University of Belgrade - Faculty of Electrical Engineering

Department of Signals and Systems

13EO54NM – Neural Networks

First project

Student: Teodora Spasojević

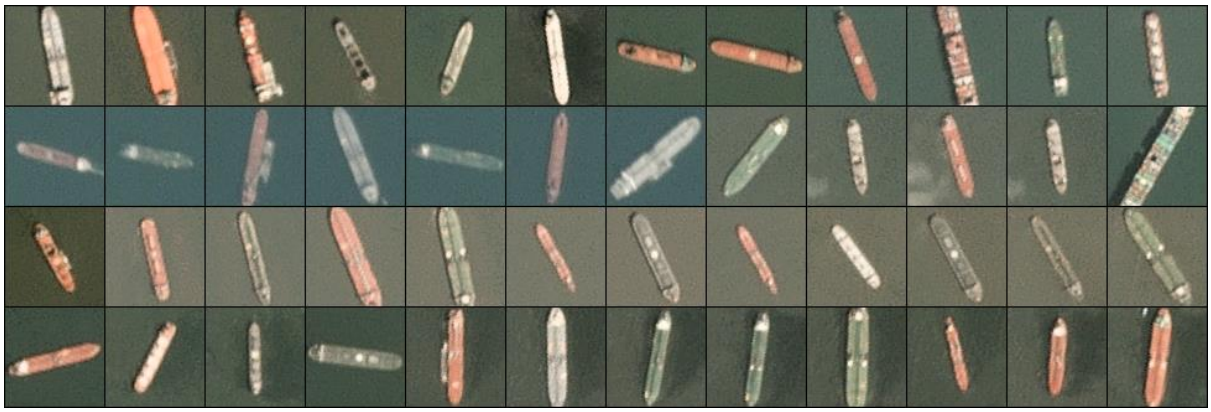
Index number: 2019/0427

February 2022.

Dataset

The dataset consists of images extracted from satellite imagery in the San Francisco Bay area and San Pedro Bay, California. It contains 4000 RGB images of 80x80 size marked with the classification "ship" or "no ship". The images are derived from PlanetScopa, with a pixel size of 3 meters.

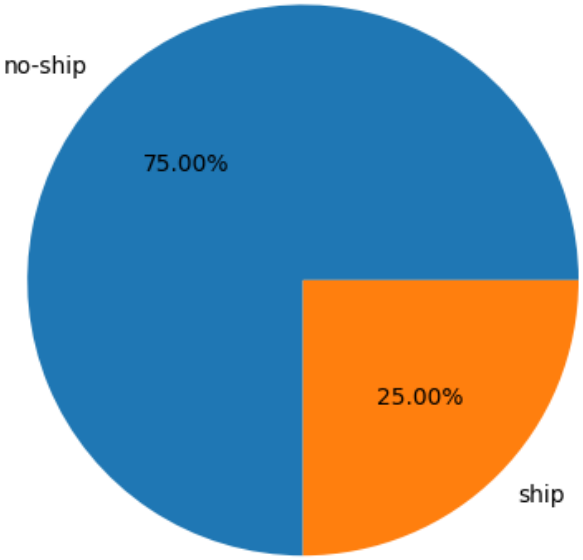
The "ship" class includes 1000 images. The images in this class are centered on the body of a ship. Ships of various sizes, orientations and atmospheric collection conditions are included. Examples of images from this class are shown below.



The "no-ship" class includes 3000 images. A third of them are randomly sampling the various characteristics of the soil cover - water, vegetation, bare earth, buildings, etc. - which do not include any part of the ship. The next third are "partial ships" containing only part of the ship, but not enough to meet the full definition of a "ship" class. The last third are images that were previously mislabeled with machine learning models, usually caused by bright pixels or strong linear characteristics. Examples of images from this class are shown below.

The goal of the project is to create a convolution neural network that will be able to recognize whether the ship is in the picture or not.

On the following page, a graphical representation of the data sharing is shown:



It can be noted that classes are not balanced. This problem is solved by oversampling (increasing the amount of data of a smaller class) or class weights method (adding weight to a smaller class when calculating a training error to increase its impact). In the code, the method used for balancing classes can be selected in the program. If AUGMENTATION flag is True, the balancing will be done using oversampling. To train through the class weights method, we need to set AUGMENTATION flag to False.

Case: AUGMENTATION = True - The number of images present in the "ship" class is increased, so there is an equal representation of classes. The current class ratio is 1:3, which means that for each image that is present in the "ship" class there are 3 images present in the "no-ship" class. This will be solved by creating 2 images per original image of the "ship" class. This will make the data set balanced. The increase in the number of images is done by augmentation, which results in obtaining different images, not just copies of existing ones. The method of augmentation used were rotating the image, cropping, adding contrast to the image, increasing the pixel value (brightening the image, darkening the image), distorting the image.

Data sharing

Instead of using train_test_split, image and label strings are randomly mixed using the same seed value of 42. This allows images and their corresponding labels to remain connected even after mixing.

70% - training set; 20% - validation set; 10% - test set.

Training and validation data sets are used to train models, while a test data set is used to test models with data unknown to the model. Unseen data is used to simulate real-world predictions, as the model has not previously seen this data. This allows developers to see how robust the model is.

Creating the model

conv_block – contains a convolution layer, layers for batch normalization and activation (the relu activation function was used). The number of filters, kernel_size, the steps to be taken are defined arbitrarily. This feature allows the programmer to create a model without having to repeat the same lines repeatedly. It also uses Python's OOP concepts which are recommended instead of coding such as C.

basic_model - creates a model using the above-mentioned function, max pooling layers and dropout layers. After selecting the number of convolutional layers, a flatten layer is introduced, along with dense layers so that the image can be classified. Flatten layer converts the map of the features produced by the convolutional layers into a single column for classification.

MaxPooling2D – reduces the spatial dimensions of the map of the hallmark produced by the convolution layer without losing information. This allows the model to become a bit more robust.

Dropout - removes the user-defined percentage of connections between neurons of successive layers. This allows the model to be robust. It can be used both in fully convoluted layers and in fully connected layers (FC layer).

BatchNormalization - normalizes the values present in the hidden part of the neural network. This is similar to the MinMax/Scaling Standard applied in machine learning algorithms. It also prevents the possibility of retraining the neural network.

Padding –fills the mark/input image map with zeros, allowing the edges to remain.

Network layout:

- Block 1:

The input layer is initialized by the Input Keras layer, which defines the number of neurons present in the input layer.
ZeroPadding is applied to the input image, so that the boundary characteristics are not lost.

- Block 2:

The first convolutional layer, starts with 16 filters and kernel size with (3.3) and steps (2.2). Padding is maintained the same, so that the image does not change spatially, until the next block in which MaxPooling appears

- Block 3 – 4:

A similar structure in both with a convolutive layer followed by a layer of MaxPooling and Dropout.

- Output block:

The map of the landmarks produced by the previous convolutional layers is converted into a single column using flatten layer and classified using the Dense layer (output layer) with the number of classes present in the dataset and the sigmoid as an activation function.

Total number of network parameters:

```
Total params: 62,530
Trainable params: 62,242
Non-trainable params: 288
```

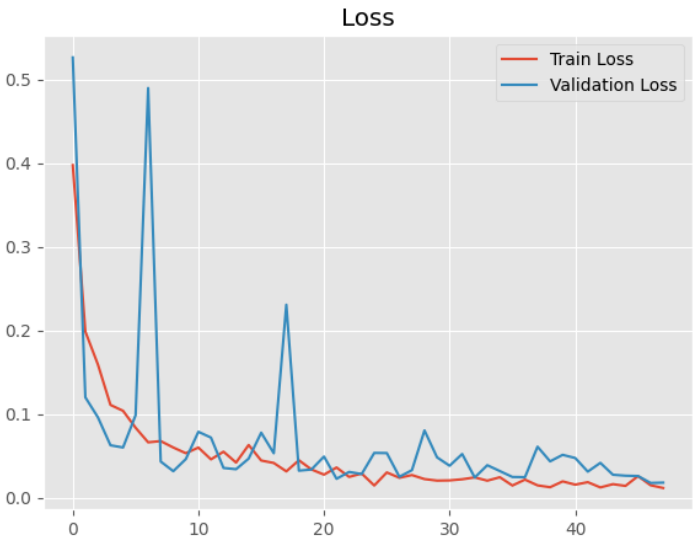
Compiling and training models

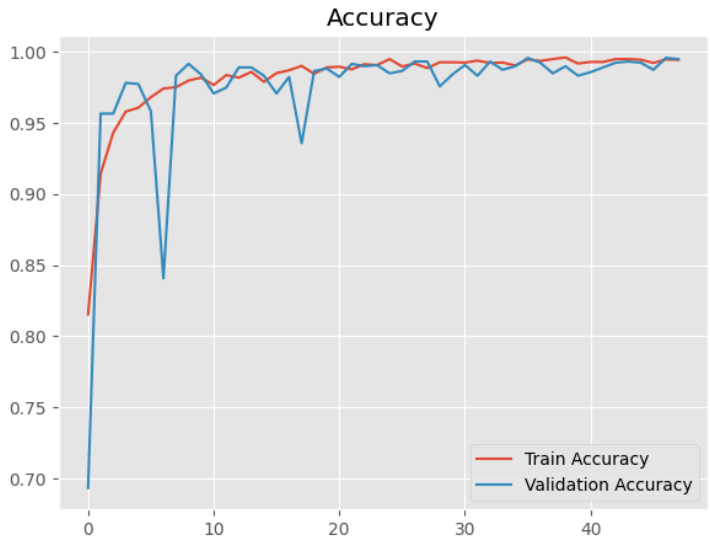
Adam's optimizer was used with a learning rate set to 1e-3. Binary crossentropy is used as a loss function since there are only two classes.

The model was trained through 50 epochs with a batch size of 16 (arbitrarily selected parmeters). The best models are stored in the file model_veight.h5 using the ModelCheckpoint function, which remembers the model with the best parameters from 50 epochs based on the maximum accuracy achieved on the validation set. At the same time, this is one of the ways to protect against the retraining of the neural network (something like early stopping, only it does not interrupt the execution of the program, but the best epoch is chosen).

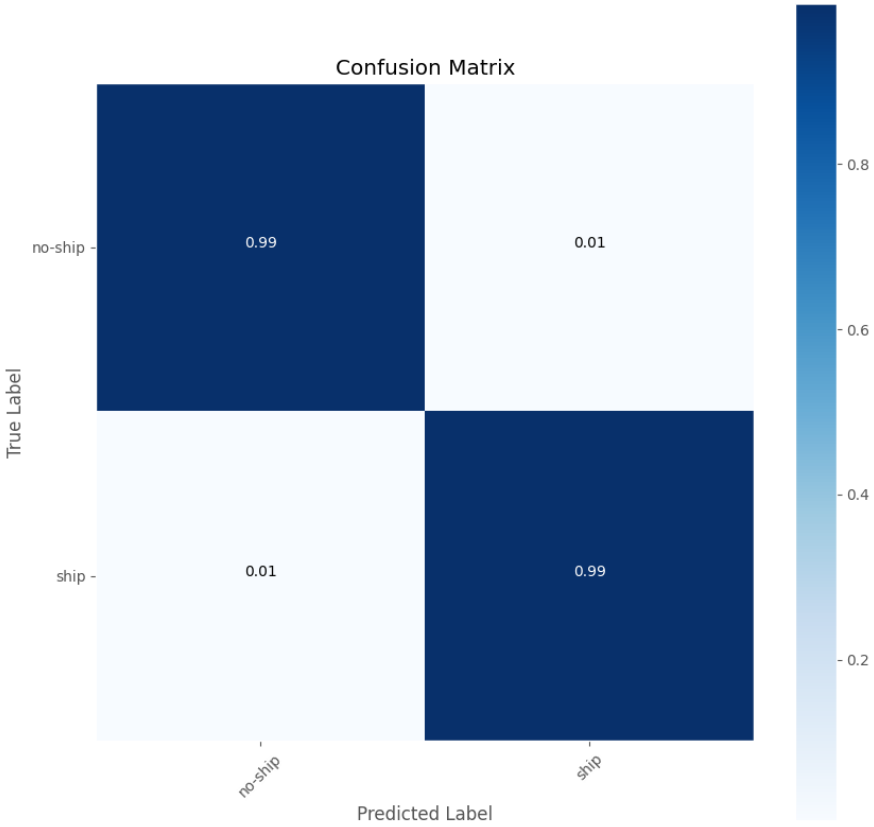
Results

Progression through the epochs:





The matrix of Confucius at the training set:



The matrix of confucius on the test set: