Convoluted Neural Networks and Iceberg Detection

Sania Bandekar and Victoria Le
Department of Computer Science, San Jose State University, California

I. Introduction

DATA regarding icebergs plays a crucial role in monitoring climate change and understanding its effects on sea levels and ecosystems. Using synthetic aperture radar (SAR) imaging combined with advanced despeckling techniques, we developed a robust methodology to analyze icebergs. This enables predictions of the effects of climate change and provides actionable insights into mitigating potential ecological and environmental impacts.

II. Methodology

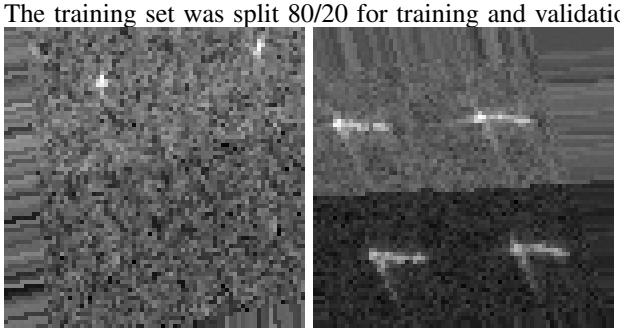
A. Research and Model Selection

Recent advancements in deep learning, particularly convolutional neural networks (CNNs), have shown great accuracy in processing satellite images. Our approach uses these techniques to achieve high precision in detecting icebergs from other things, such as ships. These methods were selected due to their effectiveness in handling large-scale datasets with noise and their ability to segment and classify images [1].

B. Data Preparation

The dataset for this project was obtained from Kaggle because of the limited availability of suitable SAR imaging datasets. The images were preprocessed to enhance important features for iceberg detection. The dataset consists of JSON files containing image data (band_1) and corresponding labels (is_iceberg). Each image is a 75x75 pixel grayscale image, and the label indicates whether the image contains an iceberg or not. The images were reshaped and normalized. The JSON files were converted to NumPy arrays to serve as input for training and validation [4]. We normalized the images to scale the pixel values between 0 and 1. This step helps the model learn more efficiently and speeds up convergence during training.

The training set was split 80/20 for training and validation.



On the left is a photo of icebergs and on the right is one of ships. These were both obtained from the JSON file.

A smaller set of the photos were used for testing after all the modifications had been made.

C. Convolutional Neural Network Architecture

- **Convolutional Layers:** The implemented CNN architecture consists of convolutional layers with batch normalization and ReLU activation. In our first model's convolutional layers, we started with 32 filters, increasing to 128 filters in subsequent layers, using 'same' padding to ensure the output has the same spatial dimensions as the input. These layers extract high-level features from the images. Experimentation with additional layers is explored in later sections.
- **Max Pooling:** There is max pooling with a 2x2 window to reduce spatial dimensions, while retaining the most critical information.
- **Model Regularization:** To prevent overfitting, we used dropout with a rate of 30 percent. Additionally, we applied L2 regularization to the convolutional and dense layers to penalize large weights and improve generalization.
- **Dense Layers:** We used a dense layer with 128 neurons, followed by a second dense layer with a single output unit. The final layer uses a sigmoid activation function, which is typically used for binary classification tasks like iceberg detection. The output will be a probability between 0 and 1, representing the model's confidence in the prediction.

D. Model Compilation

The binary classification task (distinguishing between icebergs and ships) was optimized using binary cross-entropy loss and the Adam optimizer.

E. Model Training and Evaluation

The model was trained on the preprocessed dataset, with a validation split to assess its performance during training. The final evaluation for the original model showed relatively high accuracy and numbers for loss that could be improved on. For testing accuracy, the numbers were not great, being correct only about 58 percent of the time, but with the modifications, shown later on, the model was able to improve. The training plots, as shown below, display the accuracy and reduction in loss over epochs.

Avg Training Loss: 0.3921
Avg Validation Loss: 0.7771
Avg Training Accuracy: 0.8406
Avg Validation Accuracy: 0.7026
Test Accuracy: 0.5819

The model was trained for 50 epochs with a batch size of 32. After every 32 samples, the model updates its weights.

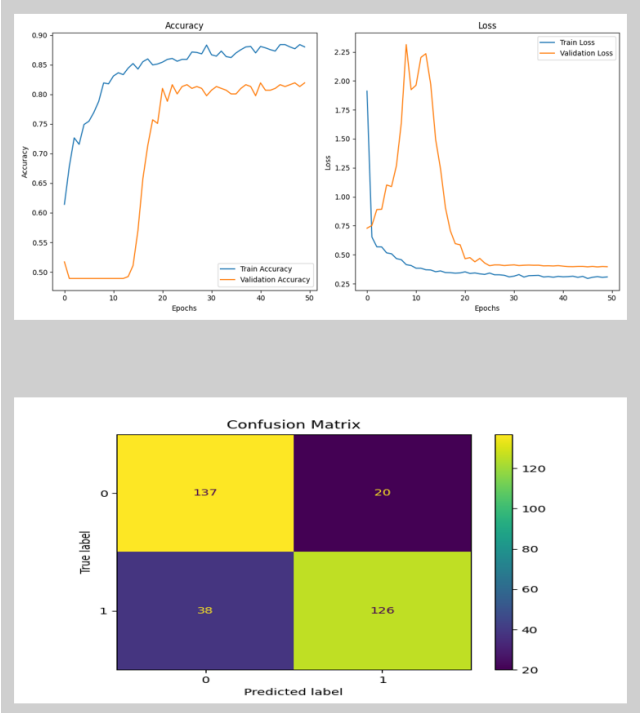


Fig. 1. Original Model Accuracy and Loss

Validation data was used to monitor the model's performance, detect overfitting, and adjust the learning rate when necessary.

To improve training stability, we used the ReduceLROnPlateau callback to reduce the learning rate by 50% if the validation loss stopped improving for three consecutive epochs.

F. User Interface Development

A user-friendly desktop application was developed using Python, allowing users to run the model and produce charts for Model Accuracy and Model Loss over epochs. There is also a page where images can be uploaded and the trained model will predict if it is a ship or not.

G. Improving the Model

To improve the model, we decided to adjust the model in certain areas and observe the effects they had on the accuracy and loss results [2] [3]. We experimented in the following areas:

- **Adjusting Number of Convolutional Layers:** Adding more convolutional layers can help the model learn more complex and abstract features, however, we made note that too complex a model could result in overfitting.
- **Adjusting Number of Epochs:** Adding more epochs can also help the model learn the patterns in the data, but like convolutional layers, adding more does not necessarily result in a better model as overfitting could occur.
- **Data Augmentation:** Data augmentation could add more variety to the dataset the model is being trained on, which could potentially improve it.
- **Adding Global Average Pooling:** Adding this layer helps reduce overfitting by reducing the spatial dimensions of the convolutional layers.

H. Challenges

Due to the limited amount of datasets for iceberg detection and the time constraints, we were unable to successfully find and test a second dataset.

In addition, we ran into many complications while executing the main feature of the application. For a brief period, when asking the model to predict an image, the result was always "Ship" even if the image we uploaded was an iceberg. When working on this issue, the problem seemed to be more on the code for the application itself, rather than the model. The code for the application was simplified to ensure the trained model was used to predict the images and results were more accurate.

III. Results and Discussion

The following charts show the accuracy and loss results after we modified the model.

A. Increased Convoluted Layers

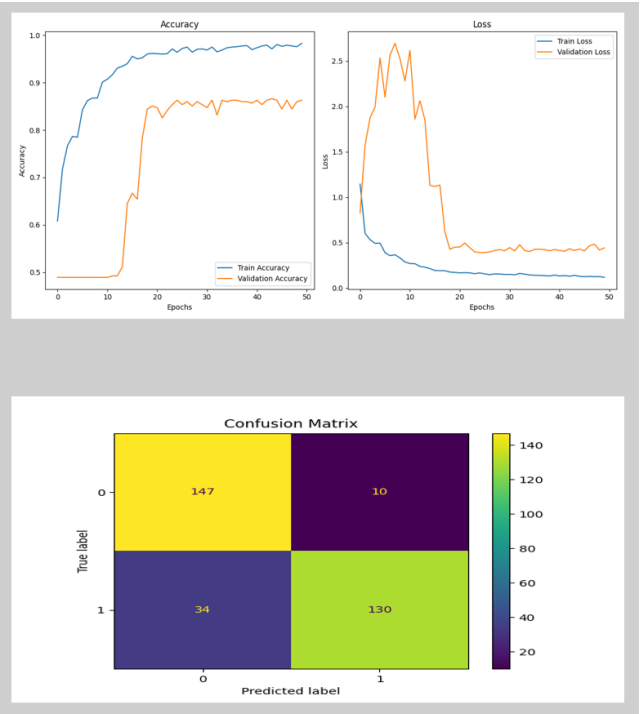


Fig. 2. Increased Convolutional Layers Accuracy and Loss

Avg Training Loss: 0.2289
Avg Validation Loss: 0.9401
Avg Training Accuracy: 0.9317
Avg Validation Accuracy: 0.7395
Test Accuracy: 0.7173

Increasing the number of layers by one seemed to result in slightly higher accuracy and loss. However, test accuracy has decreased.

B. Increasing Epochs to 75

Avg Training Loss: 0.5055
Avg Validation Loss: 0.7535

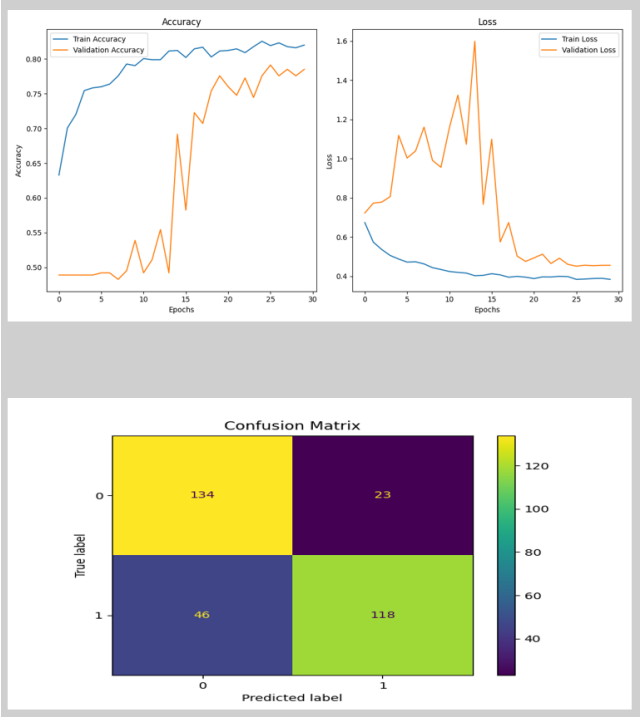


Fig. 3. Increasing Epochs to 75 Accuracy and Loss

Avg Training Accuracy: 0.7885
Avg Validation Accuracy: 0.6544
Test Accuracy: 0.6097

Increasing to 75 epochs resulted in slightly higher numbers regarding accuracy and loss, and better accuracy for testing.

C. Data Augmentation

Avg Training Loss: 0.5674
Avg Validation Loss: 0.6442
Avg Training Accuracy: 0.7206
Avg Validation Accuracy: 0.6371
Test Accuracy: 0.1681

Data augmentation did not improve the model.

D. Global Average Pooling

Avg Training Loss: 0.4033
Avg Validation Loss: 0.5841
Avg Training Accuracy: 0.8122
Avg Validation Accuracy: 0.7222
Test Accuracy: 0.3053

Global average pooling also did not improve the model.

E. Adjusting the Model Hyperparameters

Batch size was reduced from 32 to 16, depths were increased to 64, 128, 256, and dropout reduced to 0.2 from 0.3.

Avg Training Loss: 0.3289
Avg Validation Loss: 0.6752
Avg Training Accuracy: 0.8695
Avg Validation Accuracy: 0.7469
Test Accuracy: 0.6436

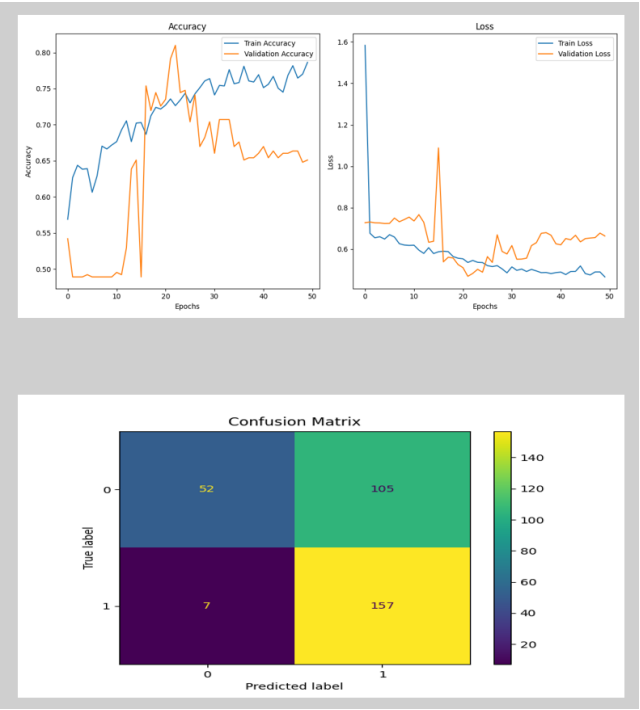


Fig. 4. Data Augmentation Accuracy and Loss

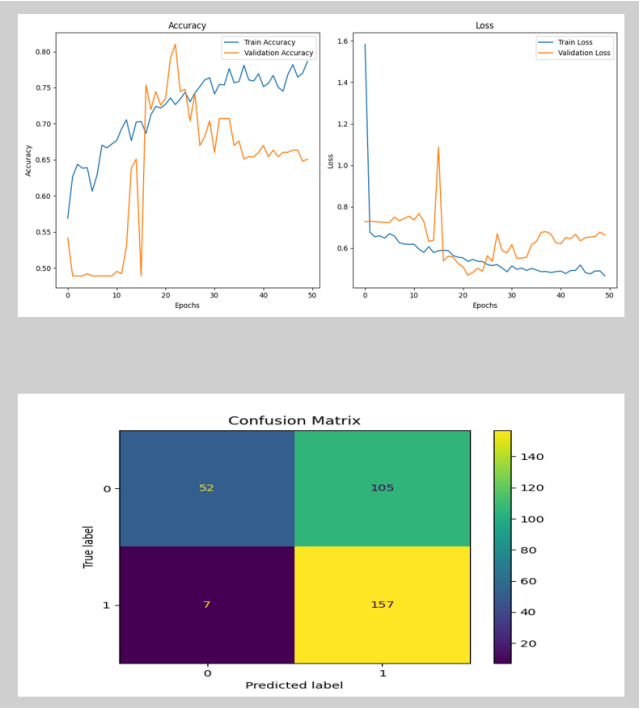


Fig. 5. Global Average Pooling Accuracy and Loss

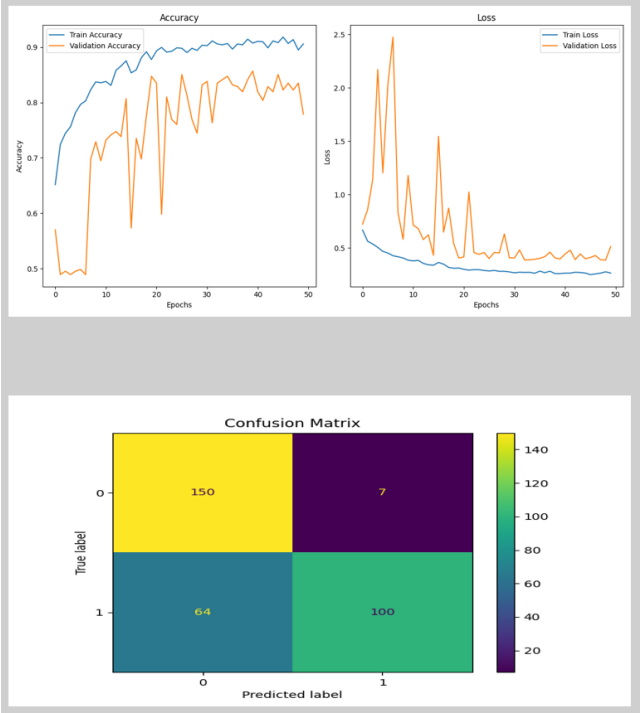


Fig. 6. Adjusted Model Hyperparameters Accuracy and Loss

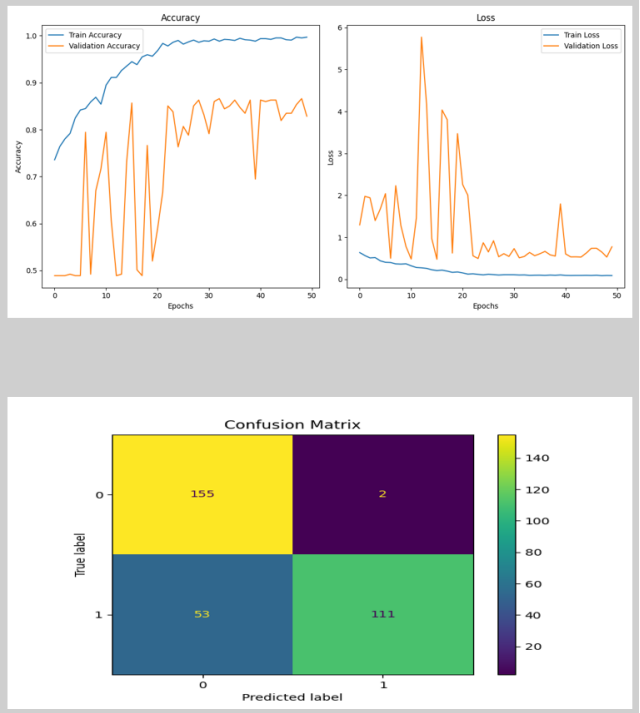


Fig. 7. Final Model Accuracy and Loss

Adjusting the hyperparameters showed a slight improvement from the original model, so we continued experimenting with other changes in these areas to get better results.

F. Final Model

4 Convolutional Layers: 128, 256, 512, 1024; 50 epochs; and with updated hyperparameters seen in previous subsection.

IV. Conclusion

Through this project, we were able to develop a desktop application for iceberg detection using synthetic aperture radar (SAR) images and convolutional neural networks (CNNs). With CNNs, we were able to classify images into two categories: icebergs and ships, with moderately high accuracy. The model demonstrated its effectiveness in detecting icebergs even with the high level of noise in the images. Additionally, various changes such as increasing the number of convolutional layers, adjusting epochs, implementing data augmentation, and incorporating global average pooling allowed us to explore how to improve the model's performance.

The results showed an improvement in training and validation accuracy and loss. Despite the challenges of overfitting due to too complex of a model, the system was still able to achieve moderately high accuracy. The developed user interface allows users to easily upload images and obtain predictions, making the application more easily accessible.

This project also demonstrates the power of deep learning in environmental research. Future work can explore additional modifications to the model and its application to more datasets. Overall, this project shows that deep learning is a very great approach to monitoring icebergs and their impact on ecosystems, helping in the prediction and mitigation of the environmental effects of climate change.

References

[1] David Hogg Anne Braakmann-Folgmann, Andrew Shepherd and Ella Redmond. Mapping the extent of giant antarctic icebergs with deep learning, 2023.
[2] Maartin Bamelis. Stack overflow, 2022.
[3] Linkedin Community. Linkedin.
[4] Kaggle. Statoil/c-core iceberg classifier challenge, 2017.