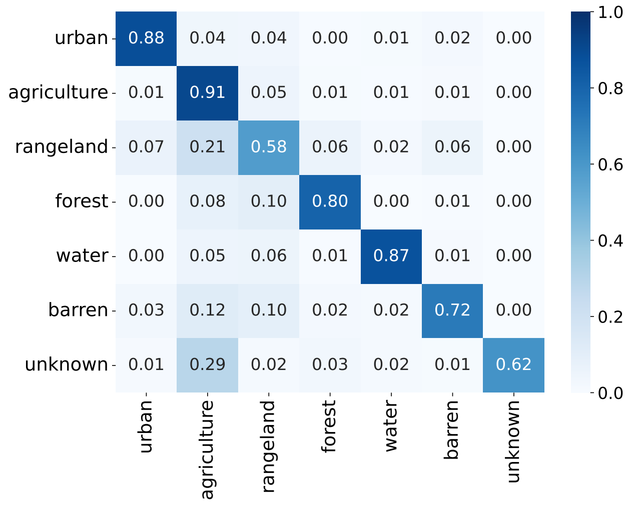

Figure 4. Confusion matrix of DeepLabV3+ with patches.

The predictions on a full-size image (2448x2448 pixels) can be made using two different strategies. The first strategy is to divide the image into smaller patches and make the model generate the semantic segmentation mask for each patch individually. The second strategy is to feed the full image to the model as input. Figure 5 presents the results of the first approach applied to a sample of the test set for both models. Figure 6 illustrates the predictions obtained using the second approach.

We observe that, in general, feeding the model with the full image typically results in predictions that do not suffer from patch patterns. Furthermore, the granularity of the predicted segmentation masks produced by our models appears to be superior to the ground truth. It is worth noting that, during model evaluation, we decided to calculate the metrics on the patches due to computational time and GPU memory constraints. Therefore, the choice of strategy will depend on the available computational resources and the requirements of the specific task.

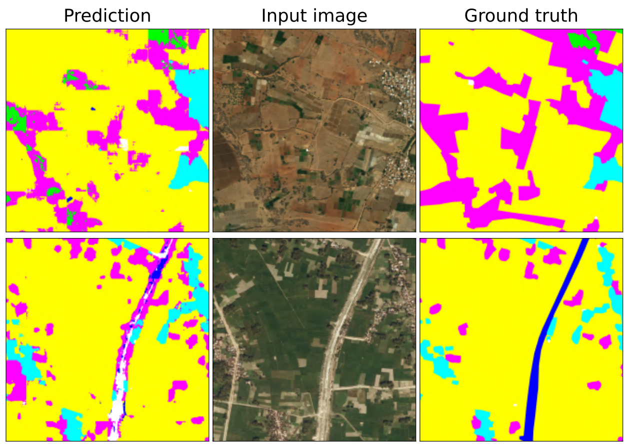

Figure 5. Patch-wise prediction for U-net (top pannel) and DeepLabV3+ (bottom pannel).

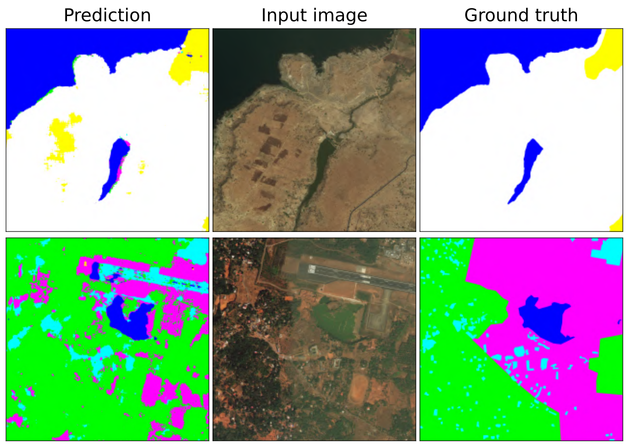

Figure 6. Prediction for U-net (top pannel) and DeepLabV3+ (bottom pannel) using full image size as input.

6. Conclusions

Semantic segmentation of remote sensing imagery can be a challenging task due to the large size and complexity of the dataset. We develop a set of strategies to improve the performance of our semantic segmentation model. First, we implement a pre-processing pipeline that takes images at different scales and split them into squared patches. Secondly, we applied median frequency balancing to weight the multi-class cross entropy loss, reducing the impact of class imbalance in the pixels. These strategies allow our implemented version of U-Net to outperform the baseline model, which does not take these considerations into account. Additionally, by using a greater number of training epochs, we are able to achieve performance comparable to the state-of-the-art model, DeepLabV3+, fine-tuned with our dataset. To further improve the prediction ability of the model, we think it might be useful to implement a post-processing pipeline, such as using Conditional Random Fields. Additionally, better annotation of the ground truth would also improve the performance. Hyperparameter tuning, such as incorporating a learning rate scheduler, could also be explored to achieve better results. In conclusion, we want to highlight that our proposed model outperforms the MIoU score of the baseline reported by the DeepGLobe Land Cover Classification challenge [14] and it is comparable to the winners of the competition.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. DeepGlobe 2018: A challenge to parse the earth through satellite images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, jun 2018.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[8] Github repository of the project code. `https://github.com/carlosgorlongaiatto/VCS_Project`.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[10] Yan Liu, Qirui Ren, Jiahui Geng, Meng Ding, and Jiangyun Li. Efficient patch-wise semantic segmentation for large-scale remote sensing images. *Sensors*, 18(10), 2018.

[11] Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing Management*, 45(4):427–437, 2009.

[14] Deepglobe land cover dataset competion results. `https://competitions.codalab.org/competitions/18468#results`.