Iceberg Detection

using Convoluted Neural Networks

Sania Bandekar & Victoria Le

Project Overview

Problem: Image classification for determining if an object in a photo is an iceberg or a ship

Goal: Build a machine learning model using a CNN for identifying icebergs from satellite images

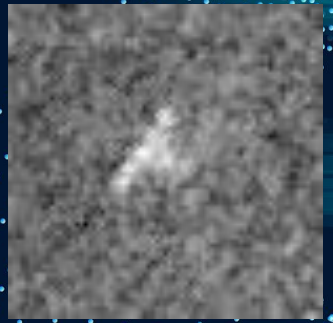
Dataset Preparation

Data Sources:

- Dataset source: JSON files
- Features: band_1 (image data), is_iceberg (label)
- Dimensions: 75x75 pixels per image
- Training set: 1604 images
- Testing set: 8424 images
    - Split 80/20 for training and validation

Preprocessing Steps:

- Load and reshape images
- Normalize pixel values



Data Loading

Datasets loaded from JSON files

Images

- Stored as arrays within the band_1 key
- Convert arrays into NumPy arrays
- Reshape them into 75x75 pixel arrays (input)

Data Normalization

Normalization Process:

- Rescaling pixel values between 0 and 1
- Helps the model learn faster and improves convergence

Model Architecture Overview

Type of Model: Convolutional Neural Network (CNN)

Layers: 3 convolutional layers followed by dense layers

Final Layer: Sigmoid activation for binary classification

Convolutional Layers

First Layer: 32 filters, 3 x 3 kernel

Second Layer: 64 filters, 3 x 3 kernel.

Third Layer: 128 filters, 3 x 3 kernel.

Padding: 'Same' to retain spatial dimensions.

Max Pooling Layers

Pooling Size

- 2 x 2 to reduce dimensionality and retain key features.

Model Regularization

Dropout: 30% to prevent overfitting.

L2 Regularization: Penalize large weights.

Dense Layers

First Dense Layer: 128 neurons with ReLU activation.

Second Dense Layer: Output layer with 1 unit, and sigmoid activation.

Model Compilation

Optimizer: Adam (learning rate = 0.001)

Loss Function: Binary cross-entropy

Metrics: Accuracy

Model Training

Epochs: 50

Batch Size: 32

Validation Data: Validation loss and accuracy

Learning Rate Adjustment

ReduceLROnPlateau

- Reduce learning rate by 50% if validation loss stagnates.

Model Evaluation

Confusion Matrix: To evaluate classification results

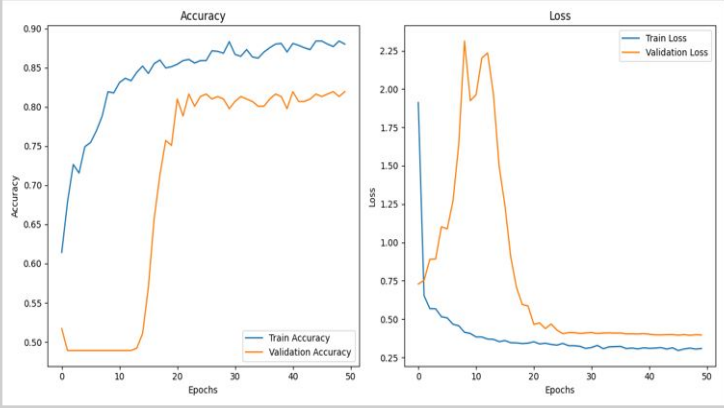
	Predicted: Iceberg	Predicted: Non-iceberg
Actual: Iceberg	True Positive	False Negative
Actual: Non-iceberg	False Positive	True Negative

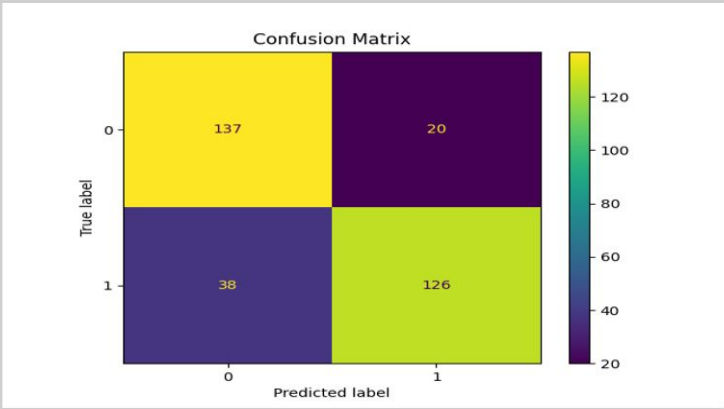
Metrics: Accuracy and Loss

Training Results: Original Model

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

Test Accuracy: 0.5819

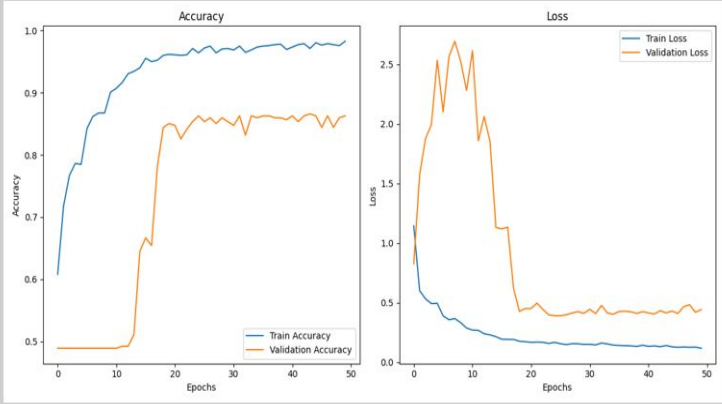


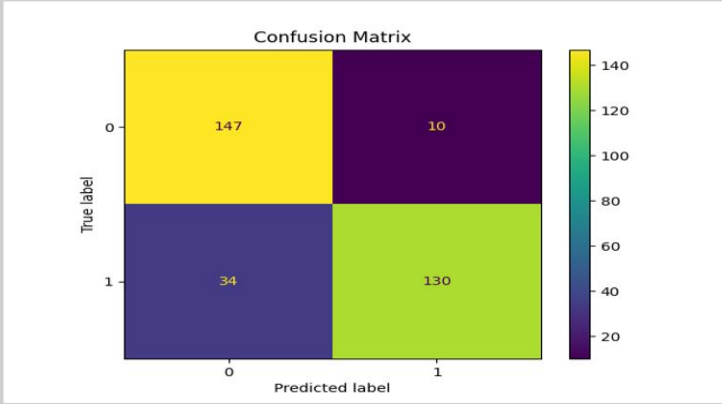


Training Results: Increasing Convoluted Layers (3 → 4)

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

Test Accuracy: 0.7173

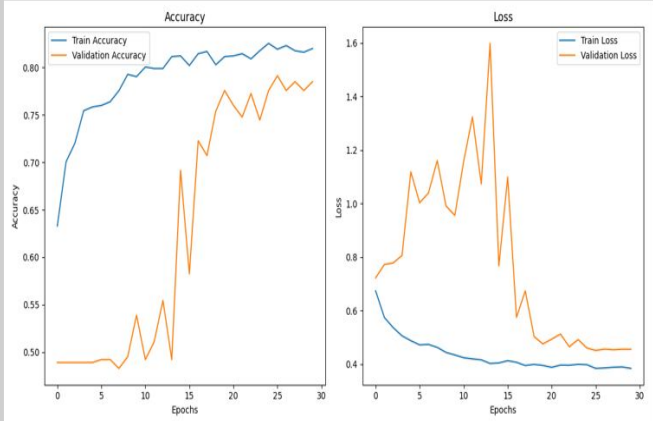


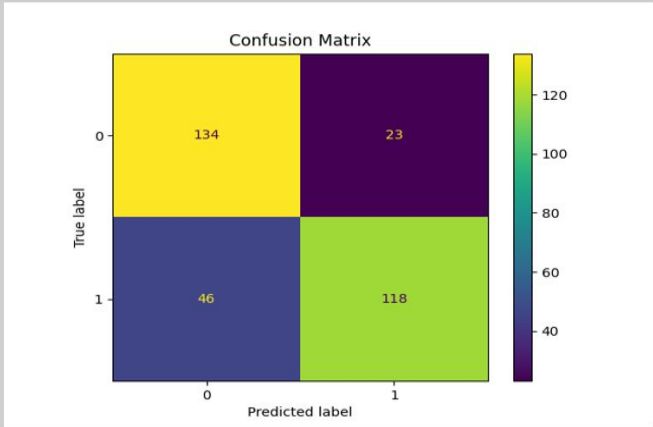


Training Results: Increased Epochs (50 → 75)

Avg Training Loss: 0.5055
Avg Validation Loss: 0.7535
Avg Training Accuracy: 0.7885
Avg Validation Accuracy: 0.6544

Test Accuracy: 0.6097

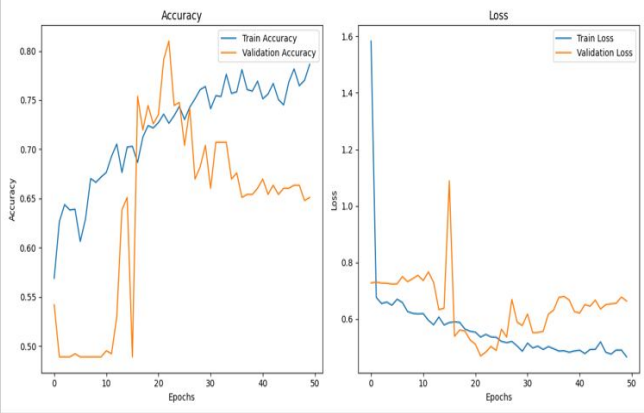


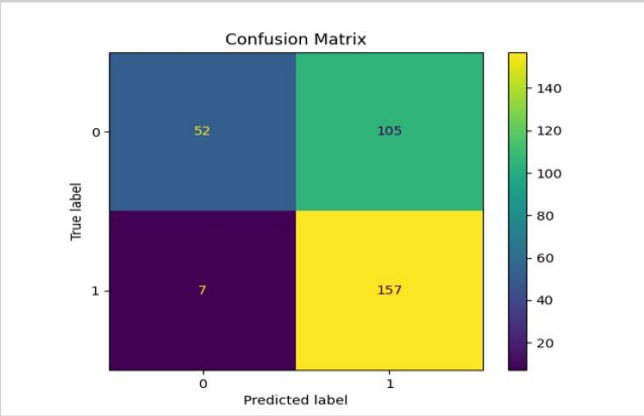


Training Results: Using Augmented Data

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

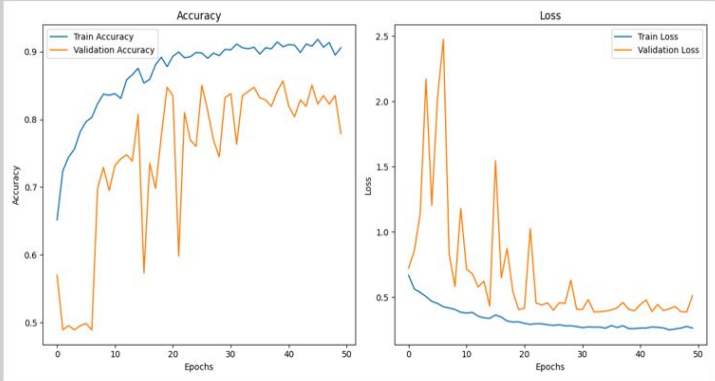
Test Accuracy: 0.1681





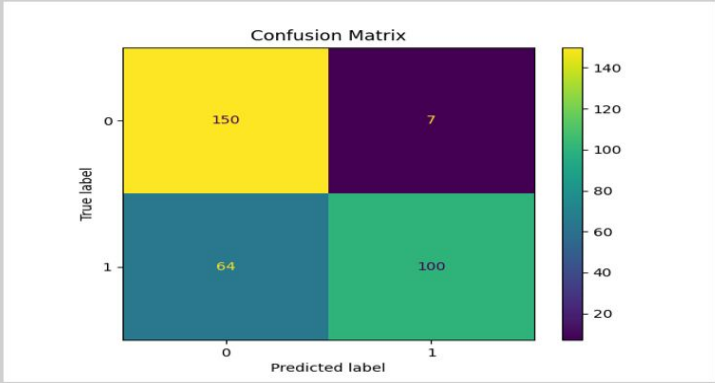
Training Results: Adjusting the Model Architecture

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

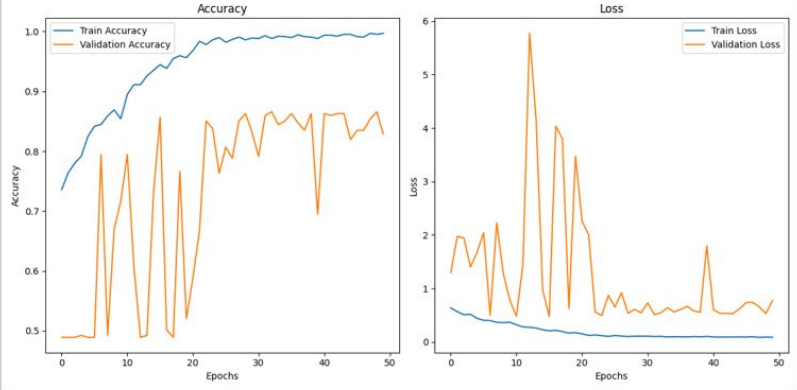


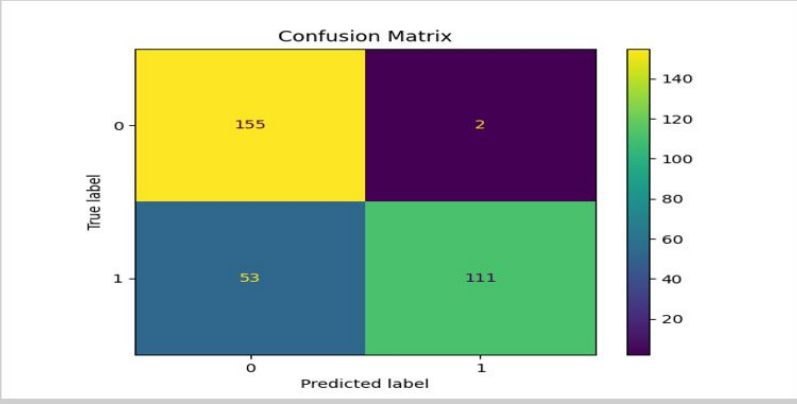
Best model (so far) :

4 Convolutional Layers: 64, 128, 256, 512 , 50 epochs, and with updated hyperparameters seen in last slide.

Avg Training Loss: 0.1992
Avg Validation Loss: 1.2652
Avg Training Accuracy: 0.9437
Avg Validation Accuracy: 0.7302

Test Accuracy: 0.7754





Demo

Challenge 1

- Brief period where all predictions on the desktop application were “ship” even if it was an iceberg

Challenge 2

- Finding another dataset to train the model on

Thank You!

Questions