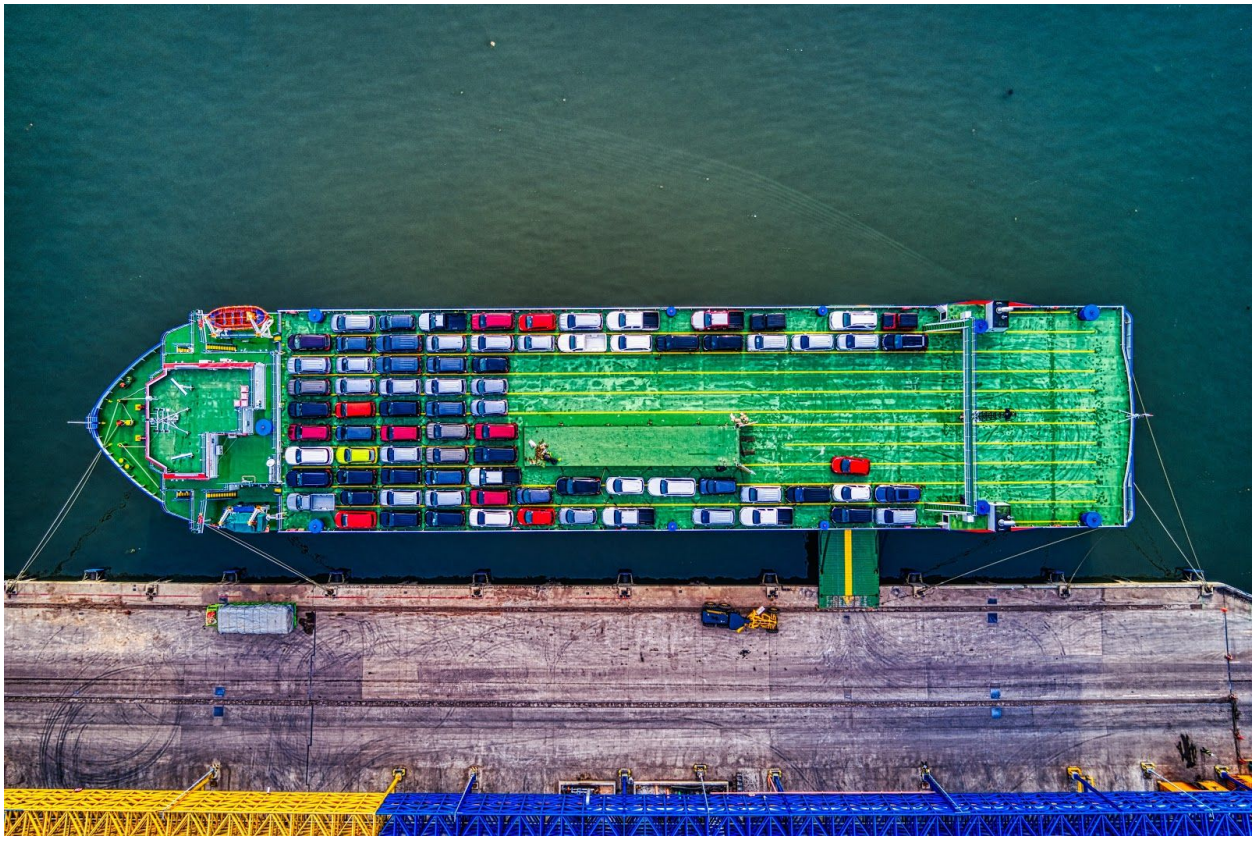


SHIPS IN SATELLITE IMAGERY

Planet labs which is a satellite imagery provider provides observation imagery of Earth’s on a daily basis. These imageries supply crucial information to various industries. By using imageries and AI, daily problems can be solved better than before.

The aim of this project is to identify the large ships in satellite images. This project can be beneficial for monitoring port activity levels and supply chain analysis.

DATA

This dataset is from Kaggle which was made available through Planet's Open California dataset, openly licensed.

Data content:

label: Valued 1 representing the "ship" class, valued 0 representing the "no-ship" class.

scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.

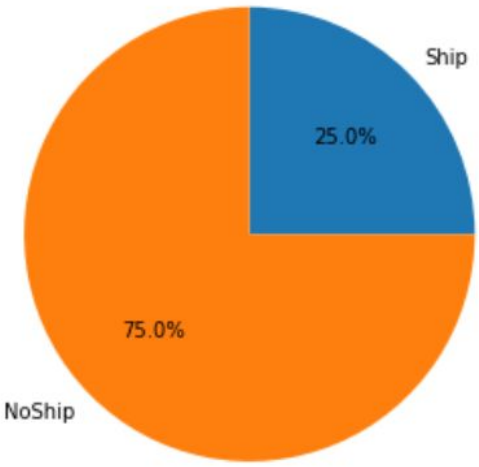
longitude_latitude: The longitude and latitude coordinates of the image center point, with values separated by a single underscore.

DATA CLEANING AND WRANGLING

This dataset includes 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue.

All features were examined to be ready for the next steps. Dataset shape, feature types were checked. Data includes 4000 images and every image is represented as a vector of length 19200 elements.

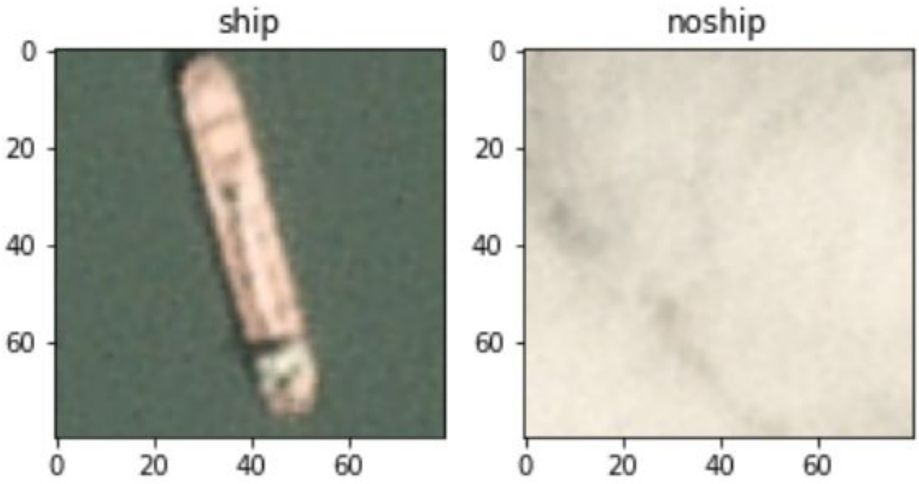
Convert data from list to array
Data and labels were converted to an array by using ‘uint8’ to represent images in the same type. The numbers for ship and no-ship were checked and data showed that:



Number of ship: 1000
Number of no-ship: 3000

Reshape data

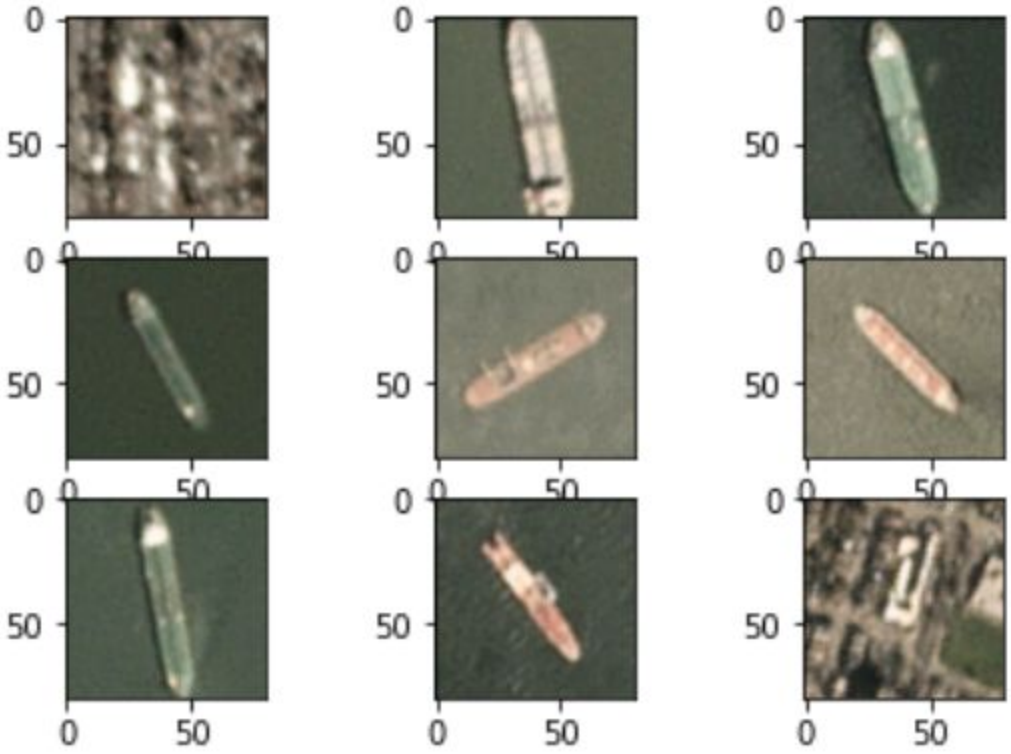
As we know, the dataset contains 4000 images. In this step, each image was represented as a vector of length 19200 elements, containing 3 layers (R, G, B) and 80x80 which is weight and height. Then the axes in the input images were transposed to the specified order, i.e., transpose function reversed the order of the dimensions. Examples after transpose:



PRE-PROCESSING

Images were normalized to the range of 0-1 by divided 255 to be ready for modeling. The 'labels' was one hot encoded by using to_categorical from Keras. This removes any unnecessary bias in the dataset.

The dataset was splitted to test and train sets as 0.8 for training and 0.2 for the test. These are the image examples of which model will use for classification:



MODELING

In this part, Keras Sequential method was used with data augmentation and without data augmentation.

First, without data augmentation:

- **Conv2D:** determines the number of kernels, which has a height and width, to convolve with the input volume. Operations produce a 2D activation map. The first required Conv2D parameter is the number of filters that the convolutional layer will learn. The filter is systematically applied to the input data to create a feature map.

- **MaxPooloing2D:** downsamples the input representation, i.e., reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

- **activation:** rectified linear activation function or ReLU is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

The model has three blocks with different convolution layer filters:

The first layer is a 2-dimensional convolution layer with 32 filters (filter size 3x3), that has an input of 80 x 80 x 3 dimensions with relu activation function and followed by a 2-dimensional max-pooling layer with 2 by 2 kernel size.

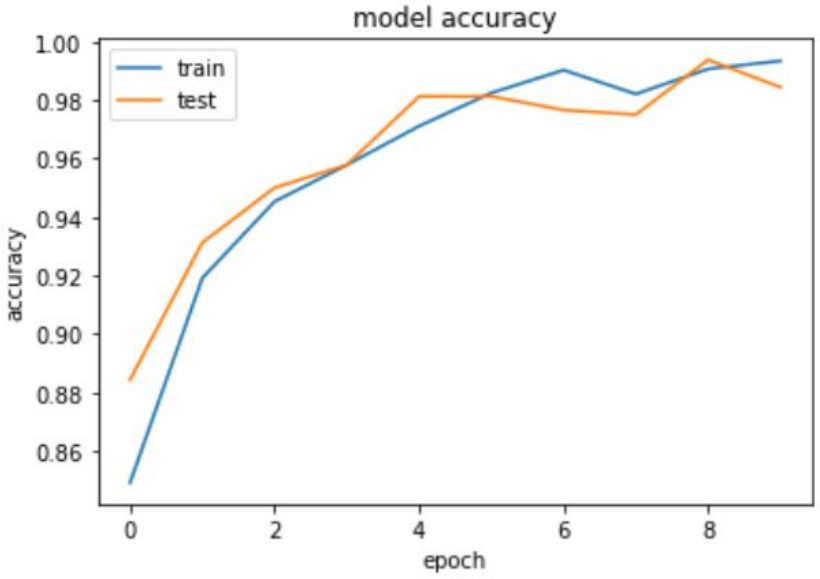
The second 2-dimensional convolution layer has 16 filters, and the third one 8 filters.

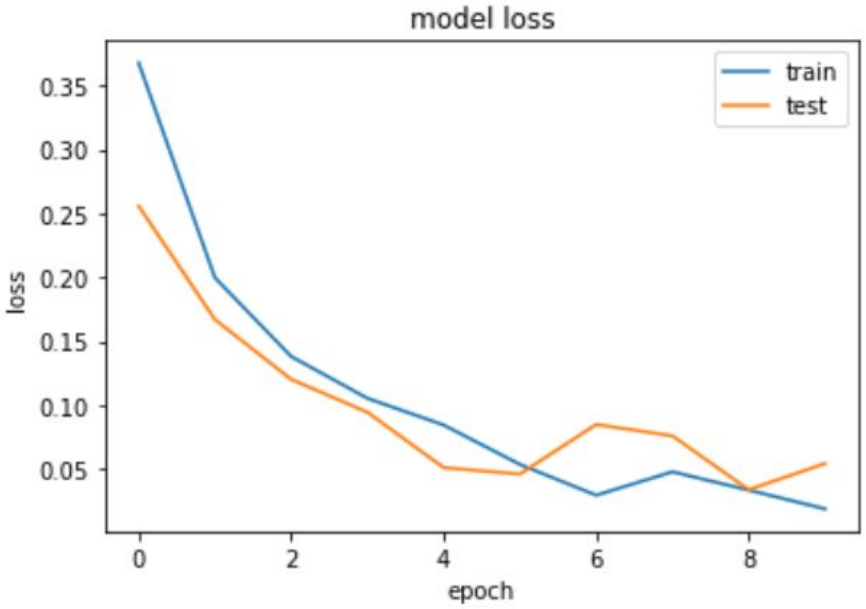
Flatten layer which converts the feature map produced by the convolutional layers into a single column.

Flatten layer followed by hidden layers of 300 and 100 neurons with a ReLU activation function.

Finally, an output layer of 2, for the probability that an input image belongs to either class ship or class no-ship.

The accuracy for without data augmentation is 98%. The accuracy and loss have good fit learning curves.

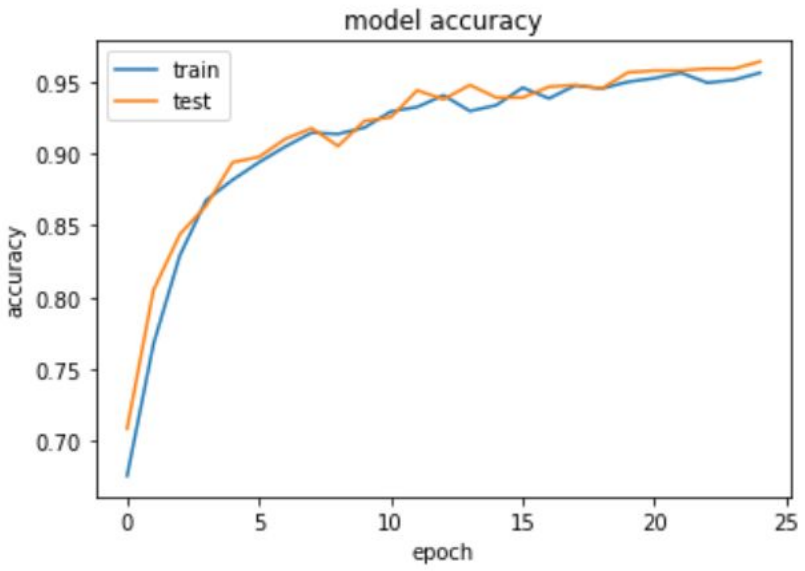


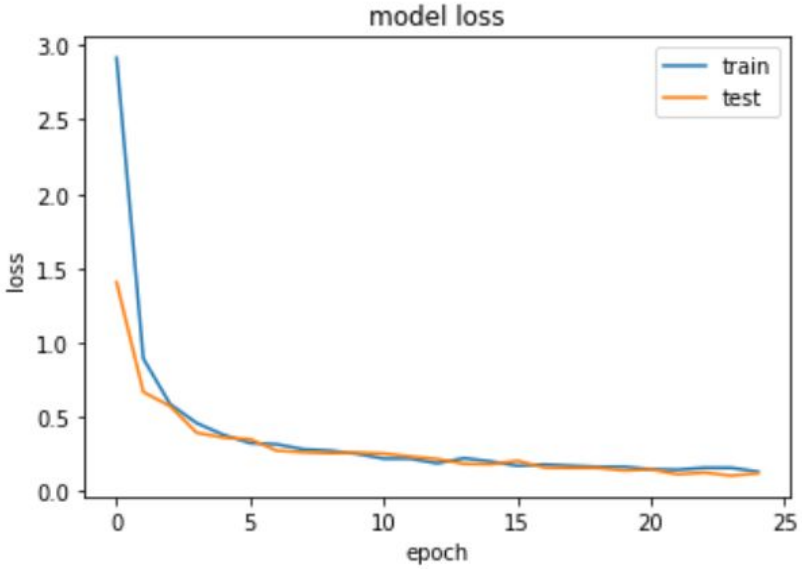


With data augmentation

Data augmentation is used to increase the amount of data by adding slightly modified copies of already existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. In this project, brightness_range, rotation_range, fill_mode, horizontal_flip, vertical_flip were used since they fit with the problem domain.

The model accuracy with data augmentation is 95%. And the accuracy and loss have good fit learning curves.





FUTURE IMPROVEMENTS

Applying different Neural Network architecture would be helpful to improve the result, especially with data augmentation.

Trying different sets of parameters for classification models would result in higher accuracy.