Satellite Image Segmentation for Ship Detection using U-net convolutional network

Péter Tamás Csatlós – TALXCE
András Kalapos – G1H5CJ

Abstract

Earth observation using satellite data is a rapidly growing field. We use satellites to monitor polar ice caps, to detect environmental disasters such as tsunamis, to predict the weather, to monitor the growth of crops and many more.

In this report we present our solution to the Kaggle Airbus Ship Detection Challenge. We are given satellite images, which might contain ships or other waterborne vehicles, the goal of the challenge is to perform semantic segmentation for ship detection on these images. The pictures might contain multiple ships, they can be placed close to each other (yet should be detected as separate ships), they can be located in ports, can be moving or stationary, etc. The pictures might show inland areas, the sea without ships, can be cloudy or foggy, lighting conditions can vary.

The large dataset containing more than a hundred thousand marine satellite images was analysed and a deep fully convolutional network based solution to perform semantic segmentation of ships on this images was presented. The network based on the U-Net architecture was successfully implemented, trained and tested. The best achieved dice coefficient on our test set was 0.714.

1. Introduction

Shipping traffic is growing fast. More ships increase the chances of infractions at sea like environmentally devastating ship accidents, piracy, illegal fishing, drug trafficking, and illegal cargo movement. This has compelled many organizations, from environmental protection agencies to insurance companies and national government authorities, to have a closer watch over the open seas.

We chose a Kaggle challenge[1] in this topic. We discovered the Airbus's ship recognition challenge as an ongoing challenge and we tried to create a solution for this problem.

In this competition we are given satellite images (more accurately sections of satellite images), which might contain ships or other waterborne vehicles. The goal is to segment the images to the "ship"/"no-ship" classes (label each pixel using these classes). The images might contain multiple ships, they can be placed close to each other (yet should be detected as separate ships), they can be located in ports, can be moving or stationary, etc. The pictures might show inland areas, the sea without ships, can be cloudy or foggy, lighting conditions can vary. The training data is given as images and masks for the ships (the later in a run length encoded format).

In the following report we will present a U-Net architecture convolutional neural network-based solution for this problem.

2. Related works

Semantic segmentation is a method to accurately identify objects on a picture to such detail that each pixel is labelled with the class of its enclosing object or region. Image classification means labelling an image to a single class, object detection results in identification, localisation and classification of certain objects. Semantic segmentation can be considered as an even finer level of inference, where even their boundary is predicted for many objects of a scene.

[1]https://www.kaggle.com/c/airbus-ship-detection

Deep neural networks had a great impact on the field of image segmentation, today as for many image processing tasks deep (convolutional) neural networks produce the best results.

Neural network based semantic segmentation has many applications in medical image processing, indoor navigation and automated driving technology [1]. It can be used to analyse the environment of an autonomous vehicle, enabling it to navigate safely from point A to B.

Satellite image segmentation has its own special use cases and properties. As we mentioned in the abstract, semantic segmentation of satellite images can be used to monitor the disintegration of polar ice caps, detect new buildings in fast growing cities, predict the weather, analyse the extent of environmental disasters (such as floods), and many more.

Satellite images usually covers a huge chunk of the surface and they could make very detailed pictures, showing many small objects. Moreover, atmospheric effects, partial clouds, and varying reflections of the sunlight can make the same object (e.g. water) look very different. This can make semantic segmentation of satellite images very difficult, for example because the neural network must label tiny objects on a huge image.

There are not any papers about satellite image segmentation focusing on ship and marine vessels, however segmentation of buildings and other objects in urban environments is a similar problem and it is well studied. The motivation behind these papers is to monitor the growth of cities in developing countries or to help installing solar panels.

Some papers [2], [3] present fully convolutional networks with image-patch or single pixel outputs. More recent works [4], [5] use encoder-decoder network architectures, sometimes referred to as U-Net or SegNet. As the network we implemented follows the U-Net architecture in the next chapter we will present this structure in more detail. [6] demonstrates the use of dilated convolution for satellite image segmentation, arguing that dilated convolution is a better way to achieve large receptive fields than using maxpool layers.

3. Network architecture

As we mentioned in the previous chapter, for image segmentation we will describe what are UNet [7] and SegNet [8] networks and why did we use them in this project.

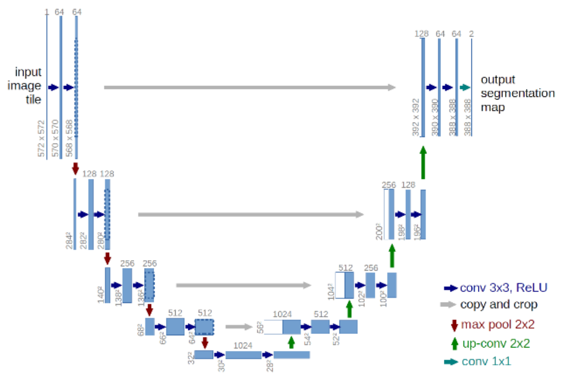

Figure 1: General U-Net architecture[2].

The U-Net was developed for biomedical image segmentation. The U name comes from the representation of the network structure (shown on figure 1.), which is defined by a contracting path and an expansive path.

The first part is the „encoder" part. It can be split into the series of encoding blocks which consist of two convolutional layers with ReLU activations and a maxpooling layer. The „decoder" part is similar to the encoder part, but in reverse. It's decoding blocks are built up of one up-convolutional layers which transform the data to higher resolution, followed by two convolutional layers.

Residual connections add the output of the encoding layers to the inputs of the decoding blocks. It helps deal with the vanishing gradient problem and in addition the decoder layer gets high resolution information from the encoder

[2] Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/u-net-architecture.png

layers, allowing predictions based on fine image details.

Our convolutional neural network is different from the network presented in the original article [7]. We use one less encoding and decoding block, and due to the imbalanced class distribution of the dataset the Dice coefficient was utilised as a loss function instead of binary cross entropy.

The Dice coefficient (also known as F1-score) is a metric to compare and measure the similarities between two sample sets. The formula is the following:

$$\frac{2|X \cap Y|}{|X| + |Y|} = 2\frac{precision \cdot recall}{precision + recall}$$

where $|X|$, $|Y|$ and $|X \cap Y|$ are the cardinalities of the two sets When used as a metric for supervised learning algorithms $|X|$ and $|Y|$ are the true and predicted outputs and their intersection, $|X \cap Y|$, is the set of correct outputs. It is commonly used in image segmentation, in cases of high class imbalance.

4. The dataset

From the competitions webpage we downloaded the training- and the test data sets and started analysing it. The training dataset consists of images and an .csv file about the images. On each picture there might be one or multiple ships, docks or land. Each image is 768x768 sized. The .csv file about the training data set contains two columns: first column contains the names of the images, second one is the run-length-encoded (RLE) mask of one ship. We read the .csv file into a *Pandas DataFrame* in order to handle it easily. It is important to note: some images appear multiple times in the dataframe, these are the pictures which contain multiple ships and each ship has a separate mask, located in separate rows.

4.1. Data analysis

After the downloading we realised how much data we have. Almost 193000 pictures in the training dataset and it's more than enough for training, if we just consider the amount of data.

We need information about the data in order to make the network learn more optimal.

So, the data analysis contains counting the empty masks (no ships on the image), number of images with multiple ships and we plotted with two pie charts.

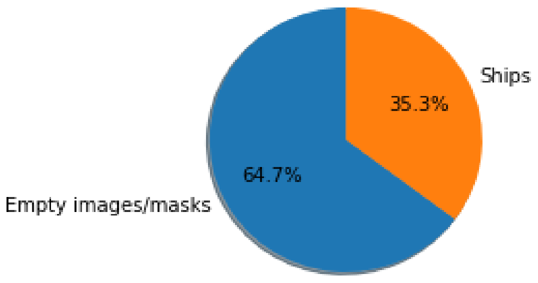

Figure 2: distribution of empty and not empty masks

According to the charts more than 75% of the images has no ships on it. If also we factor in that the ships take up only a very small area of the images (discussed in detail below) we can conclude that the two classes of this dataset is extremely unbalanced (the ratio is $\sim 1:10\,000$). If we drop all the images with no ships this ratio improves slightly (to $\sim 1:1000$), also the training time reduces significantly.

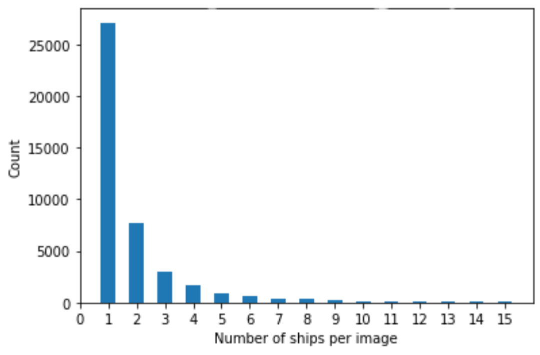

Figure 3: histogram of ship count per image

On figure 3 you can see a histogram about the number of ships on images. You can see most of the images with ships contain only a few ships. it can make the training difficult. The bar representing the number of images without ships (0 ships) is not on the diagram, because it would make even more bars invisibly small.

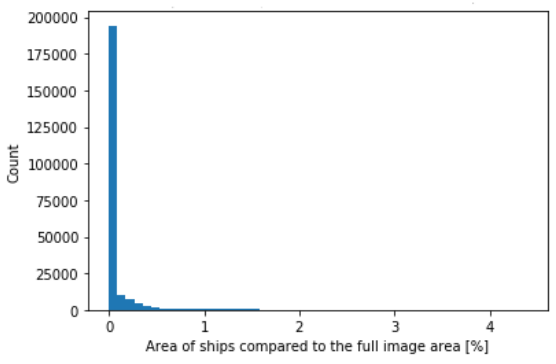

Figure 4: Ship sizes

On figure 4 you can see the ships are very small compare to the whole picture (note, that the vertical axis is in % of the full image area!). So this is another reason why training the network could be difficult.

On the next pictures there are the ships and the decoded RLE masks on them.

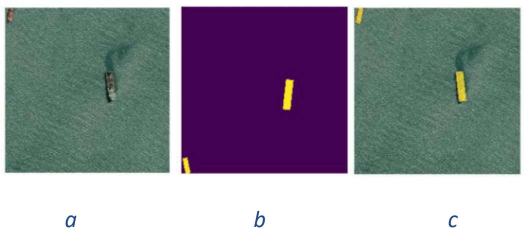

Figure 5: (a) image without mask, (b) mask, (c) image with the mask

The masks are matrices with ones and zeros therefore we have to classify the data into two classes. Zero class means there is no ship on the image and the other class means there is ship on the image.

4.2 Preparing data for learning

We split the data into three different partition. Train, validation and test data sets. We declared how much data is needed for each partition. We use 15% of the data for validation and 15% for testing. The rest is training data. Later we realised that the test and validation datasets are unnecessarily large, however for consistency we decided not to change the partitioning.

In order to process this vast amount of data we used a data generator to load and process images during training. The dataset is too large to be loaded and processed once, by using a data generator only a small portion of the images is loaded at a time. The downside of this approach is that images are reloaded and transformed again in every epoch.

5. Experiments

We implemented the model and multiple variations of it in Keras and trained them on Amazon Web Services EC2 GPUs (the Tesla K80 GPUs we used had 12GB RAM). During most trainings we used a batch size of 4 due to memory limitations. With smaller models larger batch sizes up to 16 were also used. Due to the huge size of the dataset in every epoch we only used a small portion of the training data and then (without shuffling the training data) we move on to the next set of images. By controlling the amount of batches in each epoch we are able to control the frequency of checkpoints, get quicker feedback about the training process with the price of more frequently validations meaning less time spent training.

Initially simplified network with less encoding and decoding steps smaller, filter depth was trained on downsampled images. In these experiments using binary cross entropy loss, the network was able to roughly identify ships however, as expected, due to the lack of detail in the input images the network tended to give many false positive outputs (non-ship object classified as ships). As we moved on to training on full res images with binary cross entropy the validation loss only improved in the first few short epochs, clearly not enough to learn the problem well. This is because cross entropy is not a suitable loss for the extreme class unbalance of the dataset, however some articles [4], [9] report that the dice coefficient copes much better with such problem.

As we tried to increase the network complexity from the initial simplified structure the memory requirement for the training quickly exceeded the GPU's limitations. On most of the images the ships are small and only their close surroundings should be required to identify them, thus training the network on smaller image patches shouldn't have a negative effect on the network's accuracy. We modified our training process so that only one small patch of each image is presented to the network in each batch. Even though some performance benefit in loading times might come from using multiple image patches from the same image in the same batch, this reduces the diversity of the batch so only a single patch from an image is used. Image patches are extracted with 50% overlap along their edges (area-wise the overlap is 25% for a pair of image patches)

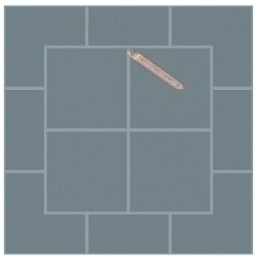
Figure 6: visualization of overlapping image patches

After multiple short trainings (5-20 epochs, 100 batches per epoch) with different training and network settings (image patch size, number of encoding, decoding steps, filter depths) we trained a network for enough epochs so that it should "see" every training image at least once. To speed up the training we initially trained the network on the full images downsampled to the input resolution (256x256 pixels) for 15 epochs, then we continued training with same sized patches of full resolution images. Using the Adam optimizer, the network was trained for 90 epochs (each with 100 batches) than for another 210 epochs with 250 batches per epoch), the full training took roughly 24 hours.

6. Results

The training history of the final model can be seen on Figure 7, it's confusion matrix on Figure 8 and some scores measuring its performance is shown in Table 1. In the appendix some example predictions with ground truth segmentation maps are presented.

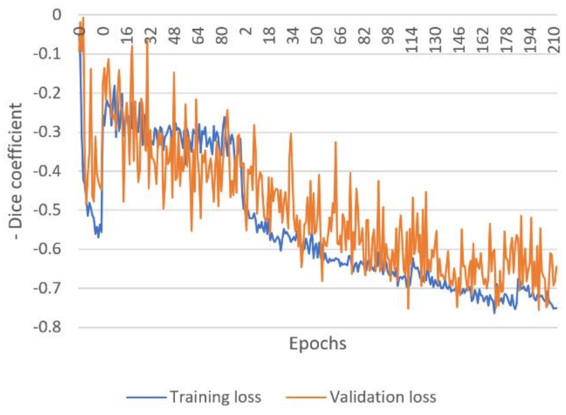

Figure 7: Training and validation loss during the training process. Note: the training was restarted where the large changes appear in the training loss curve

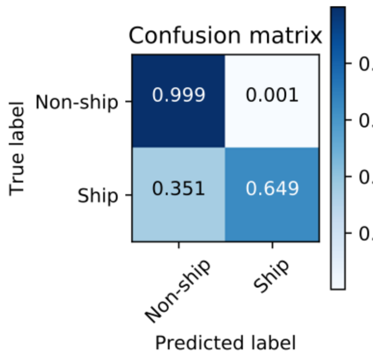

Figure 8: Confusion matrix generated on the test partition

Balanced accuracy score	0.8241
F1 similarity score/ Dice coefficient	0.7140

Table 1: Some metrics evaluated on the test partition

Among the many challenges we had to face during training the greatest one was the vast size of the dataset. With 16 image patches per batch and 250 batches per epoch one epoch takes circa 414 sec. 13 image patches were extracted from each of the 57200 training images. This means 186 batches are needed to process the dataset (containing only images with at least one ship) once, this requires 21-22 hours of training. The time-consuming training process means that, hyperparameter optimization is very slow, mistakes, and ideas which are discovered or emerge during training might not worth implementing, at the cost of restarting several hours of training.

The extreme unbalance of the two classes make this problem particularly hard to learn. Using the dice coefficient as the loss function proved to be a viable solution, however it doesn't cope well with false negative predictions, the result of which can be seen on the confusion matrix (see Figure 8).

[10] presents a new loss function to improve upon this issue, which is the combination of the binary cross entropy and the dice coefficient, however this introduces 2 new hyper-parameters, which have a serious effect on training and network performance. Another loss function which is capable of handling extreme class imbalance is the Focal loss [11]. As part of our future works on this project, we intend to experiment with both alternative loss functions.

7. Conclusions

The large dataset containing more than a hundred thousand marine satellite images was analysed and a deep fully convolutional network based solution to perform semantic segmentation of ships on this images was presented. The network based on the U-Net architecture was successfully implemented, trained and tested. The best achieved dice coefficient on our test set was 0.714. Several properties of this problem make it quite challenging to achieve good performance. To further improve our results we intend to train the network with more alternative loss functions, such as the Focal loss or the combo loss. We would also like to carry out a thorough hyperparameter optimization and explore more results in the literature to gain some comparison basis for our results.

8. References

[1] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez and J. G. Rodríguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *CoRR,* vol. abs/1704.06857, 2017.

[2] E. Maggiori, Y. Tarabalka, G. Charpiat and P. Alliez, "Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 55, pp. 645-657, 2 2017.

[3] M. Kampffmeyer, A. Salberg and R. Jenssen, "Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016.

[4] G. Chhor and C. B. Aramburu, "Satellite Image Segmentation for Building Detection using U-net," 2017.

[5] N. Audebert, B. L. Saux and S. Lefèvre, "Semantic Segmentation of Earth Observation Data Using Multimodal and Multi-scale Deep Networks," *CoRR,* vol. abs/1609.06846, 2016.

[6] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi and S. Hikosaka, "Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery," *CoRR,* vol. abs/1709.00179, 2017.

[7] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR,* vol. abs/1505.04597, 2015.

[8] V. Badrinarayanan, A. Kendall and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *CoRR,* vol. abs/1511.00561, 2015.

[9] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin and M. J. Cardoso, “Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations,” *CoRR,* vol. abs/1707.03237, 2017.

[10] S. A. Taghanaki, Y. Zheng, S. K. Zhou, B. Georgescu, P. Sharma, D. Xu, D. Comaniciu and G. Hamarneh, “Combo Loss: Handling Input and Output Imbalance in Multi-Organ Segmentation,” *CoRR,* vol. abs/1805.02798, 2018.

[11] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He and P. Dollár, “Focal Loss for Dense Object Detection,” *CoRR,* vol. abs/1708.02002, 2017.

[12] E. Shelhamer, J. Long and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, pp. 640-651, 4 2017.

8. Appendix