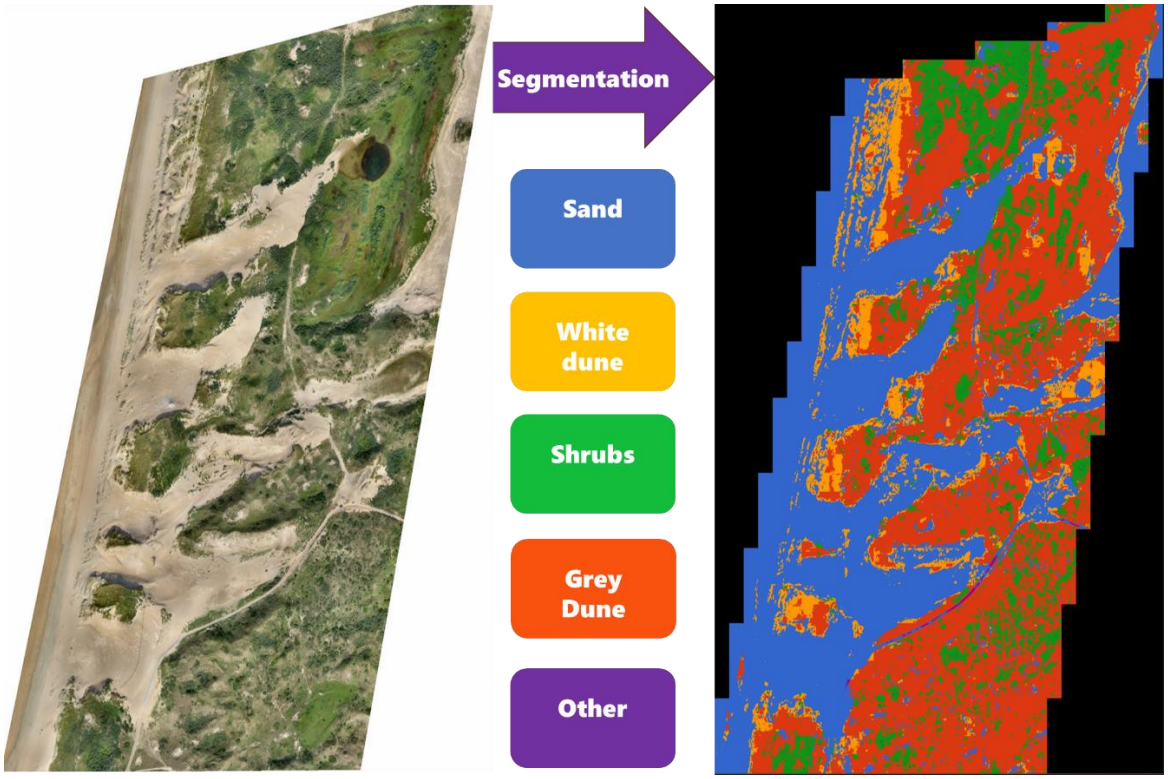
Habitat mapping from high-resolution UAV orthomosaics using Convolutional Neural Networks

Exploring Hyperparameter Tuning for optimization

Master Thesis



Name student: Tino Ouwerkerk
Student number: 1301233
Name of first Supervisor: Prof. dr. Gerben Ruessink
Name of secondary Supervisor: Dr. Timothy Price
Student number: 1301233
Date: 4 July 2023
Program: Master Applied Data Science, Utrecht University

Utrecht University

Contents

List of figures

List of tables

Summary

In this MSc research project, the feasibility of automatically deriving habitat maps from UAV derived orthomosaics using deep learning is explored. Habitat maps are crucial for preserving biodiversity in nature areas, and traditional methods of creating them are costly and time-consuming. Creating habitat maps typically involves high-quality fieldwork, which is expensive and leads to infrequent updates, even though dunes can be highly dynamic. This project focuses on the Dutch National Park Zuid-Kennemerland and addresses the problem of automating the creation of habitat maps by exploring the use of deep learning algorithms and increasing the performance by using hyperparameter tuning. In this research we show that using an annotation tool called doodler and a ResUnet with class weights from segmentation gym performs the most optimal. The habitats demonstrates that the ResUnet model outperforms the segformer and UNet models in classifying sand, grey dunes, white dunes, and shrubs in coastal dune images. However, there are still difficulties in differentiating between grey dunes and white dunes. The use of class weights during training improves overall predictions and reduces tiling issues in the fully predicted image. These findings contribute to the understanding of effective image segmentation techniques for habitat mapping in coastal dune environments in the Netherlands, emphasizing the importance of model selection and the use of class weights for improved performance. We anticipate this thesis to be a starting point in the automation of habitat mapping for coastal dunes. Some hyperparameters like image augmentation can still be tested. Also a popular trend in deep learning for images is transfer learning which might achieve even better results. Furthermore accurately automating this process can save considerate time for researchers who create habitat maps for coastal dunes and makes the tracking of biodiversity within these areas more consistent.

1. Introduction

1.1 Motivation, context and limitations

In this MSc Research project we explore the feasibility of automatically deriving habitat maps from high-resolution RGB unmanned aerial vehicle (UAV) orthomosaics (aerial image of the Earth's surface) using deep learning. Habitat maps show the geographical locations of ecosystem types with characteristic geographic, abiotic and biotic features. They are standard products used in the protection of nature areas to preserve biodiversity. The habitats that are considered in this research are sand, grey dunes, white dunes and shrubs. White dunes mainly contain marram grass with sand and can be found closer to the sea in the foredunes. Grey dunes also contain marram grass and other low vegetation. Shrubs mainly contain bushes and trees and are mostly within grey dunes.

In 1990 the Dutch government decided to maintain the position of the coastline by artificial sand nourishment, because this was not sustainable for biodiversity in the long term and was not needed for protection anymore. Due to this, intensive management of the foredunes was no longer required. Consequently, natural processes in the foredunes revived, and these changes need to be tracked (Arens et al., 2013). This is tracked by using repeat topographic survey data to examine the geomorphic response of a coastal dune system. It is expected that the revived natural progress and enhanced geomorphological dynamic alters vegetation patterns and hence requires frequent updates to habitat maps. The creation of habitat maps is often based on high quality fieldwork, which is costly and means that maps are not often updated (Kooij, 2022).

This research will focus on the Dutch National Park Zuid-Kennemerland an area where five notches were excavated in 2012–2013 within an 850-m stretch of the 20-m high established foredune. Over the years a lot changed and future monitoring is required to determine for how long the notches will stimulate aeolian dynamics and if (and when) vegetation eventually starts to regrow and enforces the degeneration of the notches (Ruessink et al., 2018).

To automatically create habitat maps deep learning is applied, which requires labels to train. This done by using a software called dash doodler. This software is a "Human-In-The-Loop" machine learning tool for partially supervised image segmentation (Buscombe, 2022). After this is done the images are preprocessed and used in deep learning segmentation models to recognize the annotated patterns using the software Segmentation Gym (Buscombe & Goldstein, 2022). The performance of the partially supervised image segmentation for annotating will be explored. Also the different models provided by Segmentation Gym will be researched and evaluated using different hyperparameters to optimize performance for our use case.

1.2 Literature overview

The main goal of this project is to evaluate the segmentation algorithms within Segmentation Gym on multiclass images of coastal dunes in the Netherlands using hyperparameter tuning. To achieve this goal the following steps need to be considered.

1. Impact of using an annotation tool like doodler on the accuracy and efficiency of annotating data for image segmentation to create a habitat map.
2. Application of image segmentation using Segmentation Gym on multiclass images of coastal dunes in the Dutch National Park Zuid-Kennemerland.
3. Tuning of hyperparameters to enhance the model's performance in image segmentation.

Image segmentation is a fundamental task in computer vision, aiming to divide an image into different regions. The accurate segmentation of multiclass images of coastal dunes is of particular interest due to its significance in environmental monitoring, land cover classification, and habitat mapping. This literature overview presents key studies related to the evaluation of segmentation algorithms within the Segmentation Gym framework for multiclass image segmentation of coastal dunes in the Netherlands, along with the exploration of effective algorithms and hyperparameters. The following steps, including the impact of annotation tools on data annotation, application of image segmentation using Segmentation Gym, and hyperparameter tuning for improved performance, will be considered.

1.2.1 Impact of dash doodler

Annotation tools play a vital role in the data annotation process, affecting both the accuracy and efficiency of annotating data for image segmentation (Buscombe et al., 2022). Labelling by hand is a time-consuming and error-prone process for uneducated labelers, which would hinder the process of the thesis. Doodler is a Semi-Automated approach for efficiently labelling images for an image segmentation task for Geoscience Applications. Doodler works by having a human ‘doodle’ lines on the parts of an image that represents a class. These classes are differentiated by color and after these lines have been drawn, doodler will fill in the rest by using deep learning algorithms. Therefore, using doodler can save a lot of time and errors. Note that in this research two individuals will label the images, which has been demonstrated by (Buscombe et al., 2022) to result in accurate and precise labels. However it still has to be verified that the concepts are clear to both individuals and that they will label in a similar fashion with the same accuracy and precision.

1.2.2 Segmentation gym

The Segmentation Gym framework, developed by (Buscombe & Goldstein, 2022) provides a comprehensive platform for evaluating segmentation algorithms. It enables researchers to annotate data collaboratively and benchmark various segmentation methods efficiently. The framework allows for the application of segmentation algorithms on multiclass images of coastal dunes, aiding in the delineation of distinct land cover classes, such as vegetation, sand and water bodies. Segmentation Gym includes a family of Unet models (Ronneberger et al., 2015) and a segformer model (Xie et al., 2021). It offers a complete pipeline of preprocessing, model training and evaluation metrics for image segmentation. The benefits of this software according to Buscombe & Goldstein (2022) are the creation of reproducible data sets and models. Also its ability for model experimentation by hyperparameter tuning through changing the config makes it easier to create a suitable model. Furthermore, the software addresses the challenge of hyperparameter tuning, which is a crucial aspect of training deep learning models.

1.2.3 Hyperparameter tuning

Hyperparameters are parameters that define the behavior of the model, and tuning them is essential for achieving the most optimal performance. However, determining values for tuning these parameters can be challenging, as different models may have varying sets of hyperparameters, and even shared hyperparameters may require different ranges or values.

Segmentation Gym assists in addressing these challenges by providing a platform for model experimentation and hyperparameter tuning. Users can modify the configuration settings, allowing for the exploration of different combinations of hyperparameters. This functionality not only makes the process easier but also helps researchers in identifying hyperparameter values for their specific segmentation tasks.

Techniques like grid search, random search, Bayesian optimization, Asynchronous Successive Halving Algorithm (ASHA) have been widely deployed to find optimal hyperparameter settings. Li, et al., (2020) conducted an extensive study on hyperparameter tuning for image segmentation, demonstrating the effectiveness of different hyperparameter tuning techniques on neural networks. However, in this research we will not focus on these algorithms, since there is a restriction in time and software. There is a restriction in software, because segmentation gym does not provide this option.

Within Segmentation Gym there is the possibility to tune the hyperparameters of a model. This can be done by using the config file and changing the settings. So this research will focus on the most influential parameters, these being the models, learning rate and batch size according to (Buscombe et al., 2022). Also the kernel size of the convolutional layers will be considered since it can increase the capabilities of the models to extract more complex features.

1.2.4 Conclusion

In conclusion, this literature overview focused on the evaluation of segmentation algorithms within the Segmentation Gym framework for multiclass image segmentation of coastal dunes in the Netherlands. The main goal of the research is to explore effective algorithms and hyperparameters for accurate segmentation. The key findings and considerations from the literature review are summarized as follows.

Doodler, a semi-automated approach for annotating images, shows potential in improving the accuracy and efficiency of data annotation for image segmentation compared to manual labelling. By allowing human experts to provide initial annotations and leveraging deep learning algorithms for filling in the remaining regions, Doodler reduces the time and errors associated with manual labeling. It has been demonstrated that multiple users using Doodler have achieved the same label accuracy and precision. However, it is essential to ensure that the users have a clear understanding of the annotation concepts and are consistent in terms of the classes to be annotated.

The Segmentation Gym framework provides a comprehensive platform for evaluating segmentation algorithms. It enables collaborative data annotation and facilitates the benchmarking of various segmentation methods. Within Segmentation Gym, the Unet models and Segformer model are available for multiclass image segmentation. Its benefits include reproducibility of datasets and models, as well as the ease of model experimentation through hyperparameter tuning.

For this research, the focus will be on the choice of models and the most influential parameters i.e., model, learning rate, batch size, and kernel size for the convolutional layers.

1.3 Research question

Main Question: How do different segmentation algorithms in Segmentation Gym perform on multiclass images of coastal dunes in the Dutch National Park Zuid-Kennemerland to create habitat maps, and which algorithms and hyperparameters are most effective?

Sub questions:

1. How does using an annotation tool like Doodler affect the accuracy and efficiency of annotating data for image segmentation?
2. What model is the most optimal for image segmentation in order to improve the accuracy on multiclass images of coastal dunes?
3. What hyperparameters will improve the model on performing image segmentation?

2. Data

In this part the data selection and preprocessing is discussed. The labels for these habitat maps were annotated by using dashdoodler (Buscombe, 2022).

2.1 Selected data exploration results

In this thesis all training validation and test data is based on the Orthomosaic NWNKern_20180917 (autumn 2018) data shown in Figure 1 available from Ruessink ,(2023). The image itself is detailed with a resolution of 1 pixel representing a square are of 0.05m by 0.05m in real life. This kind of resolution is reached by using an UAV that takes many pictures of the whole area (Ruessink, 2023). All these overlapping images are combined to create the detailed image shown in Figure 1. The image covers an area of 900*1200 meters.

The habitats that will be segmented from this image are sand, grey dunes, white dunes, shrubs and the other class. White dunes mainly contain marram grass with a bit of sand in between. Grey dunes however can also contain some marram grass, but also a lot of other vegetation and is a bit more dense. Shrubs in autumn can be distinguished by their darker green color and are mainly bushes. The other class is everything besides the other four classes which are mainly water, humans and equipment. The image mainly contains sand and grey dunes and a few white dunes and shrubs. This image was chosen since it contains multiple classes which are easy to differentiate for annotation and segmentation step. This is due to the difference in weather and seasonality which influences how the vegetation looks like. This makes it easier to differentiate between grey dunes, white dunes and shrubs.

Figure 1 Orthomosaic NWNKern_20180917 18000x24000 pixels (Ruessink, 2023)

Before annotating the image will be divided into tiles (smaller images) of 500 by 500 pixels representing 20 by 20 meters in the real world, since the image is unfeasibly big (18,000 by 24,000 pixels; 20m by 20m) for both doodler and the segmentation models. An example of such a tile is shown in Figure 2. In this image the four classes white dune, grey dune, shrubs and sand is shown.

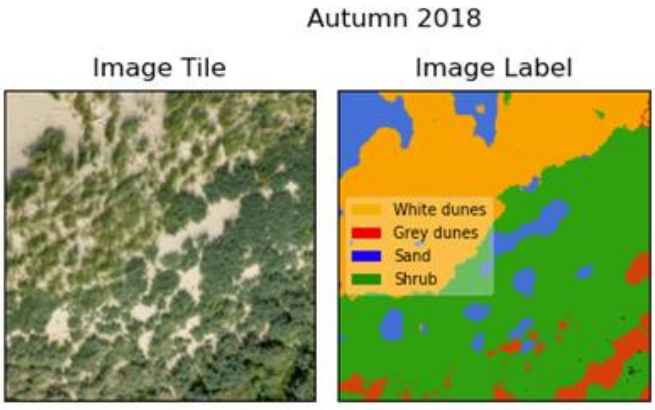


Figure 2 Annotation example

For training the choice was made to use tiles that contain multiple classes in them mainly including white dunes, grey dunes and shrubs and sand. The reason for this is because after training a model to distinguish only 2 classes, sand and vegetation, the model performed quite well. Therefore it was not needed to provide a lot of data of only sand tiles.

From these tiles a habitat map still needs to be created, this is shown in Figure 3. This is done by using the image segmentation model on every tile. After this is done each tile be put together again by using the original position of the data. This will be extracted from the .tif files that was saved during creation of these tiles. An issue that can occur by using tiles is a habitat map with lines and squares. This will become more clear in the results and will be a measure on the models performance.

Figure 3 Flowchart creation of habitat map

2.2 Data annotation

The tiles are annotated using dash doodler, a human in the loop machine learning tool which speeds up the annotation of images for segmentation tasks. In Figure 4 It is illustrated how doodler helps with simplifying the annotation of images. In this research the images will be segmented into five possible classes, namely sand, grey dune, shrubs, white dune and the other class. The other class includes everything that is not covered by the other class such as water, humans and equipment for the drones.

Figure 4 Annotating using dash doodler (Buscombe et al., 2022)

For this thesis two individuals annotated the data. To ensure that both individuals would understand the concepts and characteristics of each class label, a testing phase for annotation was performed. The consistency between the two individuals annotations was validated by using the Cohen Kappa Score. Cohen's kappa, is a statistical measure of annotator agreement. It assesses the agreement between two or more annotators when assigning labels to pixels in this case. Cohen's kappa takes into account both the observed agreement and the agreement expected by chance. The kappa coefficient ranges from -1 to 1, where a value of 1 indicates perfect agreement between raters, 0 equal to chance and less than 0 indicates an agreement worse than chance.

At first a round of 20 annotations got labelled to ensure that both individuals understood the concepts behind each class. This resulted in almost no confliction between distinguishing the classes sand, other and shrubs. However the classes white dune and grey dune were harder to distinguish after comparing the variability between our annotations. This could be explained by the similarities in the vegetation of grey and white, namely marram grass. Therefore another subset of images was selected specifically with both white and grey dunes. After annotating, these were compared and the differences were highlighted. After discussing these images with the supervisor another subset was made including the previous annotations to finally reach a valid comparative result. Eventually a Kappa coefficient of 0.84 was reached, which is sufficient for further labelling as individuals

2.3 Data preparation for analysis including motivation (integration, missing data analysis, etc.)

The dataset was split into subsets for training, validation and testing. For training and validation 102 tiles were used, which was extended to 660 images using image augmentation, this is a method that changes the data using standard operations such as rotation. The data was split fifty-fifty into training and validation. For the test set 15 tiles were selected that have not been in the training/validation set to evaluate the model.

The first preprocessing step is standardizing the images by subtracting the mean and dividing by the standard deviation. This step is important for improving model performance when using images that have different distributions from the training data (Buscombe & Goldstein, 2022). Standardizing the images reduces the impact of outlier values and helps the model transfer its learned knowledge to new images more effectively.

Secondly, both images and labels were resized to the target dimensions of 512 by 512 pixels, and also zero-padded if necessary depending on the kernel size of the convolutional layer. Zero padding is a technique used in convolutional operations where zeros are added to the borders of an input image or feature map. Its goal is to preserve spatial dimensions and keep information flow during the convolution.

Finally, the training data was augmented using standard operations, such as randomly rotating images by 0, 90, 180, or 270 degrees, width shift, height shift, zoom, vertical flip, and mirroring. The main purpose of data augmentation is providing a wider variety of image variations to train the segmentation models. This helps the model with generalization and handling different viewpoints, lighting conditions, and object orientations. Other purposes are to reduce overfitting due to a small data set and create more artificial data without redundancy to train the model. There is little to no redundancy because all oversampled augmented images are unique and randomly augmented (Buscombe & Goldstein, 2022). All specific augmentation settings can be found in the appendix Config files.

After these preprocessing steps the data was checked. An error was found within dash doodler and Segmentation Gym that causes the model to create black pixels. The cause of this is unknown and it was not possible to change these and use that data for the model. To increase performance and create better habitat maps after model predictions, the black pixels were converted to the corresponding color/class it was surrounded by. If these were all black pixels as well then the pixel was changed to the class other. An example of this conversion is shown in Figure 5.

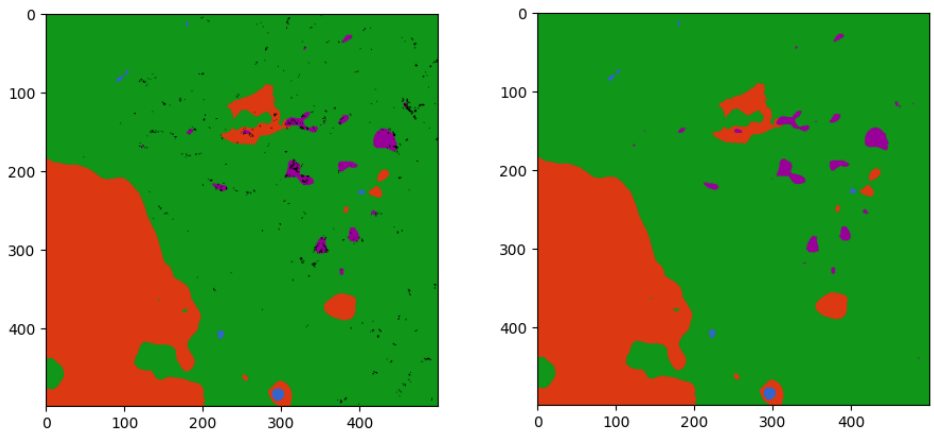


Figure 5 Conversion of black pixels to corresponding colour surrounding the pixel

After preprocessing the distribution of the training and validation data is shown in Figure 6. The distribution of the test set is shown in Figure 7. The only big difference between the training/validation and test is the distribution of sand. The set contains less pixels of sand, because this class should be easier to classify and want to test the model more on distinguishing grey dunes, white dunes and sand. It is impossible to know what the exact distribution of the original image is, but it mainly contains sand and grey dunes. An estimation was made based on the full image. The distribution of Figure 6 comes close to the original distribution. However, the sand is a bit underrepresented, but this class should be easier to classify because the RGB values should be very different from all the other classes which is not the case between the four other classes. Also since there is much grey dunes in the training/validation set, also known as a majority class, it is expected to perform well on it. White dunes and shrubs are in this case a minority class, and will make it harder to get a good performance.

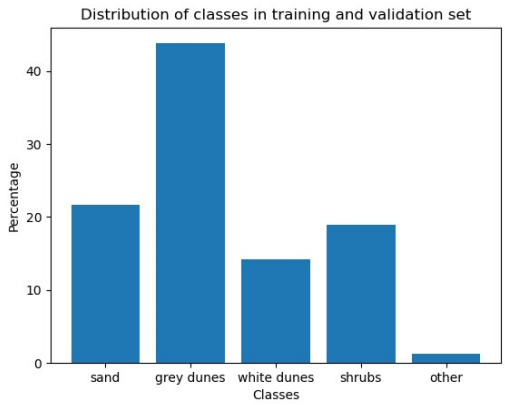


Figure 6 Distribution of training and validation [22%, 44%, 14%, 19%, 1%]

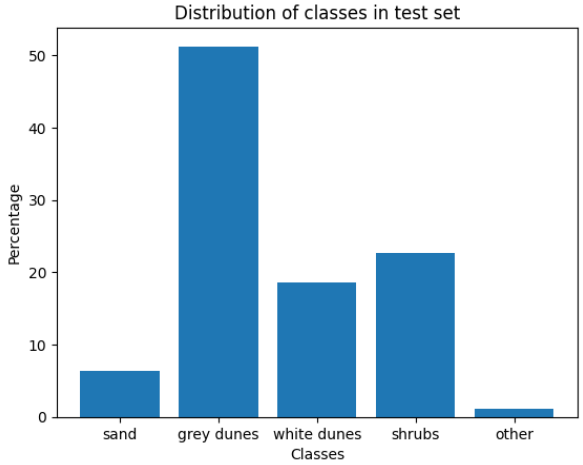


Figure 7 Distribution of test set [6%, 51%, 19%, 23% , 1%]

2.4 Ethical and legal considerations of the data

The data used in this thesis is open source and provided by the supervisor. The images sometimes also include individuals. However, the individuals in these images are not recognizable, because they are top down images and since the images were made during autumn only the individuals flying the drones to create the orthomosaics are visible in this data.

3. Methods

3.1 Translation of the research question to a data science question

Data science question: "What is the comparative performance of different segmentation models in Segmentation Gym when applied to multiclass images of coastal dunes in the Dutch National Park Zuid-Kennemerland to create habitat maps, and which models and hyperparameters performs the best?"

The question clearly specifies the objective of the analysis: evaluating the performance of different segmentation models on multiclass images of coastal dunes. The question also highlights the importance of assessing both models and hyperparameters, as the effectiveness of the segmentation process depends not only on the choice of models but also on the specific hyperparameters used.

This question stimulates a data-driven evaluation of different techniques in the context of coastal dune segmentation by comparing the performance of various models and hyperparameters. It allows for a comparison of segmentation results, enabling researchers to identify the most effective approaches for accurately segmenting coastal dunes into habitat maps. The outcomes of this analysis could provide valuable insights and guide the selection of optimal algorithms and hyperparameters for future segmentation tasks in similar geographic locations.

3.2 Motivated selection of method(s) for analysis

Segmentation gym provides four default hyperparameter settings that includes three different models. The first four models are based on these config files and the best model/config file will be used to tune the hyperparameters. The default config files are called ResUnet1, ResUnet2, Segformer and Unet. The models that will be used are Unet, ResUnet and Segformer, that means there are two Resunet models in the four default settings. The only difference is that the ResUnet 2 has a lower kernel size to extract more local features.

3.2.1 Unet

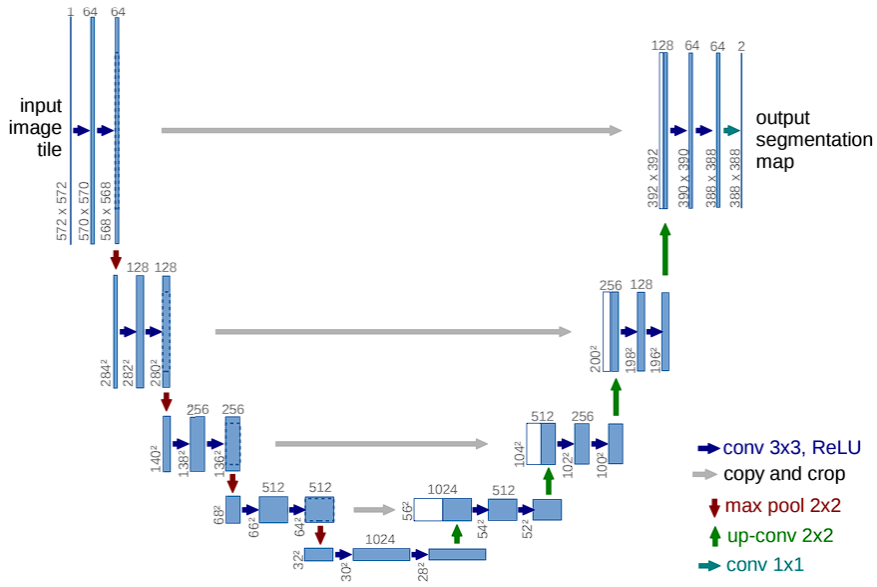


Figure 8 U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. The grey arrows indicate the skip connections. The left part of the U shape is the encoder and the right part is the decoder (Ronneberger et al., 2015).

Unet shown in Figure 8, is a convolutional neural network (CNN) architecture designed for image segmentation tasks (Ronneberger et al., 2015). It consists of an encoder-decoder structure with skip connections. The encoder part performs down-sampling operations (convolution) to capture high-level features, while the decoder part performs up-convolution to generate a segmentation mask. The skip connections connect the corresponding feature maps between the encoder and decoder, aiding in preserving spatial information. In the layer UNet model uses a 1x1 convolutional layer with a sigmoid activation function at the end of the decoder path. This produces a pixel-wise prediction mask with values between 0 and 1 for each class, indicating the probability of each pixel belonging to the target class. The output map has the same spatial dimensions as the input image, providing a segmentation mask for each pixel. Overall, the UNet strength lies in its ability to utilize skip connections and capture local and global context features. By combining these two features it creates accurate segmentation masks (Ronneberger et al., 2015).

3.2.2 ResUNet

ResUNet is an extension of the UNet architecture that utilizes residual connections (Diakogiannisa et al., 2019). In ResUNet, residual connections are added to the encoder and decoder blocks of the UNet, allowing the network to learn residual mappings rather than complete mappings. Residual connections help solve the vanishing gradient problem during training. The gradient provides essential information for optimization algorithms to iteratively update the model's parameters. Vanishing and exploding gradients hinder the effective training of deep neural networks, limiting their ability to learn complex representations and leading to poor performance. These problems are particularly challenging in deep learning, where the gradients have to propagate through multiple layers.

3.2.3 Segformer

Segformer utilizes transformers with multilayer perceptron (MLP) decoders (Xie et al., 2021). Transformers are originally used in natural language processing, but have been adapted for segmentation tasks. This was achieved by incorporating transformers into deep learning models, to effectively assess the relationships between pixels in an image, enabling more accurate and context-aware segmentation. MLP decoder aggregates information from different layers, and thus combinine both local and global features to compute segmentation masks (Xie et al., 2021). In Segmentation Gym the segformer model utilizes transfer learning. Transfer learning uses pretrained models on big datasets to reduce computation time and increase performance of models.

3.3 Motivated settings for selected method(s)

The routine for training a model is determined by the optimizer that guides training, and through the use of a deterministic learning rate scheduler. Neural networks are trained with variations of the stochastic gradient descent (SGD) algorithm, and the specific form of the optimizer is a hyperparameter. Schmidt et al.,(2021) found Adam to be a good optimizer choice for almost all Deep Learning models based on Convolutional layers. Therefore the Adam optimizer is used for the first four models to find out which model performs best. This model will be utilized to research which hyperparameters work best.

Other important model training variables specified in the configuration file are batch size, loss function, and learning rate. These tend to have a greater impact on final model accuracy than other hyperparameters such as kernel size, number of convolutional filters, and Dropout according to (Buscombe & Goldstein, 2022). Therefore, these more important hyperparameters are described in more detail below.

When training models with large datasets, they are divided into smaller batches because they can't fit into memory all at once. The batch size, which refers to the number of examples processed together, is an important factor to consider. For example, if we have a batch of six images and their corresponding labels, we evaluate the model's performance and adjust the weights based on the average difference between the model's predictions and the actual labels of those six images.

The batch size affects how well the model recognizes patterns in the data, especially when there is variability among the examples. Smaller batch sizes can handle more variability, but they may take longer to train (converge) because they require more computational resources. If the GPU memory allows, the batch size will be increased to evaluate the difference in performance. For the first four models a batch size of six will be used.

During training, the model's performance is optimized by using a loss function that measures the difference between the predicted outputs and the true labels. In the case of segmentation tasks, where the goal is to label each pixel in an image, Segmentation Gym provides various options for the loss function. In this research three common loss functions are compared: mean Dice, categorical cross-entropy (CCE), and Kullback-Leibler distance (KLD). The first four models will be trained on mean dice loss function, since it can effectively deal with class-imbalance, or the tendency for a majority class to dominate over one or more minority classes (Csurka et al., 2013).

Segmentation gym uses a learning rate scheduler that starts with a small learning rate, then quickly ramps up to a maximum, then decays exponentially. Lowering the learning rate as the model is training allows it to slowly converge on a solution. This procedure prevents the solution from getting stuck in a local minimum much higher than the global minimum. Without changing the learning rate, the model can potentially lead to suboptimal convergence or trigger early stopping criteria prematurely. In addition to a scheduler, Segmentation Gym also implements an early stopping criterion, where the model ceases training early when no improvement to validation loss is observed over a user-defined number (10) of training epochs. (Buscombe & Goldstein , 2022). For these reasons different learning rates will be researched.

This paper also researches if changing the kernel size can improve the model. Kernel size refers to the dimensions (width and height) of a filter or convolutional kernel applied to an image or feature map. Even though it should not lead to a much better performance according to (Buscombe & Goldstein, 2022), I would suggest it can improve the model, since using a lower kernel size can capture more local features. This will especially help with predicting white dunes and shrubs since these cover smaller areas and more detail compared to sand and grey dunes. However, it is important to strike a balance, as excessively reducing the kernel size can also increase sensitivity to noise or result in the detection of irrelevant features.

The last hyperparameter that will be researched is the class weights. When looking at the autumn 2018 image the data is unbalanced. Grey dunes and sand covers most of the image, so to prevent the model from predicting too much of the majority class, class weights can be utilized. When using class weights the distribution of the training set is calculated and weights are added to each class according to their distribution. If the class is in majority, the model will be rewarded less for getting it correct and if the class is in minority the mode gets punished more for getting it wrong.

3.4 Model evaluation

The following metrics were used for model evaluation f1-score, precision, recall, Mean Intersection over Union (Mean IoU). The model were also evaluated visually on the complete orthomosaics.

All these metrics were calculated using true positives, true negatives, false positives and false negatives. These will be explained by using an example with two classes, sand and vegetation. In this research negative would be referred to the incorrect classes compared to the true label.

- True Positive (TP): Model predicted positive and it is true. You predicted that the pixel is sand and it actually is.
- True Negative (TN): Model predicted negative and it is true. You predicted that the pixel is vegetation and it actually is.
- False Positive (FP) (Type 1 Error): Model predicted positive and it is false. You predicted that the pixel is sand and it actually is vegetation.
- False Negative (FN) (Type 2 Error): Model predicted negative and it is false. You predicted that the pixel is vegetation and it actually is sand.

Predicted values are described as Positive and Negative and actual values as True and False.

All these measures are displayed in a confusion matrix seen in Figure 9.

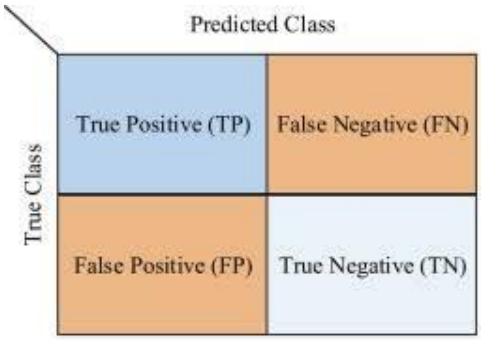


Figure 9 Example Confusion Matrix. X-axis are predicted classes and Y-axis is the actual classes. Each number (TP, TN, FP, FN) represents the percentage of that predicted class. The value should be closer to 1 for the diagonal and closer to 0 for the other values. The diagonal indicates the correct predictions.

Mean IoU measures the similarity between the predicted segmentation mask and the true mask by calculating the intersection over union (IoU) for each class and then calculating the average across all classes. The IoU, also known as the Jaccard index, is computed as the ratio of the intersection to the union of two sets. In the context of image segmentation, the IoU for a specific class is determined by dividing the area of overlap between the predicted mask and the corresponding true mask by the area of their union. Mean IoU is calculate by $TP\ /\ (TP\ +\ FP\ +FN)$.

- Recall: Think of recall as finding all the classes. A high recall means the model captures most classes present in the image. Recall is calculated by $TP/(TP + FN)$.
- Precision: Precision is about how accurate the model is when it says something is a specific class like sand. Precision is calculated by $TP/(TP + FP)$.
- F1 score: The F1 score combines both recall and precision into a single metric. It provides a balanced measure of how well the model performs in differentiating between the classes. F1 score is calculated by $2 * (Precision * Recall)/(Precision + Recall)$.
- All values are better if they are closer to 1 and worse if they are closer to 0.

An orthomosaic will represent the habitat map by letting the model predict on each tile and putting the complete image back together. This will be compared with the original image to evaluate how well the model performs on the complete image, to make sure the habitat map is accurate. The models performance can be skewed if the model only predicts the majority classes, in this case sand and grey dunes. To prevent creating a model that achieves skewed results the orthomosaic is used.

4. Results and analysis

First the four models with the default hyperparameters and three different models are compared based on the created habitat map. Afterwards the best performing models are shown and compared based on the metrics, performance and habitat maps. Lastly all metrics are displayed in a table to illustrate what different settings were compared.

4.1 Four default models

The habitat maps are shown in Figure 10, where blue is sand, orange is grey dune, yellow is white dune, green is shrubs and purple is other. In this image ResUnet2 overall performs the best. The model can distinguish the sand and grey dunes well. It does however mistake shrubs for grey dunes sometimes. In the image it is also shown that the road passing through the image is correctly predicted as the other class. However the model can't distinguish the water that is also included in the other class, this is visible in the top right at the dark blueish spot which is supposed to be water but classified as shrubs. This is most likely, because it is very shallow water which makes it transparent and shows some vegetation and sand that is on the lakebed. Also in ResUnet2 it predicts more white dunes in the foredunes which should be more correct compared to ResUnet1. However ResUnet2 also predicts more white dunes more land inward, probably because it notices marram gras near sand.

Segformer predicts a lot of the class 'other' and shrubs. It mistakes grey dunes as shrubs and sand as the other class. However it is interesting to see that both models can predict that white dunes are more towards the sea which is correct. This difference is due to the lower kernel size which makes it predict more white dunes.

UNet predicts mostly white dunes and no sand at all. It definitely performs the worst out of the 3 models.

According to these habitat maps ResUnet2 performs the best overall. Segformer and Unet will not be further researched, because it generally doesn't perform well and hyperparameters won't boost their performance to an efficient result.

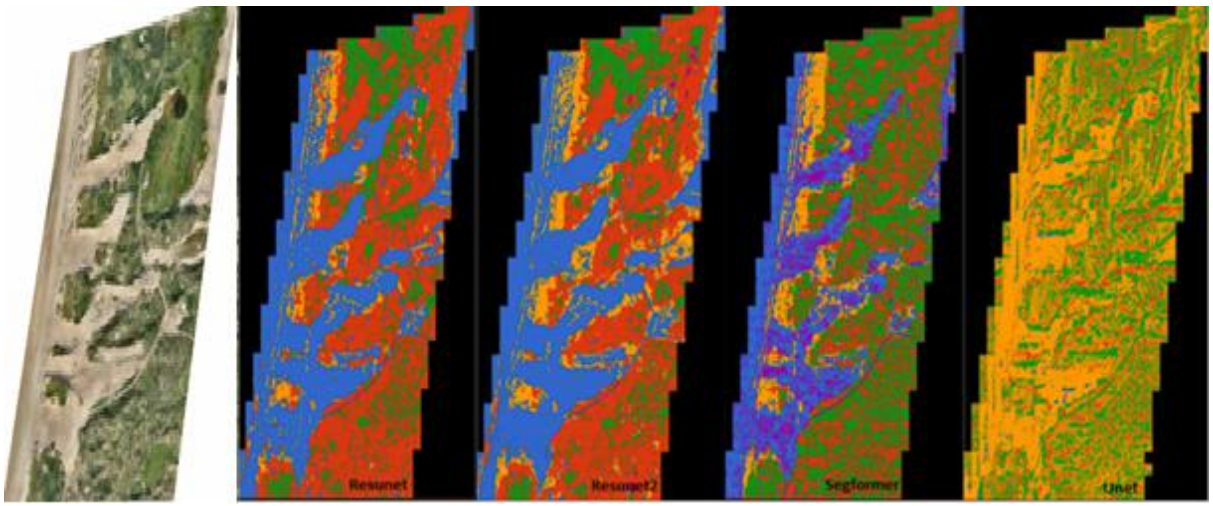


Figure 10 The four default models, original image, ResUnet1, ResUnet2, segformer and UNet. Blue=sand, orange=grey dune, yellow=white dune, green=shrubs and purple=other.

4.2 Best models

The performance of ResUnet1 and ResUnet2 are show in Figure 12 and Figure 11. ResUnet1 performs well based on true positive rate (TPR), which indicates how much of the true class is predicted correctly. The TPR on sand (0.583), grey dunes (0.845) and shrubs (0.493) performs well but worse on white dunes (0.192) and the other class (0.071). Resunet2 is better on white dunes with a TPR of 0.444. The difference between the models is a smaller kernel size for convolutional layers. This means it can extract a bit more specific detail from the images, which apparently benefits the white dune class and disrupts the performance on shrubs.

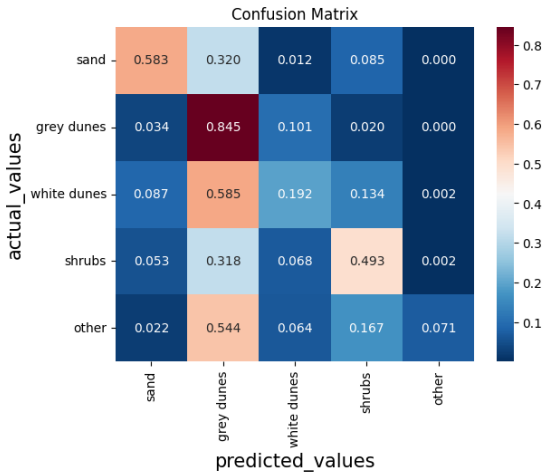


Figure 11 Confusion Matrix ResUnet (diagonal is TPR)

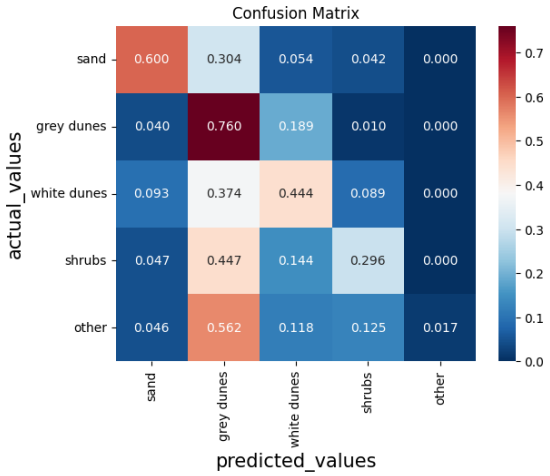


Figure 12 Confusion Matrix ResUnet2 (diagonal is TPR)

The model performs worse on other, shrubs and white dunes, the minority classes, the class weights should help the predictions on the minority class. Sand would also be a minority class, but the model does not struggle with getting sand correct. The class weights for sand is 0.80, grey dunes is 0.54, white dunes is 0.86, shrubs is 0.81 and other is 0.99. The models gets rewarded more for getting the minority class like sand, other, shrubs and white dunes right and rewards the model less for getting the majority class of grey dunes correct. The confusion matrix of the ResUnet with class weights is shown in Figure 13. It performs well on sand (0.605), grey dunes(0.845) and decently on shrubs(0.402). Its still lacking in performance on the other class and white dunes, like the other two ResUnet models. Every class gets mistaken mostly by grey dunes, seen in the predicted row of grey dunes. By adding class weights it was expected this would be lower but not much changed compared to the other two ResUnet models. Despite the limited improvement in the confusion matrix, the improvement in the orthomosiacs will show it is a better result.

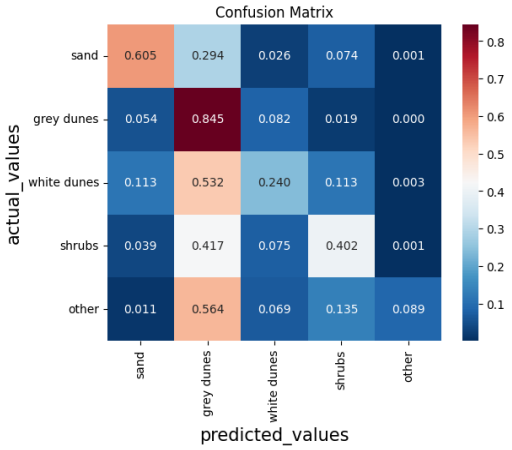


Figure 13 Confusion Matrix Resunet with classweights

In Figure 14 the original image and predictions are shown again and the last image is with class weights. The ResUnet with class weights makes better predictions close to the sea, since it mistake the shoreline as dunes less compared to ResUnet and ResUnet2. The ResUnet with weights also does not predict the class other much, but still predicts the road as the other class. However there is one big advantage in using class weights shown in a zoomed in visualization of the top left near the white dunes in Figure 15.

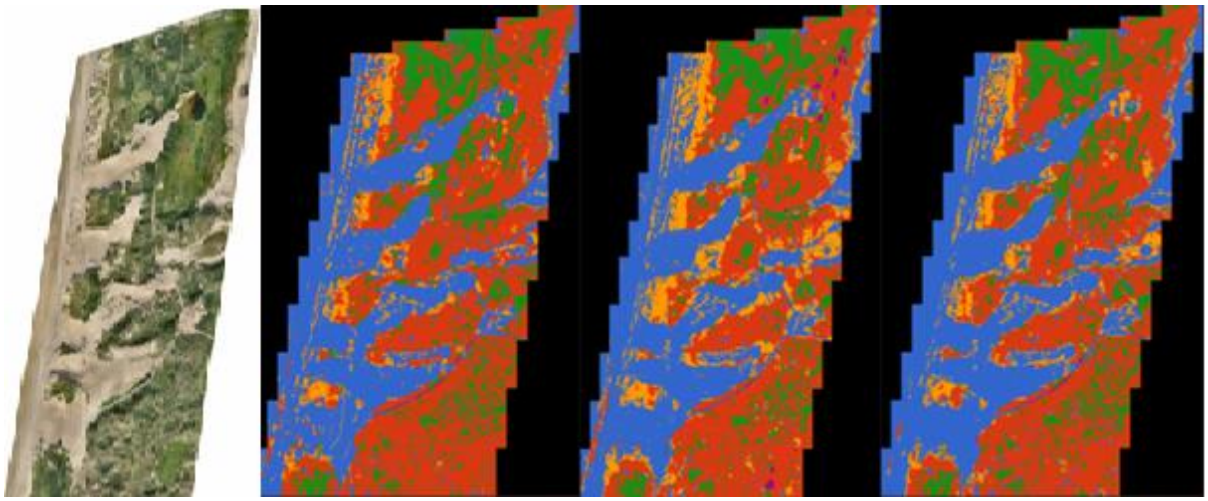


Figure 14 In order original Image, ResUnet, ResUnet2, ResUnet with class weights

It is visible in Figure 15 that the ResUnet with class weights on the right performs better. In the ResUnet1 prediction lines and squares/tiles can be seen. This is a cause of dividing up the data into tiles, however the ResUnet with weights has less issues with this. This phenomenon of tiles/squares is called tiling.

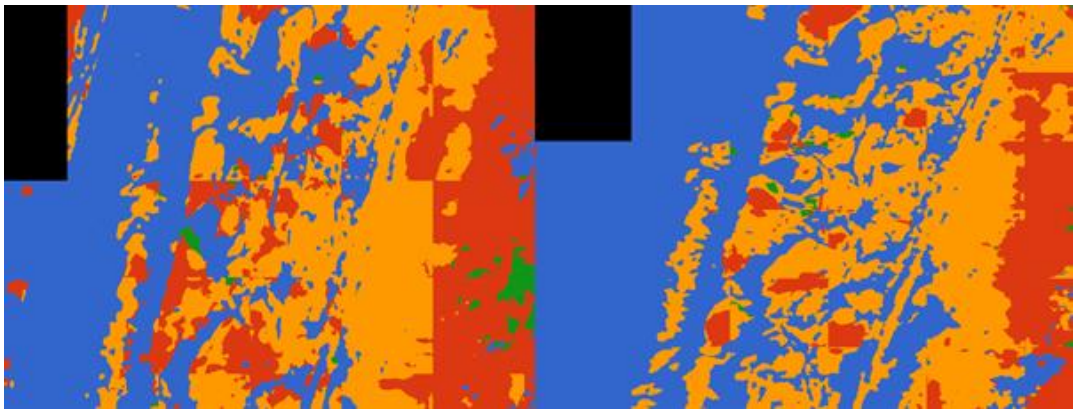


Figure 15 Two of the same tiles. To the left is ResUnet1 and to the right is ResUnet with weights

All metrics are compared in Table 1 and are all high because they are closer to 1 rather than 0. All the models perform worst on the Mean IoU metric. The lower Mean IoU score can be explained by how the metric is measured. There may scenarios where the model successfully predicts the general shape and location of the vegetation but fails to precisely capture the exact boundaries, which leads to a lower Mean IoU score. The other metrics are close to each other, even though the metrics for ResUnet with class weights is lower, I would suggest to use class weights since it prevents tiling.

Metrics/ Models	ResUnet1	ResUnet2	ResUnet with Weights
F1	0.768	0.763	0.752
Recall	0.780	0.781	0.765
Precision	0.826	0.829	0.805
Mean IoU	0.670	0.666	0.649

Table 1 Metrics for the three ResUnets

4.3 Hyperparameter Tuning

The ResUnet1, ResUnet2 and ResUnet with class weights models perform well on grey dunes/sand, but there is still a challenge in predicting white dunes and other class for these models. The model that is used for hyperparameter tuning is the ResUnet with class weights, since it performs the best. The new models will be compared by exploring different parameters and mainly focus on loss function, kernel size, batch size and learning rate, since these will impact the model the most. The loss function will influence the training of the model. The kernel size will get more specific feature maps and capture more detail, lower batch size will train the model more times on smaller amounts of data. Changing the learning rate can help the model escape a local minimum and perform way better or worse.

The results of all the different loss functions are shown in Table 2 Comparison of different loss functionsTable 2. The loss function has a lot of impact on the performance of the model, because all the metrics get way closer to 0.5 and even 0. This is a big decrease compared to the metrics of the dice loss being around 0.75. This decrease in performance is likely due to the dice function being more optimized for image segmentation tasks. Unlike the pixel-wise evaluation of loss functions like categorical cross-entropy (cat), the Dice loss function takes bigger areas around the pixels into account to calculate the loss. The next models will only be trained using dice loss, since it is the best performing loss function.

Metrics/ Models	Dice (original)	Hinge	Cat
F1	0.752	0.330	0.058
Recall	0.765	0.279	0.077
Precision	0.805	0.546	0.335
Mean IoU	0.649	0.211	0.034

Table 2 Comparison of different loss functions

The learning rate parameter uses a starting, max and minimum learning rate. The default values for the starting learning rate is $1e^{-7}$. The starting learning rate was to $1e^{-5}$, since it stayed low during the models training. The thought was that the model was stuck in here but it decreased the performance shown in Table 3. All metrics decrease with 0.3 to 0.4, which means the model performs just a bit better than a model that only predicts random values (0.2 for all metrics).

Kernel size was set from 9 to 7, because it increased the performance on white dunes seen in ResUnet2, however this also decreased the performance by 0.3 to 0.4 per metric. For batch size a larger setting was chosen since it was already low as a value of 6. The choice was mode to set it to 32 to see if this would make the model perform better, but this was not the case as it decreased performance by 0.3 to 0.5 per metric. In all the models the precision stays the best performing metric, this is due to all the models still being able to predict grey dunes well.

Metrics/ Models	Original	Kernel size	Learning rate
F1	0.752	0.365	0.337
Recall	0.765	0.388	0.371
Precision	0.805	0.472	0.498
Mean IoU	0.649	0.273	0.270

Table 3 Comparison of Kernel size, learning rate and batch size

The decrease in performance for the model with an increased learning rate can be explained by overshooting. The higher learning rate causes the model to change the parameters of the ResUnet too aggressively which makes it harder to train and converge to an optimal solution.

Increasing the batch size causes there to be fewer updates for the model to train on. With a higher batch size the variation with each update is also lower since there is more data to train on. This can cause overfitting on the training data. However, it can also be beneficial since it distributes the noise a bit better so a batch size between 6 and 32 needs to be tested.

Noise is also an issue with smaller kernel size due to the same reasons of the batch size. It might also be better for our use case to capture more global features, for example the change from sand to vegetation.

Since only one change in each parameter is tested it would be beneficial to explore more changes in settings. My suggestion would be to test a lower starting learning rate of $1e^{-8}$ and $1e^{-6}$, because $1e^{-5}$ was to low. A batch size between 2 and 32 would be better since going beyond 32 will only make it worse. My suggestion would be to use increments of the power of 2, so 2, 4, 8, 16 and 32. Also lowering the kernel size even further won't benefit the model as much, an increase of kernel size to 11 or 13 might perform better.

5. Conclusion

The three different models ResUnet, segformer and UNet performs differently where ResUnet performs the most optimal. The segformer mistakes a lot of grey dunes as shrubs and predicts the other class to often. The UNet model makes a lot of mistakes and predicts white dunes a lot and almost no sand at all. The ResUnet gives a desired result where sand, grey dunes and shrubs are clearly distinguished and have a cross-shore zonation. However it still struggles with differentiating grey dunes and white dunes. Since the ResUnet performed the best hyperparameter tuning will be performed on this model.

The biggest performance increase comes from training with class weights to help with the class imbalance in the training data. This improved the overall predictions and reduced tiling, an issue where lines and squares appear in the fully predicted image due to how the model was trained on smaller images. Testing the loss functions hinge, Cat and Kld only reduced performance substantially. Increasing the learning rate and batch size also only resulted in lower performance. Also by lowering the kernel size there is a reduction in performance. The most ideal configuration for image segmentation in the Dutch National park Zuid-Kennemerland would be to use the ResUnet default settings while adding class weights.

6. Discussion

The doodler annotation tool proved to be a great tool, as it saved a lot of time and field work. Also the precision between annotators is similar and creates easy to process labels. However there is an issue with the black pixels within the annotations. It was not possible to change these black pixels to the surrounding class before training, which eventually leads to a decrease in performance. A suggestion would be to get into contact with the developers or try to resolve this issue. An attempt was made to remove the black pixels before training, but segmentation gym could not recognize the images anymore after changing the black pixels.

This research mostly focused on exploring different models and tuning hyperparameters for those models, but it is also possible to tune on the image augmentation parameters. This can increase the performance of the ResUnet model as well, however this won't increase the performance of segformer and UNet to obtain a desired solution.

In segmentation gym it is not possible to use transfer learning for ResUnet. However this is possible using different python libraries like segmentation models (Lakubovskii, 2019). It provides four different models and uses transfer learning by providing 25 different pre-trained models. Transfer learning can be applied to image segmentation tasks by leveraging pre-trained convolutional neural networks (CNNs). These are trained on large-scale image datasets, such as ImageNet, to initialize a model's weights. This initialization provides a head start in learning general visual features, which could be helpful for this research.

Using transfer learning also reduces computation time. Training deep learning models from scratch can be computationally intensive and time-consuming, especially for large networks. The pre-trained layers in the network are already well-tuned for basic visual features, and only need to be fine-tuned on specific layers involved in the segmentation task. This training process enables quicker iterations, experimentation, and model refinement.

All in all, transfer learning can expedite the development of accurate and robust segmentation models for coastal dunes images, even when data is limited, by leveraging the knowledge and features learned from pre-training on larger and more diverse datasets (Hosna et al., 2022).

References

Arens, S. M., Mulder, J. P., Slings, Q. L., Geelen, L. H., & Damsma, P. (2013). Dynamic dune management, integrating objectives of nature development and coastal safety: Examples from the Netherlands. *Geomorphology*, 205-213. https://doi.org/10.1016/j.geomorph.2012.10.034

Buscombe, D. (2022, November 18). *dash_doodler*. From Github: https://github.com/Doodleverse/dash_doodler

Buscombe, D., & Goldstein, E. (2022). *A Reproducible and Reusable Pipeline for Segmentation of.* AGU. https://doi.org/10.1029/2022EA002332

Buscombe, D., & Goldstein, E. (2022, May 29). *Doodleverse/Segmentation Gym*. From Github: https://github.com/Doodleverse/segmentation_gym

Buscombe, D., Goldstein, E. B., Sherwood, C. R., Bodine, C., Brown, J. A., Favela, J., . . . Wernette, P. (2022). *Human-in-the-Loop Segmentation of Earth Surface Imagery.* AGU. https://doi.org/10.1029/2021EA002085

Csurka, G., Larlus, D., & Perronnin, F. (2013). *What is a good evaluation measure for semantic segmentation.* Bristol: BMVA Press. https://doi.org/10.5244/c.27.32

Diakogiannisa, F. I., Waldnerb, F., Caccetta, P., & Wu, C. (2019). *ResUNet-a: a deep learning framework for semantic segmentation of remotely sensed data.* Crawley: CoRR. https://doi.org/10.48550/arXiv.1904.00592

Hosna, A., Merry, E., Gyalmo, J., & Alom, Z. (2022). *Transfer learning: a friendly introduction.* Bangladesh: Journal of Big Data. https://doi.org/10.1186/s40537-022-00652-w

Lakubovskii, P. (2019). *segmentation_models*. From Github: https://github.com/qubvel/segmentation_models

Kooij, F. (2022). *Classification of Coastal Dune Vegetation from Aerial Imagery with a Convolutional Neural Network.* Utrecht.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., & Talwalkar, A. (2020). A system for massively parallel hyperparameter tuning. *MLSys Conference.* Austin: arXiv. https://doi.org/10.48550/arXiv.1810.05934

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical. *arXiv*, 1-8. https://doi.org/10.48550/arXiv.1505.04597

Ruessink, B., Arens, S., Kuipers, M., & Donker, J. (2018). Coastal dune dynamics in response to excavated foredune notches. *Aeolian Research*, 3-17. https://doi.org/10.1016/j.aeolia.2017.07.002

Ruessink, G. (2023, January 9). *Topographic data and orthomosaics of the Noordwest Natuurkern project.* From zenodo: https://zenodo.org/record/7515352#.ZEZja3ZBy3A

Schmidt, R. M., Schneider, F., & Hennig, P. (2021). Descending through a Crowded Valley — Benchmarking Deep Learning Optimizers. *International Conference on Machine.* Tübingen: arXiv. https://doi.org/10.48550/arXiv.2007.01547

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. *arXiv*, 1-18. https://doi.org/10.48550/arXiv.2105.15203

Appendix

Config files

All other config settings not mentioned here can be found in the results.

ResUnet1


```
{
 "TARGET_SIZE": [512,512],
 "MODEL": "resunet",
 "NCLASSES": 5,
 "KERNEL":9,
 "STRIDE":2,
 "BATCH_SIZE": 6,
 "FILTERS":6,
 "N_DATA_BANDS": 3,
 "DROPOUT":0.1,
 "DROPOUT_CHANGE_PER_LAYER":0.0,
 "DROPOUT_TYPE":"standard",
 "USE_DROPOUT_ON_UPSAMPLING":false,
 "DO_TRAIN": true,
 "LOSS":"dice",
 "PATIENCE": 10,
 "MAX_EPOCHS": 100,
 "VALIDATION_SPLIT": 0.5,
 "RAMPUP_EPOCHS": 20,
 "SUSTAIN_EPOCHS": 0.0,
 "EXP_DECAY": 0.9,
 "START_LR":  1e-7,
 "MIN_LR": 1e-7,
 "MAX_LR": 1e-4,
 "FILTER_VALUE": 0,
 "DOPLOT": true,
 "ROOT_STRING": "hatteras_l8_768",
 "USEMASK": false,
 "AUG_ROT": 5,
 "AUG_ZOOM": 0.05,
 "AUG_WIDTHSHIFT": 0.05,
 "AUG_HEIGHTSHIFT": 0.05,
 "AUG_HFLIP": true,
 "AUG_VFLIP": false,
 "AUG_LOOPS": 10,
 "AUG_COPIES": 5,
 "SET_GPU": "0",
 "WRITE_MODELMETADATA": false,
 "DO_CRF": false,
 "LOSS_WEIGHTS": false,
 "MODE": "all",
 "SET_PCI_BUS_ID": true,
 "TESTTIMEAUG": true,
 "WRITE_MODELMETADATA": true,
 "OTSU_THRESHOLD": true,
 "TF_GPU_ALLOCATOR" : "cuda_malloc_async",
```

```
 "CLEAR_MEMORY" : true
}
```


ResUnet2


```
{
 "TARGET_SIZE": [512,512],
 "MODEL": "resunet",
 "NCLASSES": 5,
 "KERNEL":7,
 "STRIDE":2,
 "BATCH_SIZE": 6,
 "FILTERS":6,
 "N_DATA_BANDS": 3,
 "DROPOUT":0.1,
 "DROPOUT_CHANGE_PER_LAYER":0.0,
 "DROPOUT_TYPE":"standard",
 "USE_DROPOUT_ON_UPSAMPLING":false,
 "DO_TRAIN": true,
 "LOSS":"dice",
 "PATIENCE": 10,
 "MAX_EPOCHS": 100,
 "VALIDATION_SPLIT": 0.5,
 "RAMPUP_EPOCHS": 20,
 "SUSTAIN_EPOCHS": 0.0,
 "EXP_DECAY": 0.9,
 "START_LR":  1e-7,
 "MIN_LR": 1e-7,
 "MAX_LR": 1e-4,
 "FILTER_VALUE": 0,
 "DOPLOT": true,
 "ROOT_STRING": "hatteras_l8_768",
 "USEMASK": false,
 "AUG_ROT": 5,
 "AUG_ZOOM": 0.05,
 "AUG_WIDTHSHIFT": 0.05,
 "AUG_HEIGHTSHIFT": 0.05,
 "AUG_HFLIP": true,
 "AUG_VFLIP": false,
 "AUG_LOOPS": 10,
 "AUG_COPIES": 5,
 "SET_GPU": "0",
 "WRITE_MODELMETADATA": false,
 "DO_CRF": false,
 "LOSS_WEIGHTS": false,
 "MODE": "all",
 "SET_PCI_BUS_ID": true,
 "TESTTIMEAUG": true,
 "WRITE_MODELMETADATA": true,
 "OTSU_THRESHOLD": true,
 "TF_GPU_ALLOCATOR" : "cuda_malloc_async",
 "CLEAR_MEMORY" : true
}
```

Segformer


```
{
   "TARGET_SIZE": [512,768],
   "MODEL": "segformer",
   "NCLASSES": 5,
   "KERNEL":9,
   "STRIDE":2,
   "BATCH_SIZE": 6,
   "FILTERS":6,
   "N_DATA_BANDS": 3,
   "DROPOUT":0.1,
   "DROPOUT_CHANGE_PER_LAYER":0.0,
   "DROPOUT_TYPE":"standard",
   "USE_DROPOUT_ON_UPSAMPLING":false,
   "DO_TRAIN": true,
   "LOSS":"dice",
   "PATIENCE": 10,
   "MAX_EPOCHS": 100,
   "VALIDATION_SPLIT": 0.5,
   "RAMPUP_EPOCHS": 20,
   "SUSTAIN_EPOCHS": 0.0,
   "EXP_DECAY": 0.9,
   "START_LR":  1e-7,
   "MIN_LR": 1e-7,
   "MAX_LR": 1e-4,
   "FILTER_VALUE": 0,
   "DOPLOT": true,
   "ROOT_STRING": "hatteras_l8_768",
   "USEMASK": false,
   "AUG_ROT": 5,
   "AUG_ZOOM": 0.05,
   "AUG_WIDTHSHIFT": 0.05,
   "AUG_HEIGHTSHIFT": 0.05,
   "AUG_HFLIP": true,
   "AUG_VFLIP": false,
   "AUG_LOOPS": 10,
   "AUG_COPIES": 5,
   "SET_GPU": "0",
   "WRITE_MODELMETADATA": false,
   "DO_CRF": false,
   "LOSS_WEIGHTS": false,
   "MODE": "all",
   "SET_PCI_BUS_ID": true,
   "TESTTIMEAUG": true,
   "WRITE_MODELMETADATA": true,
   "OTSU_THRESHOLD": true,
   "TF_GPU_ALLOCATOR" : "cuda_malloc_async",
   "CLEAR_MEMORY" : true
 }
```

Unet


```
{
 "TARGET_SIZE": [512,512],
 "MODEL": "unet",
 "NCLASSES": 5,
 "KERNEL":3,
 "STRIDE":2,
 "BATCH_SIZE": 6,
 "FILTERS":8,
 "N_DATA_BANDS": 3,
 "DROPOUT":0.0,
 "DROPOUT_CHANGE_PER_LAYER":0.0,
 "DROPOUT_TYPE":"standard",
 "USE_DROPOUT_ON_UPSAMPLING":false,
 "DO_TRAIN": true,
 "LOSS":"cat",
 "PATIENCE": 10,
 "MAX_EPOCHS": 100,
 "VALIDATION_SPLIT": 0.5,
 "RAMPUP_EPOCHS": 20,
 "SUSTAIN_EPOCHS": 0.0,
 "EXP_DECAY": 0.9,
 "START_LR":  1e-7,
 "MIN_LR": 1e-7,
 "MAX_LR": 1e-4,
 "FILTER_VALUE": 0,
 "DOPLOT": true,
 "ROOT_STRING": "hatteras_l8_768",
 "USEMASK": false,
 "AUG_ROT": 5,
 "AUG_ZOOM": 0.05,
 "AUG_WIDTHSHIFT": 0.05,
 "AUG_HEIGHTSHIFT": 0.05,
 "AUG_HFLIP": true,
 "AUG_VFLIP": false,
 "AUG_LOOPS": 10,
 "AUG_COPIES": 5,
 "SET_GPU": "0",
 "WRITE_MODELMETADATA": false,
 "DO_CRF": false,
 "LOSS_WEIGHTS": false,
 "MODE": "all",
 "SET_PCI_BUS_ID": true,
 "TESTTIMEAUG": true,
 "WRITE_MODELMETADATA": true,
 "OTSU_THRESHOLD": true,
 "TF_GPU_ALLOCATOR" : "cuda_malloc_async",
 "CLEAR_MEMORY" : true
}
```

Resunet with weights


```
{
 "TARGET_SIZE": [512,512],
 "MODEL": "resunet",
 "NCLASSES": 5,
 "KERNEL":9,
 "STRIDE":2,
 "BATCH_SIZE": 6,
 "FILTERS":6,
 "N_DATA_BANDS": 3,
 "DROPOUT":0.1,
 "DROPOUT_CHANGE_PER_LAYER":0.0,
 "DROPOUT_TYPE":"standard",
 "USE_DROPOUT_ON_UPSAMPLING":false,
 "DO_TRAIN": true,
 "LOSS":"dice",
 "PATIENCE": 10,
 "MAX_EPOCHS": 100,
 "VALIDATION_SPLIT": 0.5,
 "RAMPUP_EPOCHS": 20,
 "SUSTAIN_EPOCHS": 0.0,
 "EXP_DECAY": 0.9,
 "START_LR":  1e-7,
 "MIN_LR": 1e-7,
 "MAX_LR": 1e-4,
 "FILTER_VALUE": 0,
 "DOPLOT": true,
 "ROOT_STRING": "autumn_v2_resunet_512",
 "USEMASK": false,
 "AUG_ROT": 5,
 "AUG_ZOOM": 0.05,
 "AUG_WIDTHSHIFT": 0.05,
 "AUG_HEIGHTSHIFT": 0.05,
 "AUG_HFLIP": true,
 "AUG_VFLIP": false,
 "AUG_LOOPS": 10,
 "AUG_COPIES": 5,
 "SET_GPU": "0",
 "WRITE_MODELMETADATA": false,
 "DO_CRF": false,
 "LOSS_WEIGHTS": true,
 "MODE": "all",
 "SET_PCI_BUS_ID": true,
 "TESTTIMEAUG": true,
 "WRITE_MODELMETADATA": true,
 "OTSU_THRESHOLD": true,
 "TF_GPU_ALLOCATOR" : "cuda_malloc_async",
 "CLEAR_MEMORY" : true
}
```

Confusion Matrices

Segformer

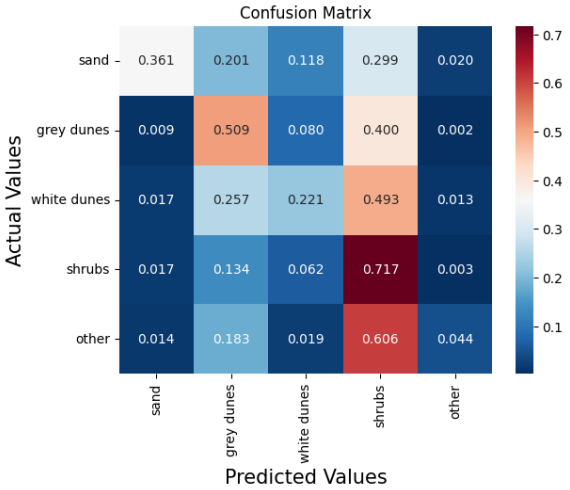


Figure 16 Confusion Matrix Segformer

Unet

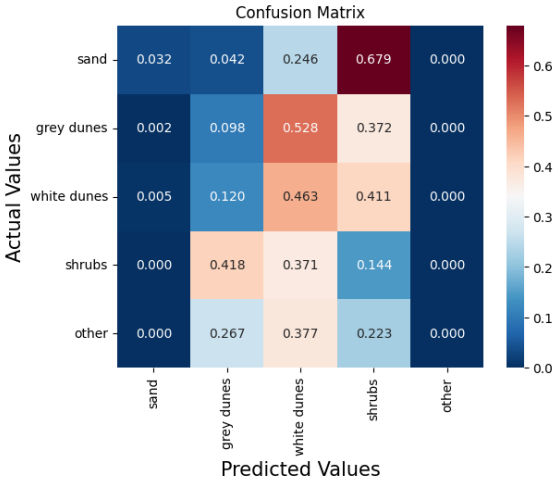


Figure 17 Confusion Matrix Unet

Resunet Hinge loss

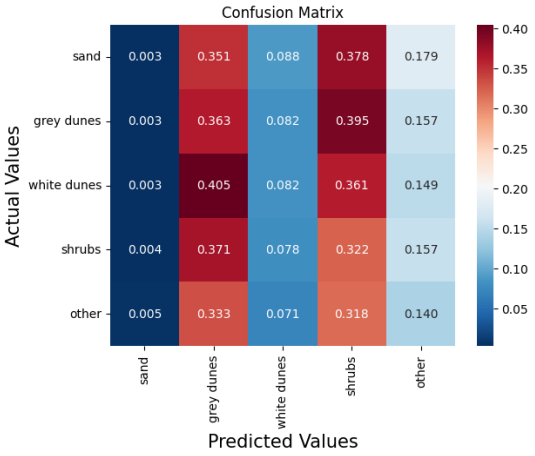


Figure 18 Confusion Matrix ResUnet Hinge loss

ResUnet Cat loss

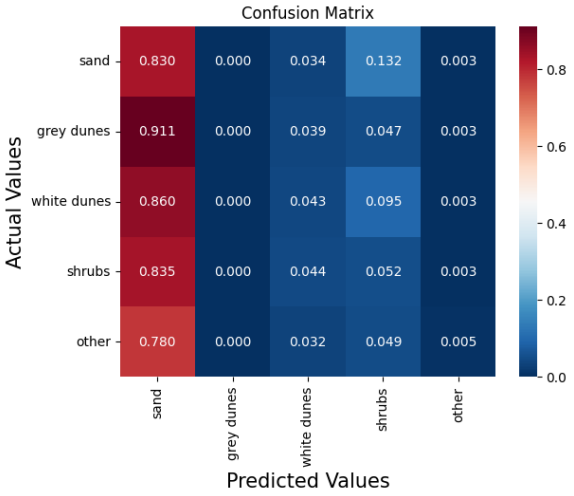


Figure 19 Confusion Matrix ResUnet Cat loss

ResUnet KLD loss

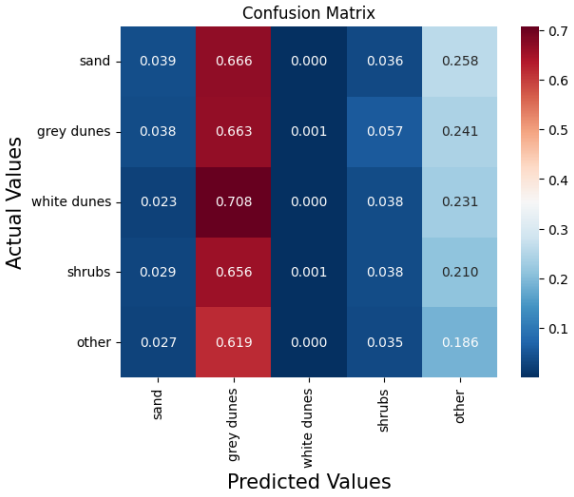


Figure 20 Confusion Matrix ResUnet KLD loss

ResUnet Learning rate

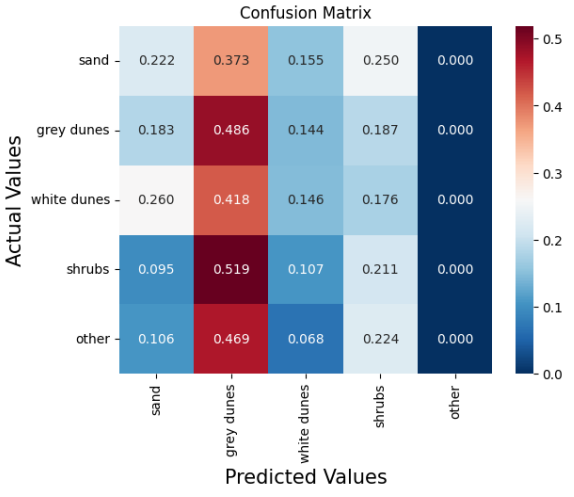


Figure 21 Confusion Matrix ResUnet Learning rate

ResUnet Batchsize

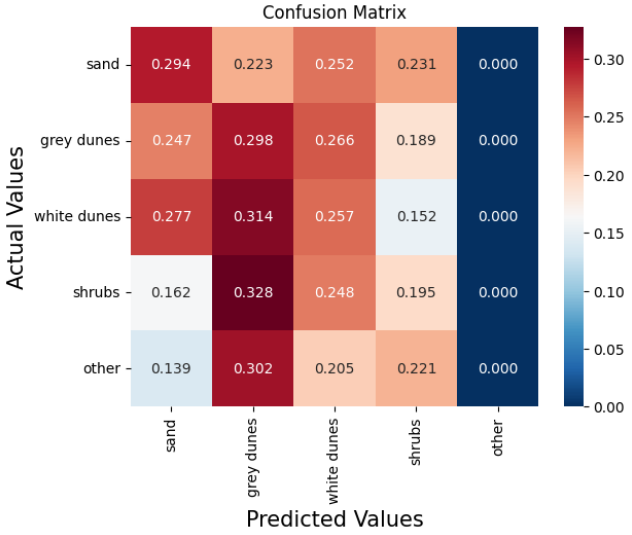


Figure 22 Confusion Matrix ResUnet batch size

ResUnet Kernel

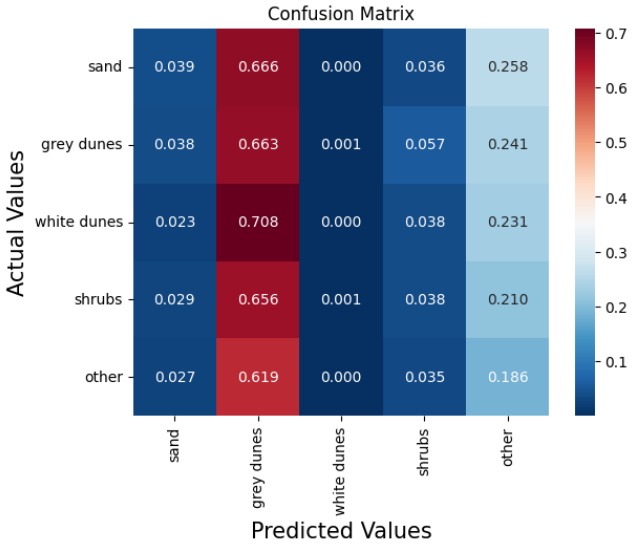


Figure 23 Confusion Matrix ResUnet Kernel