Group 27 - Point Classification on Beach Survey LiDAR Point Clouds

Austin Barnes
Scripps Institution of Oceanography
atbarnes@ucsd.edu

Raymond Young
Scripps Institution of Oceanography
ray011@ucsd.edu

Hannah Walker
University of California, San Diego
h2walker@ucsd.edu

***Abstract*—This report details the implementation and results of a Random Forest (RF), K-Nearest Neighbors (KNN), and Neural Network (NN) classifiers used to classify LiDAR point returns into land and wave classes. Pre-processing techniques including subsampling, data augmentation, and normalization as well as feature engineering for geometrical features are also included. Accuracy of over 99% was achieved on subsamples of training data while a maximum accuracy of over 83% was achieved for testing on a completely separate dataset.**

I. Introduction

The Center for Climate Change Impacts and Adaptation (CCCIA), part of Scripps Institution of Oceanography at UCSD, is one of the world's premier coastal monitoring teams. As part of their survey work monitoring and modeling coastal climates, they use truck-based mobile LiDAR (Light Detection And Ranging) which yields 3D topographic maps based on laser surface reflectance return times. The CCCIA uses LiDAR to do repeat beach and coastal cliff surveys in support of an array of research projects on beach morphology, cliff erosion, and ocean waves.

The LiDAR laser reflects best off of land, but in every coastal survey there are returns from wave surfaces. The CCCIA needs to identify and label LiDAR returns from wave surfaces because they complicate and confound true topological elevation measurements of beaches and cliffs and can also be used for wave studies of breaking waves and white water. For years, the CCCIA has been visually inspecting and manually labeling individual wave surface LiDAR returns in a time-intensive process. Our project goal was to use machine learning techniques to aid the CCCIA's research efforts by automatically classifying LiDAR points as either wave or land returns.

The input to our algorithm is a LiDAR point cloud (labeled as either waves or land for training) with five features for each point: three-dimensional (x,y,z) spatial location, intensity, and intensity cubed; in some tests we have added two additional features we have engineered. We trained three different models – KNN, RF, and a fully connected NN – to output predicted labels of waves or land for every LiDAR point.

II. Related Work

We referenced a few papers that focused on similar work to our project. First, we refer to work by Medina and Paffenroth [1] who train a KNN classifier, RF classifier, and NN applied to LiDAR data in order to classify terrestrial objects from aerial surveys. We also followed the work of Qi et al. [2] which introduced a "novel deep net architecture" called Pointnet used to classify and segment 3D point cloud data. These were especially useful for our purposes because they described various machine learning frameworks applied to non-uniformly distributed data similar to ours. We also took note of work by Li et al. [3] who took advantage of varying density patterns in LiDAR data in their convolution networks. This inspired us to include some feature engineering in our work to improve our models. Moorthy et al. provides another example of classification in the terrestrial environment using RF, XGBoost, and lightGBM machine learning algorithms [4]. Lastly, scientists at the CCCIA have demonstrated the usefulness of machine learning techniques to automatically detect beach cobbles in these same beach surveys [5].

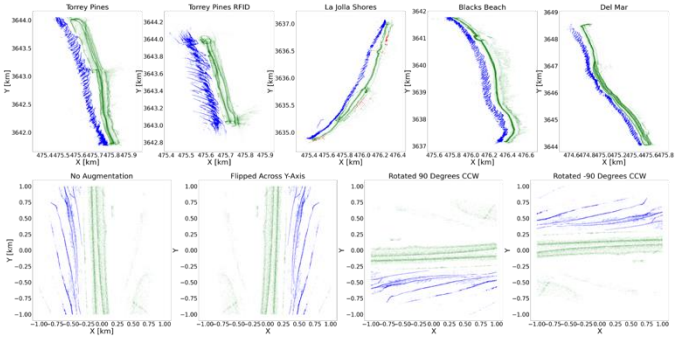

Figure 1. A summary of the 5 datasets (top) with labeled land (green), waves (blue), and noise (red, ignored). (bottom) A demonstration of 100 m segments of beach that have been normalized between (-1,1) with flipped and rotated augmentations applied.

III. Dataset and Features

A. Raw Data

CCCIA provided us five LiDAR point cloud beach surveys of southern California beaches (La Jolla Shores, Blacks Beach, Del Mar, and two from Torrey Pines). Each file contains over 90 million irregularly spaced points (>1 billion points total) with four feature attributes: three-dimensional spatial location and intensity of the laser return. The x- and y-dimensions represent the east-west and north-south positions, respectively, in units of meters referenced to NAD83(2011) epoch 2010.00 datum in projection UTM Zone 11N. The z-dimension is elevation referenced to NAVD88 Geoid 12b (close to but not quite current mean sea level). The CCCIA provided pre-labeled data so that in addition to those features, each point has a classification as either land or water as seen in Figure 1.

B. Features

It is apparent in Figure 1 that LiDAR returns from waves have strong linear features relative to the more uniformly spaced land returns. We decided to test if we could take advantage of these differing geometries using feature engineering to increase model accuracy, and created two additional features to extract the relevant geometries: for a given point we find k=40 nearest neighbors based only on x, y, and z and computed the R^2 metric

[6] to a linear fit of x and y, and a cosine similarity metric, eq. (1) [7]. Both features were designed to have values that correspond to the linearity of the neighbors of each point.

$$cos\theta = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}||\left|\left|\vec{b}\right|\right|} \quad (1)$$

C. Pre-Processing

Ideally, we would have trained and tested the models on the full surveys and dealt with the large class imbalance (>99% classified as land) by doing inverse sample weighting, but we ran into computational efficiency and memory problems with such large datasets. To deal with both the class imbalance and our computational limitations, we decided to randomly subsample the land returns so that our datasets have equal numbers of land and wave returns – because of the uniformity of land points we hypothesize this should have negligible impact on model predictive accuracy. After subsampling, we also split full beach surveys into 100 meter alongshore segments to split the data into smaller, more manageable parts to reduce computational stresses and to reduce dependency on coastline geometry and performed normalization to each of these segments individually. We scaled x- and y-dimensions to fall between –1 and 1 to preserve some symmetry of the shoreline and both the z-dimension and intensity to fall between 0 and 1. Initial tests indicated that trained models were highly dependent on the similar geometry of the beaches we were provided. To reduce model dependency on the x-dimension (east-west) and create models robust to different beach geometries, we tested models with augmented datasets in which we added flipped beaches along the y-axis to simulate an east coast beach and rotated beaches about the z-axis by 90 and –90 degrees to simulate north- and south-facing beaches, respectively.

IV. Methods

Based on Medina & Paffenroth [1], we present a comparison between 3 types of models: KNN classifier, RF classifier, and a NN. Each of these models were trained using the same features of x, y, z, and intensity and intensity^3, as well as the aforementioned R^2 value and cosine similarity metric for models that take advantage of "feature engineering." Hyperparameters for each respective model were also motivated by the results presented by [1] because they demonstrated good accuracy for a similar structure of classification problems as well as their ability to run at reasonable computation times on our hardware limitations.

Both KNN and RF models were implemented using the open source *SKlearn* package for machine learning [8] and the NN model was implemented through the *Keras* libraries [9] with *TensorFlow* [10].

A. KNN

KNN is a clustering algorithm that is based on finding the n-dimensional distance between the point one wants to classify and other points in the dataset where each dimension corresponds to a feature [11]. Specifically for classification, the test point takes on the class of the majority class of its k-nearest neighbors. Two of the most important hyperparameters in these models are the choice the number of neighbors, for which we chose k=15, and the distance metric, for which we chose the Euclidean distance metric (eq (2) where **p** and **q** are n-dimensional vectors) which represents the straight-line distance between each point.

$$\mathrm{d}(\boldsymbol{p},\boldsymbol{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} \quad (2)$$

B. RF

A RF is a general method for bagging points in multidimensional spaces based on a set of sequential segmentations based on a single dimension of the space [12]. The number of segmentations in the space is controlled by the depth of the tree. For classification, each trained tree in the forest provides a classification for the query point and the most popular choice becomes the assigned label. The number of trees is another primary hyperparameter. For this specific model define the tree depth of 20, a forest of 20 trees, and bootstrapping which is a method in which each tree is a random subset from the entire training dataset with replacement.

C. NN

Our feedforward neural network architecture consists of two fully connected layers. The first hidden layer consists of 20 neurons, each of which carries a weights matrix that is multiplied by the input (feature) matrix, adds a bias vector, and is then activated by the rectified linear unit (ReLU) function [13]. The second hidden layer consists of 15 neurons connected to all of the previous layer's outputs and is again activated by the ReLU function. The final output layer uses the softmax function (refer to eq. 6.29 in [13]) to produce probabilities of the point being classified as either land or wave. The NN classifier used the categorical cross entropy loss metric to complement the softmax output and 'adam' optimization that combines properties of AdaGrad and RMSProp algorithms [14] We used a batch size of 1000 following Medina & Paffenroth [1] but used only 200 epochs based on the accuracy and loss curves (see presentation slides) and a learning rate of 0.001 to allow for smaller incremental changes in the model training.

V. Experiments/Results/Discussion

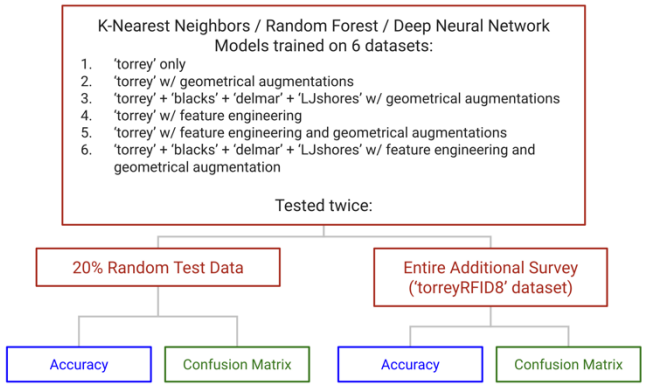

Figure 2. This chart summarizes the 6 datasets, made of combinations of specified beaches, and pre-processing steps applied to each model. It also details the types of testing presented in Table 1.

Our results detail the training of 6 different models with testing on a 20% random subsample taken from the same beaches as training with a second test performed on a completely different scan. This process is detailed in Figure 2 as well as Table 1.

For each of the tests we performed, we used two primary evaluation metrics: accuracy and a confusion matrix. Accuracy, eq (3), is defined as the percentage of total classifications that the model predicts correctly, and the confusion matrix, eq (4), differs from the standard definition in that it is normalized by the number of samples for each actual label. This gives what is normally defined as precision along its diagonals.

$$\text{Accuracy} = \frac{\#\ Points\ Correctly\ Classified}{Total\ \#\ Points} \tag{3}$$

$$[\text{Confusion Matrix}] = \begin{bmatrix} \frac{\#\ Land\ Points\ Correctly\ Classified}{Total\ \#\ Land\ Points} * 100\% & \frac{\#\ Land\ Points\ Misclassified}{Total\ \#\ Land\ Points} * 100\% \\ \frac{\#\ Wave\ Points\ Misclassified}{Total\ \#\ Wave\ Points} * 100\% & \frac{\#\ Wave\ Points\ Correctly\ Classified}{Total\ \#\ Wave\ Points} * 100\% \end{bmatrix} \tag{4}$$

As discussed earlier, we performed two different tests on each model to evaluate the model's ability to predict classifications on very similar data to the training data and on a new survey altogether. Table 1 summarizes the results of trained models with reported accuracies and confusion matrices. We have color coded the table to indicate the model algorithm for a given training/test set with highest overall accuracy (green), second highest (yellow), and lowest (red). The spread between accuracies of the algorithms varies from test to test, but the confusion matrices give a clearer picture of how algorithms are misclassifying. The KNN and RF classifiers perform the best most often; KNN appears to perform better than RF when applied to a new survey.

Tests 7-12 are identical to tests 1-6 except that the input data includes our engineered features. All model accuracies are greatly enhanced by the inclusion of these engineered features, indicating that the geometrical properties of the wave returns can be used to enhance model performance. Figure 3 shows the random forest feature importance from model 6 including the engineered features.

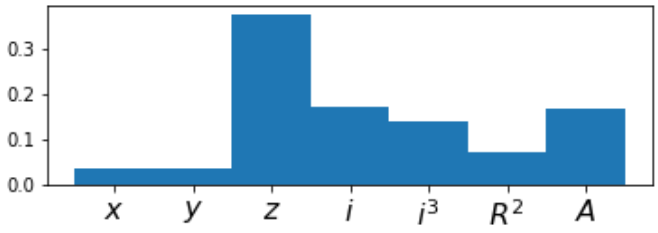

Figure 3. Feature importance for the RF classifier, including engineered features. x,y,z are the three-dimensional spatial locations, i is intensity, i^3 is intensity cubed, R^2 and A are the geometrical features described in section IIIb.

We also looked at feature importance and visually inspected classifications for the RF classifier to determine the efficacy of our data augmentations in training a more robust model that would be able to classify beaches with different geometries (all of the provided beach surveys had very similar geometries with ocean to the west and land to the east). Figure 4 shows that a random forest trained on data that did not include augmentation depends greatly on the x-dimension (east-west), and when tested on a flipped beach simulating an east coast beach, fails to classify more than 25% of the points correctly. However, when a random forest with the same hyperparameters is trained on data including the data augmentations, the x-dimension feature importance drops and the model accuracy on an east coast beach increases to around 80%. These results indicate the importance of using data augmentation to train models robust to different beach geometries.

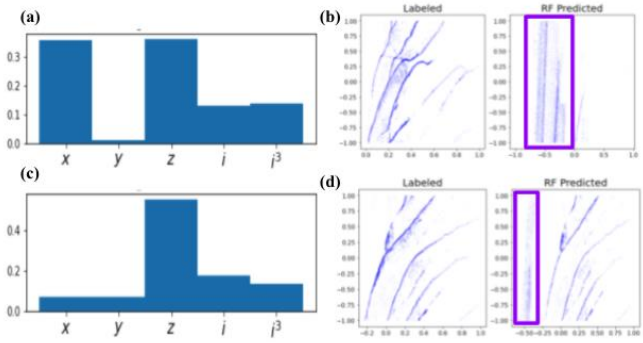

Figure 4*:* (a) Feature importance from simple RF model with one training beach and no data augmentation (b) Model failure to identify waves on 'East Coast' beach with ~25% accuracy (c) Feature importance of model with data augmentation (d) results of data augmentation model that identified waves with ~80% accuracy. Purple box marks land points misclassified as waves by model.

We tested our model algorithms with different normalization methods (not reported in Table 1) and found that the KNN and RF algorithms performed much better when the intensity was not scaled to fall between 0 and 1 but was left in raw form (with values in the tens of thousands). We hypothesize that the scaling squeezes the intensity values into a small range making differences between the land and wave points negligible and thus harder to distinguish effectively in these algorithms. In contrast, the NN algorithm completely fails if the intensity values are left in raw form and needs some sort of rescaling. This is likely due to the use of the softmax activation in the final classification output layer: with such high values of intensity, the softmax activation gives nearly equal probabilities of classification, so the model accuracy does not deviate from near 50%.

VI. Conclusion and Future Work

Our results indicate that with min/max scaling of features, either the KNN or RF models classify the LiDAR survey points most accurately between land and waves. The spatial nature of the data and separation between wave and land LiDAR returns lends itself to these models over the NN classifier. We do, however, recommend further comparisons of normalization methods that may boost NN model accuracy to similar levels. Feature engineering based on point geometry boosts model accuracy significantly regardless of the classifier and should be implemented.

As mentioned in section 3, we will also need to do model testing on full datasets that have not been subsampled to make sure that subsampling land returns for model training does not impact model accuracy. We can further explore the

hyperparameter space and more complex NN model architectures to find higher accuracy models and will use our test results to inform the CCCIA on which model architectures are likely to work best given their preferred workflow. If the CCCIA wishes to create a single model robust enough to classify points from a variety of beach surveys, we have shown that it is possible to train a model with limited input data using data augmentation techniques to simulate other beach geometries. However, if the CCCIA's goal is the most accurate point classification possible for each survey, we recommend that individual models be trained for each survey location using multiple surveys with a variety of wave conditions.

Table 1: Full summary of training data, model algorithms, testing, and result metrics.

Training Dataset(s)	Augmentation	Features	Testing	Test #	K-Nearest Neighbors (KNN)	Random Forest (RF)	Deep Neural Network (DNN)
torrey	NA	X, Y, Z, intensity, intensity^3	20% of dataset	**1**	**accuracy:** 95.343 **confusion matrix:** [95.0117 4.9882] [4.3399 95.6600]	**accuracy:** 98.674 **confusion matrix:** [98.6504 1.3495] [1.3037 98.6962]	**accuracy:** 96.933 **confusion matrix:** [97.3285 2.6714] [3.4447 96.5552]
			torreyRFID	**2**	**accuracy:** 69.722 **confusion matrix:** [83.9663 16.0336] [44.5573 55.4426]	**accuracy:** 59.59995 **confusion matrix:** [90.8451 9.1548] [71.7228 28.2771]	**accuracy:** 67.659 **confusion matrix:** [91.4483 8.5516] [56.1889 43.8110]
torrey	flipY, rot 90, rot -90	X, Y, Z, intensity, intensity^3	20% of dataset	**3**	**accuracy:** 92.717 **confusion matrix:** [92.7755 7.2244] [7.3398 92.6601]	**accuracy:** 96.345 **confusion matrix:** [96.6960 3.3039] [3.9922 96.0077]	**accuracy:** 91.924 **confusion matrix:** [92.2617 7.7382] [8.3994 91.6005]
			torreyRFID	**4**	**accuracy:** 57.974 **confusion matrix:** [58.6013 41.3986] [42.6557 57.3442]	**accuracy:** 54.682 **confusion matrix:** [68.8876 31.1123] [59.5598 40.4401]	**accuracy:** 69.233 **confusion matrix:** [56.6423 43.3576] [18.1432 81.8567]
torrey, LJshores, blacks, delmar	flipY, rot 90, rot -90	X, Y, Z, intensity, intensity^3	20% of dataset	**5**	**accuracy:** 90.639 **confusion matrix:** [87.8551 12.1448] [6.8342 93.1657]	**accuracy:** 94.320 **confusion matrix:** [94.3414 5.6585] [5.7000 94.2999]	**accuracy:** 89.822 **confusion matrix:** [88.1476 11.8523] [8.6581 91.3418]
			torreyRFID	**6**	**accuracy:** 61.893 **confusion matrix:** [67.7246 32.2753] [43.9530 56.0469]	**accuracy:** 55.920 **confusion matrix:** [80.1388 19.8611] [68.3595 31.6404]	**accuracy:** 55.149 **confusion matrix:** [72.4200 27.5799] [62.1639 37.8360]
torrey	NA	X, Y, Z, intensity, intensity^3, R^2, cosine similarity	20% of dataset	**7**	**accuracy:** 95.460 **confusion matrix:** [95.4649 4.5351] [4.5449 95.4550]	**accuracy:** 98.948 **confusion matrix:** [99.1388 0.8611] [1.2357 98.7642]	**accuracy:** 97.914 **confusion matrix:** [98.3813 1.6186] [2.5345 97.4654]
			torreyRFID	**8**	**accuracy:** 83.226 **confusion matrix:** [81.2988 18.7011] [14.8423 85.1576]	**accuracy:** 71.823 **confusion matrix:** [89.6737 10.3262] [46.0634 53.9365]	**accuracy:** 81.340 **confusion matrix:** [90.2292 9.7707] [27.5651 72.4348]
torrey	flipY, rot 90, rot -90	X, Y, Z, intensity, intensity^3, R^2, cosine similarity	20% of dataset	**9**	**accuracy:** 93.114 **confusion matrix:** [93.4639 6.5360] [7.2223 92.7776]	**accuracy:** 97.917 **confusion matrix:** [98.5331 1.4668] [2.6769 97.3230]	**accuracy:** 94.684 **confusion matrix:** [95.3036 4.6963] [5.9124 94.0875]
			torreyRFID	**10**	**accuracy:** 73.978 **confusion matrix:** [62.3234 37.6765] [14.3446 85.6553]	**accuracy:** 68.821 **confusion matrix:** [70.7017 29.2982] [33.0635 66.9364]	**accuracy:** 72.173 **confusion matrix:** [70.8835 29.1164] [26.5339 73.4661]
torrey, LJshores, blacks, delmar	flipY, rot 90, rot -90	X, Y, Z, intensity, intensity^3, R^2, cosine similarity	20% of dataset	**11**	**accuracy:** 90.828 **confusion matrix:** [88.8276 11.1723] [7.3621 92.6378]	**accuracy:** 96.167 **confusion matrix:** [96.7626 3.2373] [4.3717 95.6282]	**accuracy:** 93.661 **confusion matrix:** [94.7831 5.2168] [7.3537 92.6462]
			torreyRFID	**12**	**accuracy:** 78.176 **confusion matrix:** [70.8787 29.1212] [14.5124 85.4875]	**accuracy:** 68.122 **confusion matrix:** [75.7416 24.2583] [39.5117 60.4882]	**accuracy:** 64.132 **confusion matrix:** [68.8208 31.1791] [40.5650 59.4349]

VII. REFERENCES

[1] F. P. Medina and R. Paffenroth, "Machine Learning in LiDAR 3D point clouds," *arXiv:2101.09318 [cs]*, Jan. 2021, Accessed: Jun. 10, 2021. [Online]. Available: http://arxiv.org/abs/2101.09318

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *arXiv:1612.00593 [cs]*, Apr. 2017, Accessed: Jun. 10, 2021. [Online]. Available: http://arxiv.org/abs/1612.00593

[3] X. Li, L. Wang, M. Wang, C. Wen, and Y. Fang, "DANCE-NET: Density-aware convolution networks with context encoding for airborne LiDAR point cloud classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 128–139, Aug. 2020, doi: 10.1016/j.isprsjprs.2020.05.023.

[4] S. M. Krishna Moorthy, K. Calders, M. B. Vicari, and H. Verbeeck, "Improved Supervised Learning-Based Approach for Leaf and Wood Classification From LiDAR Point Clouds of Forests," *IEEE Trans. Geosci. Remote Sensing*, vol. 58, no. 5, pp. 3057–3070, May 2020, doi: 10.1109/TGRS.2019.2947198.

[5] H. Matsumoto and A. Young, "Automated Cobble Mapping of a Mixed Sand-Cobble Beach Using a Mobile LiDAR System," *Remote Sensing*, vol. 10, no. 8, p. 1253, Aug. 2018, doi: 10.3390/rs10081253.

[6] S. A. Glantz and B. K. Slinker, *Primer of applied regression and analysis of variance*. New York: McGraw-Hill, Health Professions Division, 1990.

[7] Sciencedirect.com. 2021. *Cosine Similarity - an overview | ScienceDirect Topics*. [online] Available at: <https://www.sciencedirect.com/topics/computer-science/cosine-similarity> [Accessed 10 June 2021].

[8] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 285–2830, 2011.

[9] F. Chollet and others, *Keras*. 2015. [Online]. Available: https://keras.io

[10] M. Abadi *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. [Online]. Available: https://www.tensorflow.org/

[11] . Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)," *arXiv:2004.04523 [cs, stat]*, Apr. 2020, Accessed: Jun. 10, 2021. [Online]. Available: http://arxiv.org/abs/2004.04523

[12] E. M. Kleinberg, "Stochastic discrimination," *Ann Math Artif Intell*, vol. 1, no. 1–4, pp. 207–239, Sep. 1990, doi: 10.1007/BF01531079.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016.

[14] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," *Machine Learning Mastery*, Jul. 02, 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (accessed Jun. 10, 2021).

VIII. Acknowledgments

We could not have done this project without the support, data, and direction of UCSD's CCCIA, especially Hiro Matsumoto, Rob Grenzeback, Adam Young, and Mark Merrifield. We would also like to thank the TAs for the ECE 228 course, especially Billy Jenkins and Brian Whiteaker, for their guidance and constant help. Thank you to our group reviewers for providing positive and constructive feedback.

IX. Contributions

A. *Austin Barnes*

I had initial discussions with Hiro Matsumoto and Rob Grenzeback from the CCCIA to decide on the project and obtain the beach surveys used for analysis, did initial data visualizations and converted the files for use in python with our chosen libraries. I worked primarily on the neural network classifier, determining hyperparameters and performing testing with different normalizations. We each contributed equally to the deliverables for this project.

B. *Raymond Young*

In pre-processing I contributed to data-augmentations on each of the normalized segments of beach. This included options to flip across each axis and rotate at arbitrary angles. For the models I worked primarily on the RF classifier producing trained models and testing. Operationally I also focused on creating a modular workflow to pre-process, train, and test our models by implementing the code written by Hannah and Austin and standardizing the inputs/outputs as well as saving relevant data and configuration information. I also helped manage our version control through Github. We each contributed equally to the deliverables for this project.

C. *Hannah Walker*

For this project I helped work on a literature review to help us determine the frameworks of methods to apply to solve our problem. I worked on the KNN classifier model training and testing by implementing and adding to code written by Ray. I also researched the application of feature engineering to our data and developed the cosine similarity feature calculation code that was later combined with the R^2 feature idea by Ray. We each contributed equally to the deliverables for this project.

X. Reply to Review

A. *Group 9*

"Summary:
In this presentation, you explain the data augmentation, preprocessing, and model clearly. Using flipping, rotating to avoid geometry dependency and to extend the dataset. The project seems great overall."
Thank you!

"Some improvements/unclear"
"Feature engineering seems to be interesting. But, is there a plot that shows the feature importance of R^2 and the cosine similarity?"
We included a feature importance plot that includes the two added feature engineering features for the random forest classifier.

"What's the reason that you decided not to do the normalization of intensity for RF and KNN models?"
This is addressed in our paper.

"Is the normalization of Z done only one segment at a time or the entire dataset? Will this make a big difference in the performance of your model?"
The z-normalization is done one segment at a time. Comparing normalization by segment versus the entire dataset would be interesting to test in future work. We were not able to proceed with this due to computational limitations but will suggest this to the CCCIA.

B. *Group 15*

"Strengths of the Project:

- The project is novel and their statement is very clear.
- Overall the project is interesting and they explain in detail about some materials including data Augmentation and their future work.
- It was really helpful that talked about their model parameters, for example, using 15 neighbors for KNN classifiers.
- It is great to test their model not only on their test data but also on the dataset.

- Great presentation from all of them."

Thanks!

"Things to improve:"

- "Based on your argument, you did not use normalization for RF and KNN. is this correct? If it is, you could have shown their performance before and after normalization. Should the models perform much better using normalization?"
    - No, we did use normalization for RF and KNN. Adding this could be useful to see how normalization impacts the performance of the model.
- "How intensity^3 can help to extract nonlinear features in your dataset? It would be clear if you explain similar to what you did for Cosine Similarity and R^2 values."
    - By cubing the value of intensity, this greatly increases values over than one while making little difference for values near one (following the trends of a cubic function). This allows for a more distinct separation between high and low intensity returns. This is a useful suggestion and when communicating this work in the future, a plot should be shown to make this clearer.
- "Could have used more models and not to rely only on the recommendations based on Medina at al. 2021."
    - This is true. We added feature engineering which was not part of Medina et al. in efforts to branch out from this paper but in the end we did very similar work to the paper due to its applicability to our problem. In future work we will make efforts to use a wider variety of models from multiple sources.
- "It would be really helpful to see performance of training all the data with feature engineering and without feature engineering."
    - Yes, that is a great idea and we added this to our results.

C. Group 24

"Team 24 explains three models, Random Forest, K-Nearest Neighbor, and a supervised deep learning model to classify LiDAR point returns off of the waves surfaces. They use 5 dataset and applied data augmentation to reduce geometry dependency. Also, they split data into subsamples to reduce dependency on coastline geometry. Then, different test data are applied to examine the accuracy of various models and the effectiveness of data augmentation. Then, they conclude that the Random Forest model has the best performance."

Strengths:

- The datasets, data preprocessing, and models are described thoroughly in the presentation.
- The data augmentation step and data splitting are great ideas to extend the dataset and improve training model
- The code demonstration is clear and easy to understand.
- The accuracy comparisons of each model using different combination of training and validation data are intuitive."

Thank you!

"Some Improvement/unclear:"

- "Why data is split into 100m alongshore segments? Will the split size affect the result? Why there is no normalization for intensity value for Random Forest(RF) and K-Nearest Neighbor(KNN)? Can you applied the same normalization for deep learning model to RF and KNN? Why or Why not?"
    - Two reasons, first splitting the beach could allow for portions of the beach to be used instead of an entire beach which initially helped with memory/computation issues. Second, by applying normalizations/augmentations to each split we were able to further expand our training set.

- "Any consideration for a more complex deep learning model?"
    - Great point. In future work we would suggest further exploring deeper/more complex deep learning models but for the purpose of this project we referenced successful previous work [1] in order to define our architecture because we were more interested in comparing and contrasting different types of algorithms instead of optimizing a single model for this project.

- "Model Description below futures will be great."
    - It is possible that we don't completely understand this question, but we have included additional descriptions of each model including a high-level explanation to each algorithm in our report.

- "Make Confusion matrix as percentage might give a clearer understanding of how each model works"
    - Great note. We took this into consideration by normalizing each row but the number of true labels and presented results this way in our report.