Automatic Detection of Internal Ocean Waves in XBand Radar Backscatter Using a Convolutional Neural Network

Wyatt Cottriel

December 9, 2023

1 Introduction

Large anomaly structures in the ocean have always been of high interest to physical oceanographers, as understanding their sources, physical manifestations, and consequences can help us better understand ocean systems as a whole. Recently, a structure of particular interest which has been observed in many near shore data acquisition missions, has been internal waves. These waves are large scale disturbances which propagate at speeds on the scales of single kilometers per hour, orders of magnitudes slower than the swell most would think of when considering ocean waves. These structures are also far more massive than smaller swells in the ocean, spanning 10's-100's of kilometers wide, having amplitudes from 10's-100's of meters deep, and are observed to have periodicities from a couple hours to days.

Due to breadth and depth of these features, internal waves in the ocean can play a big roll in the dispersion of sediments and nutrients in the water column when displacing the thermocline by such large amounts. Further, in some specifically intense events, internal waves can create such large pressure changes that they can damage equipment placed in the water column as well as ocean vessels such as submarines which may also be deeper in the water column as these waves propagate through. Even though their frequency when compared with many other ocean processes in nearshore environments may seem small, a disturbance as large as these can clearly have some large impacts on the systems they propagate through. Accordingly there is great value in being able to detect and observe them.

Some of these structures have been well documented and studied, often being attributed to simple signals like tidal modulations and their interactions with bathymetric features, whereas the dynamics of others have proven to be more elusive to oceanographers. Consequently, many labs such as the Coastal Observing Research and Development Center (CORDC) at the Scripps Institute of Oceanography have sought to collect more data on these phenomena through new and novel ways. One approach taken by CORDC, which is explored in this report, has been to use radar backscatter to attempt to recognize the pattern these features leave on the surface of the water, hopefully providing a better spatial and temporal understanding of these features. This approach has proven to be successful, however, the acquisition of this data has one fatal flaw.

Radar scans produce massive files, and when a scan can be taken almost once every second, sifting through them to search for these waves is tedious to say the least. The process of picking these waves out in massive data sets has served as a large bottleneck on research, therefore it has become of specifically high interest to attempt to develop some kind of machine learning algorithm to automate this task.

The initial task at hand is as follows: write an algorithm that can quickly and effectively sift through radar data to flag what data does and does not include wave observations. Previously an algorithm was written using support vector machines to attempt to detect waves [CMP+21]]. In this report, however, a convolutional neural network (CNN) was written and trained to complete this task as a proof of concept. To do so, processed and normalized radar data was compressed into PNGs which were then categorized by hand and fed into a CNN. This CNN has been modeled somewhat simply with very few layers after previous experimentation demonstrated a tendency for the model to over-fit. The algorithm was tasked to decide either: yes there is a wave in this scan or no there is not. Training a model such as this can be very computationally expensive and can take a lot of tuning to try to perfect it for results that can be extrapolated, therefore the purpose of this report was not to configure and train a be all end all model, but rather show with a small subset of data that the use of such an algorithm for this task is feasible, motivating wider application.

In this report, promising results were achieved. Three days worth of clean data (2146 photos) were used to successfully train the model to $\tilde{9}8\%$ accuracy for detection. Application of these results is likely narrow as the data was very clean and uniform, and more data will certainly be necessary to push the model to wider application, however this preliminary model serves as a good benchmark of possibility.

2 Data

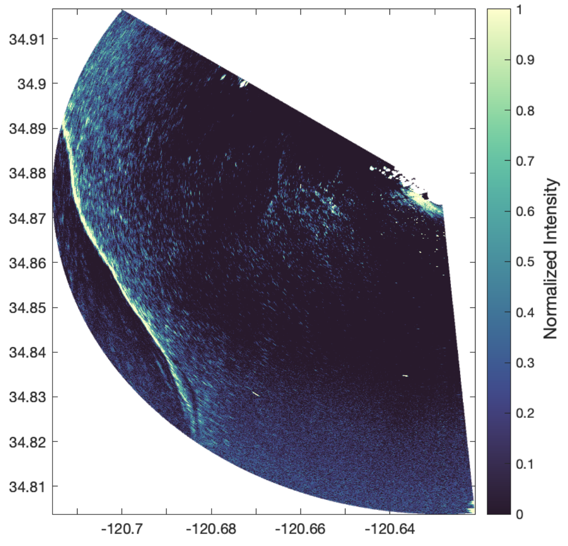


Figure 1: Example XBand backscatter plot with internal wave.

The dataset being used in this project was collected during a deployment of an XBand radar during September and October of 2017 at Point Sal near the Vandenberg Air Force Base as part of the Inner Shelf Department Research Initiative (DRI) (McSweeney et al. 2020b). The radar used was a noncoherent marine X-band Furuno radar system which operates in the microwave range, with a periodicity of 1.5s, an angular resolution of 1/10 of a degree, and a range of 7km (Celona et al. 2021).

To measure the ocean surface, the XBand radar radially disperses signals in the microwave range which can then backscatter off of steep small scale surface features on the ocean (primarily capillary waves) to then be re-received by the radar, creating a map of 'ocean roughness', an example of such data is displayed in figure 1. When the signal from the XBand is able to scatter back towards the radar at shorter angles (i.e. when the capillary waves are steeper), the intensity from that area will have a larger signature. Larger features on the ocean surface, like the mean displacement of water from an internal wave, on their own would not create an intensity pattern on the XBand radar as the angle they form with respect to the radar is too shallow. Features like internal waves however, are known to impact the amplitudes of capillary waves on the surface directly above the disturbances. When the thermocline experiences downwelling, the surface of the water is known to become rougher which would lead to a region of higher intensity on the radar. When the thermocline experiences upwelling, the surface of the water is known to become smoother which would lead to a region of lower intensity on the radar. When a disturbance like an internal wave propagates through the ocean, regardless of whether the displacement of the thermocline is positive or negative, one edge of the disturbance will experience downwelling (creating a high intensity pattern) and the other will experience upwelling (creating a low intensity pattern), again this can clearly be visualized in the intensity band in figure 1 indicating an internal wave.

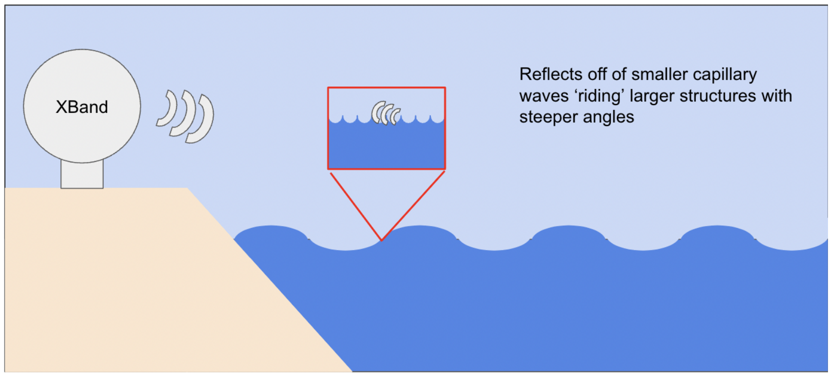

XBand
Reflects off of smaller capillary waves 'riding' larger structures with steeper angles

Figure 2: Diagram showing how XBand works.

Once the radar has collected all of its data, the data needs to be preprocessed such that it can be compressed and presented in such a way that is conducive to efficient and meaningful interpretation. For this data set, the backscatter pattern was first pushed through a denoising filters by the Marine Physical Laboratory at the Scripps Institute of Oceanography, and then the backscatter was normalized to contain only values from 0 to 1 (1 being the highest intensity and 0 being the lowest). It is also common practice to take subsequent backscatter data from the XBand and average it over 1-5 minute intervals in order to 'smooth' the data as well as decreasing the sheer volume of data (in this instance it was averaged over 1 minute intervals). At this point, the data was ready to be preprocessed for use in the CNN. In its state as raw data files, the three days worth of radar data amounted to 70 GB of data, this was much too large to efficiently read into any neural network to train it. To compress the data, a MATLAB script was utilized to take the backscatter and transform it onto a colormap (cmocean deep) and save it as a PNG. This process cut the data size down about 50x from 70 GB to 1.5 GB but was also the most expensive part of the process in terms of computational time. The PNG plots generated were stripped of axis labels, a colorbar, or axis tick marks as to retain only the information that was absolutely necessary to train the CNN. Further, the data was organized into two separate files, one labeled to contain radar scans which had visible internal waves and one which had no visible internal waves. The target labels for the data were decided by hand, so any scans which presented visual information which was too ambiguous to tell whether there was a wave or not were omitted, example images an labels from the data set can be seen in figure 3.

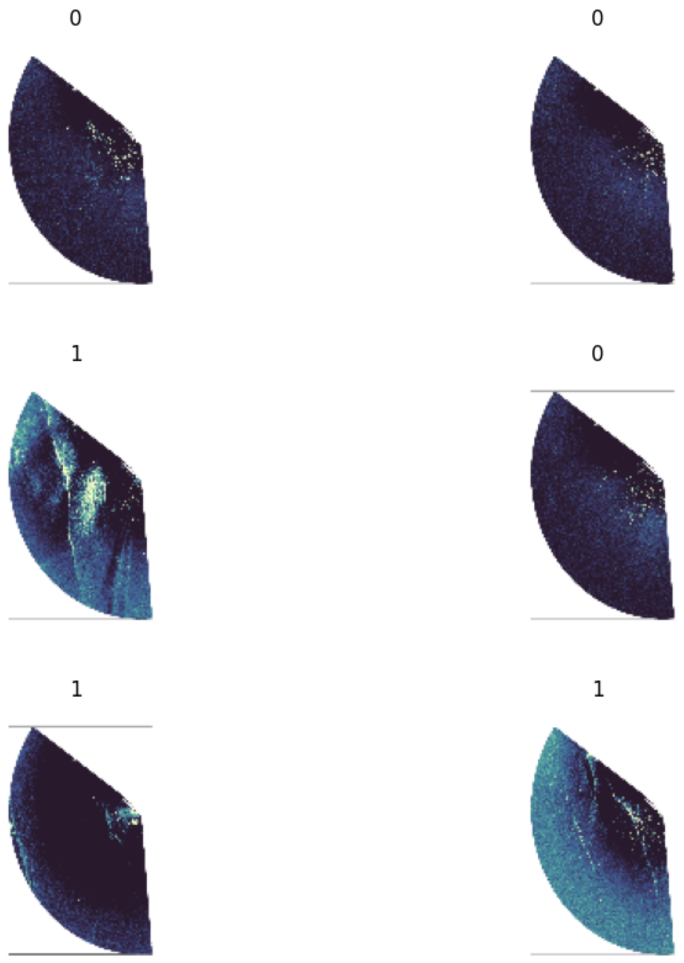


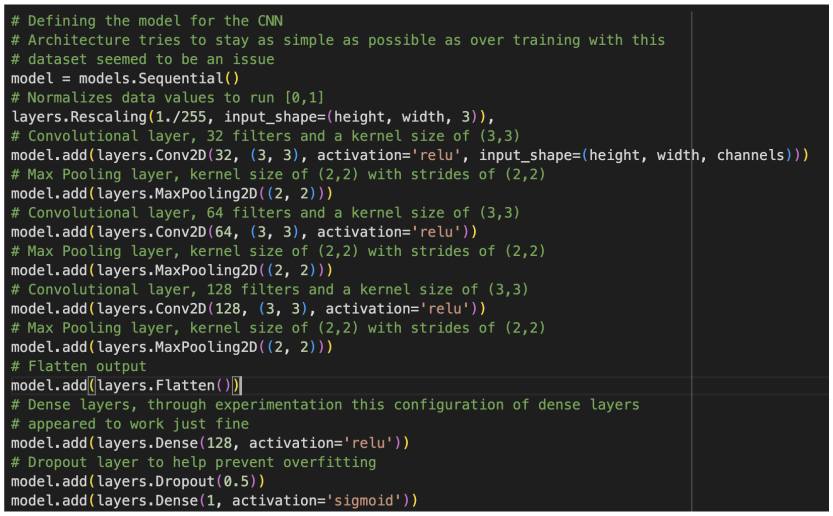
Figure 3: Example compressed images from XBand backscatter to be input into model. The 1's and 0's represent whether the data was labelled to have or not to have a wave respectively.

3 Modeling

To complete this binary image classification task, a convoluted neural network was elected as the algorithm of choice. The decision to use a convoluted neural network was made due to the complexity of the task at hand. The identification of waves in radar scans is a somewhat vague task to try to accomplish through another algorithm where the data may need to display a clear pattern everytime, or be transformed into another coherent vector space where just the wave is isolated. Often, the wave would slightly change shapes and breadth, the interference pattern of the whole image would get offset, certain regions could randomly light up (likely from gusts of wind), or clouds could interfere with parts of the scan. Neural networks excel in tasks with issues like these and thus it seemed the most logical choice for the job.

The exact architecture of the model used here was initially organized with heavy inspiration from the VGG-16 architecture [SZ15] as this algorithm has proved to be highly effective at image classification and is not as deep as some more recent algorithms, which could decrease computational complexity for this task.

However, in the initial stages of training the model showed a clear tendency to quickly over fit the data and perform poorly when tested. Due to this, the model was stripped of most layers and left with the following architecture: a rescaling layer (to normalize rgb values to go from 0 to 1), 1 convolutional layer, 1 max pooling layer, 1 convolutional layer, 1 max pooling layer, 1 convolutional layer, a final max pooling layer, and then 2 dense layers with a dropout in between (to prevent over fitting). Each layer also utilized a ReLU activation function. Highly simple, but effective. The python script written to configure the model has been included below in figure 4.



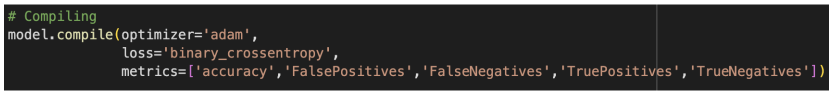
```
# Defining the model for the CNN
# Architecture tries to stay as simple as possible as over training with this
# dataset seemed to be an issue
model = models.Sequential()
# Normalizes data values to run [0,1]
layers.Rescaling(1./255, input_shape=(height, width, 3)),
# Convolutional layer, 32 filters and a kernel size of (3,3)
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(height, width, channels)))
# Max Pooling layer, kernel size of (2,2) with strides of (2,2)
model.add(layers.MaxPooling2D((2, 2)))
# Convolutional layer, 64 filters and a kernel size of (3,3)
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
# Max Pooling layer, kernel size of (2,2) with strides of (2,2)
model.add(layers.MaxPooling2D((2, 2)))
# Convolutional layer, 128 filters and a kernel size of (3,3)
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# Max Pooling layer, kernel size of (2,2) with strides of (2,2)
model.add(layers.MaxPooling2D((2, 2)))
# Flatten output
model.add(layers.Flatten())
# Dense layers, through experimentation this configuration of dense layers
# appeared to work just fine
model.add(layers.Dense(128, activation='relu'))
# Dropout layer to help prevent overfitting
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

Figure 4: Python Script Building Model Architecture.

This model accepted data with the shape (batch size, image height, image width, channels). In this particular case, the batch size used (found through some experimentation) was 32, dimensions of the images were 150 x 150, and there were three channels for the rgb information.

The model was compiled with the script in figure 5.

The Adam optimizer was utilized as it has proven effective at image classification and recognition tasks with a simple stochastic gradient descent algorithm (since the task



```
# Compiling
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy','FalsePositives','FalseNegatives','TruePositives','TrueNegatives'])
```

Figure 5: Compiling Script

at hand was just binary classification, it felt logical to try to keep things as simple as possible as to not develop an overly complex model or risk over training). Additionally, this model opts to utilize the binary cross entropy loss function.

Furthermore, the model was also configured to produce values for accuracy, false/ true positive/ negative predictions, as well as the loss function for both the training and testing data at each epoch to better understand how the model was performing at each iteration.

Once the model was configured and compiled, it was time to train it by passing the training and testing data sets to the model. With some experimentation, 7 epochs were used to train the model as its accuracy, loss, and false/true negative/ positive detections would all plateau past that, opening the door to over training the model and losing applicability of results. The metrics of this training process are in figure 6.

4 Results

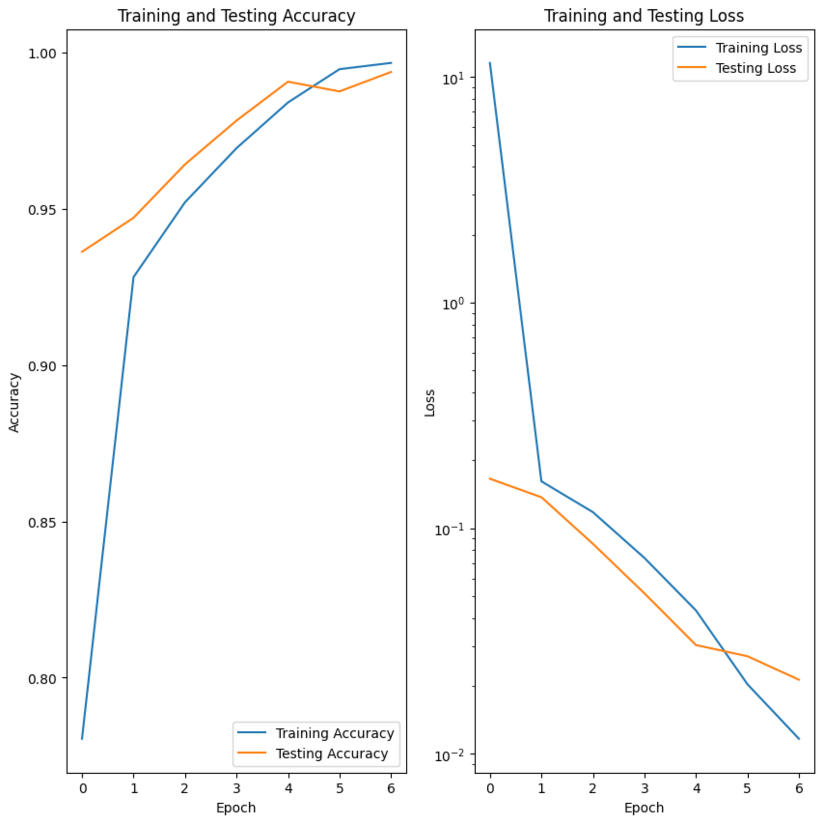


Figure 6: Figures showing the accuracy and loss metrics of the model as it trained.

The performance of the model can be seen in figure 6. The model was trained with a subset of 70% of the overall data randomly shuffled and then applied to test data (30% of the overall data) to assess the performance. Taking a look at figure 6, one can clearly see that the model very quickly started behaving with high accuracy. After just the second and third epochs both the training and testing data had accuracy scores over 90% and the loss was very quickly minimized for both. As a side note, one may notice that testing accuracy was higher than the training accuracy and the testing loss was lower than the training loss at least initially. It is important to remember that the evaluation of the testing data occurs at the end of the epoch while the evaluation of the training data is occurring during that epoch, thus the model is more 'trained' while evaluating the test data and this isn't actually so strange.

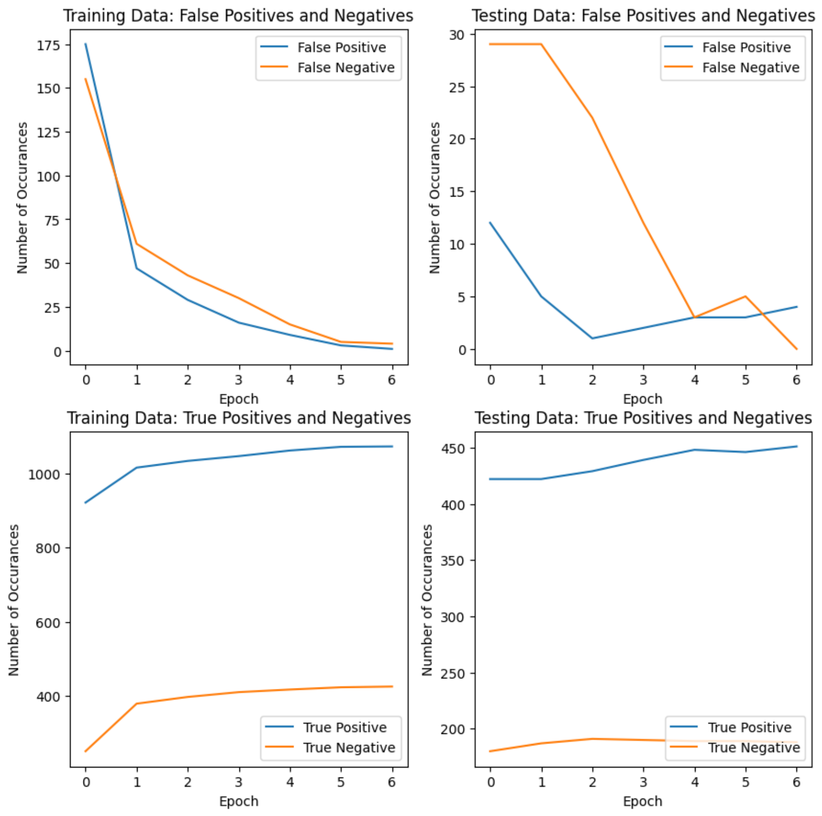
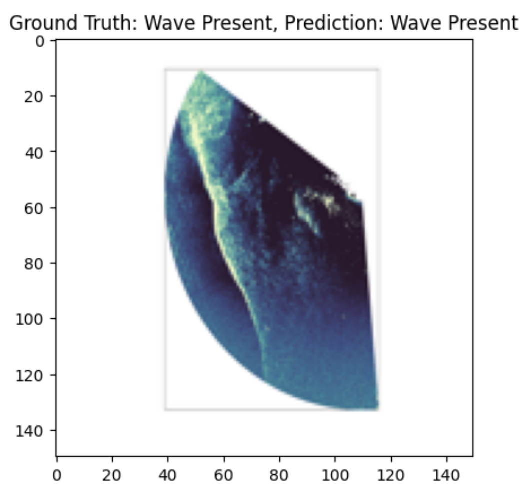


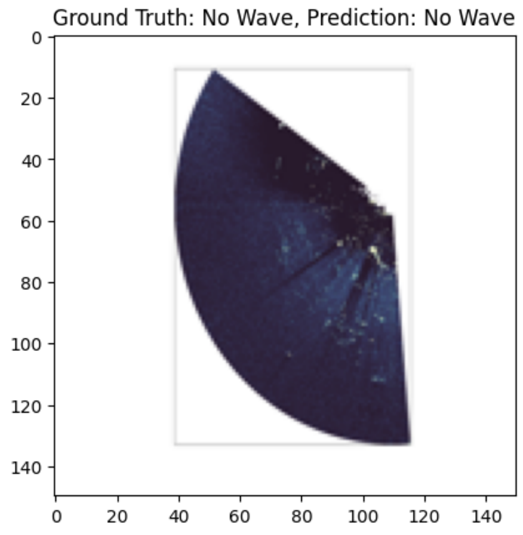
Figure 7: Figures showing the true/false negative/positive rates of the model as it trained.

To see more clearly what is going on we can take a look at figure 7 and see that the rates of both false positives and negatives in the testing and training data greatly decreased by just the 2nd and 3rd epoch for both the testing and training data and began to level out (indicating that training has achieved a kind of maturity). We can also see the inverse trend in the true positive and negative rates for both the training and testing data, beginning to level out after increasing from the 1st to 3rd epochs.

Below in figure 8 we can also see example model outputs given the input compressed images. We can see here that for both of these instances the model correctly identified whether or not a wave was present in the input.



(a) Example model prediction for radar scan with wave present.



(b) Example model prediction for radar scan with no wave present

Figure 8

5 Discussion

From the above results in figures 6-8, we can clearly see that given the provided training data set, the model was successful in creating its own regime for establishing whether inputs included a wave or not. Within 7 epochs the model was able to reach well over 95% accuracy for this data. Taking a look at our performance metrics as well, we can see trends that should be anticipated. Generally there is a slightly higher rate of false negatives than there is false positives, and this would make sense as I did include some inputs that included extremely faint waves which the model likely struggled time to time with, but impressive performance none-the-less! Taking a look at our loss and accuracy metrics we can again see that they steadily decreased and increased respectively until they plateaued, indicating improving performance as the model was training and an eventually 'maturity'.

To push the limit of the model and see how it would handle some more difficult data, some data was input that was initially omitted due to atmospheric noise to see how the model might classify it (figure 9). Labelling by hand, the best human guess classified this scan as not including a wave, however, the model classified it as including a wave. While there is some ambiguity as to whether this image may contain a wave or not, it seems clear that the model is more than likely classifying one of the clouds in the image, like in the upper left, as a wave. This indicates that while we had good overall performance on the training and testing data, more diverse data, potentially including data that is well labelled with weather, is absolutely needed for the model to be further extrapolated.

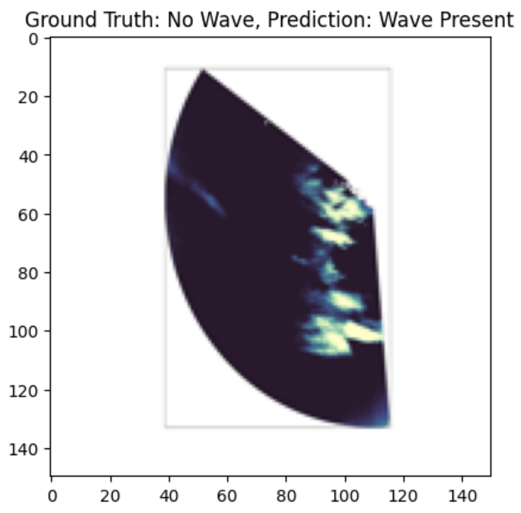


Figure 9: Figure showing output of the model for an input radar scan with atmospheric noise.

6 Conclusion

After designing and training the model, a few takeaways from this little 'case study' become clear. First, it seems as though the use of convolutional neural networks for wave detection in radar imagery is likely an option worth exploring. Within this limited data set of just three days, a model was able to be achieved which was capable of classifying images, albeit the results being very narrowly applicable.

Second, it became clear early on that more complex models were actually struggling more with the data set by very quickly over fitting (i.e. the initial exploration with a model similar to VGG-16) and simplicity came out on top in terms of performance. It would certainly be interesting to see if this result stays strong with a larger data set of radar imagery representing an even wider range of scenarios and coastline geometries from different deployments.

Finally, it should be noted that despite the promising metrics on just this test data set, there is absolutely a need for more data to train this model to actually be able to practically use it. This can be clearly seen in its immediate failure in handling a data input with any amount of noise(figure 9), such as from the atmosphere.

From here, I think it would be very valuable to continue to explore the use of CNNs for wave detection in radar imagery, and the first steps that should be taken in doing this should be a) collecting and processing more data and b) exploring different model architectures as one introduces more diverse data sets. In all, these results are certainly exciting and could potentially greatly accelerate the pace of research with instruments such as the XBand radar.

References

[CMP$^+$21] Sean Celona, Sophia T Merrifield, Tony De Paolo, Nate Kaslan, Tom Cook, Eric J. Terrill, and John A. Colosi. Automated Detection, Classification, and Tracking of Internal Wave Signatures Using X-Band Radar in the Inner Shelf. *Journal of Atmospheric and Oceanic Technology*, 38(4):789–803, 2021.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.