Master's Thesis

Utrecht University

Faculty of Geosciences
Department of Physical Geography
Earth Surface and Water

Image-based beach state classification using convolutional neural networks (CNN)

Author:
Stan Oerlemans

Supervisors:
Dr. Timothy Price
Dr. Wiebe Nijland

October 13, 2022

Final version

Abstract

Subtidal sandbars are features characterizing many sandy coasts. Unravelling their dynamics is crucial for the understanding of nearshore sediment pathways. Subtidal sandbars exhibit complex patterns, ranging from shore-parallel ridges of sand to an alongshore alternation of shore-attached bars and rip-channels that can be classified into distinct equilibrium states. Recognition and classification of these beach states are not trivial and hitherto involved manual classification or pre-defined image features. The tremendous progress in data-driven learning in image recognition has led to the first automated classification of single-barred beach states from video (Argus) imagery, using a Convolutional Neural Network (CNN). We built upon this work to extend this method for classifying beach states in a double-barred system. We used Transfer Learning (TL) to fine-tune the pre-trained network of ResNet50. Our data consisted of labelled single-bar *daytimex* images from the beaches of Narrabeen (Australia) and Duck (US). We complemented these datasets with 9+ years of daily averaged low-tide images of the double-barred beach of the Gold Coast (Australia), with individual labels for the inner and outer bars. We assessed seven different CNNs of which each model was tested on its own test data, the *self-tests*, and on the test data of the other locations, the *transfer-tests*. Three models were trained with only the single-barred data; one at Duck, one at Narrabeen and one at data of both Duck and Narrabeen. The skills of these models were comparable with previous work (F1 scores at Duck, Narrabeen and their combination of, respectively: 0.89, 0.76 and 0.78). When the model trained on the data of both Duck and Narrabeen was tested on unseen data of the Gold Coast, we achieved poor performance for both the inner and outer bar (F1 score at inner bar = 0.32 and outer bar = 0.55). In contrast, we trained two models with only the double-barred beach data, with skills in the self-tests comparable with the skills in the self-tests of the single-barred models (F1 scores of 0.73 and 0.88 trained with inner or outer bar labels, respectively). The last two models were trained with data from Duck, Narrabeen and incrementally added data of the Gold Coast, with either the inner or outer bar labels. The tests with these models showed that with an additional 20% of data, reasonable F1 scores could be achieved. Moreover, with an additional 33% of data, F1 scores comparable (within 15%) to the self-test cases at each location were achieved. It mattered which of the two bars was used for training the model. Training with the outer bar led to overall higher performances, except at the inner bar. When trained on the inner bar, only the performance on detecting the inner bar was higher than the models trained with the outer bar. Additionally, we saw that when trained with data from multiple locations, more data coming from one location did not always positively affect the model's performance. However, the larger diversity of images coming from more locations allowed the transferability to other locations.

Using a model visualization technique called Guided-CAM, we confirmed that the sandbar features consistent with the beach states were used in the predictions made by the CNN.

Keywords: Beach state, Machine learning, Deep learning, CNN, Nearshore morphology, Transfer learning, Argus, Video imaging.

Preface

This report marks the end of my MSc Thesis, written in partial fulfilment of my studies in Physical Geography. I have learned a great deal during this project, and I am grateful for the opportunity to present my research at the NCK Days 2022. I would like to thank my direct supervisors, Timothy Price and Wiebe Nijland, for their enthusiasm, knowledge and guidance. I am especially grateful for the space and support they gave me during this thesis. Special thanks to Ashley Ellenson, who, despite her travels, was willing to answer my questions and give feedback on the draft version of my thesis. At last, I would like to thank my family and friends. They were always willing to listen to my frustrations, and they helped me put things into perspective whenever needed. I feel privileged to have their unconditional support.

Stan Oerlemans

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network.
BN	Batch Normalization.
CAM	Class Activation Map.
CE	Cross Entropy.
CIL	Coastal Imaging Lab.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
D	Dissipative.
Deconv	Deconvolution.
DL	Deep Learning.
ENSO	El Niño-Southern Oscillation.
FC	Fully Connected.
FN	False Negative.
FP	False Positive.
FRF	Field Research Facility.
GAP	Global Average Pooling.
GBP	Guided Backpropagation.
GPU	Graphics Processing Unit.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
LBT	Long-Shore Bar and Trough.
LTT	Low Tide Terrace.
ML	Machine Learning.
MLP	MultiLayer Perceptron.
NN	Neural Network.
RBB	Rhythmic Bar and Trough.

Ref	Reflective.
ReLU	Rectified Linear Unit.
ResNet	Residual Network.
RNN	Recurrent Neural Network.
TBR	Transverse Bar and Rip.
Timex	Time-Exposure Images.
TL	Transfer Learning.
TN	True Negative.
TP	True Positive.

Chapter 1

Introduction

1.1 Beach States

The understanding of nearshore coastal regions has always been of interest in the coastal community. In particular, the high dynamic response of nearshore features such as sandbars, troughs and rip channels (Alexander and Holman, 2004). These features are fundamental in controlling advection, mixing and distributing nearshore nutrients and pollutants (Grant et al., 2005), as well for determining the recreational safety and erosion risk (Castelle et al., 2016; Holman et al., 2006; Tătui and Constantin, 2020). Hence, understanding the temporal and spatial variability of the nearshore morphology is essential to both scientists and coastal management experts. Wave breaking and wave-driven currents constantly rearrange nearshore sediment into complex, consistently-occurring patterns, often referred to as sandbar morphology. Moreover, Wright and Short (1984) used these patterns, which range from shore-parallel ridges to an alongshore alternation of shore-attached bars and rip channels (Price and Ruessink, 2011; Short and Aagaard, 1993), to create a widely used beach state classification scheme for single-barred coasts (Figure 1.1). They identified three basic beach types: reflective, intermediate and dissipative, composed of six beach states in total with distinct sandbar configurations. The two end members Reflective (Ref) and Dissipative (D) were identified to relate to low and high-energetic conditions. The intermediate states were identified as Long-Shore Bar and Trough (LBT), Rhythmic Bar and Trough (RBB), Transverse Bar and Rip (TBR), and Low Tide Terrace (LTT) (Wright and Short, 1984). During a full accretionary sequence, the bars migrate onshore and mainly advance through all intermediate states towards the reflective state over several days to weeks. However, the large amount of energy inherent to erosional sequences often causes the bar to jump to a higher state within hours, briefly acquiring the erosional intermediate states (Price and Ruessink, 2011).

Initially, this classification scheme was used for single-barred beaches and did not apply to multi-barred beaches. However, in a double-barred beach, the dynamics slightly differ from the dynamics of single-barred beaches. The outer bar evolves more slowly through the bar states and is typically more upstate than the inner bar. Furthermore, the outer bar rarely has been observed to reach the reflective state and often occurs as a quasi-inactive feature during low-energetic periods. On the other hand, the inner bar is far more dynamic and may progress through all the bar states (Price and Ruessink, 2011).

Eventually, the scheme of Wright and Short (1984) was extended to be applicable on multi-barred beaches, and a multi-bar state model was devised (Short and Aagaard, 1993). In this model, each bar can go through the same states as in the original single-bar model.

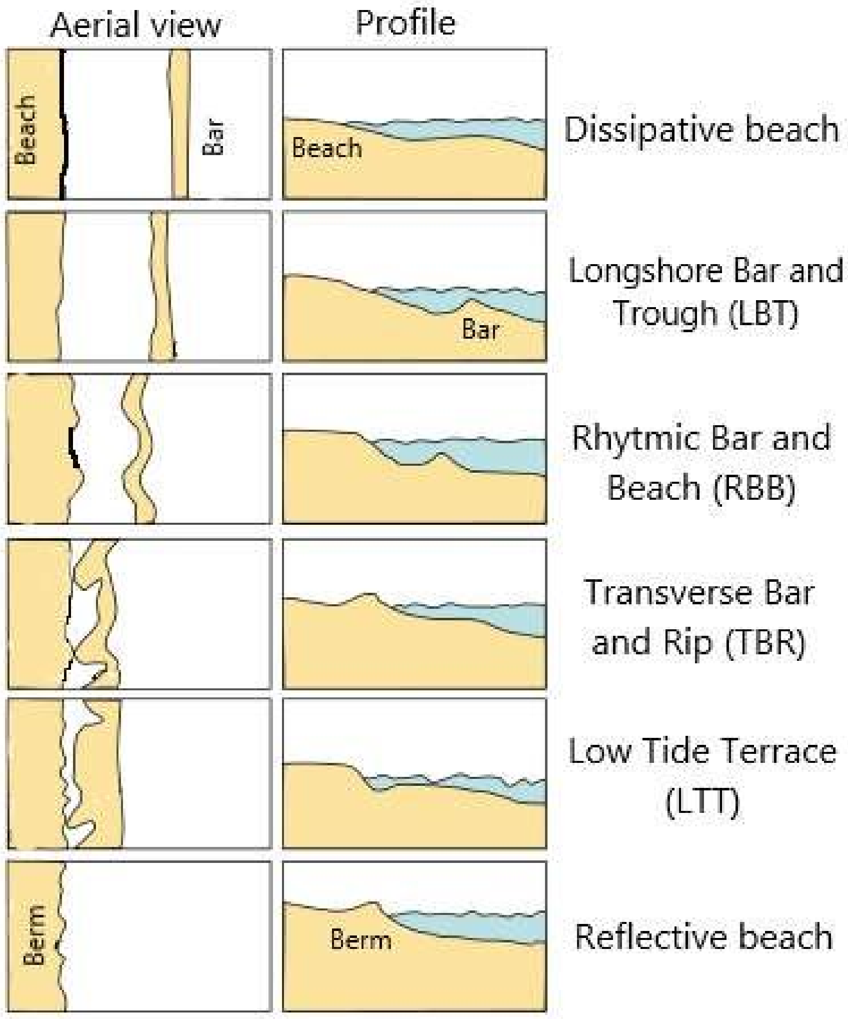


Figure 1.1: A schematic view of the six beach states. (Adopted from: Abessolo Ondoa (2020))

1.2 Nearshore Bar Research

Traditional in-situ measurements have often been used to study sandbars and their dynamics. However, this method is spatially limited and is expensive, both in terms of time and money. In addition, these field-based experiments are often limited because of the surf zone's harsh and potentially dangerous conditions (Holman and Stanley, 2007; Holman and Haller, 2013). Luckily, many optical signatures in the nearshore can be exploited. For example, variations in water reflection with the slope of the sea surface result in the visibility of ocean waves. Thus, ocean waves' period, wavelength and direction can be seen. Furthermore, wave breaking is very obvious to the eye as bright patches of foam on the water surface. Since waves tend to break in shallow water, locations of concentrated foam can be used to locate the position of submerged sandbars (Holman and Stanley, 2007).

The limitations of in-situ measurements and the benefits of optical signatures in nearshore zones led to the broad application of remote sensing techniques such as video monitoring systems, permitting frequent, spatially extensive and high-resolution data monitoring (Alexander and Holman, 2004; Aleman et al., 2017). Using these techniques for nearshore research enabled the development of innovative algorithms and methods. These algorithms and methods have been used for the extraction of geophysical signals from image and video data, providing new insights into the dynamics of nearshore morphology (Aarninkhof and Ruessink, 2002; Alexander and Holman, 2004; Davidson et al., 2007; Holman and Haller, 2013; Lippmann and Holman, 1989; Plant and Holman, 1997).

The Argus system is the most frequently used video monitoring system in nearshore research. This system was developed by the Coastal Imaging Lab (CIL) at Oregon State University. Since its inception, it has extended significantly with approximately 50 operational sites worldwide, delivering a considerable contribution to the needs of coastal research (Holman and Stanley, 2007; Bracs et al., 2016). Argus imagery has extensively been used to derive specific morphological features (Price and Ruessink, 2011). In different kinds of Argus products, such as Time-Exposure Images (Timex), white bands are visible on locations of sustained wave breaking over topographic features such as sandbars, and the resulting spatial pattern can be categorized into a beach state. In addition, previous studies used Argus imagery to derive features such as the position of the sandbar, shoreline or rip locations (Plant et al., 2006; Armaroli and Ciavola, 2011; Van Enckevort and Ruessink, 2001). However, while these previous methods successfully identify these specific morphological features from imagery, they require several preprocessing steps to extract them. Moreover, these methods are mainly fitted for specific sites, meaning they may not be successfully transferred to other locations (Contardo and Symonds, 2015). To improve our understanding of coastal processes at different sites, nearshore optical remote sensing data should be exploited in a generic and scalable way (Splinter et al., 2018; Smit et al., 2007; Holman and Haller, 2013). In addition, many of the current mathematical formulations cannot extract meaningful physical information from remotely sensed imagery, such as depth limited wave breaking. However, machine learning (ML) techniques may be a potential path forward.

ML algorithms have previously been applied in various coastal applications and have shown that due to their highly flexible mathematical formulations, they can detect physically relevant patterns in imagery. Pape et al. (2007) applied a Recurrent Neural Network (RNN) to study nearshore sandbar behaviour. Pape and Ruessink (2011) used a Neural Network (NN) to study the predictability of nearshore sandbar migration. Furthermore, Kingston et al. (2000) used an Artificial Neural Network (ANN) to produce a model that estimates cross-shore bar location from raw Argus images of double-barred beach systems. It has been suggested that an ANN is the best currently available method for sandbar extraction from imagery (Román-Rivera and Ellis, 2019; Ribas et al., 2010).

Tremendous progress has been made in ML in recent years, particularly in image recognition and classification. This progress is primarily due to the application of Convolutional Neural Networks (CNNs). CNNs are a specific class of ML that has been shown to produce a state-of-the-art performance for various image recognition and classification tasks (Gu et al., 2018; Hoshen et al., 2015; Krizhevsky et al., 2012; Parkhi et al., 2015). These state-of-the-art performances could mainly be achieved due to the use of transfer learning, a technique in which complex deep CNN architectures are pre-trained on large public datasets, such as ImageNet (Krizhevsky et al., 2012), and used in other image classification problems. In general, there are three different approaches when using an already existing CNN architecture. The first method is the conventional method of training a model from scratch. Secondly, the pre-trained parameters can be fine-tuned on the preferred dataset. And thirdly, CNNs can be used as a feature extractor only to classify new images. The approach with the best results depends on the dataset's characteristics and on the dataset on which the CNN has been pre-trained. In general, fine-tuning is the better alternative when the target dataset is small, whereas fully training the network from scratch is preferable when the target dataset is large (Castelluccio et al., 2015).

In a recent study, Ellenson et al. (2020) trained a CNN from scratch for the automated classification of single-barred beach states from Argus imagery of the coasts of Duck (US) and Narrabeen (Australia). They implemented various data combinations to train and test the models and compared their results to inter-labeller agreements. They showed that CNNs trained and tested at the same site had the comparable skill to the inter-labeller agreement with a higher overall skill at Duck than at Narrabeen. For both sites, the highest accuracy was in classifying the low-energy Ref and LTT states, while the lowest skill of the CNN was classifying the rhythmic states of RBB (at Narrabeen) and TBR (at Duck). When the CNNs trained with the data of one location were tested on the other site's data, the performance decreased drastically. However, they showed that the models trained on data from multiple locations achieved performances compared to the CNNs trained with the data from a single location, with at least 25% of the training data coming from each location.

1.3 Problem Statement and Research Questions

Ellenson et al. (2020) showed that single-barred beach systems can be successfully classified using CNNs. However, a study on the application of CNNs on double-barred beaches was not done yet. Hence, in this thesis we built upon the work of Ellenson et al. (2020) and extended their model[1] for the automated classification of each sandbar contained within the Argus imagery of a double-barred beach system. To evaluate the proposed method we answered the following key questions:

1. How do pre-trained networks, fine-tuned with single-barred beach imagery perform on the classification of beach states of double-bar beach systems?

[1]https://github.com/anellenson/DeepBeachState

2. How do pre-trained networks, fine-tuned on beach imagery of single-bar beaches and complemented with data of a double-barred beach system perform on the classification of beach states of double-bar beach systems?

These questions were answered by conducting two experiments. In the first experiment, we evaluated our model by conducting experiments similar to the experiments by Ellenson et al. (2020), using only single-barred beach datasets. Furthermore, we tested whether the same single-bar beach models could be applied to a double-bar beach. In the second experiment, we added different proportions of the double-barred beach data to the single-barred beach data and evaluated the models' performances. Both experiments are analysed by the following sub-questions:

- How are the overall model performances in terms of model learning and skills?
- How are the skills in the per-class classification of single- and combined-location models?

1.4 Thesis Outline

This thesis is organized into six chapters. Chapter 1 provides an introduction, followed by the background information in Chapter 2. Chapter 3 discusses the proposed method, including the experimental setup. Chapter 4 is primarily concerned with the results of this study. In Chapter 5, the CNNs' performances and results are presented, and eventually, we conclude with the main findings in Chapter 6.

Chapter 2

Background theory

This chapter lays the groundwork for this thesis and covers all the important background material concerning our method. First, the theory behind NNs is introduced, followed by the concepts of CNNs, and lastly, transfer learning as a training protocol is covered.

2.1 Neural Networks

A Neural Network (NN), or so-called Artificial Neural Network (ANN), is a subset of ML and forms the core of Deep Learning (DL) algorithms. In contrast with conventional ML, DL algorithms do not require any human-designed rules to operate. Rather, it uses large amounts of data to map the given input to a specific label. In addition, DL can automate the learning of feature sets and enables learning and classification to be achieved in a single shot (Alzubaidi et al., 2021). DL algorithms exist out of multilayered neural networks, inspired by the neurons that make up humans' and animals' brains and nervous systems. These systems consist of billions of neurons connected through even more connections. NNs mimic how neurons can pass information to other neurons by sending electrical signals through the connections. Together the neurons and connections enable information to be processed.

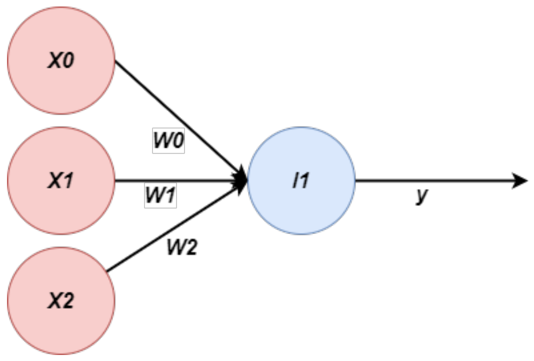


Figure 2.1: Example of an artificial neuron, where *X0*, *X1* and *X2* are the inputs *W0*, *W1* and *W2* the weights, *l1* the activation function and y the output.

Figure 2.1 shows an example of a typical artificial neuron. It consists of a set of nodes connected by edges, representing the neurons and the synapses of a biological system. Every edge is associated with a weight, the so-called learnable parameters, representing the strength of the connection between the nodes. For example, when the weight from node A to node C has a greater magnitude than the weight from node B to node C, it means that the first connection has a greater influence than the latter. In addition, every neuron has an activation function that decides whether a neuron should be activated or not. This activation function is like a threshold; if the output of a node is higher than the threshold value, the node is activated, sending data to the next layer of the network (Lipton et al., 2015). The activation function is further described in Section 2.1.1.

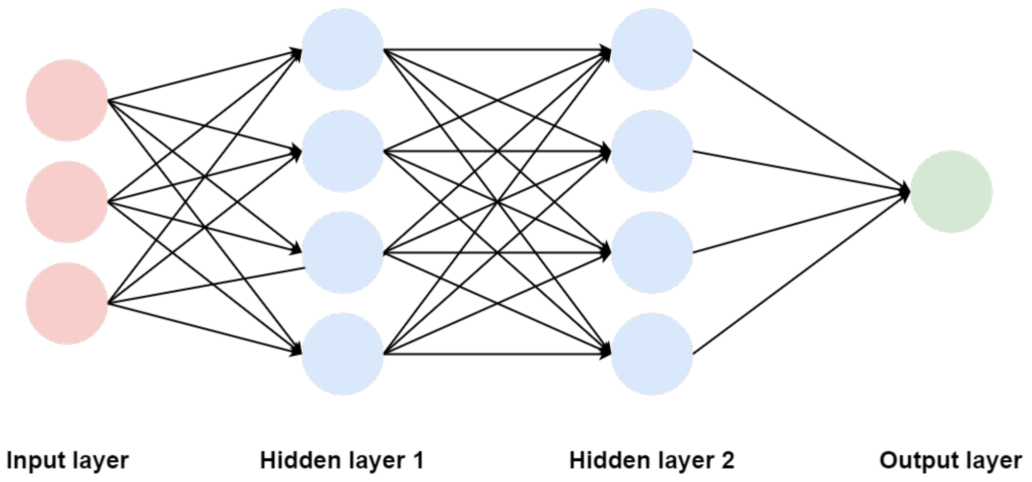


Figure 2.2: Example of a simple feed-forward neural network with an input layer, two hidden layers and an output layer. This network is fully connected, meaning that all nodes are connected to all nodes of the previous layer.

The simplest type of NN is a feed-forward NN, which is illustrated in Figure 2.2. A feed-forward NN is comprised of a set of artificial neurons, one or multiple hidden layers, and an output layer. Edges connect the nodes within a feed-forward NN to all nodes in the preceding and subsequent layer. Besides feed-forward NNs, there are many other types of NNs, each with its advantages and disadvantages. For example, RNNs are commonly used for temporal analyses for in example for natural language processing and speech recognition (Mikolov et al., 2010; Hoshen et al., 2015). In contrast, CNNs are specifically designed for spatial analyses and to process pixel data. Due to their ability of self-learning and automatically identifying relevant features while at the same time resulting in state-of-the-art classifications, they have become the most famous and commonly applied DL algorithms in classification tasks(Gu et al., 2018; Simonyan and Zisserman, 2014). Hence, in the next sections, CNNs are discussed in more detail.

2.1.1 Convolutional Neural Networks

Similar to NNs, CNNs are feed-forward networks that can be trained by backpropagation. However, where human and animal brains have inspired NNs, CNNs are more specifically inspired by the complex sequence of cells that forms the visual cortex (Hubel and Wiesel, 1962). They are designed to learn spatial hierarchies of features and therefore are often utilized in pattern recognition. They take advantage of the assumption that the input consists of three dimensions. This allowed image-specific features into the network's architecture and made the network more suited for image-focused tasks (Goodfellow et al., 2016).

CNNs differentiate from NNs in the sense that NNs are fully connected (FC). This means that all nodes are connected to all nodes in the preceding layer and that each connection in a NN has its own trainable weight. However, complex classification tasks, such as image classification, require several layers of connected neurons to be active at the same time (Castelluccio et al., 2015). Therefore, CNNs have a limited receptive field, meaning that the nodes are not connected to all nodes in the previous scene. The cells in a CNN only sense small regions rather than the whole scene, similar to a virtual cortex. In addition, CNNs apply weight sharing between connections, thus, requiring an extremely small number of parameters, simplifying the training process and speeding up the network (Goodfellow et al., 2016).

A common type of CNN is shown in Figure 2.3. It consists of numerous convolutional layers with a Rectified Linear Unit (ReLU) activation function, which precedes sub-sampling (pooling) layers and Batch Normalization (BN) while ending in a fully connected layer. The remaining part of this section covers these key concepts in more detail.

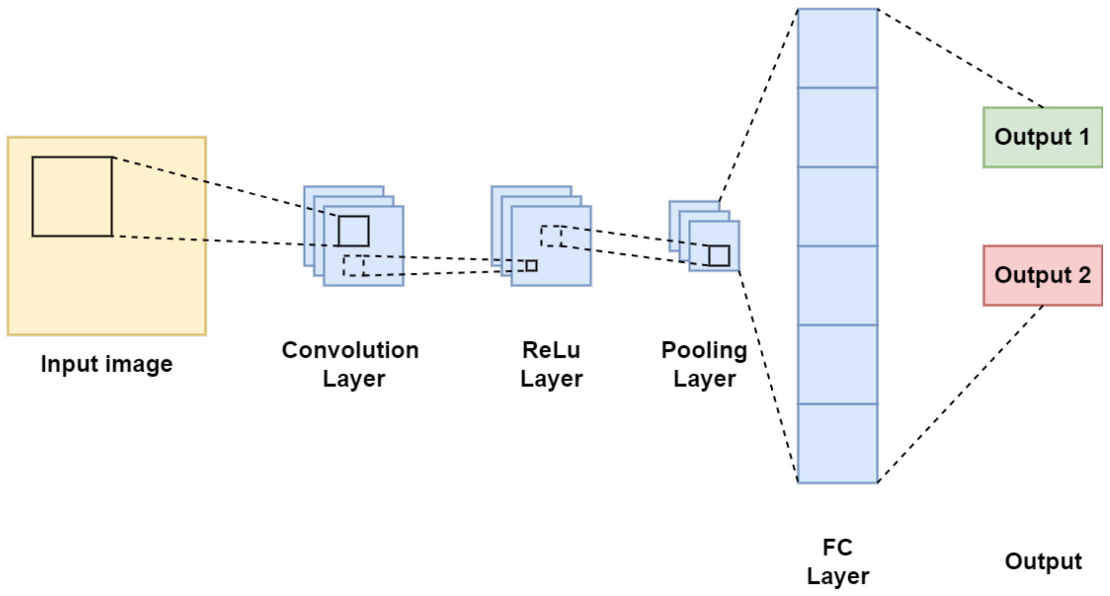


Figure 2.3: Schematic view of the building blocks of a typical Convolutional Neural Network. As input an image fed to convolutional layers, a ReLU activation function, a pooling layer and eventually a fully connected layer before the output.

Convolutional Layers

Convolutional layers are the most important building blocks of CNNs. In these layers, the majority of the computations occur. They exist of nodes that are typically arranged in three dimensions: width, height and depth, and of a collection of convolutional filters, the so-called kernels. The filters move over the regions of nodes of the preceding layer in a moving window fashion (Alzubaidi et al., 2021). A simplification of this convolutional operation is shown in Figure 2.4. In this figure, we have a 4x4 image, in which each square represents a pixel. A 2x2 filter is applied with stride 1, which determines the step size at which the filter moves over the input. Next, the dot product between the input and the filter is determined, and the corresponding values are multiplied and summed to create a single scalar value. This process is repeated until the filter has slid over the complete input. All these single-scalar values combined represent the output feature map. In the figure, the values of the filter stay consistent. However, in practice, these values are updated during network training, the so-called learning process. Eventually, the kernels can detect and distinguish patterns and features in an image.

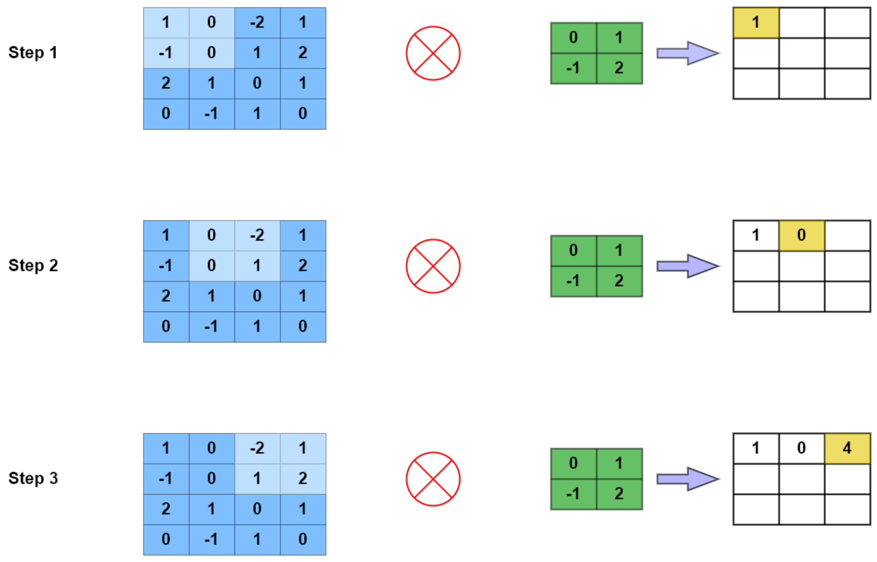


Figure 2.4: Example of the process and primary calculations of a convolutional layer. The 2x2 light-blue square represents the location of the filter moving with a stride of 1 over the 4x4 dark-blue input. The green 2x2 square represents the filter and the single-scalar output values are shown in yellow.

NNs use matrix multiplication to describe the interaction between input and output nodes. For example, in classic MultiLayer Perceptron (MLP), the input node would have affected all the output nodes (Gardner and Dorling, 1998). In contrast, CNNs use sparse interactions, which means that in the convolutional layers, the filters are made smaller than the input image. Hence, not all the input nodes interact with the output node. Due to sparse interactions in CNNs, only the nodes corresponding with the width of the kernel are affected by the input node. These sparse interactions are memory-effective, resulting in a small number of required weights or connections, and therefore the memory required to store these is also small.

Pooling Layers

Pooling layers are often placed between convolutional layers. They sub-sample the output feature maps, in which they maintain the useful information while eliminating the irrelevant information. In other words, pooling layers shrink large-size feature maps and transform them into small feature maps. Pooling layers serve two main objectives: the first is to lessen the computational cost by reducing the number of parameters and weights, and the second purpose is to control overfitting, which is further discussed in Section 2.1.1 (Gholamalinezhad and Khosravi, 2020).

Many pooling methods are available. Two pooling methods that are significant for this thesis are Max Pooling andGlobal Average Pooling (GAP). Both pooling methods are illustrated in Figure 2.5 and discussed in more detail below.

1. **Max Pooling**: During max pooling, the maximum value for the patches of a feature map is calculated and used to down-sample the convolutional output bands, reducing variability and creating a pooled (down-sampled) feature map. Max pooling is often used after a convolutional layer.
2. **Global Average Pooling**: The idea behind GAP is to generate one feature map for each corresponding category of the classification task at hand. It is designed to replace fully connected layers in CNNs (Lin et al., 2013). GAP has several benefits. First of all, this method is more native to convolutional structures by enforcing resemblances between feature maps and categories, causing the feature maps to be more easily interpreted as category confidence maps. Secondly, there is no parameter to optimize. Thus overfitting is avoided. Moreover, GAP sums out the spatial information, making it more robust to spatial translation.

For further reading on pooling methods see Gholamalinezhad and Khosravi (2020).

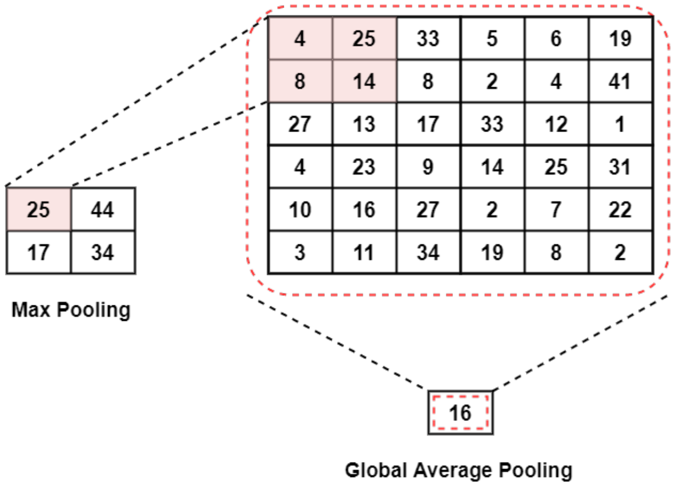


Figure 2.5: A schematic visualization of the processes of max pooling and global average pooling.

Activation Functions

We shortly mentioned activation functions in Section 2.1. The core function of this function is to map an input signal into an output signal which is in turn fed as input to the next layer. When no activation function is applied in a NN, the NN will act as a linear regression model, and the output signal will be a linear function (Sharma et al., 2017). Although linear equations are simple and easy to solve, their capabilities are limited, and they are not fitted to learn and recognize complex structures from data.

As a NN is required to learn, represent and process any data and any complex function, activation functions were added. These activation functions made the network more dynamic and allowed the extraction of complex and complicated information. Non-linear activation functions were applied after each layer with weights. The input of these functions is determined by computing the weighted summation of the neuron along with its bias. The core function of these activation functions is to map the computed input to the output in a non-linear way, enabling the CNN to learn extra-complicated structures (Alzubaidi et al., 2021). The activation functions that are important for this thesis are the Rectified Linear Unit (ReLU) activation and the SoftMax functions described below.

1. The Rectified Linear Unit (ReLU) activation function is the most commonly employed function in CNNs. The main benefit of using ReLU is that not all neurons are activated at the same time. Thereby lowering the computational load and greatly reducing the time required to train CNNs. The ReLU activation function is represented by:

$$ReLU(x) = max(0, x) \tag{2.1}$$

   The ReLU activation function converts the positive integers of the input to positive numbers and a neuron will only be deactivated when the output of the linear transformation is zero.

2. The SoftMax activation function is mainly used for multi-class classification problems and is represented by:

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}} \qquad for \qquad i = 1, ..., N \tag{2.2}$$

   In which e^{z_i} represents the non-normalized output from the previous layer, N the number of neurons in the output layer and i the location of the i-th neuron. The output of this function is the probability of all data points of each class $y \in \{0, 1\}$. When the SoftMax activation function is used in a multi-class task, then the output layer of the network will have the same number of neurons as the number of classes in the target dataset (Alzubaidi et al., 2021; Sharma et al., 2017).

For further reading on activation functions, see Sharma et al. (2017).

Fully Connected Layers

Convolutional and pooling layers are very good in extracting features from the input images. However, they are not fitted for classification tasks. Therefore, a Fully Connected (FC) layer is located on top of the convolutional and pooling layers (Figure 2.3). In an FC layer, each neuron is connected to all neurons of the previous and the next layer, receiving input from the last pooling or convolutional layer and often followed by a SoftMax function. The output represents the final CNN output, containing a number of neurons equal to the number of classes in the dataset on which the model is trained.

Loss Functions

Above, several layers are described, which play a vital role in the architecture of a CNN. Like a feed-forward NN, the training of a CNN happens by utilizing a loss function that calculates the predicted error created across the training samples, meaning that it indicates the difference between the actual values, the so-called labels, and the network's predicted values, the so-called predictions. The weights of the network's connections are trained iteratively through backpropagation and an optimization algorithm. In general, the objective of the optimization algorithm is to minimize the loss function; the smaller the loss, the better the model. The loss function important for this thesis is the Cross Entropy (CE) loss function. This function is employed to measure the CNN model's performance. Usually, CE is calculated using the probabilities calculated by an activation function such as the SoftMax activation function. In addition, CE is often used as a substitution for the square error loss function in multi-class classification tasks. It is represented by:

$$\begin{aligned} CE &= -\sum_{i=1}^{N} \vec{y_i} log(SoftMax) \\ &= -\sum_{i=1}^{N} \vec{y_i} log(y_i) \end{aligned} \tag{2.3}$$

Where N denotes the number of classes, y_i the i-th prediction class, and $\vec{y_i}$ the i-th true class. The SoftMax function is defined in Equation 2.2.

For further reading on loss functions, see Janocha and Czarnecki (2017).

Regularization

One of the key challenges of working with CNNs is model generalization. Model generalization refers to the ability of a model to adapt to new, unseen data drawn from the same source as the training data. When the performance of a model is very high on the training data, while the performance on the new data is low, we speak of *overfitting*. In contrast, when the model is unable to learn sufficiently from the training data and performs poor on both the training and the unseen data, we speak of *underfitting*.

To avoid overfitting and help the model learn a sufficient amount from the train data, model regularization has become an important aspect in image classification tasks. Model regularization is described by Goodfellow et al. (2016) as *'Any modification we make to a learning algorithm intended to reduce its generalization error but not its training error.'* Two regularization methods are significant for this thesis:

1. **Data Augmentation**: Augmentation techniques include a collection of methods to apply geometric transformations such as rotation, flipping, translation and cropping, which can be used to improve attributes, increase the amount of data and avoid overfitting. Therefore, models can perform better when these techniques are employed (Shorten and Khoshgoftaar, 2019).

2. **Batch Normalization**: To improve training Ioffe and Szegedy (2015) proposed BN as a way to reduce the internal covariance shift. The internal covariance shift is defined as the change

in network parameters during training, complicating the training of ML systems. BN makes CNNs faster and more stable by adding additional layers and performing standardization and normalization on a batch of images rather than on a single input. The process of BN speeds up the training process, handles internal covariance shift, and smoothens the loss function.

For further reading on regularization techniques, see Nusrat and Jang (2018).

2.1.2 Residual Network

The CNN's architecture is a critical factor in its performance. Hence, many architectures have been presented in the past (Khan et al., 2020; Shrestha and Mahmood, 2019). Modifications on these architectures have eventually resulted in today's state-of-the-art models (Huang et al., 2017; Simonyan and Zisserman, 2014; Szegedy et al., 2016). These modifications included structural reformulation, regularization, and parameter optimization. Moreover, the most significant advancement in performance was due to the reorganization of computer's Central Processing Unit (CPU) and the development of novel architectures. It has become a trend to increase the depth and width of networks, increasing the number of layers and neurons. However, by training with deeper NNs, several issues occurred, such as the vanishing gradient problem and the degradation problem (Pricope, 2021). These problems have been tried to solve by the development of innovative architectures.

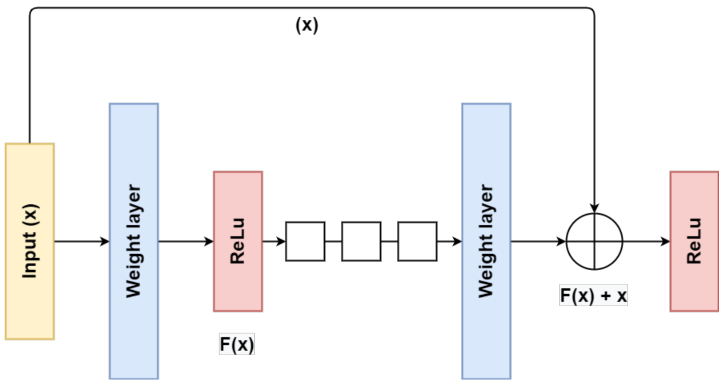


Figure 2.6: Conceptual block diagram of the ResNet architecture. The arrow denoted with x is the residual connection skipping the ReLU ($F(x)$) function and passing information directly to the next block ($F(x) + x$).

One of these architectures is the Residual Network (ResNet), of which the first version was introduced by He et al. (2016). ResNet is an architecture that differs from previously designed architectures. It was designed to create an ultra-deep network and, simultaneously, solve the problems of degradation and vanishing gradient. The novel idea of He et al. (2016) was to implement the concept of the bypass pathway in the form of residual learning, visualised in Figure 2.6. This figure displays the fundamental block diagram of the ResNet architecture with the corresponding residual connection, as designed by He et al. (2016). The residual connections in ResNet occur through identity mapping, meaning that the input is mapped to the output by an identity function represented by $(y = F(x) + x), 0)$. This function indicates that the information from a previous block (x), skips the transformations made within the blocks along the way $F(x)$ and is directly passed to a later block. This enables information to flow more efficiently through the network, allowing for better adjustment of the kernel weights and reducing training errors.

ResNet achieved state-of-the-art results; it won the first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2015 with a top-5 error rate of 3.57% (Russakovsky et al., 2015). Because of this success, ResNet has been continuously modified and enhanced, and deeper versions of ResNet were created. Nowadays, ResNet has become a family of models based on the number of layers, starting with 34 and going as deep as 1202 layers.

ResNet50

In this thesis, the architecture of ResNet50 was used, following He et al. (2016). Figure 2.7 shows a block diagram representing the architecture of ResNet50. As can be seen, the architecture consists of two parts, the feature extraction part and the classification part. The feature extraction part mainly consists of convolutional layers, whereas the classification part is an FC layer. After each convolution and before activation, BN is applied. The feature extraction part starts with a convolutional layer with a kernel size of 7x7 and 64 different kernels, all with a stride of size 2. A Max Pooling follows this layer. After the Max Pooling, four blocks connected by residual connections follow. In these blocks, we have kernels of different sizes. For example, block one consists of a 1x1, 64 kernel followed by a 3x3, 64 kernel, and at last, a 1x1, 256 kernel, repeated three times before passing to the second block. The following blocks exist out of convolutions with kernels of different sizes until we reach 49 layers. After these 49 layers, the feature extraction part is to an end. The classification part exists out of a GAP preceding the FC layer, including the SoftMax function, reaching the total amount of 50 layers. The 1000 in the FC layer corresponds to the amount of classes in the dataset on which the model is pre-trained. When employing this architecture to a new dataset, this value should correspond to the amount of classes in the target dataset.

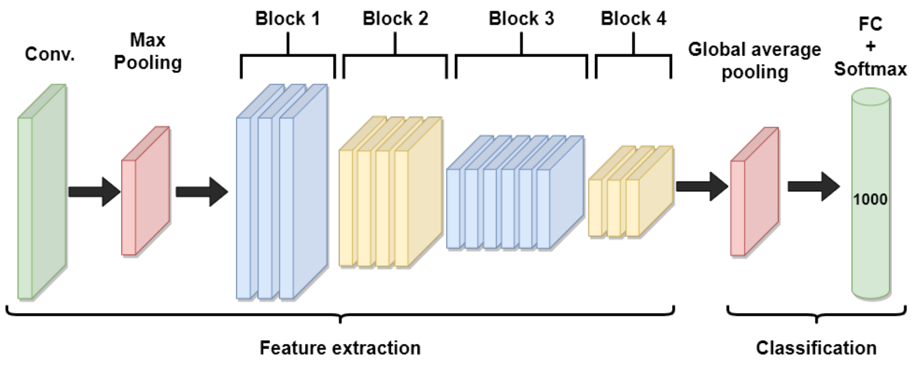


Figure 2.7: Block diagram of the ResNet50 architecture. The feature extraction part exist out of a convolutional layer, followed by a Max Pooling and 4 blocks connected by residual connections. The classification part exist of an FC layer with a SoftMax function. The 1000 in the FC layer corresponds to the amount of classes of the dataset on which the model is pre-trained.

2.2 Training Protocols

In the section above, we covered some of the key concepts related to CNN architectures. In this section, the training protocols concerning CNNs are discussed. There are several ways to train a CNN. Naturally, similar to classical supervised learning, a CNN can be trained from scratch, which means that all weights are randomly initialized before training. In training a model this way, a large amount of well-annotated data is required. However, requiring and annotating representable data has become the most common challenge associated with DL as it is costly and very time-consuming. In addition, training a model from scratch takes a relatively long time. Rather than trying to collect and annotate large datasets, it has become popular to use networks that have been pre-trained on vast datasets and use a technique called Transfer Learning (TL) to train their model (Pan and Yang, 2009).

2.2.1 Transfer Learning

The idea of TL came from educational psychology and is based on the knowledge that the human brain can apply the knowledge acquired by previous experiences to solve new problems (Weiss et al., 2016). For example, a person who wants to learn how to play the guitar would be better off with some earlier musical experience than learning to play the guitar from scratch. When this person already has some basic experience on the piano and already knows how to read music, learning the guitar would go much faster. In other words, the learning process is enhanced by using shared knowledge. A similar process is used in TL for CNNs, where models pre-trained on large datasets may already know how to detect certain features, such as edges, straight lines, and circles.

There are two different ways how this technique of TL is commonly applied. The network can be used either as a fixed feature extractor or as an initialization, respectively called *feature extraction* and *fine-tuning*. When using transfer learning for feature extraction, we only want to update the parameters of the classification layer. This is done by *'freezing'* all the weights in the feature extraction part, replacing the classifier and replacing the FC layer with a new, trainable layer. When TL is used for fine-tuning the network, the weights initialized from pre-training will be used, and all trainable parameters will be *'unfrozen'* and updated by training the model on a new dataset. The FC layer is still replaced to adjust the nodes corresponding to the number of classes for the classification task at hand (Tan et al., 2018).

Chapter 3

Methodology

In this thesis, we built on the idea of Ellenson et al. (2020) to apply CNNs for the classification of beach states. However, instead of training the architecture of ResNet50 from scratch, we used the pre-trained parameters to fine-tune the model on our datasets. The proposed method consists of five steps:

1. Data collection (Section 3.1)
2. Preprocessing the data (Section 3.2)
3. Initialize and reshape the network (Section 3.3)
4. Train and test the network (Section 3.4)
5. Model evaluation and interpretation (Section 3.5)

These five steps are covered in the next sections. Starting with the introduction of the data used in this thesis.

3.1 Field Sites and Datasets

Three different field sites were used, which are displayed in Figure 3.1. The datasets of these field sites exist out of Argus imagery processed to gray-scale images. Figure 3.2 shows examples of images from all three datasets, with each column containing images from a different site and each row containing a different beach state. The next sections outline the three different study sites, their characteristics and the corresponding datasets.

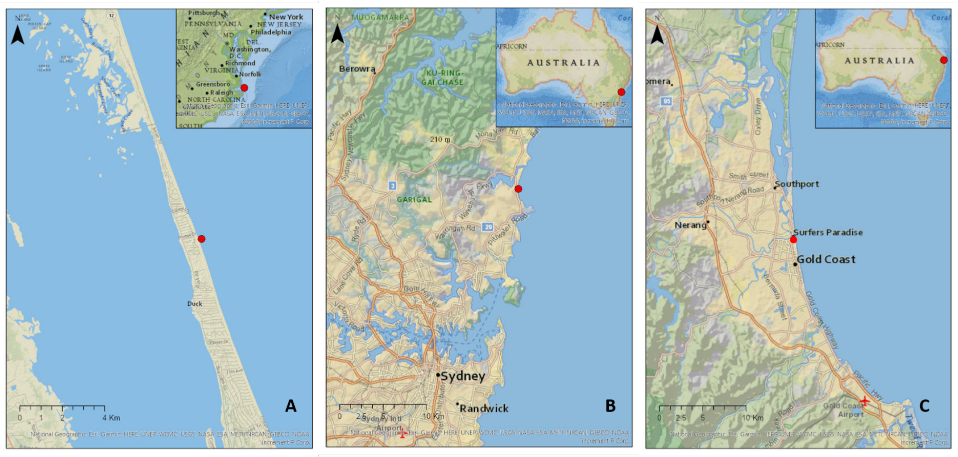


Figure 3.1: The location of the three field sites. A: Duck (US), B: Narrabeen (Australia) and C: Gold Coast (Australia).

3.1.1 Duck

Field Site

The sandy barrier beach of Duck is situated at the U.S. Army Engineer Research and Development Center Field Research Facility (FRF), North Carolina. As shown in figure 3.1A, Duck is located between two water bodies, the Currituck Sound in the west, while the Argus setup at the FRF is facing the North Atlantic Ocean in the east. In general, the beach of Duck is classified as an intermediate beach with one and, occasionally, two sandbars present. The waves coming from the North Atlantic Ocean vary seasonally, with higher incident wave energy in winter and lower incident wave energy in summer (Birkemeier et al., 1981). The annual significant wave height is 1.1 meters, with waves tending to come from the south during spring and summer and from the north during winter. In addition, winter storms consist of both extra-tropical and tropical cyclones (Lee et al., 1998). The mean spring tide range is micro-tidal at 1.2 meters.

Dataset Description

The dataset of Duck is similar to the dataset used by Ellenson et al. (2020) and consists of 680 manually classified and labelled single-bar images. The dataset was collected over 27 years, 1987-2014. The images were combined from various camera views; depending on the number of cameras installed and functional at the Argus system, this amount ranged from one to three. The view of these cameras spanned the beach area north of the FRF pier. Effects of peer lighting due to the angle of the sun or cloud cover, or a lack of wave-breaking signal due to low waves or high tide, were reduced by using *daytimex* images. These images represent the temporal mean of all frames collected during a single day, thus averaging out tidal dependencies and natural modulations in wave-breaking and removing persistent optical signatures. This eventually results in a picture of beach morphology over the surf zone (Holman and Stanley, 2007). Furthermore, the daytimex

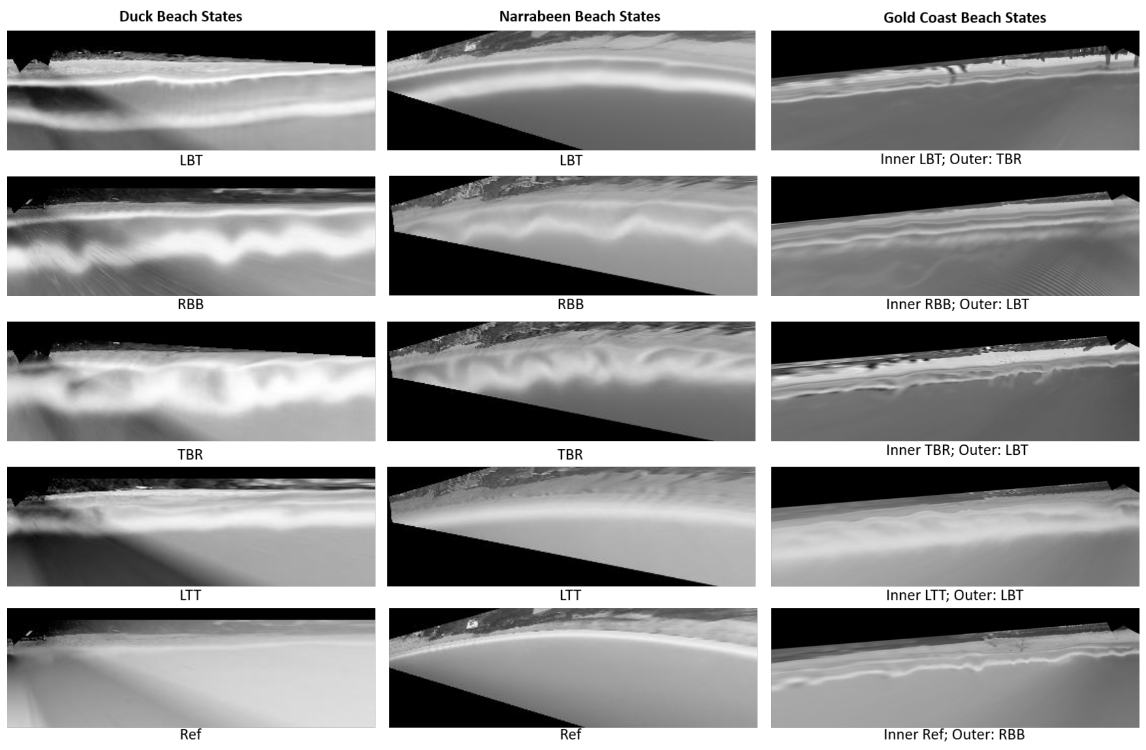


Figure 3.2: Examples of Argus imagery showing the five different beach states. Left column: Duck, middle column: Narrabeen and right column: Gold Coast. Note that the inner and outer bars of the Gold Coast have individual labels, with the outer bar labels only exhibiting the TBR, LBT and RBB states.

images were orthorectified onto a domain of 900 meters alongshore and 300 meters cross-shore, with a ground resolution of 2.5 x 2.5 meters.

The Duck dataset consists of images labelled with the beach states R, LTT, TBR, RBB, and LBT. Table 3.1a shows the class distribution of this dataset.

3.1.2 Narrabeen

Field Site

The 3.6 kilometre-long embayment of Narrabeen-Collaroy, hereafter referred to as Narrabeen, is located within the Northern Beaches region of the metropolitan Sydney, Australia (Figure 3.1B). A lagoon backs the northern half of the barrier and is connected to the ocean via a shallow, narrow inlet, which opens and closes on a regular base at the embayment's northern end. Adjacent to the southern end of the beach is a prominent headland.

The headland and the curvature of the embayment result in a distinct alongshore wave gradient

(Turner et al., 2016). The wave climate at Narrabeen is mild seasonal, with high-energy cyclones and east-coast lows more prominent in Austral winter months and low-energy swells more prominent in Austral summer. At interannual time scales, the wave climate is influenced by the El Niño-Southern Oscillation (ENSO), resulting in periods of less energetic and a more southerly wave climate. On the other hand, La Niña periods result typically in more energetic and easterly wave climates. The deep-water wave climate for the region of Sydney is of moderate to high wave energy with a mean wave height of 1.6 metres. It is dominated by long continuous period swell waves coming from an SSE direction. These swell waves are generated from mid-latitude cyclones that propagate in the southern Tasman Sea, south of Australia. Superimposed on these swell waves are storm events typically defined for this region by a significant wave height threshold of 3 metres. Tides are micro-tidal and semi-diurnal, with a mean spring tide of 1.3 metres.

The sediment is mainly uniform along the beach and consist of primarily fine to medium quartz sand with 30% carbonate materials (Short and Trenaman, 1992).

Dataset Description

The dataset of Narrabeen is based on the manually classified and labelled dataset used by Ellenson et al. (2020), with minor adjustments made to the labels. It comprises 687 images collected between 2004-2018. Compared to the data of Duck, the data of Narrabeen consist of only a single camera view, looking towards the stretch of the beach in the centre of the embayment. Similar to the data of Duck, day-Timex images were composed and orthorectified onto a domain of 900 metres alongshore and 300 metres cross-shore, with a ground resolution of 2.5 x 2.5 metres.

Table 3.1b shows the data and class distribution, in which the * denotes the differences compared to the distribution used by Ellenson et al. (2020). Similar to the dataset of Duck, the dataset consists of images labelled with the beach states R, LTT, TBR, RBB, and LBT.

Class	**Images**
Ref	125
LTT	126
TBR	140
RBB	126
LBT	163
Total	680

(a) Duck

Class	**Images *(*)***
Ref	125 *(0)*
LTT	164 *(+7)*
TBR	157 *(+8)*
RBB	106 *(-24)*
LBT	135 *(+9)*
Total	687

(b) Narrabeen

Table 3.1: Per class data distribution for a: Duck and b: Narrabeen (*Differences with comparison to the distribution used by Ellenson et al. (2020)).

3.1.3 Gold Coast

Field Site

The third dataset is from the very popular tourist destination of Surfer's Paradise, located at the northern end of the Gold Coast in South East Queensland, Australia (Figure 3.1C). The beach of the Gold Coast consists mostly of a double-barred system that is exposed to high-energy waves. In winter months, persistent SSE swells and high-energy mid-latitude cyclones result in a net annual littoral drift to the north in the order of 500.000 m^3 (Strauss et al., 2007). The mean-root-square wave height at the Gold Coast is typically about 0.8 metres. However, East Coast and tropical lows can increase the significant wave height to 1.5 metres, with an estimated return interval of 2 years for a significant wave height of 3.5 metres (Allen and Callaghan, 2000). The tide is semi-diurnal with a spring tidal range of 1.5 to 2 metres (Price and Ruessink, 2011). Furthermore, the nearshore is predominantly quartz sand with a median grain size of 0.225 millimetres, and the slope at the nearshore is approximately 0.02.

Between 1999 and 2000, a 1.2 M m^3 beach nourishment was undertaken to maintain and enhance the sub-aerial beach width (Jackson et al., 2017). The implementation of the nourishment near our study site commenced in early November 1999 and reached its southernmost extension in June 2000. The effect of the nourishment on the sandbars was most pronounced in March and April 2000. Our study site was restricted to the area south of the nourishment, and our sandbar data was not directly impacted. Furthermore, a hybrid coastal protection-surfing reef structure is located at Narrowneck, in north of our study site (Jackson et al., 2005). Hence, only the imagery containing the southern area of our study site was used.

Class	**Images**
Ref	377
LTT	1779
TBR	596
RBB	74
LBT	197
Total	3023

(a) Inner bar

Class	**Images**
Ref	0
LTT	0
TBR	1118
RBB	746
LBT	1159
Total	3023

(b) Outer bar

Table 3.2: Per class data distribution of the Gold Coast for the a: Inner bar and b: Outer bar.

Dataset Description

The dataset of the Gold Coast exists out of labelled images covering 9+ years, from 1999 to 2008. The dataset comprises 3023 low-tide timex images, representing the temporal mean of all frames collected during low-tide. Price and Ruessink (2011) used the dataset to classify the inner and outer bars individually. They identified two additional intermediate beach states, the eTBR and the rLTT, related to the dominant oblique angle of wave incidence and the multiple bar setting, respectively. For this thesis, the images corresponding to these additional bar states have been adjusted to the best alternative to fit the scheme of Wright and Short (1984).

Table 3.2 shows the class distribution for the dataset of the Gold Coast over the entire collection. Every image has two labels, one belonging to the inner and one to the outer bar. The inner bar has been labelled with the beach states R, LTT, TBR, RBB, and LBT. In contrast, the outer bar labels only exhibit the higher-energy states TBR, RBB and LBT.

3.2 Preprocessing

Before training the model, it is important that the images are in the same input format as the images on which network is pre-trained. Hence, we preprocessed the data of each location separately not to mix images.

The preprocessing phase existed out of four steps. The first preprocessing step was to process each image to meet the requirements of the pre-trained ResNet50 model. ResNet50 requires the input images to be 224 x 224 pixels. Hence, all images were resized to the target size. In addition, the images were normalized in the range $[0, 1]$. The images were split into a train, test, and validation set in the second preprocessing step.

The third and fourth preprocessing steps were only applied to the training set as they concern regularization, thus only influencing model training. The third step was to compensate for the class imbalances in the dataset. And the fourth step was to apply augmentations to avoid overfitting.

3.2.1 Data Distribution

To train and test a model, it is necessary to distribute the data into a train, test and validation set. In general, the train and validation sets are used to train and optimize the model, whereas the test sets are used to assess model performance. This distribution can be done in many ways. In the study of (Ellenson et al., 2020), only a train and test set were used in which 125 images per class per site were selected. In this thesis, we used all available images, and we first sorted our images by date before dividing them. This was done because our datasets consisted of time series in which consecutive images may be very much the same. Therefore, it was preferable not to shuffle the images, as this may result in *look-a-likes* distributed in the train, validation and test sets.

After sorting the images by date, we split our data into two sets: the first 80% and the last 20% of the data. The set of 80% was used as training data. Furthermore, the last 20% was used to create the validation and test data. All classes should be represented in the test and validation sets. Therefore the last 20% was divided in half to create the validation and test sets. This was done using a stratified split, meaning that after splitting, both sets had the same percentage of samples of each target class.

3.2.2 Oversampling

The training data was not stratified, which resulted in highly unbalanced datasets. This might adversely affect the training process leading to a bias during the prediction phase of the model (Menardi and Torelli, 2014). This could mean that an instance not previously seen by the model is more likely to be classified as the majority class. To counter this problem and balance out the bias, we over-sampled the data based on the classes' occurrence ratios in the training dataset. This ratio was calculated by dividing 1 by the amount of training images in a specific class. During training the model created copies of images with the minority classes' labels according to the calculated ratios.

3.2.3 Data Augmentation

Training a CNN requires a massive amount of data, and performance can usually be improved by adding more training data. The datasets used in this thesis were limited by the number of times each state occurred over the period spanned by the dataset and the number of images that distinctly exhibited one beach state (Ellenson et al., 2020). We increased the number of training data and, at the same time, increased regularization (Section 2.1.1) by applying augmentations to the training images while simultaneously preserving their labels.

Different augmentations were applied empirically to determine the best augmentations for our model. At first, we ran the model without applying any transformations to the training data. Depending on the increase or decrease in performance, we added or removed augmentations until we found a combination for which the model was slightly overfitting (Section 2.1.1). Table 3.3 shows the combination of augmentations and their corresponding functions that were used in this thesis. Examples of the resulting images after transformation are shown in Figure 3.3.

Augmentation	**Function**
RandomRotation(15)	Rotate the image by a 15 degree angle.
AutoAugment	Data augmentation method based on the proposed method of Cubuk et al. (2019).
RandomAffine(0,translate=(.15,.20))	Random affine transformation (translation with horizontal shift of 15 degrees, and vertical shift of 20 degrees) of the image keeping center invariant.

Table 3.3: The augmentations used in this thesis. In the left column the transformations applied and in the right column the function of the corresponding augmentation

3.3 Model Initialization

3.3.1 Network Reshaping

We trained our CNN by fine-tuning the pre-trained ResNet50 model. Before we could do this, the model's architecture needed to be reshaped. Recall the last layer of ResNet50 to be an FC layer with a SoftMax activation function (Figure 2.7). Since our model has been pre-trained on a large public dataset, the FC layer's output equals the number of classes in the dataset used for pre-training. Hence, we reinitialized this FC layer to be a linear layer maintaining the number of input features, and changing the output features to five, corresponding with the number of unique labels in our datasets.

3.3.2 Detailed CNN settings

Many of the parameter settings were used as proposed by Ellenson et al. (2020). Nevertheless, while we evaluated and tested our method, several adjustments were made to increase the computational speed and the model performance. The parameters and corresponding settings used in this thesis are shown in Table 3.4. These parameters are described in more detail in the following sections.

Stochastic gradient descent

Similar to Ellenson et al. (2020), the loss function used was the CE function as described in section 2.1.1. To minimize this function and reduce the training error, an iterative scheme, the SGD, was applied as an optimization algorithm. At each iteration, the SGD algorithm uses a single, randomly selected sample to compute the loss function's gradient and eventually determine the minimum. This was used to update the model's parameters each iteration. Since SGD uses

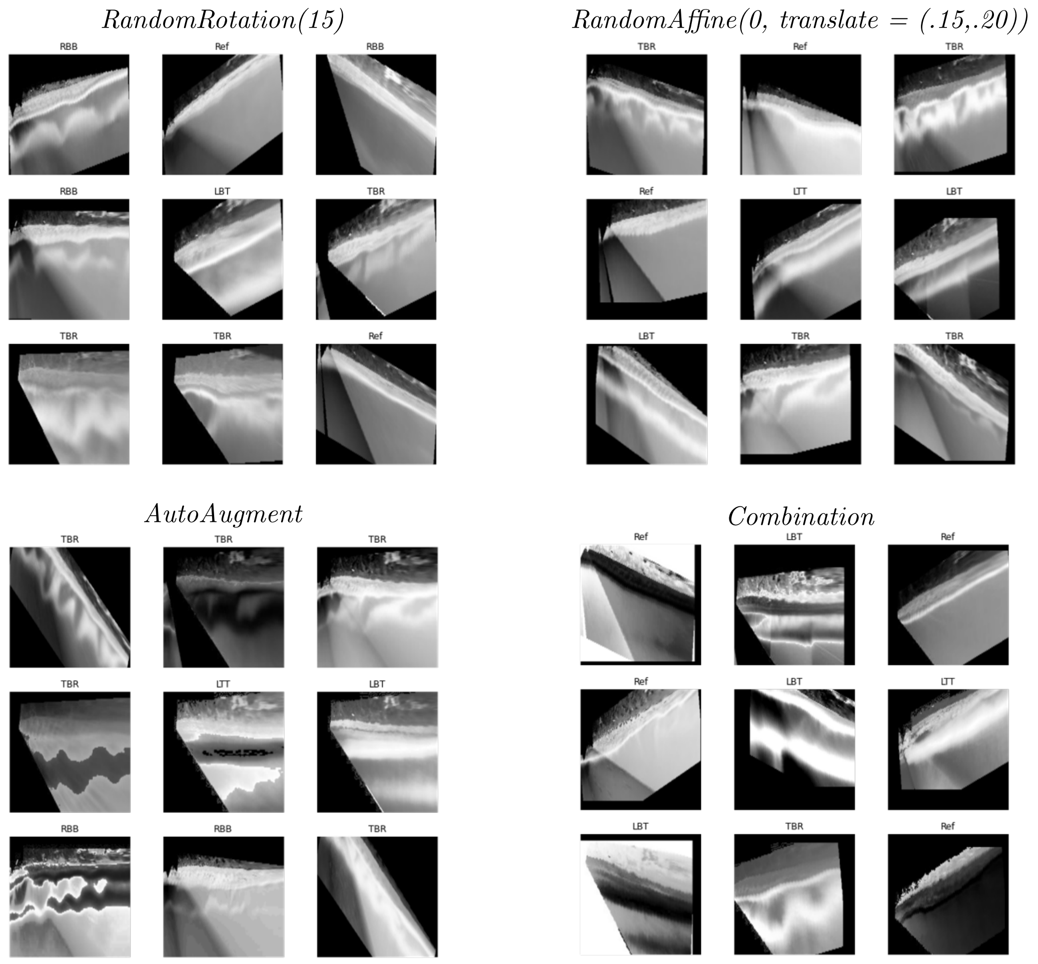


Figure 3.3: Examples of transformed images after applying: RandomRotation, RandomAffine, AutoAugment and the combination of the three.

only one sample, the path taken by the algorithm to reach the minimum is usually noisier than with the classic gradient descent method. At the same time, this noisier path caused the computational time to be much faster (Ketkar, 2017).

One Cycle Learning Rate

The amount that a weight is updated during training is referred to as the learning rate. To schedule this learning rate, we applied the OneCycleLR. This method resulted in high-speed training, using so-called *super convergence* (Smith and Topin, 2019). The OneCycleLR sets the learning rate of each parameter group according to the One Cycle Policy and changes it every batch. This policy takes the initial learning rate, which we set at 0.002, and increases it to the maximum learning rate. set to 0.01, followed by a decrease of the learning rate to a value much lower than the initial

Type	**Parameter**	**Setting**
OneCycleLR	Max momentum	0.95
OneCycleLR	Min momentum	0.85
OneCycleLR	Max learning rate	0.01
SGD	Learning rate	0.002
Network	Batch size	32
Early stopping	Patience	20

Table 3.4: The parameters and corresponding settings used in this thesis.

learning rate. At the same time, the momentum, a value which is used to achieve minimization more efficiently by adding a weighted estimate to the adjustment, cycled inversely to the learning rate with a maximum value of 0.95 and a minimum value of 0.85.

Early stopping

Each round of training and validation is called an epoch. The model was evaluated on the validation set during each epoch after training. When the performance on the validation dataset started to reduce, we wanted the model to stop training. Ellenson et al. (2020) did this by running their model for 120 epochs which generally converged after approximately 40 epochs.

We applied a method called early stopping. Early stopping was used as a form of regularization to avoid overfitting and to help find the convergence value for the lowest possible validation loss. Early stopping keeps track of the validation loss, and if the loss stops decreasing for several epochs in a row, the training stops. This amount of epochs before stopping is called patience. In this thesis, we used patience of 20 epochs.

Batch size

The batch size is the number of training images utilized in one iteration of training and validation. The batch size used in all experiments was 32, which showed the best fitting results. A smaller batch size resulted in slightly underfitting models, whereas a higher batch size resulted in overfitting models.

3.4 Experimental Setup

Conducting reliable experiments is an important step in finding answers to the defined research questions. This section describes the setup of the experiments conducted in this work.

We conducted two experiments, both following the same basic approach for training and testing. In the first experiment, we only used the training data of the single-barred beaches for training the CNNs. In the second experiment, we added the training data from the double-barred beach of the Gold Coast. Each CNN was tested on its own test data, the *self-tests*, and on the test data of the other locations, the *transfer-tests*:

1. Training the CNNs using:

- Single location training data
- Combined location training data

2. Testing the trained CNNs using:

- Self-testing
- Transfer-testing

Important to note is that the training of a CNN is stochastic due to random seeding of the original parameters, the optimization routine, and the batches of data selected as input. Because of this randomness, we trained ensembles of 10 CNNs for each training setup. The names given to the resulting models corresponding to these setups, the corresponding experiments and the data combinations on which they were trained and tested are shown in Table 3.5. In the following sections, the two experiments are discussed in more detail.

Experiment	**Model name**	**Training data**
1.	DUCK-CNN	Duck
	NBN-CNN	Narrabeen
	CombinedSINGLE-CNN	Duck & Narrabeen
2.	INNER-CNN	Gold Coast's inner bar labels
	OUTER-CNN	Gold Coast's outer bar labels
	CombinedINNER-CNN	Duck, Narrabeen & Gold Coast's inner bar labels
	CombinedOUTER-CNN	Duck, Narrabeen & Gold Coast's outer bar labels

(a) The two experiments, the corresponding models and the data combinations used to train them.

Experiment	**Model name**	**Test data**			
		Self-tests	*Transfer-tests*		
1.	DUCK-CNN	Duck	Narrabeen		
	NBN-CNN	Narrabeen	Duck		
	CombinedSINGLE-CNN	Duck & Narrabeen	Duck	Narrabeen	Gold Coast
2.	INNER-CNN	Gold Coast	-		
	OUTER-CNN	Gold Coast	-		
	CombinedINNER-CNN	Duck, Narrabeen & Gold Coast	Duck	Narrabeen	Gold Coast
	CombinedOUTER-CNN	Duck, Narrabeen & Gold Coast	Duck	Narrabeen	Gold Coast

(b) The two experiments, the corresponding models and the data combinations on which they were tested.

Table 3.5: The tables with the two experiments, the corresponding models and the data combinations on which they were trained *a* and tested *b*.

3.4.1 Experiment 1: Single-bar beach models

In the first experiment, only the single-barred beach data of Duck and Narrabeen was used to train the models, similar to the experiments of Ellenson et al. (2020). However, we should note that they used 100 training images per state per location to train the model, where we fed all available training images of the corresponding datasets to the model. As can be seen in Table 3.6, the DUCK-CNN was fed with 544 training images, and the NBN-CNN was fed with 549 training images. Both amounts correspond to 80% of the total amount of images in the dataset. In addition, in Table 3.6a and Table 3.6b, we see the distribution of the number of images from each class in the training, validation and test sets for Duck and Narrabeen, respectively. In addition, the weights used for oversampling the training data are shown in both tables. The models resulting from these single-location runs, the DUCK-CNN and NBN-CNN, were self-tested and transfer-tested, as can be seen in Table 3.5b.

Class	**Training**	**Weights**	**Validation**	**Test**
Ref	105	0.0095	10	10
LTT	94	0.0106	16	16
TBR	110	0.0091	15	15
RBB	89	0.0112	18	19
LBT	146	0.0068	9	8
Total	544	-	68	68

(a) Per class distribution of Duck

Class	**Training**	**Weights**	**Validation**	**Test**
Ref	91	0.0110	17	17
LTT	146	0.0068	9	9
TBR	126	0.0079	16	15
RBB	87	0.0115	9	10
LBT	99	0.0101	18	18
Total	549	-	69	69

(b) Per class distribution of Narrabeen

Class	**Training**	**Weights**	**Validation**	**Test**
Ref	196	0.0051	27	27
LTT	240	0.0042	25	25
TBR	236	0.0042	31	30
RBB	176	0.0057	27	29
LBT	245	0.0041	27	26
Total	1093	-	137	137

(c) Per class distribution of Duck and Narrabeen combined.

Table 3.6: The per class data distribution of a: Duck, b: Narrabeen and c: Duck and Narrabeen combined. The weights are calculated by dividing one by the number of training images in a specific class.

In the second part of Experiment 1, we trained the CNN with the training data of both Duck and Narrabeen, and 1093 images were fed to the CNN to train the CombinedSINGLE-CNN. Table 3.6c shows the merged per class data distribution and the weights used for oversampling the training data. The resulting CombinedSINGLE-CNN was self-tested and transfer-tested on several testing sets (Table 3.5b). To compare the skill of the CombinedSINGLE-CNN to DUCK-CNN and NBN-CNN, the CombinedSINGLE-CNN was tested on the test data of Duck and Narrabeen. In addition, the CombinedSINGLE-CNN was tested on the test data of the Gold Coast to assess the skill of a model trained with only single-barred beach data on unseen data of a double-barred beach system. This was assessed by comparing the the ratios in which the inner and outer bars would be correctly classified with the skills of the other tests of the CombinedSINGLE-CNN.

3.4.2 Experiment 2: Double-bar beach models

In Experiment 2, we assessed the performance of models trained using additional data from a double-barred beach. Table 3.7 shows the per class distribution and the weights used for oversampling the training data of the Gold Coast's inner and outer bar labels. First, we trained two 'single-location' models by feeding the 2418 training images with labels corresponding to either the Gold Coast's inner or outer bar (Table 3.5a), creating the INNER-CNN and OUTER-CNN. These models were only self-tested to be used as benchmarks and to assess their performance in classifying beach states. We did not train a model using the labels corresponding to the Gold Coast's inner and outer bar combined. In that case, we would have created a multi-label dataset, and our algorithm is only suited for single-label classification.

Class	**Training**	**Weights**	**Validation**	**Test**
Ref	306	0.0033	35	36
LTT	1502	0.0007	139	138
TBR	431	0.0023	83	82
RBB	55	0.0182	9	10
LBT	124	0.0081	37	36
Total	2418	-	303	302

(a) The per class distribution of the Gold Coast's inner bar labels, as used for the INNER-CNN.

Class	**Training**	**Weights**	**Validation**	**Test**
Ref	0	-	0	0
LTT	0	-	0	0
TBR	884	0.0011	117	117
RBB	547	0.01	100	99
LBT	987	0.0116	86	86
Total	2418	-	303	302

(b) The per class distribution of the Gold Coast's outer bar labels, as used for the OUTER-CNN.

Table 3.7: Data distribution for the Gold Coast's: a: inner bar labels and b: outer bar labels.

In the second part of this experiment, the recognition and classification of beach states of the double-barred beach of the Gold Coast were evaluated as a function of training data composition. The, CombinedSINGLE-CNN resulting from Experiment 1, was used as a base model. The Gold Coast's data, with either the inner or outer bar labels, was added incrementally as a percentage of the total training data fed to the CombinedSINGLE-CNN, creating the CombinedINNER-CNN and CombinedOUTER-CNN (Table 3.5a). In other words, the CombinedSINGLE-CNN was fed with 1093 images. 10 runs were performed for both cases in which the inner or outer bar data was added in 10% increments, thereby increasing the total amount of training data with $1093 * 0.1109$ images each run. During this process the ratio between the classes changed with each increment added, hence the weights for oversampling were automatically updated. The increase in training data continued until the total amount of training data was doubled (Table 3.8).

Important to note that in the case where we added data incrementally, the data distribution was done differently from the previous experiments. Instead of dividing the Gold Coast's data into two sets of 80% and 20%, the training data varied concerning the number of images needed for training. At the same time, the test and validation sets were constant and equal to the amount of validation and test images coming from the other two datasets.

The overall performance of all runs was assessed to determine the best performing CNNs. These CNNs were used for further evaluation and used for testing. The testing included self-testing and transfer-testing on the test data of the Gold Coast, Duck and Narrabeen individually and both single-bar test datasets combined (Table 3.5b).

3.5 Model Evaluation and Interpretation

This section discusses the performance metrics used to evaluate the models. In every kind of ML, the performance metrics are a significant part of the model. These metrics help determine if the model is making any progress and put a number on it.

As classification tasks are very discrete, discrete ways to evaluate them are required. For this purpose, many metrics have been developed, of which each metric evaluates the model's performance differently. We used the loss function discussed in Section 2.1.1 to measure the model's performance during training. Furthermore, we used four performance metrics to analyse the trained CNNs' skills

Class	**Training**										
	+0%	*+10%*	*+20%*	*+30%*	*+40%*	*+50%*	*+60%*	*+70%*	*+80%*	*+90%*	*+100%*
Ref	196	210	224	237	251	265	279	293	306	320	334
LTT	240	308	375	443	512	579	647	715	783	851	918
TBR	236	255	275	294	314	333	353	372	392	411	431
RBB	176	178	181	184	198	189	191	194	196	198	201
LBT	245	251	256	262	255	273	278	284	290	299	301
Total	1093	1201	1311	1420	1530	1639	1748	1858	1967	2076	2185

(a) The per class distribution of the training data used to train the CombinedINNER-CNN.

Class	**Training**										
	+0%	*+10%*	*+20%*	*+30%*	*+40%*	*+50%*	*+60%*	*+70%*	*+80%*	*+90%*	*+100%*
Ref	196	196	196	196	196	196	196	196	196	196	196
LTT	240	240	240	240	240	240	240	240	240	240	240
TBR	236	275	316	356	396	436	476	516	555	596	635
RBB	176	201	225	250	275	299	324	349	374	398	423
LBT	245	289	334	378	423	468	512	557	602	646	691
Total	1093	1201	1311	1420	1530	1639	1748	1858	1967	2076	2185

(b) The per class distribution of the training data used to train the CombinedOUTER-CNN.

Table 3.8: Training data distribution of a: CombinedINNER-CNN and b: CombinedOUTER-CNN.

on the test sets: *accuracy*, *precision*, *recall* and the *F1 score*. Additionally, to validate if the model was training on the features of interest, we applied model visualization using class activation maps. These concepts are discussed in the following sections.

3.5.1 Performance Metrics

Training and Validation Loss

To assess the model's performance during training, we used a measure known as loss. This loss quantifies the error produced by the model and was calculated by the loss function discussed in Section 2.1.1. We used the training loss as a metric to assess how our model fitted the training data. In other words, it assesses the model's error on the training set. This training loss was measured after each batch. On the contrary, we used the validation loss as a metric to assess the performance of our model on the validation set. The validation loss was measured after each epoch rather than after each batch.

To assess the performance of our models, we visualized both the training and validation loss together on a graph. When both the validation loss and training loss decrease and when they are almost equal, this would indicate an optimal fit. However, when they start to diverge, the model started to overfit.

Confusion Matrices

It is important to view the per class predictions in evaluating a model's performance. For this purpose, confusion matrices were used. A confusion metric is technically not a performance metric. However, they are fundamental for other metrics and, therefore, are discussed in this section.

A confusion matrix in a classification task is a tabular representation of the per-class predictions of the model. It is widely used as it gives insight into the errors made by the model and the type of errors. To understand the confusion matrix, we created one with *Yes* and *No*. In the confusion matrix, each cell represents an evaluation factor, as shown in table 3.9.

		Predicted	
		Yes	**No**
Actual	**Yes**	TP	FN
	No	FP	TN

Table 3.9: Confusion matrix for the classification of Yes or No.

These evaluation factors can be described as:

- **True Positive (TP)**: TP denotes the number of cases in which we predicted *Yes* correctly.
- **True Negative (TN)**: TN signifies the number of cases we predicted *No* correctly.
- **False Positive (FP)**: FP is the number of cases we predicted *Yes* where it is *No*.
- **False Negative (FN)**: FN denotes the number of cases we predicted *No* where it is *Yes*.

Accuracy

One often-used metric is accuracy. It is often defined as a system's quality or state of being correct and is especially essential when considering datasets with balanced classes. It can be computed through the number of correct predictions divided by the total number of predictions multiplied by 100. Alternatively, when represented by the evaluation factors:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} * 100 \tag{3.1}$$

However, accuracy could give an unreliable representation of the classification performance in imbalanced data sets. Hence, we used the following three metrics to assess the overall and per-class performances.

Precision

The precision metric is the proportion of positive test results that are TP, the so-called positive predictive value. The precision can be computed through:

$$Precision = \frac{TP}{TP + FP} = \frac{\textit{Yes}\text{ predicted correctly}}{\text{All predicted as }\textit{Yes}} \tag{3.2}$$

The precision has a range of $[0, 1]$. A small precision value indicates a high number of FP in the classifier. Following up on any positive results with a more reliable test may be necessary to obtain a more accurate assessment. On the other hand, a high precision value indicates that the model can make an accurate classification. However, a downside of the precision is that the precision cannot measure FN.

Recall

The recall, or so-called true positive rate, is calculated by the proportion of actual positives which are correctly classified. In terms of an equation:

$$Recall = \frac{TP}{TP + FN} = \frac{Yes \text{ classified correctly}}{\text{Total } Yes} \tag{3.3}$$

Similar to precision, the recall is in the range of [0, 1]. A recall towards 1 indicates that a model can classify accurate between correctly and incorrectly labelled images. However, a low recall score signifies that the model has a high number of FN, possibly resulting from an imbalanced class or un-tuned model parameters. One downside of the recall metric is the fact that it is not able to measure FP.

F1 score

The recall and precision both have a downside of not measuring FN or FP, respectively. The F1 score uses both precision and recall and is, in fact, the harmonic mean between the two. In this metric, the recall and precision are evenly weighted and is mainly used when the class distribution in the dataset is imbalanced while the FN and FP are evenly important. However, the F1 score can be used for per class evaluation as well. The F1 score is represented by:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{3.4}$$

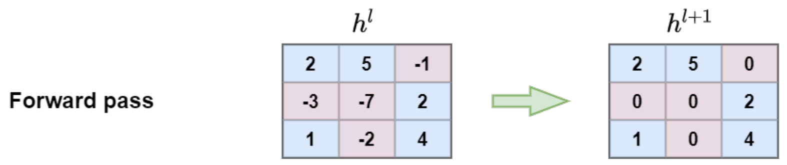
When the F1 score is high, it signifies a high precision and recall, indicating a good balance between the two and giving good results on imbalanced classification challenges. A low F1 score, on the other hand, is not very useful as it only indicates that the model performance was poor over the entire test set and did not indicate whether we have a low precision or low recall.

3.5.2 Model Visualization

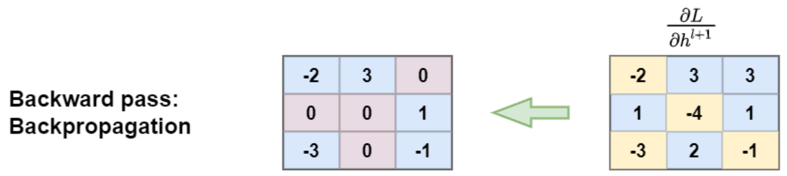
CNNs are excellent in extracting features from images and performing classification tasks. However, an important aspect of computer-aided classification is understanding why a CNN model interprets what it interprets. Only when we understand the model, we can put our efforts into addressing its shortcomings. However, their lack of decomposability into separate components makes CNNs hard to interpret. As interpretability is a key factor in building trust for the practical application of a model, we used model visualization to validate if the models have trained on the features of interest (Simonyan and Zisserman, 2014). We used guided backpropagation (GBP) in combination with a Class Activation Map (CAM). This combination is a way to measure the spatial support of a particular class in an image. It is particularly useful for model interpretation and not specifically for model evaluation (Selvaraju et al., 2017).

Guided Backpropagation

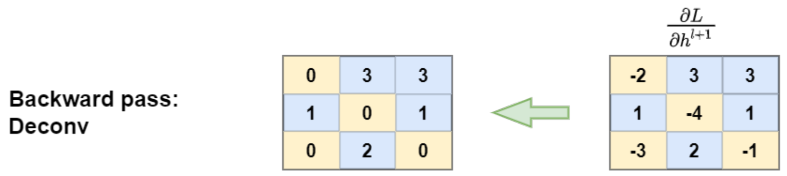
GBP uses the feature extraction part of the CNN model to visualize fine-grained details in the image. In general, it focuses on specific pixels associated with relevant spatial structures (Zeiler and Fergus, 2014; Springenberg et al., 2014). For example, the spatial structures associated with the linear or curved shapes of sandbars or shorelines.



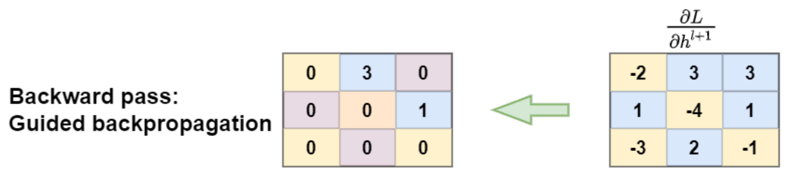
(a) ReLU Forward Pass: Pass the values which are greater than zero using Equation 3.5. In which the right 3x3 square represents the filter h^{l+1} used in b, c and d.



(b) ReLU Backward Pass: Pass the values which are greater than zero in the filter of a using Equation 3.6.



(c) Deconvolution: Pass the values backward for the values in the filter of a that were greater than zero using Equation 3.6.



(d) Guided Backpropagation: Taking the intersection of the concept of Backward pass and the Deconv using Equation 3.6.

Figure 3.4: Example of the workflow of how guided backpropagation works. The red squares represent the negative pixels put to zero during vanilla backpropagation and the yellow represent the pixels which are put to zero during Deconv.

To perform GBP, we have to undo all convolutions. This is done by an inverse matrix multiplication with ReLU as an activation function. During forward passing an image, the ReLU function dismisses all negative values and sets them to zero (Figure 3.4a):

$$h^{l+1} = max\{0, h^l\} \tag{3.5}$$

With h^{l+1} representing the filter and h^l the input image. This forward pass causes the negative gradients to be zero as well. This is also referred to as vanilla backpropagation (Figure 3.4b):

$$\frac{\partial L}{\partial h^l} = [\![h^{h+1} > 0]\!]\frac{\partial L}{\partial h^{l+1}} \tag{3.6}$$

The vanilla backpropagation is followed by Deconvolution (Deconv). Deconv applies the interpretation of the gradients as can be seen in Figure 3.4c. The negative gradients (the yellow squares) are suppressed and the positive gradients are kept as relevant. A ReLU is applied during backward passing, dismissing the negative values (Figure 3.4d). The outcome of this process is a saliency map with clear visualization of the relevant features (left figure in Figure 3.5).

Guided CAMs

Another commonly used visualization technique is CAM. The idea behind CAM is to identify pixels that have the largest contribution to classifying an image as a specific class (Zhou et al., 2016). This can be done by projecting back the weights of the output layer on the convolutional feature maps obtained from the last Convolutional Layer. The dot product of the extracted weights from the final layer and the feature map is calculated to produce the CAM. A CAM on its own is often not able to highlight fine-grained details like pixel-space gradient methods. It is often unclear from the coarse heat-map what specific features a prediction is based on (middle figure in Figure 3.5).

We used an element-wise multiplication of GBP and CAM to create a Guided-CAM, using the best features of both (Selvaraju et al., 2017). The resulting saliency maps are both high-resolution and class-discriminative (right figure in Figure 3.5).

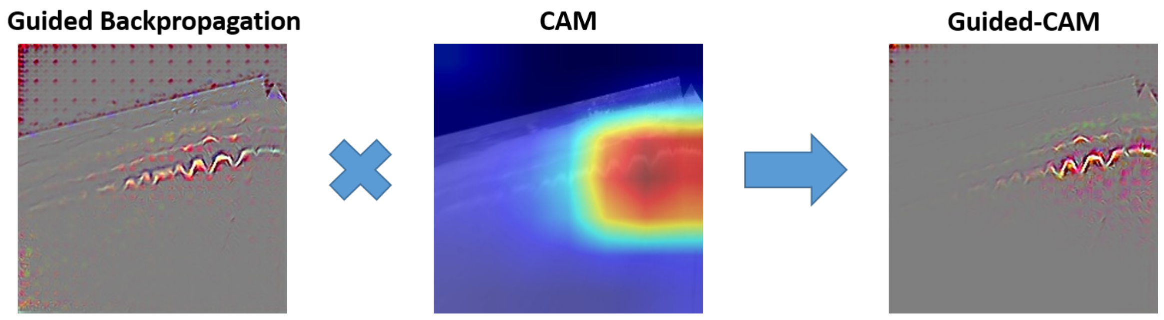


Figure 3.5: Example of an element-wise multiplication of the GBP and a CAM, resulting in a Guided-CAM.

3.6 Implementation Details

Operations performed by CNNs, i.e. training and prediction, are often computationally expensive and can be very time-consuming. However, these operations can significantly be accelerated using a Graphics Processing Unit (GPU). In this thesis, all computational work and experiments were done on Google Colaboratory (Google Colab)[1]. Google Colab is a product of Google Research that allows writing and executing arbitrary python codes online. In addition, Google Colab provides a 13 GB of RAM, an NVIDIA Tesla K80 12 GB GPU, and an Intel(R) Xeon (R) CPU @ 2.30 GHz processor.

[1]https://colab.research.google.com/

The code used in this thesis is presented online and publicly available use[2].

[2]https://colab.research.google.com/drive/1O5STQM2WYIDGzDm2YMlGRZcIdo$_d4sS3?usp = sharing$

Chapter 4

Results

This chapter presents the results obtained by the experiments described in Section 3.4. The results of Experiment 1, consisting of only single-barred data (Section 3.4.1), and experiment 2, with additional double-barred data (Section 3.4.2), are presented separately in terms of:

1. Overall performance:
    - Training and validation loss
    - F1 scores
2. Per-class performance
    - Single-location models
    - Combined-location models

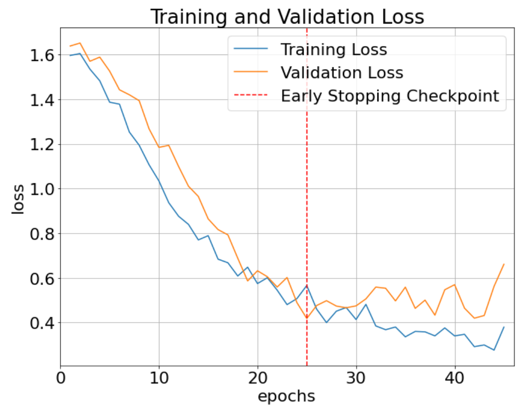
4.1 Experiment 1: Performance of single-bar beach models

4.1.1 Overall Performance

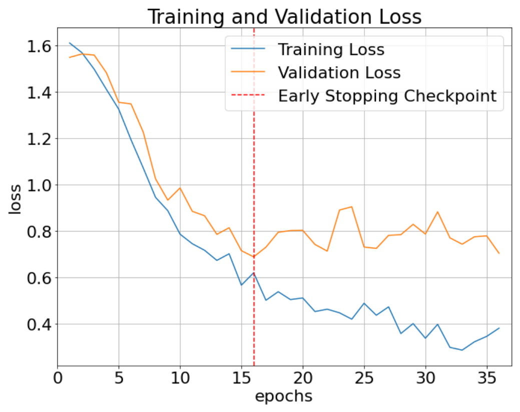
Performance in terms of loss

Figure 4.1 shows the results in terms of training and validation loss. Preferably we wanted to stop training just before a model started to overfit. This is the case when the training and validation losses started to diverge. Hence, we implemented the early stopping algorithm to stop training when this divergence started to occur.

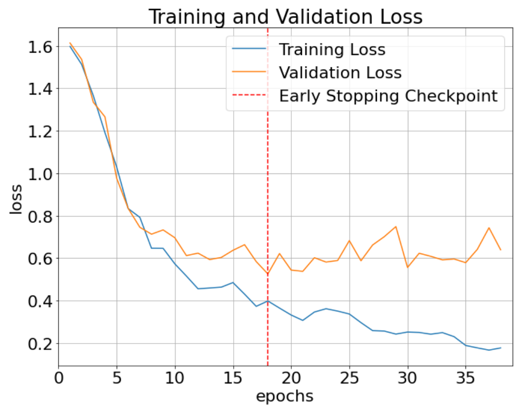
Comparing all three models trained with single-bar data, the DUCK-CNN (Figure 4.6a) trained for the highest number of epochs (25) and reached the lowest validation and training loss before starting to diverge. The low losses indicate that this model had a better fit compared to the NBN-CNN (Figure 4.6b) and CombinedSINGLE-CNN (Figure 4.1c). In other words the performance of DUCK-CNN was better on both the training and validation sets.



(a) Training and validation losses of DUCK-CNN.



(b) Training and validation losses of NBN-CNN.



(c) The training and validation losses of CombinedSINGLE-CNN.

Figure 4.1: The training and validation losses of training the models a: DUCK-CNN, b: NBN-CNN and c: CombinedSINGLE-CNN. On the y-axis the loss and on the x-axis the number of epochs. The red dotted line indicates the early stopping point.

Results in terms of F1 scores

Figure 4.2 shows a box plot summarizing the F1 scores obtained by the first experiment (Section 3.4.1). The boxes show the values between the first and third quartile of the 10-member ensembles, with the line in the boxes representing the median. The whiskers show the rest of the distribution within 1.5 times the quartile range. In general, we see consistent results with the models trained from scratch by Ellenson et al. (2020), with the highest performance in the cases of the self-tests. Compared to their work with F1 scores of 0.59 and 0.80 for Narrabeen and Duck, respectively, we achieved F1 scores of 0.76 and 0.89.

When transfer-tested, we see for both single-location models a lower performance. Testing NBN-CNN on the test set of Duck resulted in a maximum F1 score of 0.66, a decrease of 13% compared to the self-test. With DUCK-CNN transferred to the test data of Narrabeen, we see a decrease of 18%, reaching a maximum F1 score of 0.73. We see a larger decrease when transferring DUCK-CNN to Narrabeen than when transferring NBN-CNN to Duck. This could indicate that the correlation of sandbars and beach states at Narrabeen was more informative than at Duck. However, this could also be caused by the difference in the optical signatures of the shorelines between the Argus images of Duck and Narrabeen. Further discussed in section 5.1.

The model trained on both the training data of Duck and Narrabeen, the CombinedSINGLE-CNN, reached a maximum F1 score of 0.78 on the self-test. In addition, the CombinedSINGLE-CNN slightly outperformed the NBN-CNN on the test set of Narrabeen with an increase of approximately 2%, reaching an F1 score of 0.78. However, the performance of the CombinedSINGLE-CNN on the Duck test set decreased by 3%, in comparison with the DUCK-CNN, reaching a maximum F1 score of 0.85. However, both differences are negligible as they are relatively small and therefore could be a result from the randomnesses during model training.

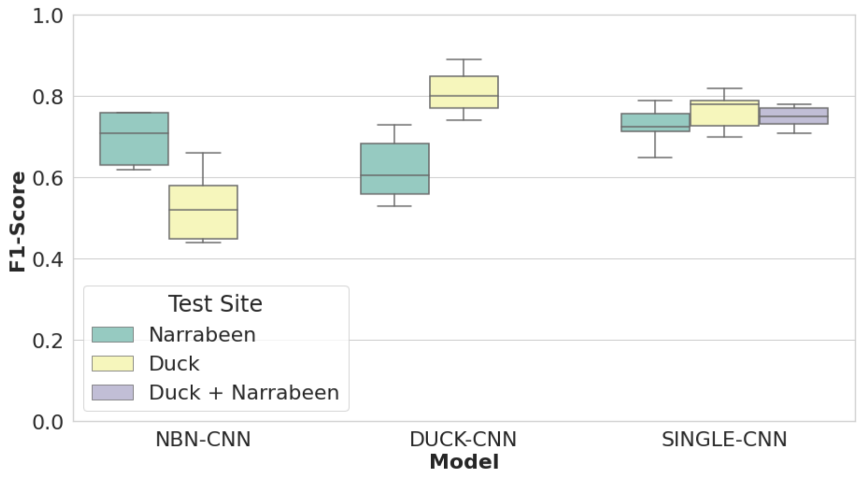
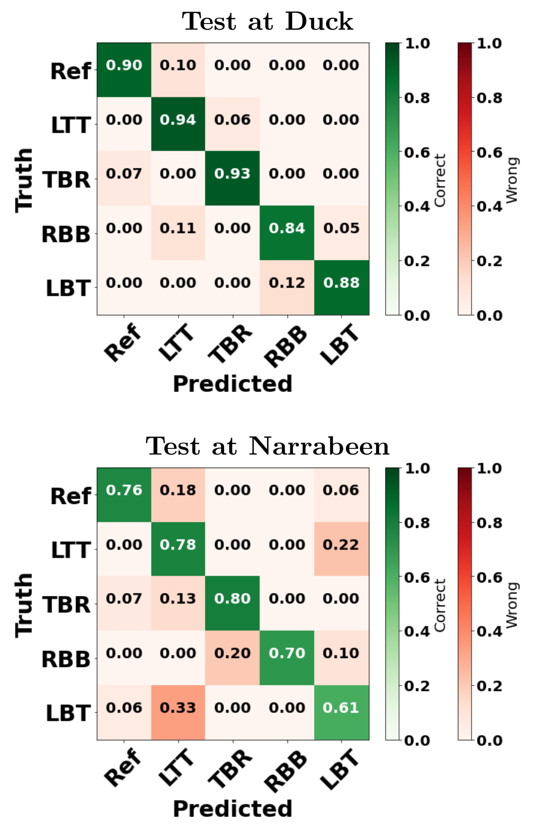


Figure 4.2: Box plot displaying the F1 scores from tests at single- and combined-location datasets using the models trained in Experiment 1 with only single-barred training data. The boxes show the values between the first and third quartile, the line in the boxes shows the median, and the whiskers show the rest of the distribution within 1.5 times the quartile range.

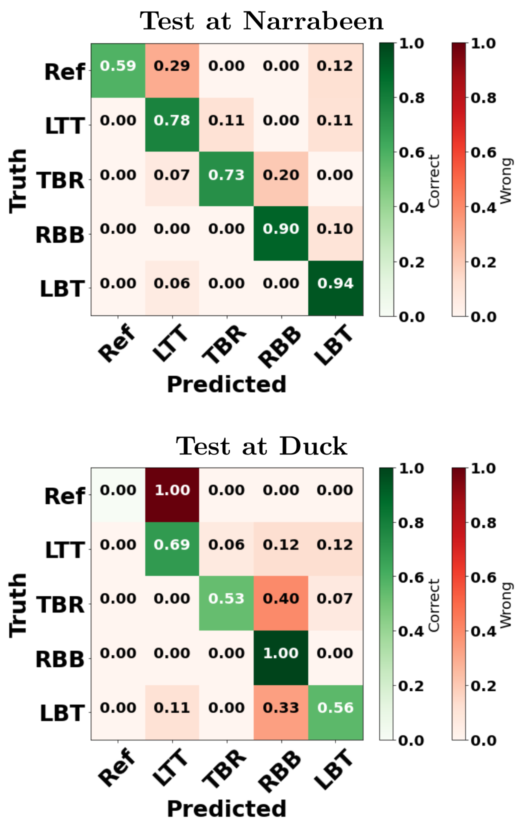
4.1.2 Per-class Performance

Single-location models

Figure 4.3 shows the normalized confusion matrices for each of the four single-location tests of Experiment 1 (Figure 3.5b). Normalized means that each of the classes is represented as having 1.00 samples. Thus, the sum of each row represents 100% of the images in this class. The F1 scores were calculated according to Equation 3.4, where the diagonal represents the recall calculated by Equation 3.5.1, and the precision was calculated by Equation 3.2.



(a) *DUCK-CNN*



(b) *NBN-CNN*

Figure 4.3: Normalized-confusion matrices for each single-location test of Experiment 1. *a*: The matrices for the DUCK-CNN. *b*: The matrices for the NBN-CNN. The predicted per class label is on the horizontal axis, and the vertical axis is the true per class label. Red indicates wrong predictions, and green indicates the correct predictions.

For the self-tests of the DUCK-CNN, we achieved per class F1 scores ranging between 0.88-0.93, with the highest score for the TBR state and the equally lowest scores in the LTT and LBT states. The class with the lowest precision is the LTT state, meaning that images are often misclassified as LTT. In this case, that accounts most often for images with the RBB and Ref states. In comparison, the per-class F1 scores of the self-test of the NBN-CNN are slightly lower than the self-test of the DUCK-CNN. They range between 0.61-0.87, with the highest value corresponding to the LBT class and the lowest to the LTT class. The major confusions happen between the linear states, Ref, LTT and LBT, and between the rhythmic states, TBR and RBB.

For the transfer-test of the DUCK-CNN, the per-class F1 scores decreased for the majority of the states compared to the self-test. We see the F1 scores ranging between 0.52-0.83. From the confusion matrix's diagonal, we can see that the recall for all classes has decreased. The main confusions occur in the adjacent states and between the linear states, in which LBT and LTT are most often confused.

With F1 scores ranging between 0.00-0.77, the case of transfer-testing the NBN-CNN at Duck becomes interesting. The first significant observation is that the model performed worst on the Ref class, similar to the self-test of NBN-CNN. However, the NBN-CNN failed to predict any images as Ref class in this case. Instead, the images of the Ref class are all classified as LTT. In contrast, LTT was often mispredicted as TBR, RBB or LBT. In addition, we see large misclassifications for TBR, of which 40% was classified as RBB, and for LBT, with 33% classified as RBB. This could be explained by the fact that they are adjacent states. However, the low precision, combined with the high recall percentages, suggests that the model was over-predicting RBB. When this is the case, this could also be caused by class imbalances. This is further discussed in Chapter 5.

Combined-location model

Single-barred beaches

In section 4.1.1, we saw an overall F1 score of 0.78 for the CombinedSINGLE-CNN, an F1 score of 0.78 when tested at Narrabeen and an F1 score of 0.85 when tested at Duck. In figure 4.4, we see the normalized confusion matrices corresponding to these tests. When tested at the test data of Duck, the CombinedSINGLE-CNN achieved F1 scores ranging between 0.79-0.95, with the confusions more concentrated on adjacent classes. The lowest performances are achieved in the LTT and TBR states. These states are relatively often confused with each other. A possible cause is that both LTT and TBR states correspond with bar welding and rip currents, with the only difference in the size and amount of rip currents (Further discussed in Chapter 5).

Compared with the NBN-CNN, the CombinedSINGLE-CNN performed slightly better on the test set of Narrabeen, with F1 scores ranging between 0.58-0.93. The highest F1 score is achieved for the TBR state and the lowest for the LTT state. In addition, there is an increase in misclassifications, where the NBN-CNN confused real TBR images with RBB states, the CombinedSINGLE-CNN confused real RBB images with LBT images. Interesting to see is the high performance for the Ref state. At the transfer-test of the DUCK-CNN at Narrabeen, we noticed that it failed to classify the Ref state, whereas the CombinedSINGLE-CNN even managed to reach a higher F1 score for the Ref class than the self-test of the NBN-CNN. This suggests that the performance increased with a larger diversity of images.

When tested at the combined test set, the CombinedSINGLE-CNN achieved a similar overall performance compared to the self-tests of NBN-CNN and DUCK-CNN, with F1 scores ranging between 0.69-0.88. The CombinedSINGLE-CNN has the most classification errors within adjacent classes and the linear states. The highest misclassifications correlate with the confusions as occurred when tested at the test data of Narrabeen, with confusions made for the real RBB images as LBT states and real LBT as LTT.

Double-barred beaches

As described in section 3.4.1, we tested the CombinedSINGLE-CNN on the test data of the Gold Coast to assess the performance of the single-barred model on unseen, double-barred data. The corresponding confusion matrices are shown in Figure 4.5. When looking at the per bar performances, the model achieved F1 scores of 0.32 and 0.55 for the inner and outer bar, respectively. For the inner bar, the per-class F1 scores range between 0.00-0.39, with the highest value for the LTT and the lowest for both Ref and RBB states. The lowest-performing states are under-predicted,

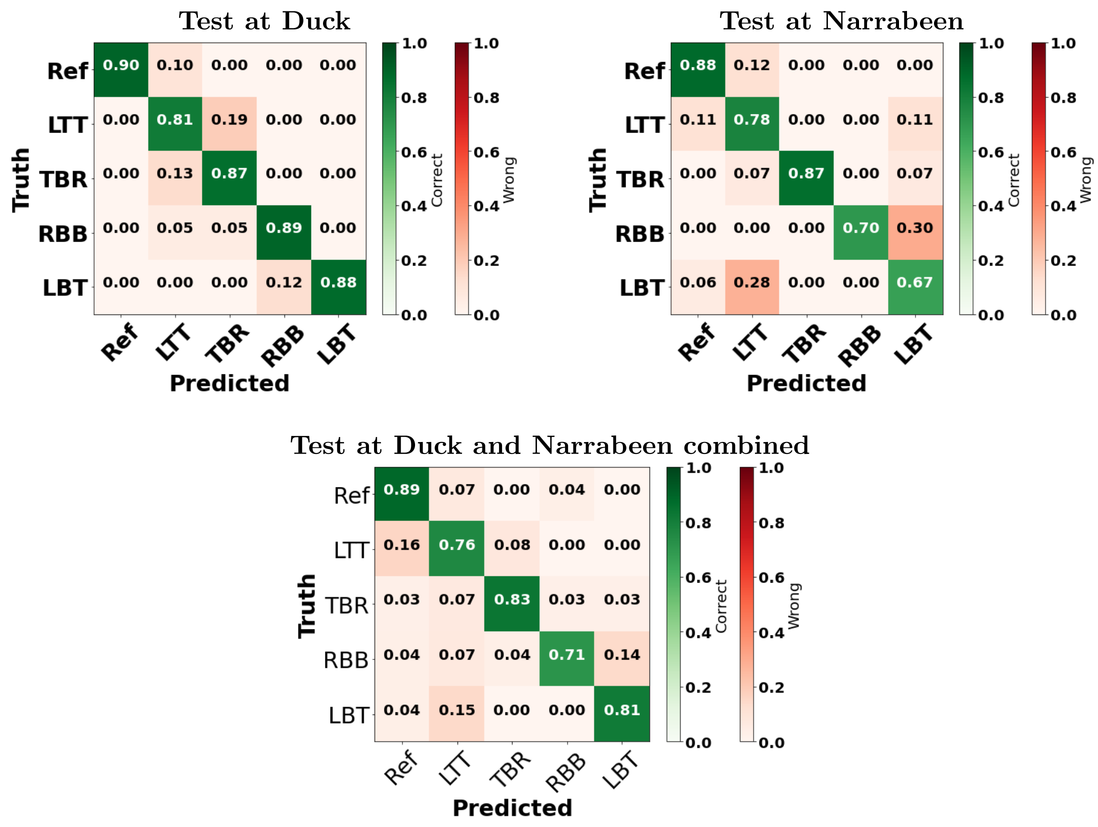


Figure 4.4: Normalized confusion matrices for the CombinedSINGLE-CNN tests. Transfer-tests on the test data of Duck and Narrabeen and self-test on the test data of Duck and Narrabeen combined (Table 3.5b).

while all other classes are over-predicted. Interesting is that this distribution correlates with the imbalances within the training set (Table 3.6c). (Further discussed in Chapter 5).

For the outer bar, the F1 scores range between 0.00-0.78, whereas the lowest values correlate to the low-energy states of Ref and LTT, which have not been observed in the outer bar. In addition, the highest performance is achieved for the TBR state. The major confusion is made for the RBB state, which is often confused with the LBT state.

4.2 Experiment 2: Performance of double-bar beach models

In this section, the performances of the models trained with data from the double-barred beach of the Gold Coast are evaluated as a function of training data composition. The models corresponding to this experiment can be seen in Table 3.5. The INNER-CNN and OUTER-CNN correspond to the models trained with only the training data of the Gold Coast, with either the labels of the inner or outer bar, respectively. In addition, the CombinedINNER-CNN and CombinedOUTER-CNN consist of the training data of Duck and Narrabeen, similar to the CombinedSINGLE-CNN,

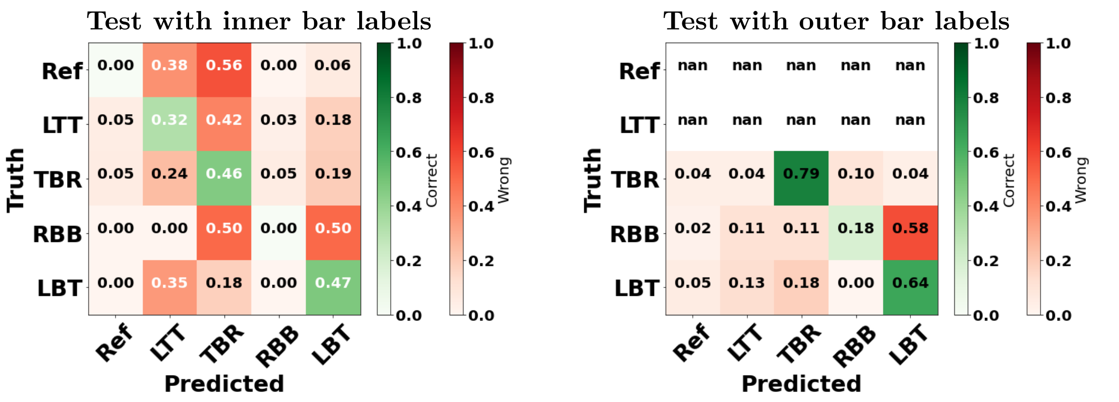


Figure 4.5: Normalized confusion matrices for the CombinedSINGLE-CNN tests at the test data of the Gold Coast. Left: the per-class performance on the inner bar. Right: the per-class performance on the outer bar.

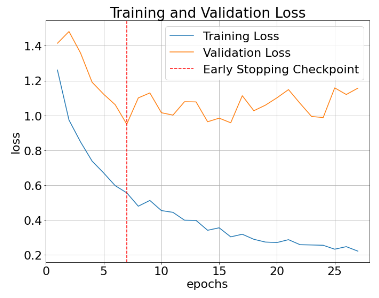
with additional training data of the Gold Coast, with either the labels of the inner or outer bar, respectively. The amount of training data coming from the Gold Coast is added in increments of 10% as described in Experiment 3.4.2.

4.2.1 Overall Performance

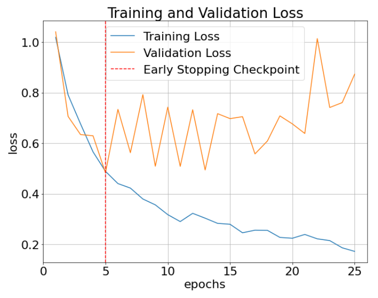
Performance in terms of loss

Figures 4.6 a&b show the graphs containing the training and validation losses for the INNER-CNN and the OUTER-CNN. The distance between the training and validation losses for the INNER-CNN is relatively large and both losses start to diverge quickly. Both the large space and the quick divergence suggests that the model is overfitting. On the other hand, the model for the outer bar, the OUTER-CNN, did a better job. It reached a relatively low training and validation loss before starting to diverge.

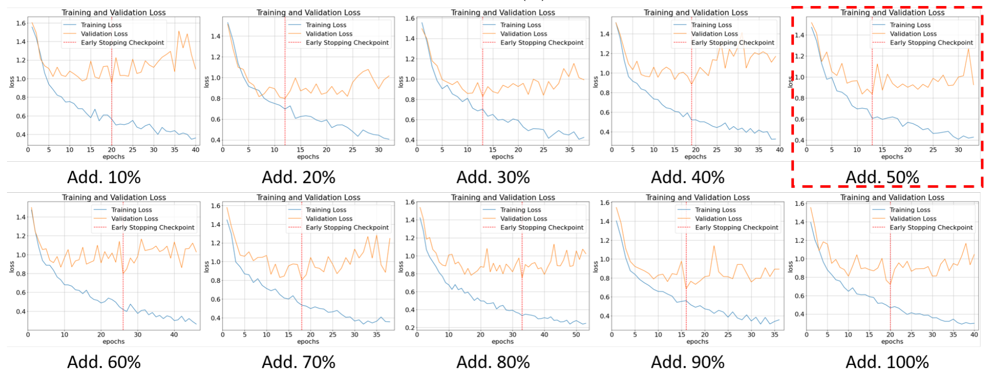
For the CombinedINNER-CNN and CombinedOUTER-CNN, the Gold Coast's training data was incrementally added with either the inner or outer bar labels (As described in Section 3.4.2). For both CNNs, Figures 4.6 c&d show the loss graphs for all ten increments. For the case of training the CombinedINNER-CNN, the graphs corresponding to the training sets wit 10%, 40%, 60%, 70%, 80%, 90% or 100% additional data, show very quick diverging charts. This quick divergence suggests that these models are probably overfitting. However, the graphs corresponding to an additional amount training data of 20%, 30% and 50% show less quick converging losses, suggesting better performance. In comparison with the CombinedINNER-CNN the charts for the CombinedOUTER-CNN are less quick diverging and reach lower losses. In particular, the graphs relating to the 30%, 40% and 90% additional data reach relatively low losses before diverging. Hence, we expected one of these models to achieve the best performance.



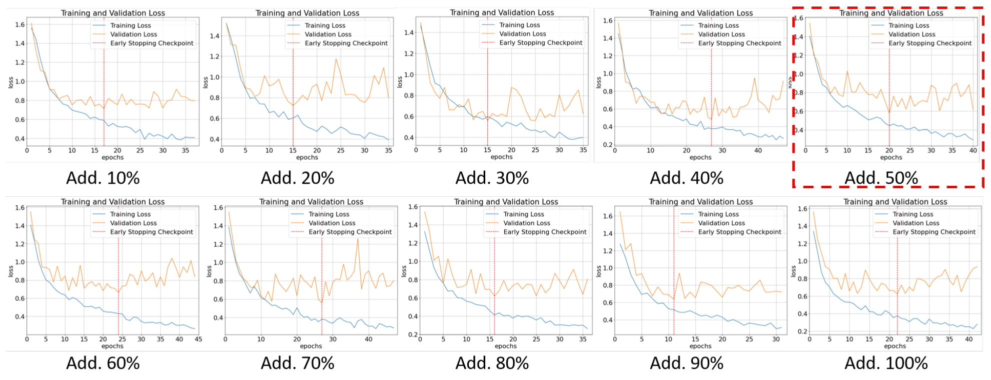
(a) Training and validation loss for INNER-CNN



(b) Training and validation loss for OUTER-CNN



(c) Training and validation losses for the models with the incrementally added data to train the CombinedINNER-CNN



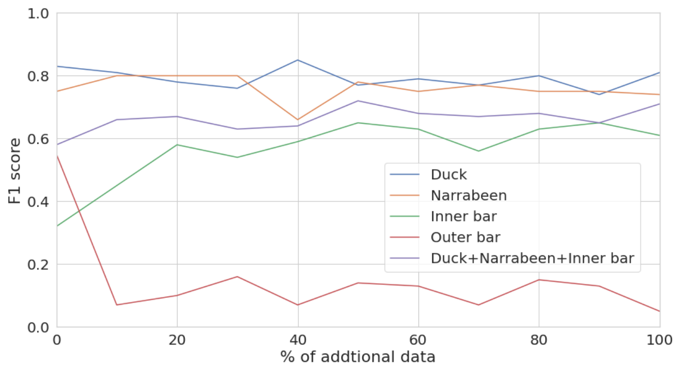
(d) Training and validation losses for the models with the incrementally added data to train the CombinedOUTER-CNN

Figure 4.6: The training and validation losses of the models fed with double-barred beach data.a: INNER-CNN, b: OUTER-CNN and c&d the CombinedINNER-CNN and CombinedOUTER-CNN, with data added in increments of 10%. The red dashed squares indicate the models with the best results.

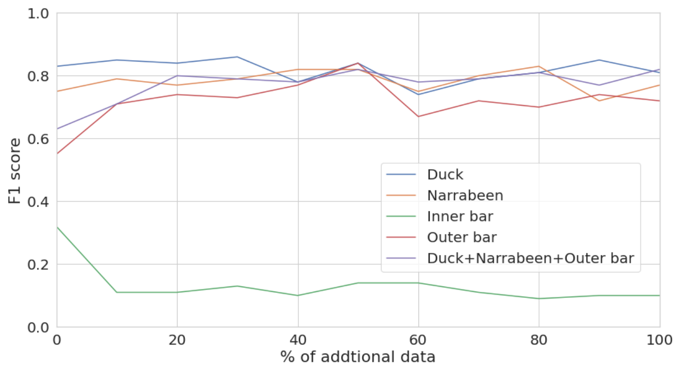
Results in terms of F1 scores

As described in Section 3.4.2, four 10-member ensembles resulted from training CNNs with the train data of the Gold Coast (Table 3.5a). First, we trained two benchmark models by feeding the model only data of the Gold Coast with either inner or outer bar labels, the INNER-CNN and OUTER-CNN. We achieved an overall good performance on the self-tests with F1 scores of 0.73 and 0.88, respectively.

Secondly, we trained the CombinedINNER-CNN and CombinedOUTER-CNN with data of the Gold Coast added with increments of 10%. The performances of both models, in terms of F1 scores, per increment of data added are shown in Figures 4.7a&b. As the loss charts in Figures 4.6c&d already suggested, the performances of the models varied with the different amounts of training data. In both cases, when tested on the various test sets (Table 3.5b), we see that for the training set ratios, which consisted of at least 20% of additional data, the F1 scores remained within 15% of the maximum skills. Important to note is that the CombinedSINGLE-CNN, the model trained on both single-barred training sets, functions as the ground model. This means that the '0% of data added' in Figure 4.7 represents the performance of the CombinedSINGLE-CNN.



(a) Performance of CombinedINNER-CNN



(b) Performance of CombinedOUTER-CNN

Figure 4.7: F1 scores with respect to percentage of additional data of the Gold Coast with labels coming from the inner(a) or outer bar(b).

In the case of the CombinedINNER-CNN, we see a hard increase in the performance on the inner bar directly when data is added. From a minimum of F1 score of 0.33 in the case without data of the Gold Coast, to a maximum F1 score of 0.65 for the models with 50% and 90% of additional data. This maximum is a slight decrease compared to the performance of the INNER-CNN. At the same time, the performance on classifying the outer bar significantly decreased from an F1 score of 0.55 to 0.07, after which it remained between 0.07-0.16.

In addition, when tested at Duck, we see a maximum F1 score comparable to the DUCK-CNN and CombinedSINGLE-CNN, of 0.85 for the case with 40% of additional data. For Narrabeen, we achieved a maximum F1 score of 0.80 with 30% of additional data, a small increase compared to the NBN-CNN and CombinedSINGLE-CNN. Furthermore, the self-test of the CombinedINNER-CNN reached a maximum of 0.72 with 50% of additional data. The percentages of additional data are consistent with our expectations from the training and validation losses in Figure 4.6c. The all-around best performance for all increments, averaged over the locations, was achieved with 50% of additional data. This corresponds to $\frac{1}{3}$ of the data coming from the Gold Coast.

In the case of adding data with labels of the outer bar and training the CombinedOUTER-CNN, we see similar behaviour as the CombinedINNER-CNN. Directly when adding data, we see a hard increase in performance on the outer bar, from a minimum of 0.55 to a maximum F1 score of 0.84 for the model with 50% of additional data. Which is comparable to the result of the OUTER-CNN as the difference is less than 5%. At the same time, the performance on classifying the inner bar significantly decreased from an F1 score of 0.32 to 0.11, after which it remained between 0.1-0.14.

When tested at Duck, we see a maximum F1 score, comparable to earlier F1 scores at Duck, of 0.86, with 30% of additional data. The same accounts for Narrabeen, where we achieved a maximum F1 score of 0.82 with 50% of additional data. The self-test of the CombinedOUTER-CNN reached 0.82 with 50% and 100% of additional data. When comparing results with the loss graphs

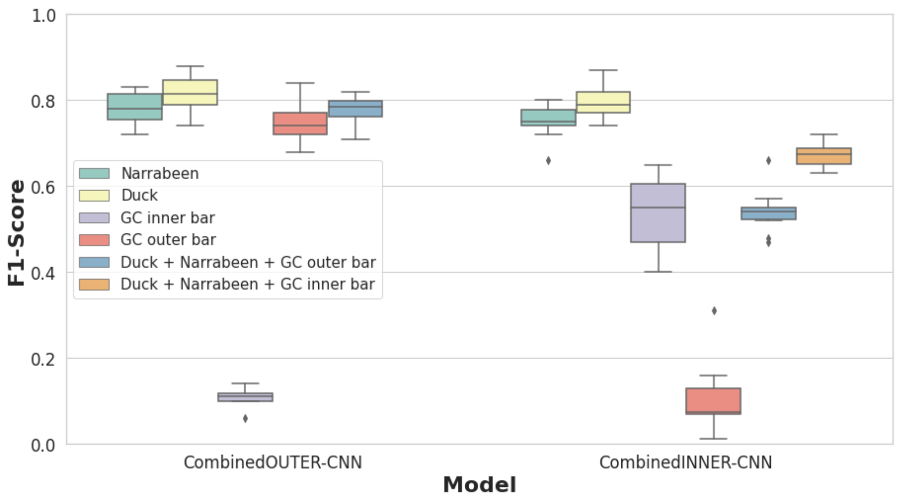


Figure 4.8: F1 score performance values from tests of the CombinedINNER-CNN and the CombinedOUTER-CNN for the self-tests and the transfer-tests at Duck, Narrabeen, the Gold Coast.

of Figure 4.6d, only the 30% fits our expectations. However, similar to the CombinedINNER-CNN, the all-around best performance of the CombinedOUTER-CNN was achieved with 50% of additional data coming from the Gold Coast.

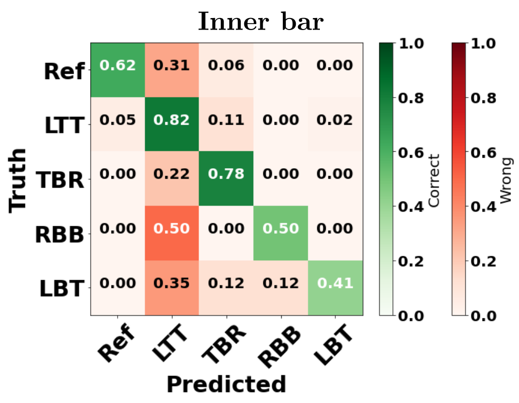
Figure 4.8 shows a box plot summarizing the F1 scores resulting from the tests conducted with the CombinedINNER-CNN and CombinedOUTER-CNN (Table 3.5b). Compared to the single-barred models, we see similar results for the performance on Duck and Narrabeen. In which the performance of Duck is slightly higher than the performance on Narrabeen. Furthermore, for the performance on the Gold Coast, the CombinedOUTER-CNN has overall better performance than the CombinedINNER-CNN, suggesting that the outer bar's signature is more clear and therefore easier to classify. In addition, we see that the model fed with data with the inner bar labels performs well on the inner bar and very poor on the outer bar, and vice versa. When tested on the test sets of Duck, Narrabeen, and the Gold Coast combined, both CNNs showed similar behaviour as we saw when tested on the Gold Coast and the CNNs performed better when the bar was involved on which the specific CNN was trained.

4.2.2 Per-class Performance

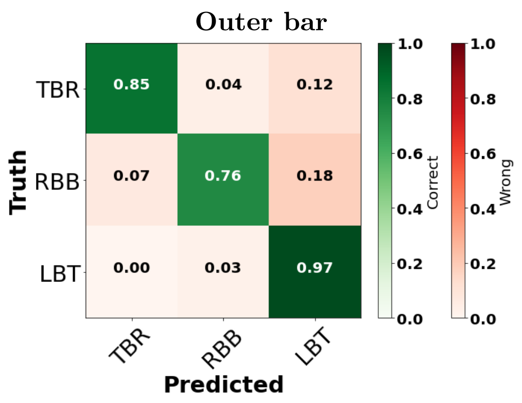
Single-location models

Figure 4.9 shows the confusion matrices for self-tests of the INNER-CNN and OUTER-CNN. In the case of the INNER-CNN, we achieved per class F1 scores ranging between 0.50-0.76, with the highest score in the classification of both LTT and TBR states and the lowest score in the RBB state. The class with the lowest recall is RBB, of which 50% of the images were classified as LTT. Moreover, 31% of the Ref images, 22% TBR and 35% of the LBT images are also classified as LTT.

In the case of the self-test of the OUTER-CNN, we achieved F1 scores of 0.84 for LBT, 0.83 for RBB, and 0.89 for TBR. The only major confusions occurred in the state with the lowest precision, the LBT state, in which true RBB and true TBR images are relatively often classified.



(a) INNER-CNN



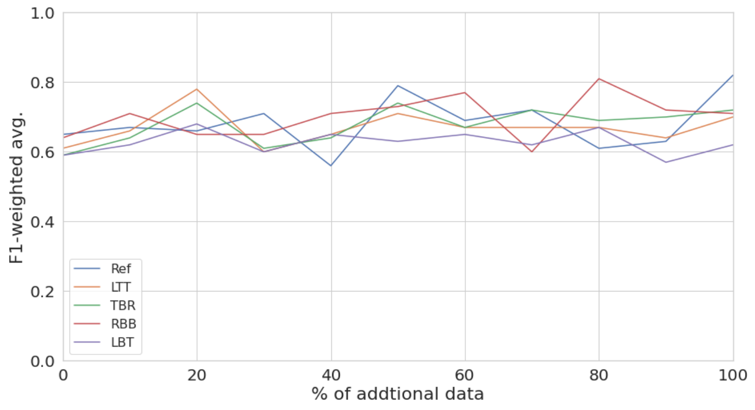
(b) OUTER-CNN

Figure 4.9: Normalized confusion matrices for the self-tests of the INNER-CNN (a) and OUTER-CNN (b).

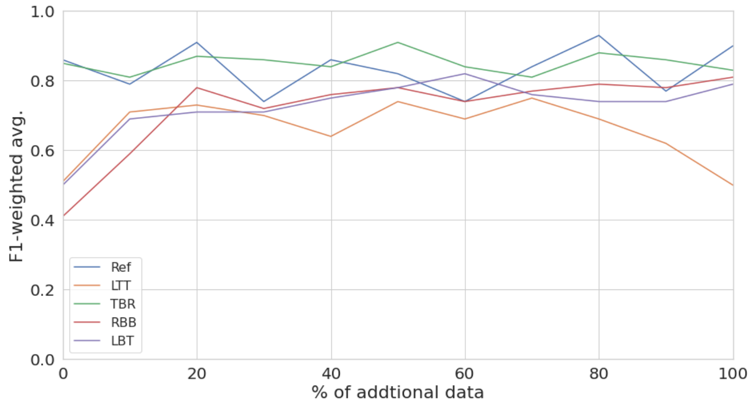
Combined-location models

The per-class performances of the CombinedINNER-CNN and CombinedOUTER-CNN have been evaluated in two ways. Figure 4.10 shows the per-class performances relative to the amount of additional data, whereas Figure 4.11 displays the confusion matrices for the models trained with 50% of additional data.

When looking at the per-class performances of the CombinedINNER-CNN, relative to the amount of additional data (Figure 4.10a), we see similar behaviour for all classes. The F1 scores are fluctuating between F1 values of approximately 0.6 and 0.8, with the largest fluctuations in the RBB and Ref states. Interesting to see is that these two classes correlate with the classes in which the least images are available in the CombinedINNER-CNN training set (Table 3.8a). In addition, as we would expect from the corresponding loss plots of Figure 4.6, the highest all-around performance is achieved for the 50% of additional data.



(a) Per class performance of CombinedINNER-CNN

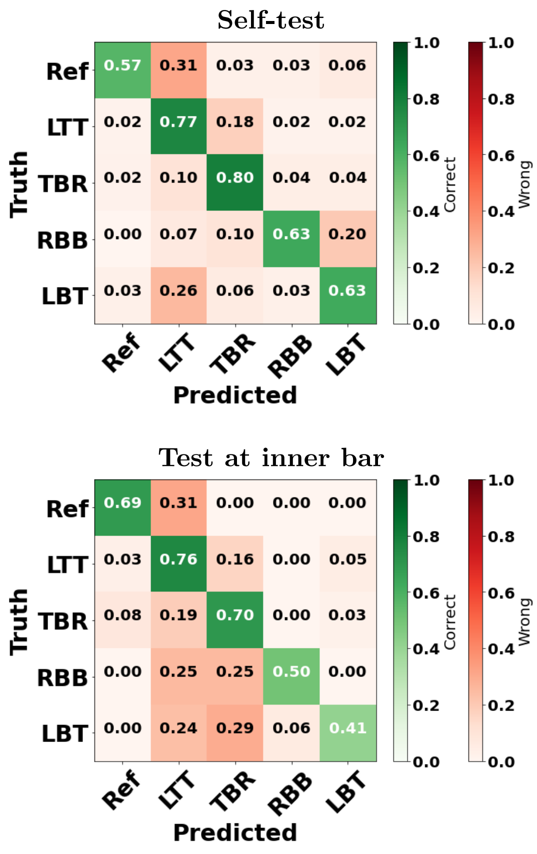


(b) Per class performance of CombinedOUTER-CNN

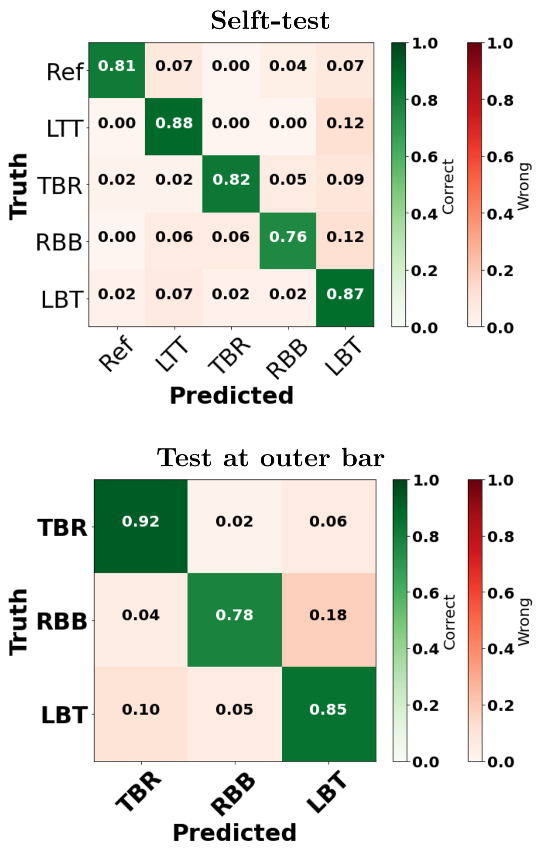
Figure 4.10: Per class F1 scores with respect to percentage of additional training data from the Gold Coast with either inner(a) or outer bar(b) labels.

In the case of the CombinedOUTER-CNN (Figure 4.10b), we see a clear difference compared to the CombinedINNER-CNN. The per-class performances vary a lot, with relatively low performances for the RBB, LBT and LTT states when 0% of data is added. In addition, the RBB, TBR and Ref classes are heavily fluctuating compared to the other classes. Again, these correlate with the classes in which the least images are available in the training set (Table 3.8b). Remarkably, the Ref class has a higher performance than the other classes while no Ref images are added. Furthermore, the performance of the LTT is relatively low compared to the other classes, and the LTT performance decreases significantly after 70% of data is added. For the CombinedOUTER-CNN the all-around highest performance at 50% is harder to confirm from Figure 4.10b, compared to the case of the CombinedINNER-CNN.

The fluctuations correlating to the minority classes of both the CNNs' training sets, suggest that the imbalances within the training sets had, despite the application of oversampling, a larger influence than anticipated.



(a) *CombinedINNER-CNN.*



(b) *CombinedOUTER-CNN.*

Figure 4.11: Normalized confusion matrices for the tests of CombinedINNER-CNN and CombinedOUTER-CNN.

In the second way of evaluating the CombinedINNER-CNN and CombinedOUTER-CNN, we looked at Figure 4.11. This figure shows the confusion matrices corresponding to the tests of the CombinedINNER-CNN and CombinedOUTER-CNN, both models with 50% of additional data.

For the self-test of the CombinedINNER-CNN (Figure 4.11a), we achieved per class F1 scores ranging between 0.65-0.75, with the highest score for the TBR state and the lowest for the LBT state. In addition, many misclassifications occurred between adjacent states, with one major exception in which LBT was classified as LTT.

In the case that we tested the CombinedINNER-CNN at the test set of the inner bar (Figure 4.11a), we achieved F1 scores ranging between 0.50-0.75. The highest F1 score for the LTT state and the lowest for the LBT state. Compared to the self-test of the INNER-CNN, the precision of TBR has decreased significantly. Hence, in this case, images are more often misclassified as TBR. On the other hand, the precision of RBB has increased. We have decreased F1 scores for LTT, TBR and LBT, resulting in overall lower performance on the inner bar for the CombinedINNER-CNN (0.68) compared to the INNER-CNN (0.72).

The self-test's predictions of the CombinedOUTER-CNN are very accurate (Figure 4.11b). The F1 scores within the range 0.78-0.87, with the highest for the TBR state and the lowest for the LBT state. The lowest precision of 0.71 corresponds to the LBT state, as most misclassifications are predicted as LBT.

When the CombinedOUTER-CNN is tested at the outer bar (Figure 4.11b), we achieved F1 scores very similar to the self-test of the OUTER-CNN with F1 scores of 0.80 for LBT, 0.84 for RBB, and 0.91 for TBR. The lowest precision is for the LBT state, indicating that many images are classified as LBT; this accounts especially for the RBB state. On the other hand, we see that many images of the LBT state are classified as TBR. Notable is the fact that when we tested the CombinedSINGLE-CNN on the outer bar, we saw predictions in the Ref and LTT states (Figure 4.5), while the outer bar has no labels belonging to those states. In the case of the CombinedOUTER-CNN, we used the same training data from the single-barred beaches as used in the CombinedSINGLE-CNN but now with additional data from the Gold Coast. When testing the CombinedOUTER-CNN on the outer bar, the model showed an improved performance with only predicting the three states included in the outer bar's dataset.

Chapter 5

Discussion

The research questions proposed in the first chapter of this thesis are: *How do pre-trained networks, fine-tuned with single-barred beach imagery perform on the classification of beach states of double-bar beach systems?* and *How do pre-trained networks, fine-tuned on beach imagery of single-bar beaches and complemented with data of a double-barred beach system perform on the classification of beach states of double-bar beach systems?* In the sections below, the answers to these questions are provided in terms of overall and per-class performances. For the interpretation of the results of CNNs it is important to keep in mind that these data-driven models do not need any explicit representation of physical processes and variables. The results are discussed in section 4, which are supplemented with Guided-CAMs for model interpretation in figures 5.1, 5.2 and 5.3. The evaluation and interpretation of the results is followed by the limitations, challenges and future perspectives.

5.1 Results Evaluation and Interpretation

5.1.1 Single-bar models

Single-bar beaches: Duck and Narrabeen

Our classification algorithm provides an extension of the Ellenson et al. (2020) approach to a double-barred beach setting. We applied transfer learning to fine-tune the pre-trained ResNet50 network rather than training it from scratch. To validate our model's performance, we compared the results on single-barred beaches from this study with the results of Ellenson et al. (2020). We found that the overall performances are comparable for both studies. Suggesting that CNNs pre-trained on large public datasets and fine-tuned with only single-bar beach imagery can correctly classify beach states in single-bar systems. We found relatively small differences in skills, with the only significant difference in the performance of the models tested at Narrabeen. Ellenson et al. (2020) reported F1 scores of 0.80 and 0.59 for the self-test of Duck and Narrabeen, respectively, and a maximum F1 score of the combined dataset of 0.69. In comparison, in this thesis we achieved F1 scores of 0.89 and 0.76 for the self-tests and a maximum F1 score of 0.78 for the combined dataset.

Consistent with the work by Ellenson et al. (2020), we found that many misclassifications made by the CNNs trained with data from Duck and Narrabeen mostly can be attributed to states that are adjacent to one another in the classification scheme of Wright and Short (1984). Besides these misclassifications, we found more misclassifications consistent with the work by Ellenson et al. (2020). These included confusions between RBB and LBT, both corresponding to an offshore sandbar with a distinct trough, and between TBR and LTT, both corresponding with bar welding. In addition, TBR and LTT may both include rip currents, with the differentiating factor being the large number and size of rip currents present in TBR. Additionally, we found more misclassifications in states with similar optical signatures, such as the linear and the rhythmic states. These misclassifications indicate that the models have more difficulty with states with similar morphology.

As expected from the overall performances, the number of misclassifications made at Narrabeen was typically higher than at Duck. The model's lower skill at Narrabeen indicates that classifying beach states at Narrabeen is more complicated than Duck. Ellenson et al. (2020) stated that a possible explanation could be the difference in optical signatures of the shoreline. In general, the CNN identifies the shoreline and offshore features when classifying beach states. In Argus timex imagery, due to swash motions, Duck's shoreline is consistently identifiable by higher image intensities. On the contrary, shoreline detection at Narrabeen happens in two ways: by higher image intensities associated with swash motions or by lower intensities associated with wet, dark sand (Pianca et al., 2015). Due to the lack of a consistent optical signature of the shoreline at Narrabeen, the separation between shoreline and sandbar is less evident than at Duck. Therefore, the labelling and prediction difficulty is enhanced. This is consistent with our results in which the classes of the linear states, in particularly the Ref/LTT classes and the LBT class, are confused, with the main difference in the distance between the shoreline and the sandbar.

Several factors can explain the differences in model performances between the work by Ellenson et al. (2020) and this study. One of these factors may be our different approach in data distribution, resulting in different images in our training and test sets. Furthermore, as described in section 3.1.2, during this study, the labels of some images in the dataset of Narrabeen were adjusted. Thus, in the hypothetical case that we had the same images in our training and test sets, this would mean that the labels would have been different. Another factor that has affected the model performance is the difference in methodology, and more specific, the difference in preprocessing, parameter settings and training protocol. Even though the same model architecture was used, we used a different training method. Ellenson et al. (2020) trained their models from scratch, whereas we fine-tuned the models. In addition, we used a different learning rate decay scheduler for super-fast learning, cycling through the learning rate during training rather than performing a sequential update. It would be interesting to compare the train and validation losses of the models from the work of Ellenson et al. (2020) with our models to evaluate the training speed and progresses of both protocols. However, this is not possible as these losses have not been reported by Ellenson et al. (2020).

During preprocessing, we split the data into three sets (train/validation/test) and optimized our model on the validation set instead of splitting it into two sets (train/test) and optimizing the model on the training set. Furthermore, we resized our images to the required input format of 224 x 224 instead of 512 x 512. We used different augmentation methods and applied weights for oversampling. Consequently, training CNNs requires setting many hyper-parameters (Alzubaidi et al., 2021). Hence, using several different parameters in this study will undoubtedly have influenced the performance. All these factors of model architecture, preprocessing, parameter settings, and training protocols are of significant importance for the performance of a model on a specific task. Therefore, it is hard to draw any conclusions from the differences between the results of this study and the work by Ellenson et al. (2020). However, these differences do underline the importance of making codes publicly available for evaluation and further research.

Figures 5.1 and 5.2 show the Guided-CAMs for Duck and Narrabeen, respectively. In the first column, we see the resized, raw images, in the second, third and fourth columns, the Guided-CAMs computed with the CombinedSINGLE-CNN, CombinedINNER-CNN and CombinedOUTER-CNN. These Guided-CAMs show us the pixels that played major roles in classifying these images. When looking at the second column, the Guided-CAMs computed using the CombinedSINGLE-CNN, we see for both Duck and Narrabeen highlights on the sandbars and simultaneously much noise around it. The highlights of the sandbar features suggest that the model was training on the preferred features of single-barred beaches.

Double-bar beach: Gold Coast

In order to answer the first research question, the CombinedSINGLE-CNN has been applied to the dataset of the Gold Coast. To assess the performance on this dataset, it is important to recall that the Gold Coast dataset consists of images that have separately been labelled for the inner and the outer bar. Applying a model with as output a single class will thus result in only a single prediction. Hence, when applying the model to the dataset of the Gold Coast, the model only classifies the bar with the most prominent features, which in one image may be the inner bar and in another image the outer bar.

Using the CombinedSINGLE-CNN, we achieved F1 scores of 0.32 and 0.55 on the inner and outer bars, respectively, which means that when the CombinedSINGLE-CNN is tested on the test data of the Gold Coast, just $\sim 32\%$ of the inner labels and $\sim 55\%$ of the outer labels is correctly predicted. The per-class performances showed that for the inner bar, the images primarily were classified as LTT, TBR or LBT. For the outer bar, this was primarily the TBR and LBT states.

The misclassifications at the inner bar may be due to the distance to the shoreline. The Argus imagery of the Gold Coast consists only of low-tide images. This means that sometimes the inner bar is often close to the shoreline and, therefore possibly confused as LTT. Additionally, sometimes the inner bar is not even visible at all. In these cases, the classification would probably be based on the outer bar, of which there are only TBR, RBB and LBT states in our dataset. Thus, the probability of being classified as one of these three states is relatively high. Furthermore, for the misclassifications of the outer bar, we primarily see RBB states classified as LBT, which may be due to similarities in morphology.

The poor overall performance suggested that the algorithm required additional data from the Gold Coast to improve transferability performance. The fact that the outer bar is more often correctly classified implies that the model has based its predictions more often on the outer bar's features than the inner bar's features. When looking at the Guided-CAMs corresponding to the CombinedSINGLE-CNN in Figure 5.1, the sandbars of the Gold Coast are just slightly highlighted and surrounded by much noise. From the F1 scores, we would expect the outer bar to be more highlighted than the inner bar. However, it is hard to tell by the highlighted pixels on the features of which sandbar the models have been trained.

All these results imply that the models, fine-tuned with only single-bar data, cannot properly classify the double-barred beach of the Gold Coast.

5.1.2 Double-bar models

Our second research question was related to double-barred beaches: *How do pre-trained networks, fine-tuned on beach imagery of single-bar beaches and complemented with data of a double-barred beach system perform on the classification of beach states of double-bar beach systems?* To answer this question, we conducted a second experiment (Section 3.4.2). In this experiment, we assessed the transferability of the CombinedSINGLE-CNN by adding additional data from the Gold Coast to train four different CNNs. This section discusses the results of this experiment and visualizes the models using Guided-CAMs.

Single dataset

We conducted tests using the INNER-CNN and OUTER-CNN. Table 3.7 already showed that the labels corresponding to the inner bar comprised all states used in this thesis, whereas the labels of the outer bar comprise only the higher-energetic states: TBR, RBB and LBT. The overall performances of the self-test of the INNER-CNN and OUTER-CNN were promising, with F1 scores of 0.73 and 0.88. These F1 scores suggest that models trained with data from the Gold Coast are able to classify and even discriminate the inner and outer bars.

We found that when the INNER-CNN was tested at the Gold Coast it heavily over-predicted the LTT and, slightly, the TBR states of the inner bar. For the OUTER-CNN this was slightly the case for the LBT state. We compared these states with the ratios of occurrence in the training datasets and found that these states correspond with the classes of which most images were available. This suggests that the misclassifications in these cases were mainly caused by imbalanced datasets, rather than similar morphology.

In Table 3.7 this imbalance is clearly visible. The training data of the inner bar was heavily imbalanced, with the frequencies of Ref/LTT/TBR/RBB/LBT were 13%/62%/18%/2%/5%. Additionally, the training data of the outer bar was moderately imbalanced, with the frequencies of TBR/RBB/LBT were 37%/23%/40%. Despite the weights applied to the classes in the preprocessing step, the per-class F1 scores indicated that the imbalances still affected the results, with the highest values for the majority classes and the lowest values for the minority classes. This influence is, in general, visible in the per-class recall and precision scores, with the majority class having high recall and low precision scores. For the inner bar, these were 0.62/0.82/0.78/0.50/0.41 and 0.77/0.71/0.74/0.50/0.88 for recall and precision, respectively. This trend is also visible for the recall and precision scores of the outer bar with, respectively, 0.85/0.76/0.97 and 0.94/0.92/0.73. These values confirm that imbalances in our dataset have influenced our results and suggests that the oversampling did not have the desired effect.

This can be explained by the method used to optimize our model. During training, the accuracy of the validation set is consistently monitored and recorded. As we optimized our models concerning the accuracy, only the model with the highest accuracy was eventually saved. Optimizing a model concerning the accuracy disadvantages the minority classes as accuracy only measures if a sample is predicted correctly and does not care about the per-class performance. Thus, a model optimized for accuracy tends to predict the majority class more quickly than the minority class. Correcting this imbalance using weights works only to improve the model to a certain level, and it never completely removes the imbalances. An alternate method would be to optimize the model for loss instead of accuracy.

Combined datasets

The models fine-tuned with solely single-bar data were not able to properly classify double-bar beach states. On the other hand, models fine-tuned with solely double-bar data were able to classify double-bar beach states properly. In the work by Ellenson et al. (2020) was suggested that with at least 25% of training data is from a specific site, comparable F1 scores to the self-test cases of each location can be achieved. Hence, another test was conducted to assess the transferability of a model with data coming from both single and double-bar beach data.

When data from the Gold Coast, with the inner (outer) bar labels, was added to the combined single-bar dataset, the performance of the combined model on classifying the inner (outer) bar increased significantly up to rates comparable with the single-bar self-tests. At the same time, the performance on the outer (inner) bar decreased to an almost insignificant rate. This implies that the model is able to discriminate the inner and outer bar. In addition, the model's performance was better when fine-tuned with additional data from the Gold Coast labelled with the outer bar than when using the labels of the inner bar.

In Figure 4.7 we saw for both cases that at least 20% of data coming from the Gold Coast was required for reasonable transferability of the model to the new sight. This is lower than the suggested minimum of 25% by Ellenson et al. (2020). Moreover, we saw that, for both cases the best performance was achieved with an additional 33% of data from the Gold Coast. Simultaneously, the performance on the single-bar datasets of Duck and Narrabeen remained quite similar compared to the performance of the CombinedSINGLE-CNN. The fact that there does not seem to be much difference in the performance at Narrabeen and Duck after adding data of the Gold Coast indicates that increasing the diversity of images does not necessarily increase the overall skill at the original locations. However, it allows predictions at an alternate location.

From analysing the per-class performance of the CombinedINNER-CNN (Figure 4.11), we found the same misclassifications as that we found in earlier tests. The self-test of the CombinedINNER-CNN showed misclassifications between LTT and TBR, similar to the test of the CombinedSINGLE-CNN at Duck, and it showed over-predictions for LTT and TBR, consistent with the self-test of the INNER-CNN. Moreover, when testing the CombinedINNER-CNN at the inner bar, the over-prediction of the LTT and TBR states increased compared to the self-test of the INNER-CNN. However, they decreased compared to the test of the CombinedSINGLE-CNN at the inner bar. This implies that datasets of Duck and Narrabeen contributed to the over-prediction of these classes. In addition, these results suggest that training with additional data from a new location does not cancel out the original errors. Rather, these errors become relative small as the total amount of test data increases.

In addition, the fact that the per-class performance with the labels of the outer bar was better than those with the inner bar labels may be caused since the inner bar evolves faster between states than the outer bar (Price and Ruessink, 2011). As the Argus images have been averaged, more rapid evolutions would result in wider white bands, and therefore, the classification difficulty would be enhanced. While the outer bar evolves more slowly between states, the white bands are possibly more distinct.

Looking at the Guided-CAMs generated by the CombinedINNER-CNN and CombinedOUTER-CNN, third and fourth columns respectively in Figures 5.1, 5.2 and 5.3, we see a significant difference with the Guided-CAMs generated by the CombinedSINGLE-CNN. In all cases the sandbars are more highlighted with less noise. This implies that the sandbars play a more important role in the classification. Despite the fact that the F1 scores do not suggest that the diversity of images increases the performances, the Guided-CAMs indicate that the models are able to better extract the preferred features. In the case of Duck and Narrabeen there is less noise and the sandbars are better highlighted. In the case of the Gold Coast it is clear that models based there classification on the sandbars. However, as both sandbars are highlighted it is still hard to distinguish the difference between the predictions for the inner and outer bar. It seems that the outer bar is more highlighted than the inner bar, however, as not for all images the Guided-CAMs are compared, no conclusions can be derived from this observation. As answer on our research question, we can state that models trained with at least 20% of additional data from a double-barred beach system are able to classify beach states in a double-bar beach system at a promising rate.

In the cases where we trained our models with additional data of the Gold Coast, we stopped adding data when the training data consisting of data from Duck and Narrabeen was doubled. Hnce, only 36% of the total amount of the Gold Coast's available training data was used to train the model. In most classification tasks, training the model on more data should improve the model's performance. However, as we saw from Figures 4.6 and 4.10, in the case of using all three datasets, higher amounts of data coming from one location, did not always positively affect the performance of the model. Especially when working with data from different sites, one study site may contain features unique to this specific site, such as an artificial reef, a specific alongshore current, or the unique water color. A majority of images coming from this site would probably result in the model training on such a unique feature. This would result in either overfitting on this site, or the fact that wrong features will be correlated with specific labels, thus decreasing the performance on the other sites. However, as the additional data is from a similar domain as the classification task at hand, it could still be used to pre-train the network before applying it with transfer learning to fine-tune the model using the training data.

The influence of the class imbalances in the datasets, particularly when the labels of the inner bar were used (Tables 3.2a&3.8a), are visible in the corresponding confusion matrices (Figures 4.9 and 4.11a&c), where the majority class, in this case LTT, is constantly over-predicted. The question is whether it matters for the outcome of this thesis. Some classification problems demand a very good precision or recall for one specific class, while not caring about the other classes. In contrast, another classification problem may aim to achieve high overall accuracy and does not care about the per-class performances. As for this thesis, the main objective was not directly to find the optimal parameter settings or the best preprocessing steps, and, the experiments mainly served as a concept of proof for the validity of CNNs for the classification of double-bar beach systems, the imbalances do not affect the answers on the research questions. However, future projects should define their purpose and requirements before collecting data and training the model (Further discussed below).

5.2 Challenges and Limitations

CNNs have been proven to outperform human classification performance in both general and more specialized classification problems (He et al., 2016; Krizhevsky et al., 2012). However, we encountered several limitations and challenges with using CNNs. CNNs require a huge amount of well-annotated training data to be optimally utilized. Moreover, as mentioned before, collecting representative data and accurate labelling of this data is very costly. In this thesis, we identified several issues as challenges.

Firstly, our amount of data is limited by the number of times each state occurred over the spanned period. Furthermore, depending on the environment of the different sites, some states were more common than other states, resulting in imbalanced datasets (Ellenson et al., 2020).

Secondly, in labelled datasets, just like the ones used in this thesis, there exist some uncertainties concerning the quality of the labels. In this thesis, we used the classification scheme of Wright and Short (1984). This scheme assumes that instantaneous beach morphology, as observed by Argus imagery, can be categorized in discrete states. However, beaches may exhibit sandbars with unique morphology, such as the rLTT and eTBR (Price and Ruessink, 2011), which are not covered in this classification scheme. In addition, sandbars exhibit a continuum of shapes as they evolve between configurations and may never reach a true equilibrium (Wright and Short, 1984; Price and Ruessink, 2011). Since there are no strict rules outlining each beach state, the identification of a beach state on a specific image may vary due to labeller perception. This perception may be different between different persons or even differ for the same person from day to day (Ellenson et al., 2020). To limit the perception bias, the datasets used in this thesis have mostly been labelled by a single person per dataset. During labelling, Ellenson et al. (2020) and Price and Ruessink (2011) used the most dominant state for determining the label. However, as the morphology evolves between states, the sandbar shape can exhibit characteristics of adjacent states in one image, thus faring in-between states or falling within more than one class (Wright et al., 1985; Price and Ruessink, 2011; Armaroli and Ciavola, 2011). Hence, a significant question is whether the optical signatures in Argus imagery and the classification scheme of Wright and Short (1984) are sufficient for classifying beach states using CNNs?

For the case of the optical signatures in Argus imagery, it might be argued that the model could be augmented with more detailed information concerning sandbar behaviour, such as wave height and wave incidence (Price and Ruessink, 2011; Pape et al., 2009). In the case of the classification scheme of Wright and Short (1984), it may be argued that the model should be expanded to discriminate more sandbars (Lippmann and Holman, 1990). Alternately, using the Argus imagery as a time series would be a major improvement to train models on temporal variability to predict beach states instead of solely classifying them. In that case, a RNN, designed for temporal analyses, could be a better option compared to a CNN (Pape et al., 2007; Pape and Ruessink, 2011).

5.3 Future work

The results presented in this thesis have proven that CNNs are able to perform beach state classifications on double-bar beach systems using Argus imagery. However, as discussed in the previous section, the limitations still offer many possibilities for improvement. These improvements can be made in many ways, for example by optimizing the model on loss instead of accuracy, the determination of the optimal hyper-parameter settings, or by considering other CNN architectures. Only the ResNet50 architecture has been applied to beach state classification. However, many more architectures are available (Alzubaidi et al., 2021). Several of these architectures were already included in the algorithm while developing it for this study. These architectures include VGG, InceptionV3, Alexnet, Densenet, and Squeezenet.

Other major improvements could be made in the collection of well-annotated data. More diverse training data often improve the model performance or make the model applicable to other locations. In this thesis, we used only data from three different Argus sites. However, the data from 47 more operational Argus systems are available. Using more data from different locations would be a step toward creating a universally applicable model for the automated classification of beach states. This study suggests that the current models would be able to classify the beach states of unseen beaches up to a certain level. However, we expect the performance on a new site to increase drastically when at least 20% and optimal (33%) of the total training data would come from this new site. Nevertheless, we still expect the classifications to be influenced by imbalanced training datasets.

Furthermore, in this study, we updated all trainable parameters using fine-tuning. As we now have models trained on Argus imagery to classify beach states, it would be interesting to see if the current models can function as the new 'pre-trained' networks. These pre-trained networks could, in turn, be employed in the transfer learning process in which only the classification part of the CNN is updated, the so-called feature extraction.

Moreover, as the labelling of new data is very costly, future research could benefit from the implementation of pseudo-labelling or model-assisted labelling to enhance the labelling process (Lee et al., 2013; Masubuchi et al., 2020; Zhu, 2005).

Using models to classify beach states in an automated way drastically reduces the amount of manual labelling. However, using models in this way also increases the significance of label quality. Imperfect labels could negatively affect the model's training process and, with it, the performance. Consequently, the information correlating to a specific classification scheme may be insufficient for the practical implication of these models. Hence, future research could enhance the label quality as well. In this thesis, we used a single-label prediction. Thus, the algorithm will give one classification output. However, a multi-label classification could give the possibility to label images with more than one label and give multiple outputs. Hence, the application of a multi-label classification (Wang et al., 2016) could partly resolve the challenges of alongshore variability and evolving bars. In addition, it would be interesting to see how a multi-label classification would handle the classification of double-bar systems when employed with both inner and outer bar labels.

In this thesis, we used a CNN for image classification and, more specifically, the classification of beach states from Argus imagery. However, this model could be used for other classification tasks and enhance coastal research. For example, the model could be applied to other morphological or hydrodynamic characteristics of coastal regions. For example, the level of pollution is of significant importance for coastal ecosystems, the kind of vegetation is important for dune and coastal management (Danielsen et al., 2005), and the location of rip currents is of major importance for recreational safety (Barbosa de Araújo and da Costa, 2008; Hanley et al., 2014). In practice, models for classifying these other characteristics can be constructed similar to the model used in this thesis.

Lastly, the possibilities for the use of CNNs are huge. Investigating these possibilities may result in major achievements in coastal research. Future research should be conducted on a wider application of CNNs, such as object detection and localization (Jiang et al., 2018) or object tracking (Bashir and Porikli, 2006). de Silva et al. (2021) used object detection and localization for rip currents. However, when applied to sandbars, the shape of a sandbar in a specific beach state could be learned through segmentation. The algorithm would be able to localize multiple beach states alongshore and cross-shore. Moreover, instead of using day averaged images, all images could be used as a time-series video. Using object tracking, a CNN could track the sandbar evolution.

We should keep in mind that, before deploying the model, we should consider the end-user's requirements and adapt the model to it in future research. Different CNN architectures or the dataset on which a model is pre-trained can differentiate between moderate or good performances on specific tasks. Furthermore, the evaluation metrics depend on the task at hand. The metrics used in this thesis were specifically valuable for multi-class image classification. However, other applications such as image segmentation would require different metrics.

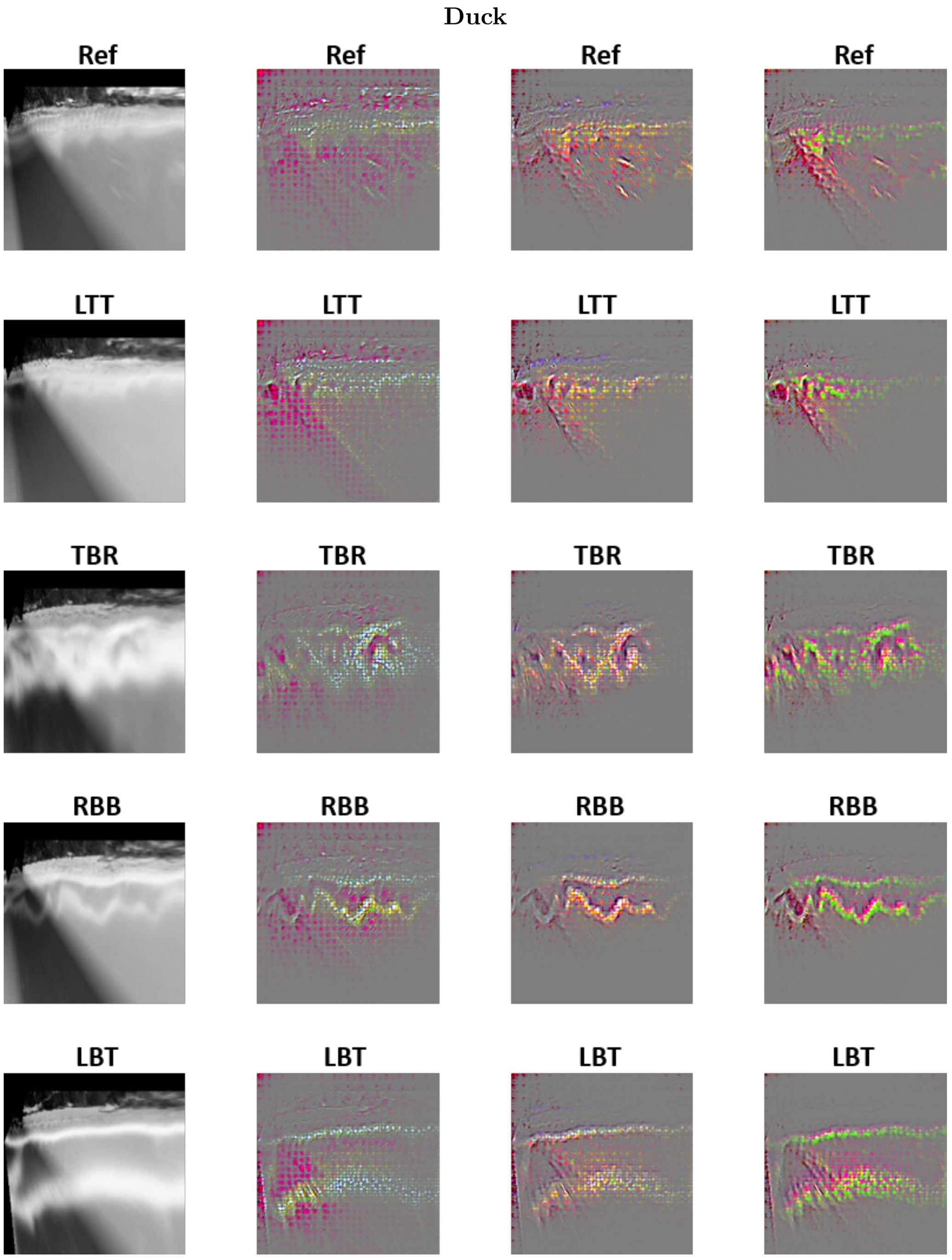


Figure 5.1: Guided-CAMs of Duck generated by guided-CAMs. In the left column are the resized raw images with corresponding beach states. The other columns are the Guided-CAMs generated using CombinedSINGLE-CNN, CombinedINNER-CNN and CombinedOUTER-CNN (l.t.r.), with corresponding predictions.

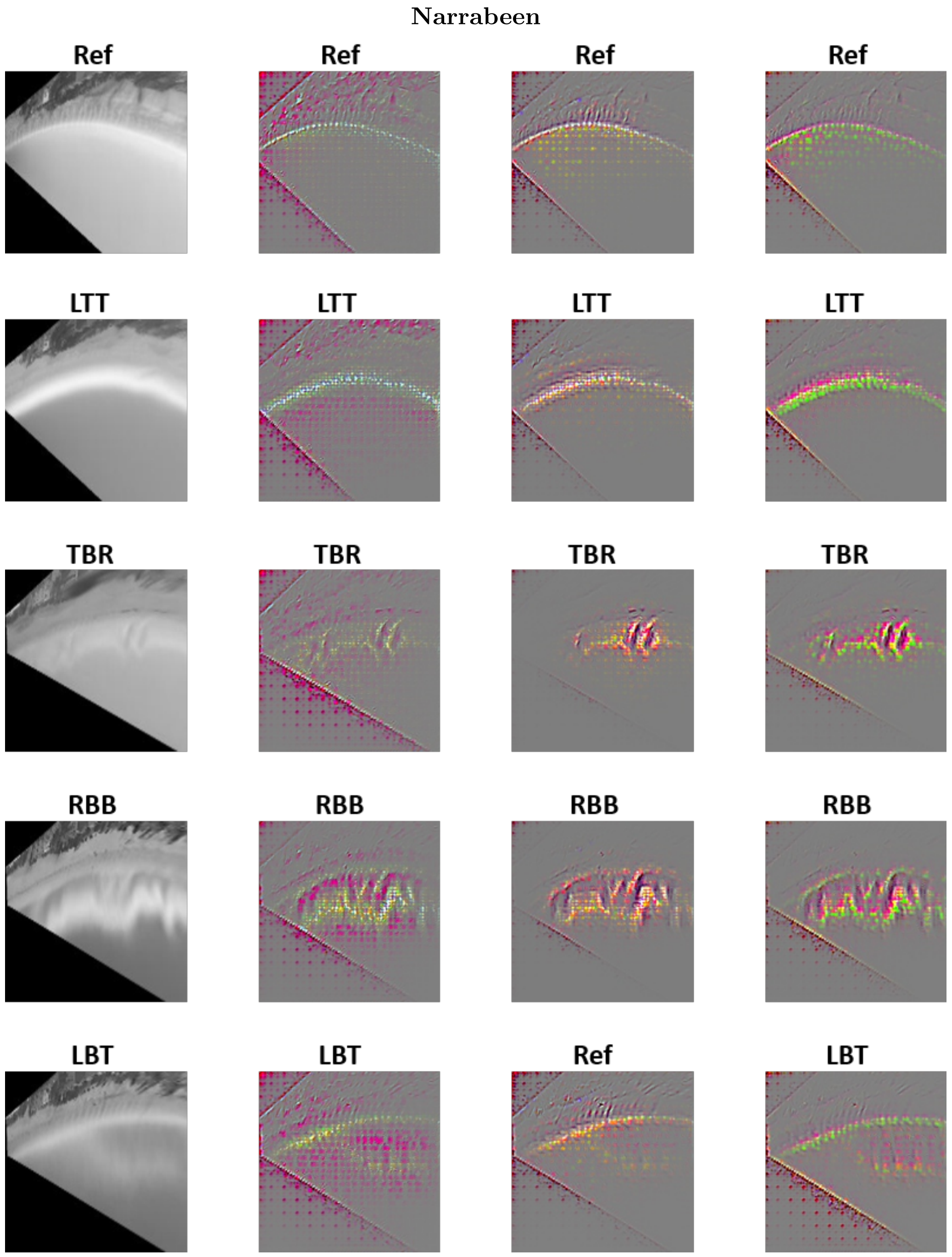


Figure 5.2: Guided-CAMs of Narrabeen generated by guided-CAMs. In the left column are the resized raw images with corresponding beach states. The other columns are the Guided-CAMs generated using CombinedSINGLE-CNN, CombinedINNER-CNN and CombinedOUTER-CNN (l.t.r.), with corresponding predictions.

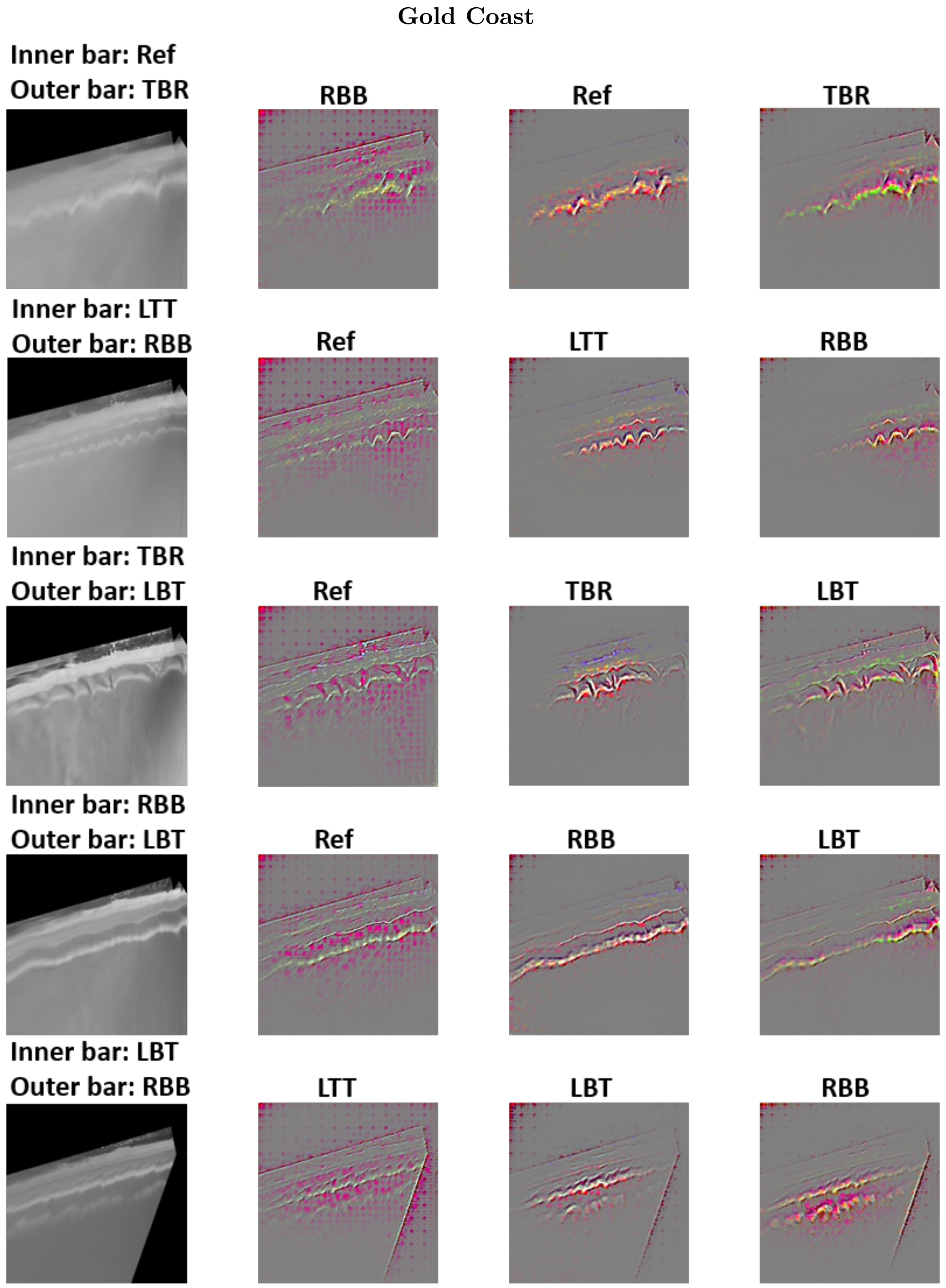


Figure 5.3: Guided-CAMs of the Gold Coast generated by guided-CAMs. In the left column are the resized raw images with corresponding beach states. The other columns are the Guided-CAMs generated using CombinedSINGLE-CNN, CombinedINNER-CNN and CombinedOUTER-CNN (l.t.r.), with corresponding predictions.

Chapter 6

Conclusion

This thesis built upon previous work by Ellenson et al. (2020) on the classification of beach states in single-barred systems using a convolutional neural network. The objective was to extend the previous work from solely the classification of single-barred beaches to the classification of beach states of double-barred beaches. Rather than training the model from scratch, we used transfer learning to fine-tune the pre-trained ResNet50 network. Datasets from three different beaches were used, the single-beach images from Duck, North Carolina, USA and Narrabeen-Collaroy, NSW, Australia, and double-bar beach images from the Gold Coast, Australia.

We conducted two experiments in which we trained and assessed seven different CNNs. Each model was tested on its own test data, the *self-tests*, and on the test data of the other location, the *transfer-tests*. Three models were trained with only the single-barred data; one at Duck, one at Narrabeen and one at data of both Duck and Narrabeen. We achieved F1 scores of 0.76 at Narrabeen, 0.89 at Duck and 0.78 for the combined model. When the model trained on the data of both Duck and Narrabeen was tested on the Gold Coast, we achieved poor performance for both bars (F1 score at inner bar = 0.32 and outer bar = 0.55). Consistent with Ellenson et al. (2020), this suggested that the algorithm requires additional data from a new location to improve the transferability performance.

In a second experiment, additional data from the Gold Coast was used for training. We trained two models as benchmarks, with only the double-barred beach data, and two models with data coming from Duck, Narrabeen and incrementally added data of the Gold Coast, with either the inner or outer bar labels. The skills of the self-tests of these models were comparable with the skills in the self-tests of the single-barred models of Duck and Narrabeen, with F1 scores of 0.73 and 0.88 for the models trained with either inner or outer bar labels, respectively.

The tests with the models trained with data of Duck, Narrabeen and incrementally added data from the Gold Coast showed that with at least an additional 20% of data, reasonable F1 scores could be achieved. Moreover, with an additional 33% of data, even F1 scores comparable (within 15%) to the self-test cases at each location were achieved. On the single-bar data, these models achieved F1 scores at Narrabeen and Duck of respectively, 0.80 and 0.85 for the model with labels from the inner bar, and F1 scores of 0.82 and 0.86 for the model with the outer bar labels. On the double-barred data, these models achieved F1 scores of 0.65 and 0.84 for the inner and outer bars, respectively.

Both experiments showed that the major misclassifications happened in adjacent beach states and beach states with similar morphological characteristics. Furthermore, the natural class imbalances in the datasets often caused the majority classes to be over-predicted.

It mattered which of the two bars was used for training the model. Training with the outer bar led to overall higher performances, except at the inner bar. When trained on the inner bar, only the performance on detecting the inner bar was higher than the models trained with the outer bar. Additionally, we saw that at least 20% and optimally 33% of data should come from each location when trained with data from multiple locations. More data from one location did not always positively affect the model's performance. However, the larger diversity of images allowed the transferability to more locations.

In future research, more factors such as wave height and alongshore and cross-shore variability should be considered with the application of CNNs for beach state classification. In addition, more data from different locations would benefit the model performance. They would be a step in the right direction for creating a universally applicable model for classifying beach states worldwide. Moreover, the application possibilities of CNNs are tremendous and could be very valuable for coastal research in the near future.

Bibliography

Aarninkhof, S. and Ruessink, G. (2002), Quantification of surf zone bathymetry from video observations of wave breaking, *in* 'AGU Fall Meeting Abstracts', Vol. 2002, pp. OS52E–10.

Abessolo Ondoa, G. (2020), Response of sandy beaches in West Africa, gulf of Guinea, to multi-scale forcing, PhD thesis.

Aleman, N., Certain, R., Robin, N. and Barusseau, J.-P. (2017), 'Morphodynamics of slightly oblique nearshore bars and their relationship with the cycle of net offshore migration', *Marine Geology* **392**, 41–52.

Alexander, P. S. and Holman, R. A. (2004), 'Quantification of nearshore morphology based on video imaging', *Marine geology* **208**(1), 101–111.

Allen, M. and Callaghan, J. (2000), *Extreme wave conditions for the south Queensland coastal region*, Environmental Protection Agency.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M. and Farhan, L. (2021), 'Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions', *Journal of big Data* **8**(1), 1–74.

Armaroli, C. and Ciavola, P. (2011), 'Dynamics of a nearshore bar system in the northern adriatic: A video-based morphological classification', *Geomorphology* **126**(1-2), 201–216.

Barbosa de Araújo, M. C. and da Costa, M. F. (2008), 'Environmental quality indicators for recreational beaches classification', *Journal of Coastal Research* **24**(6), 1439–1449.

Bashir, F. and Porikli, F. (2006), Performance evaluation of object detection and tracking systems, *in* 'Proceedings 9th IEEE International Workshop on PETS', pp. 7–14.

Birkemeier, W. A., DeWall, A. E., Gorbics, C. S. and Miller, H. C. (1981), A user's guide to cerc's field research facility., Technical report, COASTAL ENGINEERING RESEARCH CENTER FORT BELVOIR VA.

Bracs, M. A., Turner, I. L., Splinter, K. D., Short, A. D., Lane, C., Davidson, M. A., Goodwin, I. D., Pritchard, T. and Cameron, D. (2016), 'Evaluation of opportunistic shoreline monitoring capability utilizing existing "surfcam" infrastructure', *Journal of Coastal Research* **32**(3), 542–554.

Castelle, B., Scott, T., Brander, R. and McCarroll, R. (2016), 'Rip current types, circulation and hazard', *Earth-Science Reviews* **163**, 1–21.

Castelluccio, M., Poggi, G., Sansone, C. and Verdoliva, L. (2015), 'Land use classification in remote sensing images by convolutional neural networks', *arXiv preprint arXiv:1508.00092* .

Contardo, S. and Symonds, G. (2015), 'Sandbar straightening under wind-sea and swell forcing', *Marine Geology* **368**, 25–41.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q. V. (2019), Autoaugment: Learning augmentation strategies from data, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 113–123.

Danielsen, F., Sørensen, M. K., Olwig, M. F., Selvam, V., Parish, F., Burgess, N. D., Hiraishi, T., Karunagaran, V. M., Rasmussen, M. S., Hansen, L. B. et al. (2005), 'The asian tsunami: a protective role for coastal vegetation', *Science* **310**(5748), 643–643.

Davidson, M., Van Koningsveld, M., de Kruif, A., Rawson, J., Holman, R., Lamberti, A., Medina, R., Kroon, A. and Aarninkhof, S. (2007), 'The coastview project: Developing video-derived coastal state indicators in support of coastal zone management', *Coastal Engineering* **54**(6-7), 463–475.

de Silva, A., Mori, I., Dusek, G., Davis, J. and Pang, A. (2021), 'Automated rip current detection with region based convolutional neural networks', *Coastal Engineering* **166**, 103859.

Ellenson, A. N., Simmons, J. A., Wilson, G. W., Hesser, T. J. and Splinter, K. D. (2020), 'Beach state recognition using argus imagery and convolutional neural networks', *Remote Sensing* **12**(23), 3953.

Gardner, M. W. and Dorling, S. (1998), 'Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences', *Atmospheric environment* **32**(14-15), 2627–2636.

Gholamalinezhad, H. and Khosravi, H. (2020), 'Pooling methods in deep neural networks, a review', *arXiv preprint arXiv:2009.07485* .

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep learning*, MIT press.

Grant, S. B., Kim, J. H., Jones, B. H., Jenkins, S. A., Wasyl, J. and Cudaback, C. (2005), 'Surf zone entrainment, along-shore transport, and human health implications of pollution from tidal outlets', *Journal of Geophysical Research: Oceans* **110**(C10).

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. et al. (2018), 'Recent advances in convolutional neural networks', *Pattern Recognition* **77**, 354–377.

Hanley, M., Hoggart, S., Simmonds, D., Bichot, A., Colangelo, M., Bozzeda, F., Heurtefeux, H., Ondiviela, B., Ostrowski, R., Recio, M. et al. (2014), 'Shifting sands? coastal protection by sand banks, beaches and dunes', *Coastal Engineering* **87**, 136–146.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

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

Holman, R. A., Symonds, G., Thornton, E. B. and Ranasinghe, R. (2006), 'Rip spacing and persistence on an embayed beach', *Journal of Geophysical Research: Oceans* **111**(C1).

Holman, R. and Haller, M. C. (2013), 'Remote sensing of the nearshore', *Annual review of marine science* **5**, 95–113.

Hoshen, Y., Weiss, R. J. and Wilson, K. W. (2015), Speech acoustic modeling from raw multichannel waveforms, *in* '2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 4624–4628.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. (2017), Densely connected convolutional networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4700–4708.

Hubel, D. H. and Wiesel, T. N. (1962), 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *The Journal of physiology* **160**(1), 106.

Ioffe, S. and Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', PMLR, pp. 448–456.

Jackson, L. A., Tomlinson, R. and Nature, P. (2017), '50 years of seawall and nourishment strategy evolution on the gold coast', *Australasian Coasts & Ports* .

Jackson, L. A., Tomlinson, R., Turner, I., Corbett, B., d'Agata, M. and McGrath, J. (2005), Narrowneck artificial reef; results of 4 yrs monitoring and modifications, *in* 'Proceedings of the 4th International Surfing Reef Symposium'.

Janocha, K. and Czarnecki, W. M. (2017), 'On loss functions for deep neural networks in classification', *arXiv preprint arXiv:1702.05659* .

Jiang, B., Luo, R., Mao, J., Xiao, T. and Jiang, Y. (2018), Acquisition of localization confidence for accurate object detection, *in* 'Proceedings of the European conference on computer vision (ECCV)', pp. 784–799.

Ketkar, N. (2017), Stochastic gradient descent, *in* 'Deep learning with Python', Springer, pp. 113–132.

Khan, A., Sohail, A., Zahoora, U. and Qureshi, A. S. (2020), 'A survey of the recent architectures of deep convolutional neural networks', *Artificial intelligence review* **53**(8), 5455–5516.

Kingston, K., Ruessink, B., Van Enckevort, I. and Davidson, M. (2000), 'Artificial neural network correction of remotely sensed sandbar location', *Marine Geology* **169**(1-2), 137–160.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**, 1097–1105.

Lee, D.-H. et al. (2013), Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, *in* 'Workshop on challenges in representation learning, ICML', Vol. 3, p. 896.

Lee, G.-h., Nicholls, R. J. and Birkemeier, W. A. (1998), 'Storm-driven variability of the beach-nearshore profile at duck, north carolina, usa, 1981–1991', *Marine geology* **148**(3-4), 163–177.

Lin, M., Chen, Q. and Yan, S. (2013), 'Network in network', *arXiv preprint arXiv:1312.4400* .

Lippmann, T. C. and Holman, R. A. (1989), 'Quantification of sand bar morphology: A video technique based on wave dissipation', *Journal of Geophysical Research: Oceans* **94**(C1), 995–1011.

Lippmann, T. and Holman, R. (1990), 'The spatial and temporal variability of sand bar morphology', *Journal of Geophysical Research: Oceans* **95**(C7), 11575–11590.

Lipton, Z. C., Berkowitz, J. and Elkan, C. (2015), 'A critical review of recurrent neural networks for sequence learning', *arXiv preprint arXiv:1506.00019* .

Masubuchi, S., Watanabe, E., Seo, Y., Okazaki, S., Sasagawa, T., Watanabe, K., Taniguchi, T. and Machida, T. (2020), 'Deep-learning-based image segmentation integrated with optical microscopy for automatically searching for two-dimensional materials', *npj 2D Materials and Applications* **4**(1), 1–9.

Menardi, G. and Torelli, N. (2014), 'Training and assessing classification rules with imbalanced data', *Data mining and knowledge discovery* **28**(1), 92–122.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J. and Khudanpur, S. (2010), Recurrent neural network based language model., *in* 'Interspeech', Vol. 2, Makuhari, pp. 1045–1048.

Nusrat, I. and Jang, S.-B. (2018), 'A comparison of regularization techniques in deep neural networks', *Symmetry* **10**(11), 648.

Pan, S. J. and Yang, Q. (2009), 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering* **22**(10), 1345–1359.

Pape, L. and Ruessink, B. (2011), 'Neural-network predictability experiments for nearshore sandbar migration', *Continental Shelf Research* **31**(9), 1033–1042.

Pape, L., Ruessink, B. G., Wiering, M. A. and Turner, I. L. (2007), 'Recurrent neural network modeling of nearshore sandbar behavior', *Neural Networks* **20**(4), 509–518.

Pape, L., Ruessink, G. and Kuriyama, Y. (2009), Models and scales for nearshore sandbar behavior, *in* 'Proceedings Of Coastal Dynamics 2009: Impacts of Human Activities on Dynamic Coastal Processes (With CD-ROM)', World Scientific, pp. 1–13.

Parkhi, O. M., Vedaldi, A. and Zisserman, A. (2015), 'Deep face recognition'.

Pianca, C., Holman, R. and Siegle, E. (2015), 'Shoreline variability from days to decades: Results of long-term video imaging', *Journal of Geophysical Research: Oceans* **120**(3), 2159–2178.

Plant, N. G. and Holman, R. A. (1997), 'Intertidal beach profile estimation using video images', *Marine Geology* **140**(1-2), 1–24.

Plant, N. G., Todd Holland, K. and Holman, R. A. (2006), 'A dynamical attractor governs beach response to storms', *Geophysical Research Letters* **33**(17).

Price, T. and Ruessink, B. (2011), 'State dynamics of a double sandbar system', *Continental Shelf Research* **31**(6), 659–674.

Pricope, T.-V. (2021), 'An analysis on very deep convolutional neural networks: Problems and solutions', *Studia Universitatis Babes-Bolyai, Informatica* **66**(1).

Ribas, F., Ojeda, E., Price, T. D. and Guillén, J. (2010), 'Assessing the suitability of video imaging for studying the dynamics of nearshore sandbars in tideless beaches', *IEEE transactions on geoscience and remote sensing* **48**(6), 2482–2497.

Román-Rivera, M. A. and Ellis, J. T. (2019), 'A synthetic review of remote sensing applications to detect nearshore bars', *Marine Geology* **408**, 144–153.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al. (2015), 'Imagenet large scale visual recognition challenge', *International journal of computer vision* **115**(3), 211–252.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D. (2017), Grad-cam: Visual explanations from deep networks via gradient-based localization, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 618–626.

Sharma, S., Sharma, S. and Athaiya, A. (2017), 'Activation functions in neural networks', *towards data science* **6**(12), 310–316.

Short, A. D. and Aagaard, T. (1993), 'Single and multi-bar beach change models', *Journal of Coastal research* pp. 141–157.

Short, A. D. and Trenaman, N. (1992), 'Wave climate of the sydney region, an energetic and highly variable ocean wave regime', *Marine and Freshwater Research* **43**(4), 765–791.

Shorten, C. and Khoshgoftaar, T. M. (2019), 'A survey on image data augmentation for deep learning', *Journal of big data* **6**(1), 1–48.

Shrestha, A. and Mahmood, A. (2019), 'Review of deep learning algorithms and architectures', *IEEE access* **7**, 53040–53065.

Simonyan, K. and Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Smit, M., Aarninkhof, S., Wijnberg, K. M., González, M., Kingston, K., Southgate, H., Ruessink, B., Holman, R., Siegle, E., Davidson, M. et al. (2007), 'The role of video imagery in predicting daily to monthly coastal evolution', *Coastal engineering* **54**(6-7), 539–553.

Smith, L. N. and Topin, N. (2019), Super-convergence: Very fast training of neural networks using large learning rates, *in* 'Artificial intelligence and machine learning for multi-domain operations applications', Vol. 11006, International Society for Optics and Photonics, p. 1100612.

Splinter, K. D., Harley, M. D. and Turner, I. L. (2018), 'Remote sensing is changing our view of the coast: Insights from 40 years of monitoring at narrabeen-collaroy, australia', *Remote Sensing* **10**(11), 1744.

Springenberg, J. T., Dosovitskiy, A., Brox, T. and Riedmiller, M. (2014), 'Striving for simplicity: The all convolutional net', *arXiv preprint arXiv:1412.6806* .

Strauss, D., Mirferendesk, H. and Tomlinson, R. (2007), ‘Comparison of two wave models for gold coast, australia’, *Journal of Coastal Research* pp. 312–316.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 2818–2826.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. and Liu, C. (2018), A survey on deep transfer learning, *in* ‘International conference on artificial neural networks’, Springer, pp. 270–279.

Tătui, F. and Constantin, S. (2020), ‘Nearshore sandbars crest position dynamics analysed based on earth observation data’, *Remote Sensing of Environment* **237**, 111555.

Turner, I. L., Harley, M. D., Short, A. D., Simmons, J. A., Bracs, M. A., Phillips, M. S. and Splinter, K. D. (2016), ‘A multi-decade dataset of monthly beach profile surveys and inshore wave forcing at narrabeen, australia’, *Scientific data* **3**(1), 1–13.

Van Enckevort, I. and Ruessink, B. (2001), ‘Effect of hydrodynamics and bathymetry on video estimates of nearshore sandbar position’, *Journal of Geophysical Research: Oceans* **106**(C8), 16969–16979.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. and Xu, W. (2016), Cnn-rnn: A unified framework for multi-label image classification, *in* ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 2285–2294.

Weiss, K., Khoshgoftaar, T. M. and Wang, D. (2016), ‘A survey of transfer learning’, *Journal of Big data* **3**(1), 1–40.

Wright, L. D. and Short, A. D. (1984), ‘Morphodynamic variability of surf zones and beaches: a synthesis’, *Marine geology* **56**(1-4), 93–118.

Wright, L. D., Short, A. D. and Green, M. (1985), ‘Short-term changes in the morphodynamic states of beaches and surf zones: an empirical predictive model’, *Marine geology* **62**(3-4), 339–364.

Zeiler, M. D. and Fergus, R. (2014), Visualizing and understanding convolutional networks, *in* ‘European conference on computer vision’, Springer, pp. 818–833.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. and Torralba, A. (2016), Learning deep features for discriminative localization, *in* ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 2921–2929.

Zhu, X. J. (2005), ‘Semi-supervised learning literature survey’.