Machine Learning Segmentation Case Study: Land Usage and Land Coverage

Leo Richard Hermann Giesen
University of Munster
Munster, North Rhine-Westphalia, Germany
leo.giesen10@uni-muenster.de

Johannes Kauffmann
University of Munster
Munster, North Rhine-Westphalia, Germany
johannes.kauffmann@uni-muenster.de

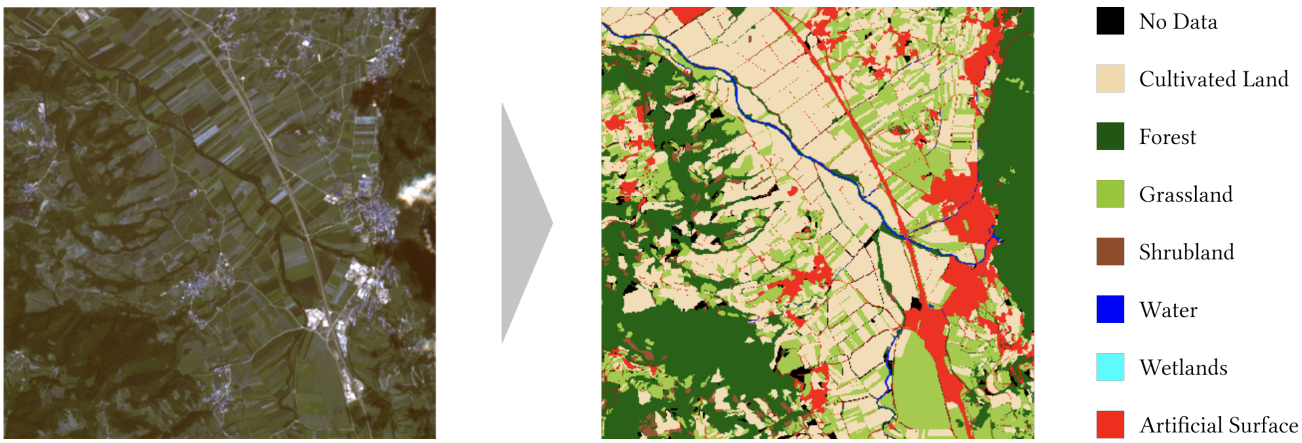

Figure 1: Case Study Semantic Segmentation Task

ABSTRACT

In this case study, a supervised deep learning algorithm is presented with the objective of semantically segmenting a landscape into ten predefined land usage and coverage classes. This is achieved by a sliding window approach with a three-dimensional convolutional neural network resulting in an accuracy of 73%. At the hand of seven steps of machine learning projects, the functionality and characteristics of the algorithm are outlined and how it may be improved upon.

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

machine learning, deep learning, semantic segmentation, convolutional neural networks

ACM Reference Format:
Leo Richard Hermann Giesen and Johannes Kauffmann. 2021. Machine Learning Segmentation Case Study: Land Usage and Land Coverage. In *Proceedings of Deep Learning with Python (Specialization Module'21).* Deep Learning with Python, Munster, Germany, 5 pages.

1 CONTEXTUALIZATION

The Information Systems specialization module 'Deep Learning with Python' held by Prof. Fabian Gieseke and Moritz Seiler, is supplemented by a machine learning case study. Its project work offered a realistic glimpse into machine learning tasks through the practical application of various theoretical concepts presented in the specialization module. However, advanced concepts such as U-Net were applied, because of the limited scope and time of the project.

The machine learning algorithm utilizes a sliding window approach in a three-dimensional convolutional neural network. It is trained on the training set, tested on the public test set, and finally evaluated on the hidden test set. The latter two include larger images for a segmentation of a landscape. The task falls under supervised learning because the algorithm draws on labeled data. More specifically, it belongs to the category of semantic segmentation classes are assigned to pixels of an image [4].

2 METHODOLOGY

The Jupyter Notebook of this project is written in Python and executed in Google Colab, which accessed the training and testing data via a mounted version of Google Drive. Frameworks such as TensorFlow and Keras were used in the machine learning implementation, i.e., they facilitate one-hot encoding, data augmentation for the sequential model and its layers and callbacks. Further libraries include for instance, NumPy, Matplotlib and scikit-learn for supporting high-level mathematical operations, plotting, and defining optimal class weights.

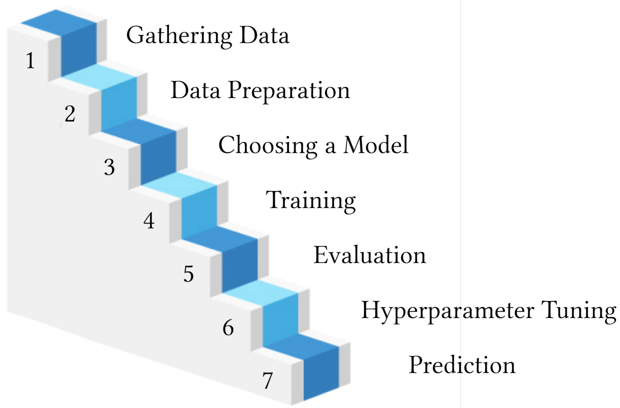

Figure 2: Adjusted from Seven Steps of Machine Learning. Web Illustration by Vaishali Advani, via Great Learning [1]. (https://bit.ly/3C0fBqp).

A machine learning project can be divided into seven steps as visualized in Figure 2, which are explained and subsequently examined in the context of this case study. First, data needs be gathered so that the algorithm can learn from it. Second, the data must be prepared to eliminate any unintended biases, for example the data is shuffled as the data instance order may affect the model's decisions. Third, the best model is determined and chosen using the validation set. Fourth, the chosen model is trained on the training data in order to perform an accurate prediction. Fifth, the model is evaluated regarding its generalization, which expresses its proficiency in application to new data. Sixth, the hyperparameters, which control and steer the learning process and define the amount of regularization, are tuned to further increase the model's accuracy. Seventh, the model is fully developed and applied in practice by performing a prediction on the test set [6].

3 APPLICATION AND RESULTS

3.1 Gathering Data and Data Exploration

These seven steps guide this project and their application in the case study is examined in-depth. The first step of gathering data has already been performed before the project. The data preparation requires an exploration of the data. Hence the shape of the training dataset is inspected. The 'bands' object with a shape of (10000, 12, 33, 33, 6) comprises 10.000 patches, where the 33x33 pixel images are portrayed in red, green, blue and three near-infrared spectroscopic channels resulting in a total of six color channels. Each satellite image is covered over the course of twelve months, which may increase the overall accuracy. The central pixel of each of these images is labeled, which is stored in the 'lulc' data with a shape of (10000,).

A profound understanding of the gathered data was achieved by visualizing the landscapes and taking six color channels and the eleven different class labels into account. The class zero called 'no data' represents a label, which could not be assigned to a specific kind of biome. This could be the case when a cloud covers the landscape. It turns out that class two 'cultivated land' is largely overrepresented with almost 6.000 instances, and classes six 'wetlands', seven 'tundra' and ten 'snow and ice' do not occur at all. This means that there is no training data available for them and consequently they cannot be predicted. The other classes occurred between 58 and 1308 times, which is observed in Figure 3.

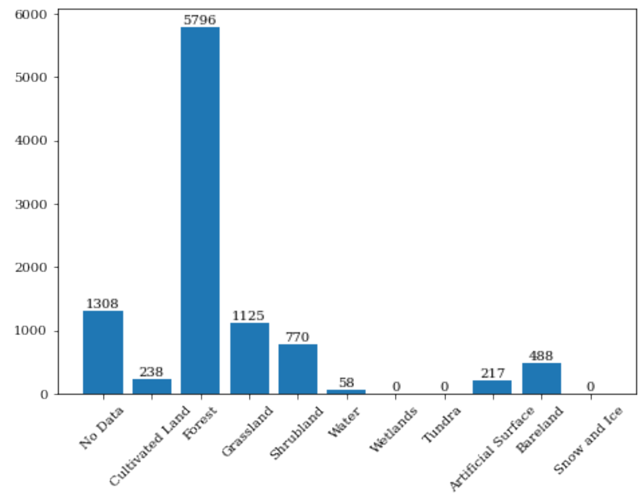

Figure 3: Class Label Frequency of Occurence in Data Set.

3.2 Data Preparation

The imbalance of class occurrence is fatal because the accuracy is not representative of the actual performance. I.e., the algorithm would predict the overrepresented class two 'cultivated land' most of the time and be correct about (5.796/10.000 =) 57,96% of the time. However, this would not be a realistic prediction as other classes are neglected, and the model is not penalized accordingly. To punish the model correctly, class weights are automatically calculated and initialized in a dictionary with a predefined function. The non-present classes can be completely ignored in the prediction or hold a class weight of zero.

Before working with the data set, it is imperative to split the data set into a training (64%), validation (16%) and test set (20%), because if the same instances of the training set are also used in testing, the model predict correctly without demonstrating its learning. After that, the data is converted to a TensorFlow dataset object to make the runtime TPU (Tensor Processing Unit) of Google Colab work more efficiently with the data.

The datasets are subsequently prepared for the model fitting: First, the class labels are one-hot encoded in order to use the categorical cross entropy as a loss function. After that, the TensorFlow prepare function was used to shuffle, augment, batch and prefetch all data [8]. As mentioned above, shuffling the data is necessary for the model to avoid the recognition of unwanted biases. The data augmentation TensorFlow preprocessing layers randomly flip the image horizontally and vertically and rotate it to provide a higher quantity of data for the model to learn from.

3.3 Choosing a Model, Training and Evaluation

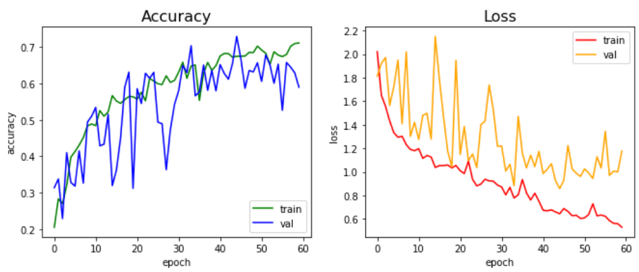

Figure 4: Accuracy and Loss of Machine Learning Algorithm.

Various architectures such as [12] were considered and inspired the final model architecture, which was designed from scratch to provide high flexibility. A 3D convolutional neural network is required because of the additional dimension of months in the data set. As displayed in Figure 4, the lower layers consist of three main stacks of two 3D convolutional layers followed by a 3D max pooling and batch normalization layer. The higher layers comprise a flattening and three dense layers in combination with dropouts.

Various architectures such keras were considered and inspired the final model architecture, which was designed from scratch to provide high flexibility. A three-dimensional convolutional neural network is required because of the additional dimension of months in the data set. As displayed in Figure 4, the lower layers consist of three main stacks of two 3D convolutional layers followed by a 3D max pooling and batch normalization layer. The higher layers comprise a flattening and three dense layers in combination with dropouts.

The compiled model uses categorical cross entropy as a loss function and Adam as an optimizer. Three callbacks are defined for model training and fitting. The first one is a learning rate scheduler managing and reducing the learning rate by multiplying it with $e^{-0,1}$ after the 10^{th} epoch [9]. The second is an early stopping callback, which prevents overfitting by stopping the training process when the validation accuracy is not improving after a patience of 15 epochs [4]. The third callback is a checkpoint, which automatically saves the best model in the h5-format. This is a hierarchical data format, which stores and organizes large amounts of data in a compact form [10].

The model is trained for a maximum of 100 epochs with a batch size of 32. The training stopped at the 68^{th} epoch due to early stopping. After training, the fitted model is evaluated using the test set. The accuracy results in 72,44% and the loss is 54,75%. The evaluation discloses a significant fluctuation of the accuracy and loss as displayed in Figure 5. This may originate from consecutive cloudy images, which may negatively impact the model's weights and lower accuracy.

3.4 Hyperparameter Tuning and Prediction

The hyperparameter tuning step is skipped as the focus of the case study lied on the general approach of the given semantic segmentation task and not on achieving the highest accuracy possible. However, it is advised to perform hyperparameter tuning if the machine learning algorithm is planned to be deployed. The final prediction is performed with a sliding window approach on the public and hidden test set, whose images measure 500x500 and 1500x1500 pixels in size. This approach predicts the central pixel of a 33x33 pixel block around it, exactly as in the training set. The sliding aspect is added, so that this block for the prediction slides across the image to predict every pixel of every row and column (see Figure 6).

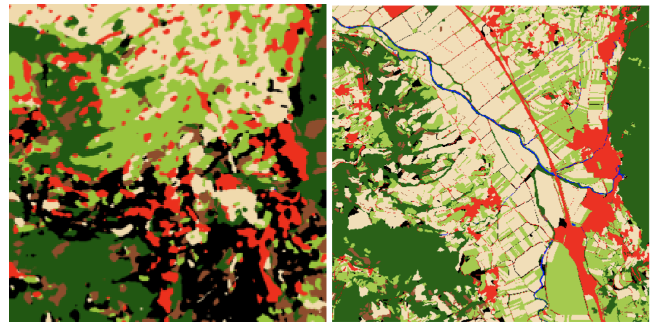

Figure 5: Comparison of Prediction and Solution in the Public Test Set.

The problem arises that the edge pixels are not surrounded by pixels on every side, which is required for the prediction box of 33x33 pixels. This is resolved by zero padding, which places a $((33-1)/2 =)$ 16-pixel margin of zero values ('no data') around the whole image. As a result, the sliding window approach produces an output prediction with the same size as the input. The output 2D prediction array is visualized using an RGB-color-map, where each label has a corresponding color.

4 DISCUSSION

4.1 Data Preparation

There are various options and alternative approaches, which may be considered in each of the previously mentioned seven steps of the machine learning process. For instance, three classes cannot be predicted because no data for them exists. This could be solved by gathering data from further satellite images. The class instance imbalance was solved with initializing corresponding class weights, but it could have also been achieved by duplicating data instances to account for the difference in occurrence. However, the former is simpler to implement and faster to train because of fewer training instances.

The inspection of data also revealed that clouds pose a problem as they complicate the prediction. The color channel values are scaled between zero and one to foster the model's learning from the data. Though, there are satellite images with color channel values greater than one, which might the case because of the cloud's high reflection of the sun light. In these cases, normal scaling would scale down other color channel values so much that their information is lost, which would prevent the model from learning. These data instances could be manually removed using data cleaning, which could potentially increase the model's accuracy. However, this is not tested because this step is very time consuming, and it would decrease the quantity of data.

The data augmentation is performed with four TensorFlow preprocessing layers, which are part of the model, which results in a longer training duration, but the model also benefits from GPU acceleration because the data augmentation layers run synchronously with the rest of the model's layers.

4.2 Choosing a Model and Evaluation

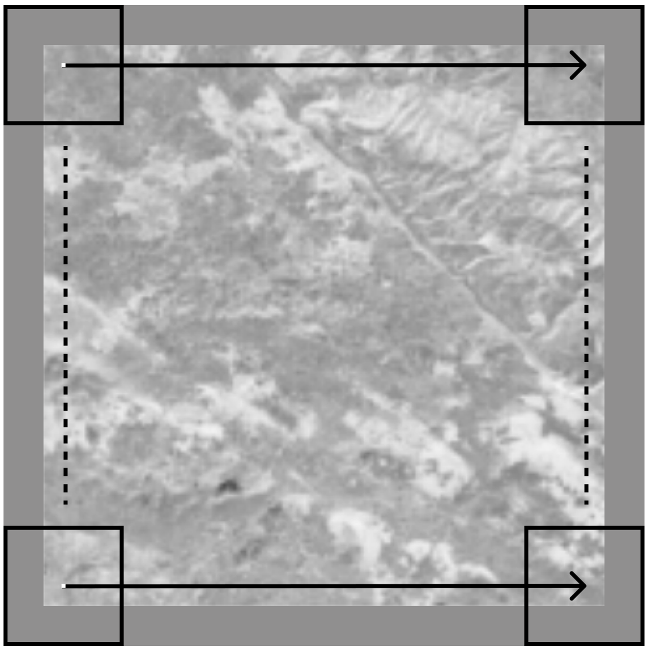

Figure 6: Sliding Window Approach.
Illustration Abstracted From Pixel-Specific Placement.

Alternatively, transfer learning could be utilized, which improves the model's performance using a pretrained model from a similar context. This is because the model's architecture is improved, which leads the model to start off with a much higher accuracy. For instance, the neuron's weights are initialized closer to their optimum and consequently the error is smaller. Though, the unusual input shape of the segmentation task hinders transfer learning as no comparable 3D convolutional neural networks could be found online during our research. Nevertheless, high performant model architectures such as ResNet or U-Net with no pretrained weights might still improve the algorithm.

Initially, the model only comprised few layers and relatively few parameters. Further model testing revealed that more convolutional 3D layers make the model more powerful and increase accuracy, because the previous model version with just under 700.000 parameters underfitted and did not have the capacity to capture the landscape's complexity.

As mentioned above, the categorical cross entropy is used as a loss function. Instead, the model could have also used sparse categorical cross entropy as a loss function, which produces a category index of the most likely matching *class*. In comparison, categorical cross entropy displays each *class probability* in a one-hot encoding vector. This leads to two major advantages of sparse categorical cross entropy as it saves memory space and may reduce the training time. The probability vector is beneficial in situations, where the classes are not mutually exclusive, and one instance may have multiple classes. However, this is not the case in this project, thus sparse categorical cross entropy is advised to be used.

In alternative to accuracy, other types of performance measurement could have been selected, which are not affected by the imbalance of class instances. For instance, a confusion matrix or a receiver operating characteristic (ROC) curve are better in evaluating the quality of the model in this situation without applying class weights.

4.3 Prediction

The naive sliding window approach results in a long overall image prediction because each pixel is predicted separately, and it takes 42,4 ms to predict one. The prediction process could be speeded up, for instance a whole row could be predicted in one iteration by passing multiple 33x33 pixel cutouts. In that case, NumPy arrays slow down the prediction process because the whole array needs to be loaded, which is not the case for a python list. Therefore, the prediction of a row could be appended to the final prediction.

Apart from the naive sliding window approach, a convolutional sliding window approach could be utilized to lower the prediction duration. This is the case, because it is a one-shot approach, which does not iterate over the rows and columns in two for-loops [2]. Moreover, one could make use of innovations such as FCN, U-Net, Mask R-CNN, which are significantly more performant than the current approach in this project [3, 5, 7].

Moreover, replication or reflection padding could be applied instead of zero padding, because they offer more significant information to the prediction of the outer pixels by mirroring the outer pixels into the padding [11].

5 CONCLUSION

In conclusion, the objective of semantic segmentation of landscapes from satellite images is achieved with a satisfactory accuracy. However, there are still options to be tested out to improve the performance. Especially, alternatives to the model's architecture and possible frameworks should be shed a light on.

ACKNOWLEDGMENTS

Regular exchange of ideas and possible solutions with Marten Jostmann and Gerrit Kruse.

REFERENCES

[1] Vaishali Advani. 2021. What is Machine Learning? How Machine Learning Works and Future of It? Retrieved 2021-07-27 from https://www.mygreatlearning.com/blog/what-is-machine-learning/

[2] Rohan Arora. 2020. Convolutional Implementation of the Sliding Window Algorithm. Retrieved 2021-08-01 from https://medium.com/ai-quest/convolutional-implementation-of-the-sliding-window-algorithm-db93a49f99a0

[3] Priya Dwivedi. 2019. Semantic Segmentation — Popular Architectures. Retrieved 2021-07-27 from https://towardsdatascience.com/semantic-segmentation-popular-architectures-dff0a75f39d0

[4] Aurélien Géron. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2 ed.). O'Reilly Media, Inc. 30–34, 203–204 pages.

[5] James Le. 2021. How to Do Semantic Segmentation Using Deep Learning. Retrieved 2021-07-27 from https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/

[6] Juan Cruz Martinez. 2020. Seven Steps of Machine Learning. Retrieved 2021-07-28 from https://livecodestream.dev/post/7-steps-of-machine-learning/

[7] Anil Chandra Naidu Matcha. 2021. A 2021 Guide to Semantic Segmentation. Retrieved 2021-07-27 from https://nanonets.com/blog/semantic-image-segmentation-2020/

[8] TensorFlow. 2021. Data Augmentation. Retrieved 2021-07-31 from https://www.tensorflow.org/tutorials/images/data_augmentation

[9] TensorFlow. 2021. Early Stopping. Retrieved 2021-07-31 from https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

[10] The HDF Group. 2021. The HDF5 Library and File Format. Retrieved 2021-07-31 from https://www.hdfgroup.org/solutions/hdf5

[11] Chris Versloot. 2020. Using Constant Padding, Reflection Padding and Replication Padding with TensorFlow and Keras. Retrieved 2021-08-01 from https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/#what-is-replication-padding

[12] Hasib Zunair. 2020. 3D Image Classification From CT Scans. Retrieved 2021-08-01 from https://keras.io/examples/vision/3D_image_classification/