Sand Engine beach state assessment by applying machine learning on massive Argus image data

Alex De Jong[1,*], Roderik Lindenbergh[1], Sander Vos[2], Daan Hulskemper[1]

[1] Dept. of Geoscience and Remote Sensing, Delft University of Technology, Delft, The Netherlands - a.dejong-9@student.tudelft.nl, (r.c.lindenbergh, d.c.hulskemper)@tudelft.nl
[2] Dept. of Hydraulic Engineering, Delft University of Technology, Delft, The Netherlands - s.e.vos@tudelft.nl

Keywords: Semantic segmentation, coastal dynamics, autonomous camera systems, illumination variation, beach processes

Abstract

Dynamic beaches world-wide are monitored by so-called Argus cameras. Their automatic capturing results in large databases of $\sim$ 30’ interval coastal images acquired during different illumination conditions. We present a lightweight method to automatically extract sand and supporting classes from $\sim$ 1 million Argus images, spanning $\sim$ 10 years, of the Sand Engine, The Netherlands, a nature-based solution for beach erosion. The workflow consists a series of neural networks. First, a ResNet18 model selects images of sufficient quality. Second, 24 pixel-wise shallow multi-layered perceptrons (MLPs) classify pixels into 5 classes, Water, Foam and Eolian, Wet and Armored Sand. The 24 MLPs correspond to 12 cameras with two different lighting conditions. The results from the 24 MLPs are used as pseudo labels for 24 CNNs that improve the initial classification by including spatial awareness. These 24 CNNs are fused in one ensemble CNN, robust to camera choice and lighting condition. At this stage, still a separate CNN is used to additionally detect vegetation pixels. Results show in most cases good agreement with human interpretation, with an overall accuracy of $\sim$ 88%. Most promising is that $\sim$ 150.000 images per day can be processed on a high-end consumer PC at a quality difficult to obtain by a human operator. Future work should focus on exploiting the large database of results for improving our understanding of dynamic processes at this challenging environment. As the workflow is generic in nature, it should be easily applicable for other image based monitoring databases.

1. Introduction

The Sand Engine is a mega nourishment at the sandy beach coast just south of The Hague, The Netherlands (De Schipper et al., 2016), compare Figure 1, right. Created in 2011, it covered initially 2×1 km. Originally, it featured a beach lake, a lagoon, and a shore connected sand plate, over time it got partially covered by Marram grass, (Hulskamp et al., 2025). The Sand Engine consists of a range of tidal and near-shore morphological features. Starting from the landward side lies a prominent foredune, densely vegetated with sea buckthorn. Moving seaward, a secondary dune area of loose sand is present, dominated by marram and couch grass. A broad flat shell bed extends over most of the shelf directly adjacent to the sea. Finally, the seaward margin consists of an intertidal zone with periodically saturated sand and wave behaviour.

Since its construction, it has been a highly dynamic environment, continuously reshaped by wind, sea, and human intervention, (Vandebroek et al., 2017, Rutten et al., 2018). Its dynamic evolution has been monitored by different remote sensing techniques, (Lindenbergh et al., 2025). Notably, a so-called Argus tower was built, (Holman and Stanley, 2007), see also Figure 1, left. This Argus tower is 40 m tall and contains 12 cameras observing the 360 $^\circ$ surroundings of the tower. Figure 1, right, shows how each of the 12 cameras covers an angular part of 30°, with some overlap, of the surroundings of the towers. Each camera automatically took an image every 30 minutes of 2448×2048 pixels from 2013 until 2025 resulting in a dataset of $\sim$ 4.5 terabyte consisting of $\sim$ 1 million images. Globally, various other Argus towers have been applied to monitor phenomena such as Phaeocystis blooms along the Catalan coast, (Arin et al., 2014), and coastal responses to extreme weather in Zhoushan, China, (Lianqiang et al., 2019). In contrast, the Sand Engine site has been comparatively underexplored.

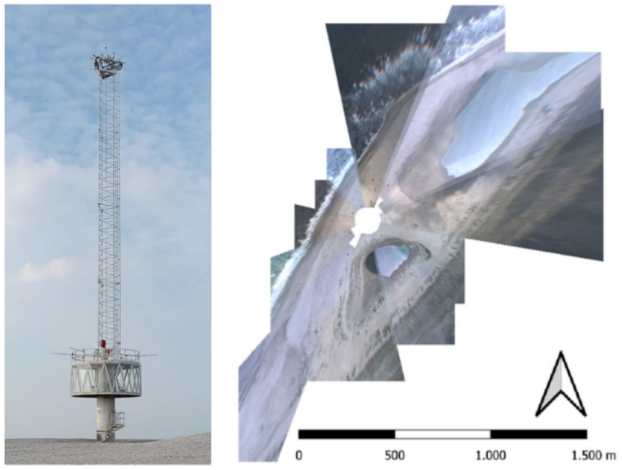

Figure 1. **Left.** Former Argus tower on the Sand Engine. The 12 used cameras are mounted on the top of the tower. **Right** Mosaic of example images from each of the 12 cameras.

Goal of this study is to develop automated deep learning methodology, (Rombado et al., 2024) to effectively exploit this huge database by automatically segmenting good quality images into classes meaningful for studying beach and dune morphodynamics, like armoured and aeolian (easily transportable) sand. At a later stage, resulting class dynamics, (Rabehi et al., 2023) will be compared to results from other sensing techniques (satellite imagery, UAV-LiDAR) and linked to forcing data (wind, waves,

* Corresponding author

bulldozer presence) to better understand sand dynamics, (McFall et al., 2024).

2. Data and Methods

2.1 Data description

The ARGUS tower (Figure 1, Left), was equipped with a circular array of 12 PtGrey Flea2 FL2G-50S5C cameras, fitted with Fujinon 12.5 mm HF12.5SA-1 lenses. These cameras captured images with a resolution of 2448 $\times$ 2048 px at 96dpi. Images are captured during daytime, based on archived sunrise and sunlit times. A mosaic of one image of each of the 12 cameras is shown in Figure 1, right. The ARGUS tower was dismantled in March, 2026.

The ARGUS dataset contains 12 years of images acquired from 12 cameras, captured at 30 minutes intervals, resulting in a dataset of over 1.2 million daytime images. Images were available in ’Snap’, ’Timex’, ’Rundark’, ’Dark’, ’Bright’ and ’Var’ (Variation) formats. ’Snap’ images capture a standard photo at the exact time of the set interval. ’Dark’ and ’Bright’ images record the minimum (darkest) and maximum (brightest) values for each pixel in each 30-minute window. ’Rundark’ images are dark-frame calibration images used to quantify sensor noise. ’Var’ images generate an index of pixel change values per channel over the recording window. Finally, ’TimeX’ images collect a time-average of all frames in the recording window. Only ’TimeX’ images were used in the workflow, as they are not affected by moving objects while preserving colour information, as demonstrated in figure 2. Figure 2 demonstrates that moving objects, like a walking person, are highlighted in ’Dark’ images, as the person presence results in locally dark pixels at changing locations in consecutive image frames.

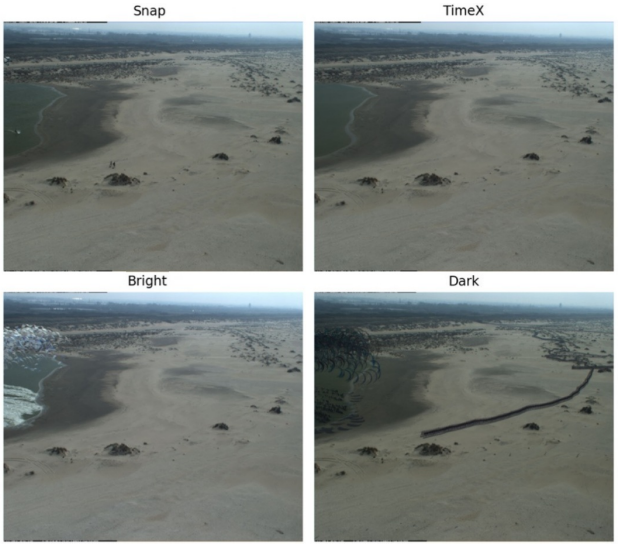

Figure 2. ARGUS image products. Top. Left ’Snap’ Image. Right ’TimeX’ image. Bottom. Left ’Bright’ Image. Right ’Dark’ Image. The workflow in study uses TimeX images, a composite image were each pixel represents the median of a series of consecutive input images. Notably TimeX composition effectively removes moving objects.

2.2 Method overview

The presented workflow focuses on efficient automated classification and segmentation of the large database of ARGUS Sand Engine ’TimeX’ imagery. It consists of five steps, with additional substeps.

Step 1: a ResNet18 CNN selects valid images by removing underexposed, overexposed, blurry, or obscured images. In addition, valid images are categorised into two exposure classes, BrightClear and DimClear. The result of Step 1 consists of $2 \times 12 = 24$ batches of images: for each of 12 cameras a batch of BrightClear and DimClear images is obtained.

Step 2: generates 24 different classification models, one for each batch. Each model assigns each pixel of each input image in one of the five classes: ’AeolianSand’, ’ArmouredSand’, ’WetSand’, ’Water’, and ’Foam’.

Step 3: one common CNN model is derived with an ensemble distillation of the final 24 batch models from Step 2. This strategy provides one single model that is light enough to process all images efficiently and good enough to provide reasonable results.

Step 4: Currently, an additional Step 4 is used to train an extra CNN for detecting vegetation pixels.

Step 5: images are projected onto a horizontal ground plane, enabling alignment with satellite imagery and Cartesian area computations.

2.3 Step 1, valid image selection

The Sand Engine is located on a North Sea beach, and exposed to a heterogeneous range of lighting and weather conditions. To reduce lighting challenges for later modeling and filter out invalid data, a model was applied to classify images accordingly. An off-the-shelf Resnet18 CNN model, (He et al., 2016, Al-Haija et al., 2020), was trained on a set of 2000 user-labeled images, downsized to 600 $\times$ 600 pixels. Two groups of classes were distinguished. The first group consists of images with ground cover that was considered to be easily distinguishable by a human user, and consisted of two classes, ’BrightClear’ and ’DimClear’, compare Figure 3. BrightClear images were well illuminated and had ideal conditions, while DimClear had flatter light caused by cloud cover, but can still be considered adequately illuminated for further processing. The second group consists of imagery considered to be too challenging for further consideration and consisted of the classes ’Overexposed’ (lens flares, whiteouts), ’UnderExposed’ (very dim), ’Obscured’ (high entropy due to e.g. raindrops on the lens), and ’Blurry’ (fog, unfocused).

2.4 Step 2, semantic segmentation

Images that successfully passed Step 1 provide input for semantic segmentation. Output consists of per-pixel class labels for three sand classes, ’AeolianSand’, ’ArmouredSand’, ’WetSand’, and two other classes, ’Water’, and ’Foam’. The sand classes were chosen, as notably ’AeolianSand’ corresponds to loose sand that is easily transported by wind; on the other hand, ’ArmouredSand’ consists of more fixed sand with pebble or shell coverage, found at area undergoing erosion, while ’WetSand’ is not expected to be shaped by wind, (Bauer et al., 2009). ’Water’ is an obvious dynamic class in the area, while ’Foam’ was added at a later stage to avoid confusions near the water boundary. These classes separate stable and unconsolidated sediment areas for transport interpretation.

Semantic segmentation is performed in two sub-steps, with the goal to simultaneously optimize quality and computational performance. In the first sub-step a shallow MLP is used to extend sparse, manually obtained, training data to a larger set of pseudo labels. In the second sub-step this extended training data set is exploited to train a deep UNet, (Ronneberger et al., 2015).

Step 2a, 2×12 shallow, feature based MLP models The motivation to start with a shallow MLP is that it requires less training than a small CNN to generate moderately accurate results, (Nurunnabi et al., 2021). In addition, user labels were typically biased towards easily identifiable patches, resulting in spurious shape biases in preliminary CNN results. In contrast, the MLP model has no spatial awareness, and is therefore better able to create unbiased labels. To improve robustness, for each feature, the difference in feature value at the pixel considered between the previous and next time step is incorporated as well, (Barbero-García et al., 2023). Hence, each feature below has 3 time steps, one in the past, one current, and one future. The following 3×7 pixel-wise features were generated:

3×3 RGB: 3 color values, providing baseline for classes like water and vegetation

3×1 Intensity: normalized sum of R+G+B, to separate reflectivity from colour changes.

3×1 Saturation: obtained by converting RGB to HSL, (Garg et al., 2022), to decouple colour from lighting intensity changes

3×1 Lightness: obtained by converting RGB to HSL, proxy for surface roughness

3×1 Shannon entropy: Proxy for small-scale surface texture, (Naidu et al., 2018). To account for the distance from the camera and high texture range, two search radii of 3px and 9px were used and the maximum value was selected. The Shannon entropy, $H(X)$, of pixel X is defined as:

$$H(X) = -\sum_i p(x_i) \ln p(x_i) \tag{1}$$

with $p(x_i)$ the relative frequency of outcome x_i, where i enumerates the pixels in the neighborhood considered, The negative sign in Eqn. 1 ensures entropy is positive. A low entropy typically corresponds to a smooth surface and indicates predictable outcomes, while e.g. texture transitions are expected to correspond to high entropy.

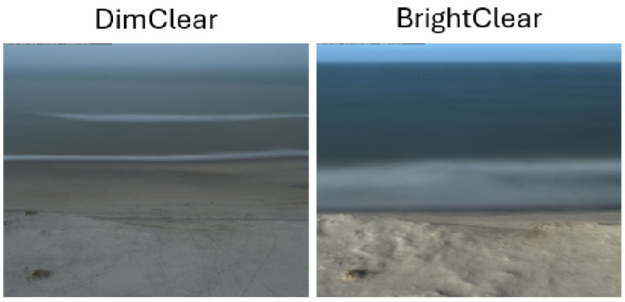

Figure 3. Image classified as valid are further classified into two lighting classes: **Left** ’DimClear’, and **Right** ’BrightClear’, corresponding to low and high contrast conditions respectively.

The classification model is a fully connected Multi Layered Perceptron (MLP), (Bishop and Bishop, 2023). The network consists of an input layer with 21 dimensions, followed by 4 hierarchically decreasing hidden layers with neuron structure (512, 256, 128, 64). Each layer is equipped with a Rectified Linear Unit (ReLU) activation function to introduce non-linearity. Between hidden layers, dropout layers are incorporated to mitigate overfitting by randomly deactivating 20% of neurons, promoting robust feature representations. This architecture was chosen to balance model capacity and computational efficiency. The output layer has a number of units corresponding to output classes and produces raw logits. These logits are passed through a softmax-based cross-entropy loss function for training. To improve generalization and prevent overconfident predictions, label smoothing is applied within the loss. Data is shuffled once per epoch to ensure stochasticity while avoiding redundant in-memory optimization. Network parameters were optimized with the Adam optimizer, which employs adaptive learning rates and momentum to achieve stable and efficient convergence.

Ground truth was collected by a human operator labeling sparse sample polygons of classes, per camera and lighting condition. Polygons were manually selected from 100 random images per class and camera-lighting combination, with some overlap between sampled images. The user defined patches were applied as extraction mask. Pixels within sample polygons aggregated a 9×9 local patch to reduce noise and provide a robust representation of the local spatial context.

Step 2b, 24 deep UNet models For each of the 24 camera-lighting combinations, a CNN UNet model was generated. The pseudo-labels resulting from Step 2a were used to train student UNet CNNs to incorporate spatial patterns. This allows the models to learn camera and lighting-specific conditions. UNets were trained directly on source RGB images, with MLP Hard-Max labels weighted by confidence as teachers.

The segmentation model used in this step is a convolutional UNet with an encoder-decoder architecture with skip connections, (Ronneberger et al., 2015). The encoder has 4 convolutional blocks with channel progressions (32, 64, 128, 256), each performing two 3×3 convolutions followed by batch normalization and ReLU activation to extract hierarchical feature representations. The bottleneck expands the channels to 512 for maximal feature richness before the decoder successively upsamples through transposed convolutions with reversed channel configuration. The full model has approximately 7 million parameters. The output layer uses a 1×1 convolution to produce logits for each semantic class. Weighted cross-entropy loss is used for class imbalance handling, and mixed precision training with gradient scaling improves computational efficiency. Optimization is performed with AdamW, balancing convergence stability and generalization.

For data preparation, a patch extraction-based pipeline was applied. As the models need to learn texture-based classes, full-resolution samples were used. As processing full images at once was computationally not feasible the dataset was prepared by sampling 256 × 256 patches from full images. Class boundaries on pseudo-label maps were detected with a morphological gradient approach. The label map is dilated and then eroded with a small rectangular kernel, with boundary pixels identified by differences in these regions. This was used to specifically target semantic edges from the label maps, to ensure difficult edge conditions were well represented. Additionally, randomly

selected patches are collected to prevent edge bias and teach general class signatures. Patches are assigned a dominant class for balance tracking, allowing for automatic prioritization of underrepresented classes. Border and random patches are distributed 50/50. On average, this resulted in 15000 patches per fragment, with approximately 4 patches extracted per image. These images were stored as compact parquets, resulting in ≈ 10GB of training data. This was split at a patch level between 80% training and 20% validation. At training, all patches are precomputed into PyTorch tensors and held in cache, resulting in near-optimal GPU use.

Step 3, combined final UNet model Step 2b resulted in 24 specialized UNets, each tuned to one of 24 camera-lighting combinations. By performing an ensemble distillation, common inter-camera overlaps support classification while degradative camera biases are modeled out, (Noothout et al., 2022). Additionally, due to cleaner labels, a smaller UNet can be used to decrease inference overhead without significant performance degradation.

Applying a similar patch-based system as discussed in Step 2b, random patches of raw RGB images are collected. For each data patch, the teacher UNet fragment with matching camera and lighting class performs a segmentation and saves a per-pixel logit map, given a weight of 70%. To improve generalization without introducing excess noise, 3 other randomly selected models of the same lighting class also perform a segmentation, making up the other 30% weight. Logits are used in contrast to pseudo labels, as the goal is to incorporate a wide range of biases over cameras and average them out, (Chen et al., 2025).

2.5 Step 4, vegetation UNet

The MLPs from Step 2a tended to treat vegetation as a catch-all class for weakly identifiable textures, due to its high range of signal and comparative rarity. Thus, a binary segmentation UNet was trained, directly on user-labelled vegetation patches.

The vegetation segmentation U-Net employs a light-weight architecture with ≈ 2 million parameters, using GroupNorm for improved stability with small batch sizes. The model aims to distinguish sparse vegetation pixels from visually similar background. First, the output layer is initialized with a strong negative bias (∼1.5), producing an initial sigmoid probability of ∼0.18, which forces the model to earn positive predictions rather than defaulting to them. Second, the training employs a two-phase loss strategy: an initial BCE warmup (10 epochs) followed by a combined focal loss (α= 0.25, γ = 2.5) and Dice loss, where the focal loss incorporates an asymmetric false positive penalty (1.5×) that specifically targets incorrect vegetation predictions. The change in loss regime resulted in improved and more stable validation dice scores.

To tune training patches, a Sobel filter, (Gonzalez and Woods, 2018) was run on all user generated vegetation polygons. This applied erosion to the label geometry, wrapping rough hand labels to the nearest vegetation edge, by identifying the texture + color boundary and cropping the polygon to that shape. These masks automatically extracted a background context label polygon that was 2.5 × larger than the vegetation patches. Additionally, 200 pure background patches were separately labeled. To account for label inaccuracy, an uncertainty gradient was set over boundary edges. Boundary regions are assigned soft labels (0.5) with reduced loss weighting (0.4), relaxing the optimization constraint where class boundaries are inherently ambiguous. To improve generalisation, the following training augmentations were applied: translations, rotations, flips, and contrast variation. Finally, post-training threshold tuning on the validation set identifies an optimal decision boundary (typically 0.65–0.75 rather than the default 0.5) that maximises precision while maintaining acceptable recall.

2.6 Step 5, orthographic projection

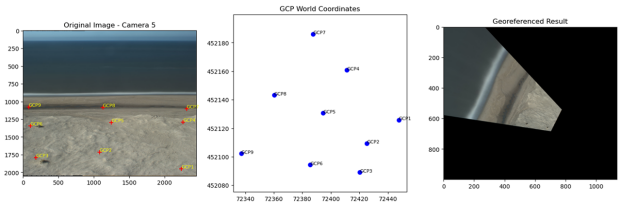

Figure 4. Example Homographic Projection of a Camera 5 image. The image on the left is projected onto a horizontal plane, as shown on the right, using the ground control points shown in the middle image.

As a final step, output classified photos were projected to a geospatially usable ground plane. This was performed using GNSS located ground control points and their corresponding pixel location on survey images, cf. Figure 4. Using RANSAC, images are then warped onto a map plane (RD-NAP EPSG:28992). A 50 % total maximum projected range radius was applied to automatically crop projections to usable resolutions. Transformation errors were typically under 1 meter.

2.7 Validation

The final classification model was evaluated with 200 hand labeled images per camera, spread across lighting conditions. Images used for evaluation were selected from years other than the year 2020 of the training dataset and had a minimum calendar day separation of 3 days to avoid similarity. Accuracy, mean intersection over union (IoU), F1 and Dice-score were used as evaluation metrics. In addition, a qualitative inspection was performed.

2.8 Libraries

The image segmentation pipeline was implemented in Python 3.12 and built around a GPU-accelerated workflow, It is using PyTorch (torch 2.10.0+cu128) with CUDA 12 support (cupy-cuda12x 13.6.0). Input images were preprocessed and augmented using Albumentations (2.0.8), OpenCV (opencv-python-headless 4.9.0.80), and Pillow (12.0.0). The workflow was supported by numerical and scientific libraries including NumPy (1.26.4), SciPy (1.17.1), and scikit-image (0.26.0). Geospatial data handling and raster processing were performed using Rasterio (1.5.0), Shapely (2.1.2), and PyVista (0.47.0). Data management and analysis relied on Pandas (2.3.3) and Dask (2026.1.2) for scalable processing, with compression management via h5py (3.15.1) and Zarr (3.1.5).

3. Results

3.1 Computational performance

All processing was performed on a consumer-grade workstation, with an Intel i9 14th gen CPU, Nvidia RTX5070 and 32GB

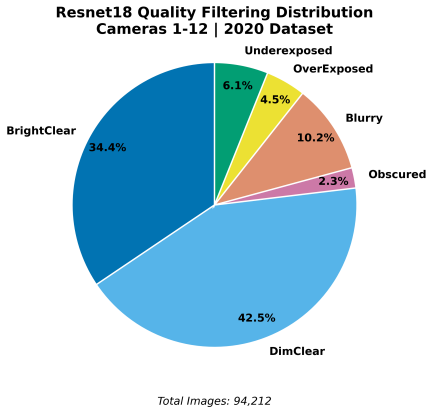

Figure 5. Overview of the ResNet18 classification into image quality classes. The two blue portions represent images used for semantic classification.

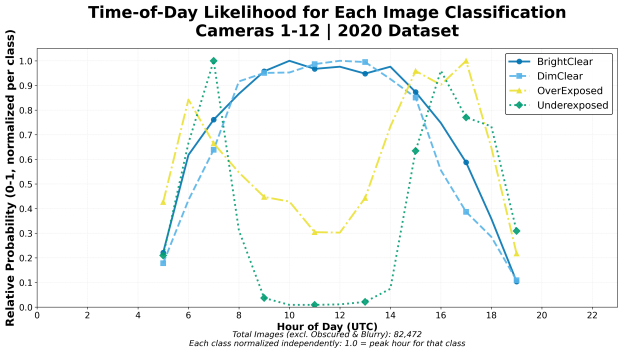

Figure 6. Temporal Distribution of Lighting Classes.

of DDR5 RAM. MLP fragments typically took under 15’ to train, and performed inference at ∼1.6 images per second. 2nd-generation CNN fragment teaching took under 30’ to train from precomputed classification maps, with approximately sub 1-second inference times on raw RGB. Finally, due to the overlapping teacher outputs, ensemble distillation had a larger training set, resulting in 6-8 hour training times. The final model had an inference rate of approximately 3 images per second.

3.2 Step 1, valid image classification results

Figure 5 provides an overview of the classification of all input images into quality classes. The two blue portions together represent the 76.9 % of the images that are classified by the ResNet18 CNN as being of high enough quality to enter the semantic segmentation stage. The remaining 23.1 % of the images was disregarded for different reasons as indicated in the pie chart. The temporal distribution of the different quality classes is shown in Figure 6. Low quality imagery is notably acquired at the start and end of the day, although ’OverExposed’ imagery is also acquired throughout the day and contains images affected by solar artifacts, like sun glint.

3.3 Step 2a, Individual MLP results

Some example results of the MLP classification are shown in Figure 7. Recall that 24 different MLP models were trained, one for ’DimClear’ and one for ’BrightClear’ imagery for 12 different cameras. Figure 7 shows examples from Camera 1 and 10. respectively. Note that results are often quite granular,

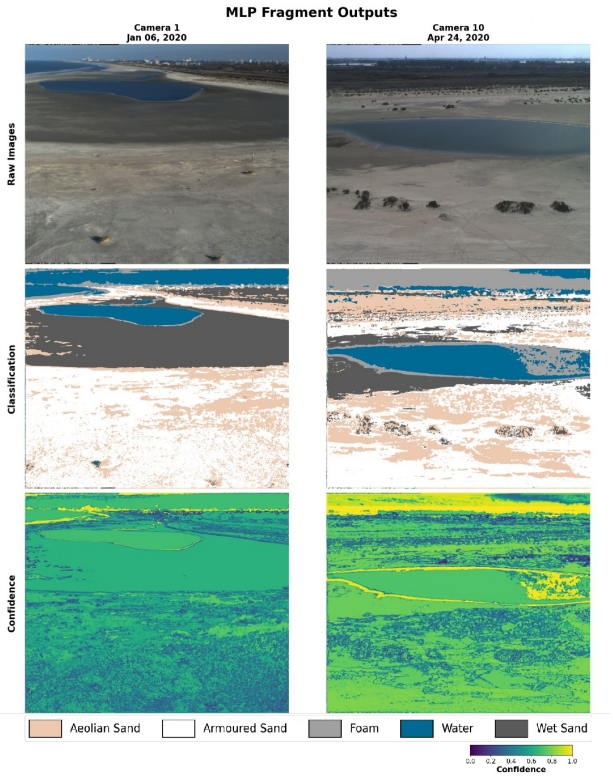

Figure 7. Example MLP Classifications. **Top.** Raw TimeX RGB **Middle**. HardMax MLP Labels **Bottom** Label Confidence Heatmap. Note that vegetation is not a target class in the MLP classification.

which is the consequence of using MLPs at single pixel level: although spatial neighborhoods are used in feature generation, no spatial awareness is incorporated in the MLP architecture. Still, the MLP results do a reasonable job in segmenting obvious image patches. The bottom row show pixel-wise confidences. Confidence is partly class dependent, e.g. ’Foam’ seems more confident, while ’Armoured Sand’ and ’Aeolian Sand’ often have lower confidences. These two classes are alternating in the foreground of the two example images, and are also not easy to distinguish for a human observer.

3.4 Step 2b, Individual CNN results

For inference, a batched scanning tile method was used. Tiles of 256 × 256 pixels were processed in parallel, with batch sizes dependent on IO speeds of the hardware. The tiles were given a stride of 192px to avoid edge artifacts. In practice, the process was bottle-necked by IO limitations due to poor Windows/Python multi-core utilization, rather than raw GPU compute. An inference speed of approximately 1.5 image per second was achieved, similar to teacher MLPs.

When evaluating CNN outputs, there is a significant increase in the quality of classifications. Conjunctively, these factors all indicate that the model is outperforming the teacher model. The UNet student quickly learns from confident labels and is predicting ground truth accurately, where the MLP is failing.

3.5 Ensemble CNN results

The ensemble CNN model was evaluated on an unseen validation set to assess its segmentation performance. Figure 8

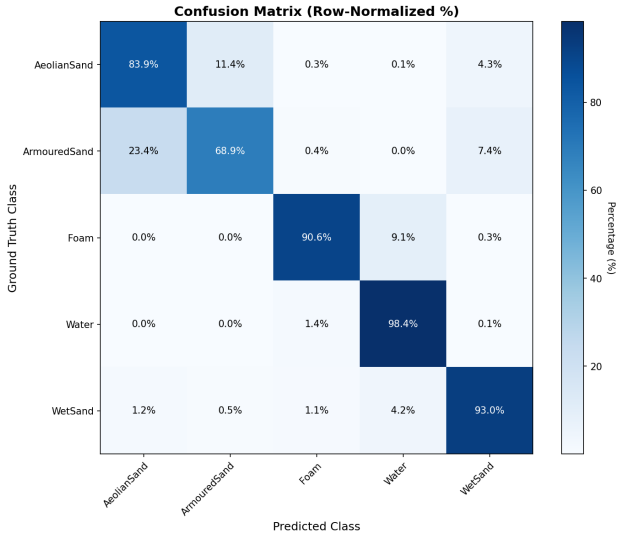

Figure 8. Confusion Matrix of the ensemble CNN model. The overall accuracy equals 88.46 %. Most confusions occur between aeolian and armoured sand classes.

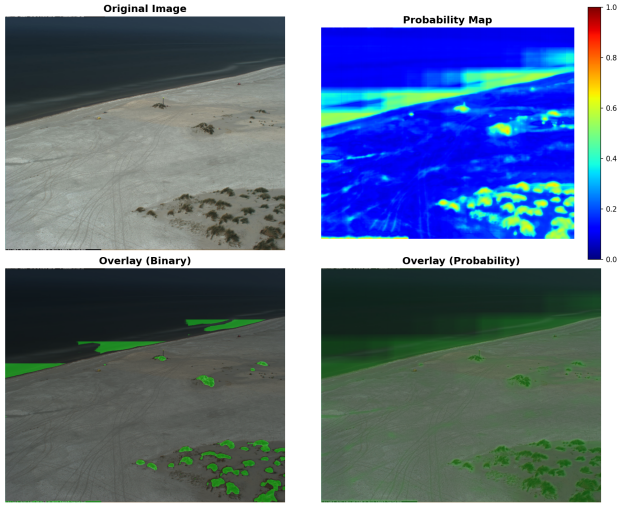

Figure 9. Example vegetation classification results. Top row; left: input image, right: predicted vegetation probability. Bottom row; left: binary classification results overlaid on input image. Green pixels are classified as vegetation. Right: predicted vegetation probability overlain on input image. Vegetation is picked up well, but water near the shoreline gets sometimes confused as vegetation as well.

shows the confusion matrix of the final model. Quantitative metrics indicate strong overall performance, with an overall accuracy of 88.46 %, a mean Intersection over Union (IoU) of 0.71, and both mean Dice and F1 scores of 0.78. These results demonstrate that the model is effectively capturing the target structures and generalizing well beyond the training data. Again, most misclassifications occurred between AeolianSand and ArmouredSand. Visual inspection of predicted segmentation shows that the UNet consistently produces more accurate boundaries compared to the rough user labels, particularly along object edges. Note though that the validation dataset is quite small, so quality results should be interpreted with care.

3.6 Results, vegetation model

An example result of the CNN vegetation model is shown in Figure 9, including probabilities. The final vegetation model reached an overall Dice score of 0.8 across all cameras and lighting conditions. Visual inspection of test results showed generally good performance, with most false negatives occurring on small vegetation patches. These small patches are often so small that human inspection is impossible. Most false positives occurred in water boundary areas as seen in Figure 9, where watery pixels are falsely assigned to the vegetation class. Overall visible vegetation, typically occurring on top of small sand dunes is captured well by the model.

4. Overall results

This section summarizes the performance of the ensemble U-Net classification model including a separate vegetation detection step. In addition, it shows two examples of linking the massive classified image database to coastal processes. Figure 10 shows representative source images alongside their corresponding classified outputs across all camera sites, offering a visual overview of the model’s general performance. Note that the results in Figure 10 are outcomes of the ensemble CNN model, on which the outcomes of an additional CNN step to identify vegetation are superimposed. The ensemble model is designed to combine efficiency and robustness. Efficiency is obtained by first, using pseudolabels obtained by each of 24 MLPs fed by features and trained with very limited ground truth data. Spatial awareness is added in 24 CNNs fed by pseudo labels output by the MLPs. Robustness over different cameras and lighting conditions is obtained by creating one ensemble CNN combining all individual CNNs. The results in Figure 10 show that in general the model is able to effectively classify input imagery from different cameras and different lighting classes in relevant geomorphological classes. In most cases, the foreground is performing better. Partially this is the result of not having an ’air’ class included, while in addition the background sea or air shows little texture for the model to work on anyway.

From a computational point of view, the first aspect to mention is the manual labeling involved. For each of 24 MLPs, only $\sim$ half an hour was spent on creating ground truth labels, resulting in a total of less then two working days for labeling, by a single person. The ResNet18 model took $\sim$ 15’ for classifying all input imagery according to lighting condition. Training each of 24 teacher CNNs takes an hour each; training the MLPs is faster, but requires additional feature engineering, which takes about an hour per year of features. The ensemble CNN took $\sim$ 6 hours to train, and classifies about 3 images per second. The vegetation UNet took $\sim$ 30’ to train, with an inference speed of $\sim$ 3 images per second.

4.1 Water Tracking Results

On the Sand Engine, water acts as a powerful morphodynamic forcing agent, eroding and transporting sand. The segmented database enables high-resolution mapping of water coverage with fine temporal granularity, allowing detailed analysis of water dynamics over time. For physical validation, the normalised power spectral density (PSD) of water coverage and tide height (measured at Hoek Van Holland) was calculated, shown in Figure 11. The analysis revealed strong correlations in the PSD peaks, particularly at 7 and 14 hours, corresponding to the two primary phases of the Dutch tidal cycle. The variations in water coverage can be notably used to explain or identify water

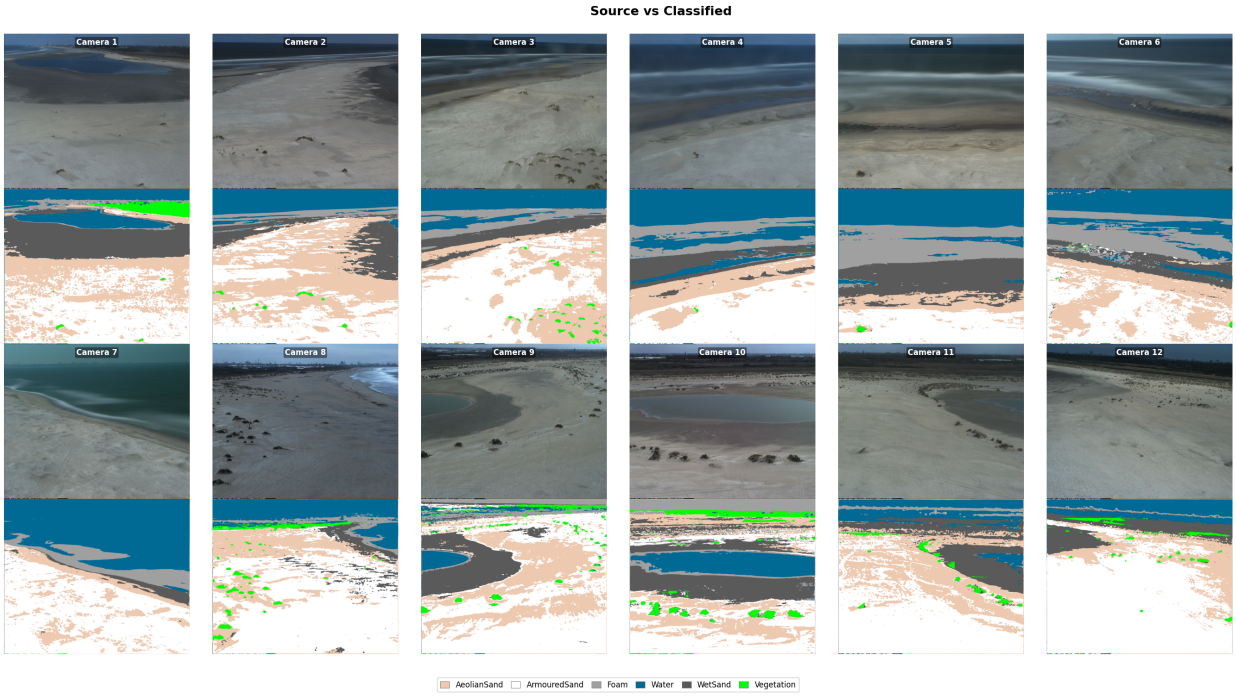

Figure 10. Results of the automatic classification of Argus Sand Engine imagery as obtained by the ensemble CNN model plus CNN vegetation. One example is shown for each of 12 different cameras. In general, best results are obtained in the foreground.

surge/flooding induced erosion or deposition in the area. Note that similar results have been obtained using a classic approach, (Wijnberg and Holman, 1997), at a different location in The Netherlands.

4.2 Storm Surge detection

Storm surges that raise local water levels above the protective berm result in rapid inundation of the inland beach plain. At the Sand Engine, these result in abrupt changes in the volume and area of the inland beach lakes (Figure 12, top). These floods erode the beach plain and transport mud and organic sediment across the surface. By monitoring spikes in the water area for camera 1, surge floods were automatically found and recorded, compare also (Pan et al., 2022). Over a sample year of 2020, there were 10 detected events, predominantly in autumn and winter (Figure 12, bottom).

5. Conclusions

A lightweight neural network method is presented to automatically classify images from a large set of Argus imagery into morphodynamic classes. The first step distinguishes suitable images from images with issues, like being overexposed. The second step consists of building 24 shallow pixel-wise MLP models using human defined features and limited training data. These MLPs are used to generate pseudo-labels, that are fed as training data into 24 spatial-aware CNNs. Next these 24 CNNs are used to generate one ensemble CNN, that is robust enough to take images from any of 12 cameras and 2 lighting conditions and classify their pixels into the five classes ’Aeolian Sand’, ”Armoured sand’, ’Foam’, ’Water’ and ’Wet Sand’. An additional CNN is used to detect vegetation pixels. The results show that the proposed solution is capable of handling years of hourly coastal imagery within a day at a quality we believe is challenging to obtain by a human operator. Ease of use of the workflow could still be improved by incorporating the vegetation detection step directly into the 24 MLP + 24 CNN steps.

Two first examples indicate possible future use cases: water extent resulting from the image analysis well aligns with independent tide data, while anomalies in water extent area can be directed linked to local flooding induced by storm surges. Future analysis could focus on more subtle change patterns in different sand classes, corresponding availability of sand for sand transport and its relation to vegetation dynamics and understanding 3D topographic variations measured using UAV laser scanners or the like at e.g. monthly intervals.

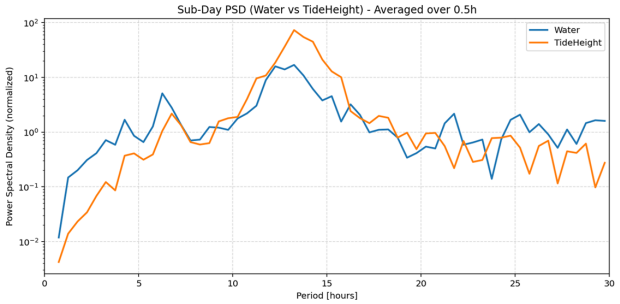

Figure 11. Power Spectral Density graphs derived from 2 independent sources. Blue: Image based water coverage. Orange: measured tide height at the nearby gauge station of Hoek van Holland.

Acknowledgments

This research has been partly supported by the Netherlands Organization for Scientific Research (NWO, grant no. 20014) as part of the Open Technology Programme.

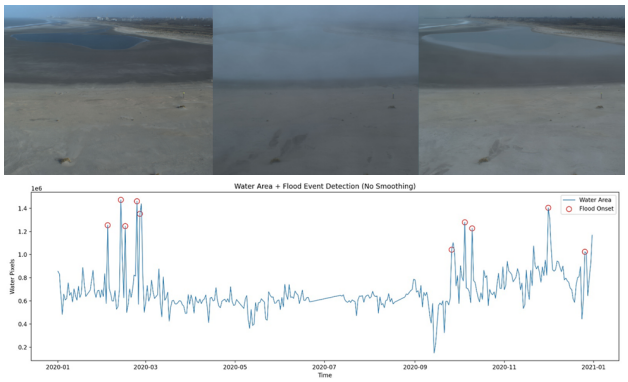

Figure 12. **Top:** Camera 1 images: Before, during and after storm surge. **Bottom:** Storm surges in 2020, automatically detected from analyzing water area extent in semantic segmentation results of Camera 1 imagery. Storm surges are indicated by red circles in a time series of the water area per image in 2020.

References

Al-Haija, Q. A., Smadi, M. A., Zein-Sabatto, S., 2020. Multi-class weather classification using ResNet-18 CNN for autonomous IoT and CPS applications. *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 1586–1591.

Arin, L., Almeda, R., Sampedro, N., Reñé, A., Blasco, D., Calbet, A., Camp, J., Estrada, M., 2014. Foam events due to a Phaeocystis bloom along the Catalan Coast (NW Mediterranean). *Harmful Algae*.

Barbero-García, I., Kuschnerus, M., Vos, S., Lindenbergh, R., 2023. Automatic detection of bulldozer-induced changes on a sandy beach from video using YOLO algorithm. *International Journal of Applied Earth Observation and Geoinformation*, 117, 103185.

Bauer, B., Davidson-Arnott, R., Hesp, P., Namikas, S., Ollerhead, J., Walker, I., 2009. Aeolian sediment transport on a beach: Surface moisture, wind fetch, and mean transport. *Geomorphology*, 105(1), 106-116.

Bishop, C. M., Bishop, H., 2023. *Deep Learning - Foundations and Concepts*. Springer.

Chen, J., Ma, Y., Dai, W., Chen, X., Li, S., 2025. Exploring Beyond Logits: Hierarchical Dynamic Labeling Based on Embeddings for Semi-Supervised Classification. *IEEE Access*, 13, 54611-54621.

De Schipper, M. A., De Vries, S., Ruessink, G., De Zeeuw, R. C., Rutten, J., Van Gelder-Maas, C., Stive, M. J., 2016. Initial spreading of a mega feeder nourishment: Observations of the Sand Engine pilot project. *Coastal Engineering*, 111, 23–38.

Garg, A., Pan, X.-W., Dung, L.-R., 2022. LiCENt: Low-light image enhancement using the light channel of HSL. *IEEE Access*, 10, 33547–33560.

Gonzalez, R. C., Woods, R. E., 2018. *Digital image processing*. Pearson, New York, NY.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 770–778.

Holman, R. A., Stanley, J., 2007. The history and technical capabilities of Argus. *Coastal Engineering*, 54(6-7), 477–491.

Hulskamp, R. L., Pregnolato, M., De Vries, S., 2025. Dynamics of engineered coastal dune landscapes at the Zandmotor. *Discover Geoscience*, 3(1), 244.

Lianqiang, S., Junli, G., Haijiang, L., Qinghua, Y., 2019. Application progress and prospect of Argus system in beach research in China. *Advances in Earth Science*, 34(5), 552.

Lindenbergh, R., Dewez, T., Hulskemper, D., 2025. Assessing 3D morphological dune changes using medial axes. *Proc. 6th Joint International Symposium on Deformation Monitoring (JISDM 2025), Karlsruhe, 07.-09.04.2025*, 1–8.

McFall, B. C., Young, D. L., Whitmeyer, S. J., Buscombe, D., Cohn, N., Stasiewicz, J. B., Skaden, J. E., Walker, B. M., Stever, S. N., 2024. SandSnap: Measuring and mapping beach grain size using crowd-sourced smartphone images. *Coastal Engineering*, 192, 104554.

Naidu, M., Rajesh Kumar, P., Chiranjeevi, K., 2018. Shannon and Fuzzy entropy based evolutionary image thresholding for image segmentation. *Alexandria Engineering Journal*, 57(3), 1643-1655.

Noothout, J. M. H., Lessmann, N., van Eede, M. C., van Harten, L. D., Sogancioglu, E., Heslinga, F. G., Veta, M., van Ginneken, B., Isgum, I., 2022. Knowledge distillation with ensembles of convolutional neural networks for medical image segmentation. *Journal of Medical Imaging*, 9(5), 052407.

Nurunnabi, A., Teferle, F. N., Li, J., Lindenbergh, R., Hunegnaw, A., 2021. An efficient deep learning approach for ground point filtering in aerial laser scanning point clouds. *ISPRS Archives*, 43, 31–38.

Pan, Y., Ayoub, N., Schneider-Kamp, P., Flindt, M., Holmer, M., 2022. Beach Wrack Dynamics Using a Camera Trap as the Real-Time Monitoring Tool. *Frontiers in Marine Science*, 9.

Rabehi, W., Larabi, M. E. A., Benabbou, O., Kreri, S., Dellani, H., 2023. Sandy beach mapping using a deep learning approach: potential method for automated monitoring of Algerian coastal erosion. *Journal of Coastal Research*, 39(5), 949–959.

Rombado, L., Orescanin, M., Orescanin, M. M., 2024. Uncertainty-aware aerial coastal imagery pattern recognition through transfer learning with ImageNet-1K variational embeddings. *IEEE Access*, 12, 130866–130883.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, Springer, 234–241.

Rutten, J., Ruessink, B. G., Price, T. D., 2018. Observations on sandbar behaviour along a man-made curved coast. *Earth Surface Processes and Landforms*, 43(1), 134-149.

Vandebroek, E., Lindenbergh, R., Van Leijen, F., De Schipper, M., De Vries, S., Hanssen, R., 2017. Semi-Automated Monitoring of a Mega-Scale Beach Nourishment Using High-Resolution TerraSAR-X Satellite Data. *Remote Sensing*, 9(7).

Wijnberg, K. M., Holman, R. A., 1997. Cyclic bar behavior viewed by video imagery. *3rd Coastal Dynamics Conference 1997*, American Society of Civil Engineers, 375–384.