Iceberg Detection and Tracking using Machine Learning

Sania Bandekar and Victoria Le

Introduction

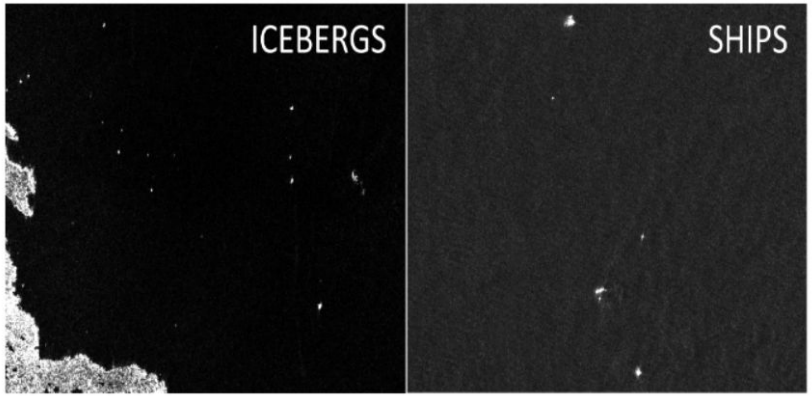
- Many researchers have utilized machine learning in order to detect icebergs and analyze them as they have a significant influence on oceanic properties.

- Recent advancements in deep learning, convolutional neural networks (CNNs) in particular, have been very useful for iceberg detection from satellite images.

Dataset Preparation

- The project uses an existing dataset from Kaggle due to the limited availability of adequately sized iceberg satellite images.
  - The dataset contains radar images of both icebergs and ships to help train the model on distinguishing from both iceberg and non-iceberg objects.

- Data was processed with scikit-learn



Convolutional Neural Network (CNN)

- After reviewing the types of neural networks, we decided to use CNN as it is the most suitable architecture for image processing and classification
    - CNNs are designed to maintain the local spatial structure of images and similar data
    - Learns through progressive levels of abstraction which allows them to recognize patterns in early layers and more complex patterns in deeper layers

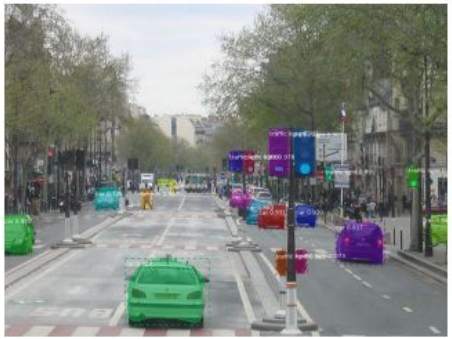
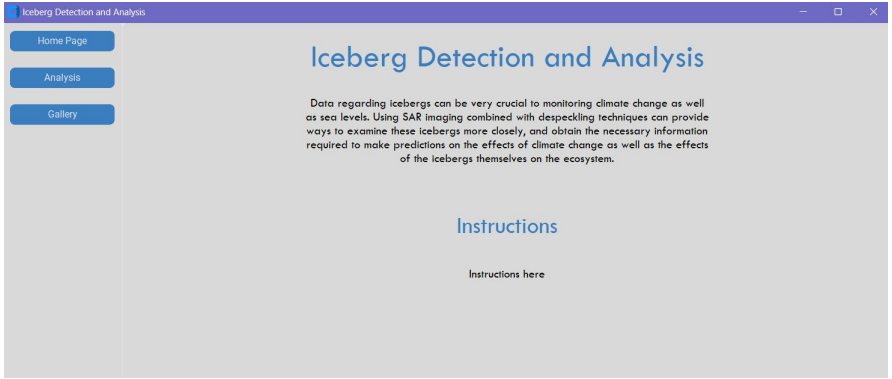


Figure 14.28: Illustration of object detection and instance segmentation using Mask R-CNN. From https://github.com/matterport/Mask_RCNN. Used with kind permission of Waleed Abdulla.

Progress

- Data Preparation
- Picked a neural network to use: CNN
- Training and Testing
- Development of User Interface

```
model = Sequential()
model.add(Conv2D(conv_depth_1, (kernel_size, kernel_size), input_shape=(75, 75, 1), padding='same', kernel_regularizer=l2
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Conv2D(conv_depth_2, (kernel_size, kernel_size), padding='same', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Conv2D(conv_depth_3, (kernel_size, kernel_size), padding='same', kernel_regularizer=l2(weight_decay)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Flatten())
model.add(Dense(dense_1, kernel_regularizer=l2(weight_decay)))
model.add(Activation('relu'))
model.add(Dropout(drop_out))

model.add(Dense(dense_2, activation='sigmoid'))

# Compile the model
optimizer = Adam(learning_rate=0.001)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-5)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```



Next Tasks to complete

- Determining best model accuracy and loss by average
- Integrating Backend with Frontend UI
- If possible: Iceberg Size Calculation with U-Net
    - We plan to look into adding an additional feature to our application where users can upload a higher quality image of an iceberg

References

thank you