Neural Networks to Recognize Ships on Satellite Images

Svitlana Popereshnyak[1], Anastasiya Vecherkovskaya[2], and Liubov Ivanova[1]

[1] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37, Prospect Beresteiskyi, Kyiv, 03056, Ukraine*
[2] *Taras Shevchenko National University of Kyiv, 24, Bohdana Gavrylyshyn str., Kyiv, 02000, Ukraine*

Abstract
In the course of the work, various digital image processing algorithms were analyzed in detail, and special attention was paid to their use in solving the actual problem of recognizing ships on satellite images. Significant results were achieved in this area through the development of software that allows for solving high-precision recognition tasks based on a specially designed and trained convolutional network. The software development stage also included experimental studies and performance evaluation of the developed solution, which allows us to objectively determine the efficiency and potential capabilities in the context of a particular task. The implementation of the results obtained can help improve the quality and speed of ship recognition on satellite images, which is of great importance in various fields, including maritime and environmental monitoring. The general methodology and developed algorithms can also be applied in other areas of image processing and computer vision. As a result of our research, we are confident in the effectiveness and prospects of using the obtained developments to solve specific problems of object recognition on large volumes of satellite images.

Keywords
Neural networks, machine learning, software.

1. Introduction

Every year, a huge amount of parallel work and data that needs to be processed in a certain way appears in various fields. These tasks are of the same type, repetitive, or require constant human concentration to control or search for an object. Various software products and machine learning algorithms are used to simplify people's work and reduce time.

Today, machine learning algorithms are in active use

Many scientists have studied the use of neural networks to solve image recognition problems. The paper [1] reviews the main methods for solving computer vision problems of classification, segmentation, and image processing implemented in CV systems.

In [2], the convolutional properties of an autoencoding neural network for object detection in an image are considered. For training and testing, datasets were generated in the form of two-dimensional images with three color channels.

Ship images have been studied by scientists from all over the world, in particular, [3] presents a sequence of image processing algorithms suitable for detecting and classifying ships from nadir panchromatic electro-optical imagery. In [4], an algorithm was developed to classify ships according to size using image processing. The image of the ship was captured by a stationary camera.

Classification and segmentation of ships by analyzing satellite images will help in searching for objects without human intervention, because there are many seas and oceans, and people will not need to look through every square kilometer to find it. This will help to find objects faster and reduce the cost of human labor. This will help to control the delivery time of certain ships carrying

CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine
EMAIL: spopereshnyak@gmail.com (S. Popereshnyak); vecherkovskaia90@gmail.com (A. Vecherkovskaya); liubov.ivanova555@gmail.com (L. Ivanova)
ORCID: 0000-0002-0531-9809 (S. Popereshnyak); 0000-0003-2054-2715 (A. Vecherkovskaya); 0000-0002-9082-928X (L. Ivanova)

cargo. Or to control water borders so that an unwanted object does not cross certain boundaries. These methods will also help to control and locate enemy ships [5–7].

The work aims to study algorithms and develop software for searching for ships in a satellite image of the sea area [8, 9].

2. General Overview of Algorithms and their Comparison

Convolutional neural networks can be used to classify and segment digital images, which are specially designed to handle large amounts of data such as images. Convolutional neural networks use convolution to detect local features in images and pooling to reduce the dimensionality of the image. They can be successfully used to classify objects in an image and segment the image into separate clusters.

Convolutional neural networks are a type of neural network commonly used for image and video processing. These networks are used to automatically detect image features and characteristics such as borders, shapes, and textures.

The main difference between convolutional neural networks and fully connected networks is that convolutional networks use convolutions to process input images instead of treating each pixel of an image as a separate input (Fig. 1).

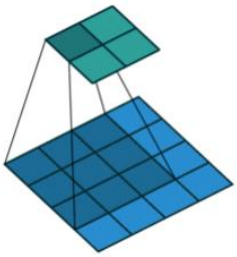

Figure 1: Reducing the size of a digital image using convolution

In Fig. 1, the bottom square is the input photo. And the top one is the new look of the photo after going through the convolution (kernel). And the lines connected between the squares are the convolution that transforms the objects. In other words, with the help of convolution, we reduce the dimensionality of a digital image for a particular purpose, which allows us to keep important features and discard unimportant ones.

Usually, one convolution is used more than once. To make it easier to understand the architecture of a neural network, we use the word neural layer, which is a certain stage of a neural network.

Convolutions are filters that slide over the input image and perform multiplication and summation operations to create a feature map (Fig. 2).

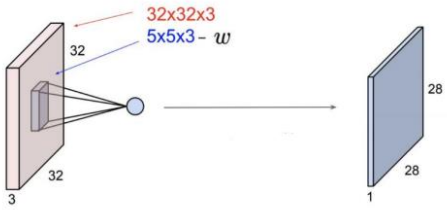

Figure 2: Using filters

Fig. 2 on the left shows how convolutions work with an image. The size of the photo is 32×32×3, where 32×32 is the size of the pixels in the horizontal and vertical directions, and 3 is the number of filters. Filters are colors, for example, red, green, and blue.

The letter w denotes the convolution itself and its 5×5×3 size. The size of the convolution is specified by the programmer.

On the right side, this is a view of the feature map to get:

$$w^T x + b\,, \tag{1}$$

where w^T is the transposed convolution, x is the part of the image that is highlighted by the convolution range, and b—is the value that the artificial model is looking for.

The main advantages of convolutional networks are that they can automatically detect and utilize local features in an image, which reduces the number of parameters that need to be trained and provides faster and more efficient performance. One of the advantages of convolutional neural networks is that they can effectively recognize local features in images, such as corners, edges, and textures, reducing the number of parameters and computational complexity compared to fully connected neural networks. In addition, convolutional neural networks can automatically learn useful features, reducing the need to manually select features to use.

Some of the disadvantages of convolutional networks include high computational

complexity and the ability to overlearn training data. Also, convolutional networks can have a complex architecture, which can make them difficult to understand and develop. The disadvantages of convolutional neural networks are the requirement for a large amount of data for training, as well as the difficulty of understanding and interpreting them. In addition, convolutional neural networks can tend to overlearn, especially when there is not enough data to train.

Compared to fully connected networks, convolutional networks are usually better for image processing because of their specialized filters that can detect different types of features. This reduces the number of parameters that need to be trained and improves training efficiency.

You can also use autoencoders to segment images and obtain embeddings for image classification. In addition, contrastive learning can be used to train embeddings that can be used for image classification.

All of these methods can be successfully used for image classification and segmentation, depending on the specifics of the task and the availability of data.

Below is a comparative table of different algorithms (Table 1):

Table 1
Advantages and disadvantages of different neural networks

Algorithm	Advantages	Disadvantages
Random forest	Fast learner. No need for retraining. Interpretability.	It will not be able to segment the image. It can only be categorized.
Boosting	Works well with big data. Quite accurate in classification.	Overtraining and high computing power load are possible.
Fully connected neural networks	With the right design, it offers high precision and the ability to parallelize machining.	Suitability for retraining. Resource requirements. When segmenting, you need to create a large number of layers—there will be a problem with gradient attenuation.
Autoencoders and embedders	It also works well with noise and image generation capabilities, reducing data size.	Resource requirements and data quality are dependent, on the need for data pre-processing.
Contrastive learning	No need for data labels, efficient use of data, Increased resilience to change, and less dependence on model architecture.	High resource requirements, difficult to set up, requires a lot of data.
Convolutional neural networks	They are effective because they were created for image classification and segmentation. Multi-level learning, with the right architecture, is very accurate.	Requires a large amount of data, requires computing resources, interpretation, data dependence

After researching the types of algorithms for classification and segmentation, since neural networks can be used as a designer, we chose the convolutional neural network algorithm as the basis for segmenting the image into 2 classes - ship and non-ship. To do this, we need to create an auto-encoder that will reduce the size of the input image and then increase it, leaving only ships in the image. It performs the task of this work best, there is also a large number of digital images, and a GPU accelerator will be used to train the algorithm, which will speed up the cloud solutions learning process many times over.

3. Input Data Analysis

The main type of input data is a digital photo taken from a satellite, they are in jpg format. There is also another type of data—masks, and coordinates of ships on the photos, if they are there, then their format is CSV. The first type of data, photographs, looks like this (Fig. 3):

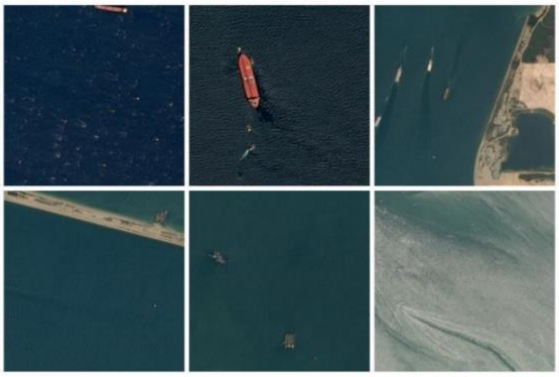

Figure 3: View of input digital images

These photos show that the size of ships and their number vary. There are also variants of photos where there are no ships, but there are other objects, such as islands, piers, or a part of the land. And the photos show that the color of the water is different from each other.

All photos are 768 by 768, but the program also has a case that converts another size to this one. The data is displayed as masks (Fig. 4).

ImageId	EncodedPi...
00003e153.jpg	
0001124c7.jpg	
000155de5.jpg	264661 17 265429 33 266197 33 266965 33 267733 33 268501 33 269269 33 270037 33 270805 33 271573 33 ...
000194a2d.jpg	360486 1 361252 4 362019 5 362785 8 363552 10 364321 10 365090 9 365858 10 366627 10 367396 9 368165...
000194a2d.jpg	51834 9 52602 9 53370 9 54138 9 54906 9 55674 7 56442 7 57210 7 57978 7 58746 7 59514 7 60282 7 6105...

Figure 4: The appearance of masks for digital images

The total number of objects to be studied, is 192555 photos, but the program artificially creates additional images from the initial set, for example, it takes one image and rotates it vertically or horizontally, and this adds more objects, but it may be that the model will overlearn on this set, so you need to immediately select the number of photos that the model has never seen. Therefore, in addition to 192555 objects, we took 15600 more to test the model.

When training a model, the distribution of objects is very important. There are cases when there is an imbalance of classes, as in our case (Fig. 5). This is critical because it is easier for the model not to find ships than to find them, so different methods need to be used. In this process, we first took a smaller number of photos with no ships by about 2 times and generated a large number of artificial images on those digital images that have ships.

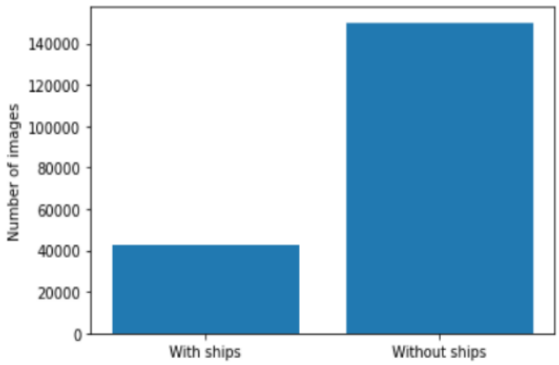

Figure 5: Distribution of images with and without ships

The number of digital images with ships is 42556. The number of digital images without ships is 149999. The number of ships in a digital image is also important (Fig. 6). Since the model has to understand what data it is working with, it will adapt to the data for training. Fig. 6 shows that most digital photos have one ship each and the distribution is similar to a logarithmic distribution. Since the program will generate a large number of artificial images, this distribution is not a problem.

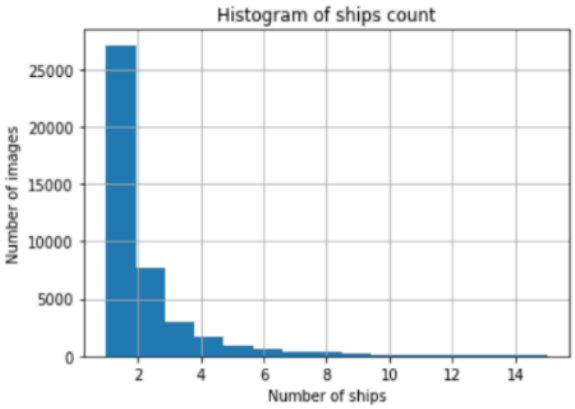

Figure 6: Distribution of the number of ships in the images

To understand how to use masks, you can look at Fig. 7:

Figure 7: Applying masks to an image

In this case, the yellow translucent color was chosen to show the location of the ships—their coordinates. The segmentation task in the program is to search for ships in a digital image and select them—to find the coordinates, i.e. masks.

If we look at Fig. 8, we can see that the distribution is exponential in the test set of photos, and most of the photos have one ship. However, the rest of the number gradually and smoothly decreases, which allows us to correctly estimate the model on the test set.

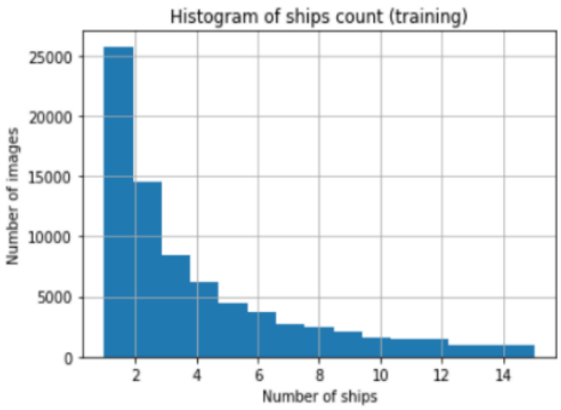

Figure 8: Distribution of the number of ships in the test sample

Data processing in machine learning is an important process, as proper processing can boost the model's results very highly, and incorrect processing can greatly reduce the results and quality of the model. However, a common practice for working with images is to artificially create additional images based on the input ones.

Below is a description that creates the AirbusDataset class, which allows you to use various methods to generate images:

```
class AirbusDataset(Dataset):
    def __init__(self, in_df, transform=None, mode='train'):
        grp = list(in_df.groupby('ImageId'))
        self.image_ids =  [_id for _id, _ in grp]
        self.image_masks = [m['EncodedPixels'].values for _,m in grp]
        self.transform = transform
        self.mode = mode
        self.img_transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
```

Figure 9: Code listing

The input to the class object is 3 parameters:

1. In_df is an array with a link to photos, and masks.
2. Transform is an array of functions or objects of the processing class, they are given below.
3. Mode has 2 modes 'train' and 'validation', when you select one of them, it will generate new photos to the database with a specific sample.

As an output, the class will generate new objects to the database—new photos.

4. Description of the Metrics Used

The choice of metric is also important, as the metric is involved in model training. There are always two important problems in such tasks.

It is necessary to choose either the model will use the most accurate classification, but this may lead to the fact that the model will sometimes classify islands, waves, or other objects as ships. Or choose the other option when the cost of error is critical and the model will repeatedly not classify different objects,

but this will result in very small ships (boats) not being found or not all points being classified in the same ship group.

For this work, we chose the second type, because this program should help users, so we don't want to distract them with unnecessary noise. Therefore, the BCEJaccardWithLogitsLoss metric was chosen.

The BCEJaccardWithLogitsLoss metric is a combination of two metrics: Binary Cross-Entropy (BCE) and Jaccard coefficient, which are used to evaluate the quality of binary semantic segmentation. This metric is a good fit because we need to segment whether a pixel is a ship or not.

BCE measures the correspondence between the predicted and true pixel values of an image. It does this by calculating the cross-entropy between the predicted and true pixel distributions. The higher the BCE, the less accurate the prediction is. The general algorithm for calculating the BCE metric:

1. Select the initial vector op $v^{(0)}$; take $t = 1$.

2. Next, you need to generate a random sample, $X_1 \ldots, \ldots, X_N$ з $f(X;\ v^{(t-1)})$.

3. Solve for $v^{(t)}$, where

$$v^t = argmax\ \frac{1}{N}\sum_{i=1}^{N} H(Xi)\frac{f(Xi\ ;\ u)}{f(Xi\ ;\ v^{(t-1)})})\ log\ f(X_i\ ;\ v) \qquad (2)$$

When convergence is achieved, stop, otherwise, t is increased by 2, and proceed to step 2.

The Jaccard coefficient measures the similarity between the predicted and true values. This is done by calculating the overlap area between the sets of pixels corresponding to the predicted and true regions in the image. The higher the Jaccard coefficient, the more accurate the prediction is.

The Jaccard coefficient measures the similarity between sets and is defined as the measure of the common part divided by the measure of the union of the sets:

$$J(A,B) = \frac{|A\ \cap\ B|}{|A\ \cup\ B|} = \frac{|A\ \cap\ B|}{|A| + |B| - |A\ \cap\ B|} \qquad (3)$$
$$0 \le J(A,B)\ \le\ 1$$

where A і B is a predicted value and a real value.

When A and B are both empty, then$J(A,B) = 0$.

The Jaccard coefficient is also used to find similar texts in a large corpus of documents. BCEJaccardWithLogitsLoss combines BCE and Jaccard coefficients into a single loss function that combines their advantages. It is used to minimize errors when training a semantic segmentation model.

5. Software and Neural Network Architecture

Creating the architecture is an important step because the number of layers is very important. If there are a lot of layers, the model will take a long time to learn and overlearn on test data, and if there are few layers, the model may not learn and perform poorly on test data.

The architecture consists of 2 main parts for auto-encoding. The first is an encoder, i.e., reducing the dimensionality of the object, for which the down_block class was created. The other part is the inverse of the previous one. That is, you need to expand a small object to the size of the input image—this is the decoder process, which allows you to create a new object—a matrix of 0 and 1. Where 1 is the location of the ship on the map, and 0 is a segmented non-ship in the digital image.

To create a neural network, you need to create a connection and combine the previous blocks correctly. The NN_Ship_Detection class is responsible for this.

You can also see its basic appearance in Fig. 10.

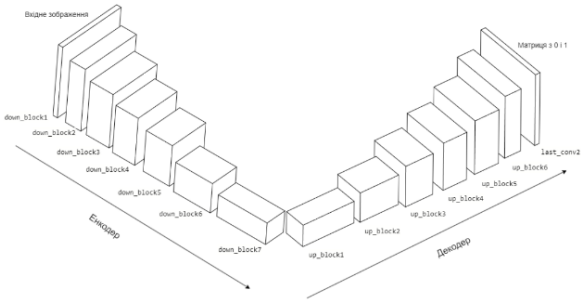

Figure 10: General view of the architecture

The architecture consists of 14 different blocks. 7 is down_block, 6 is up_block, and the last one converts the object into a matrix with the size of the input image. There is also an activation function between each block - ReLu. The blocks work as follows:

1. down_block1 is the image is added and the block increases the number of filters from 3 to 16.
2. down_block2 increases the number of filters from 16 to 32.

3. down_block3 increases the number of filters from 32 to 64.
4. down_block4 increases the number of filters from 64 to 128.
5. down_block5 increases the number of filters from 128 to 256.
6. down_block6 increases the number of filters from 256 to 512.
7. down_block7 increases the number of filters from 512 to 1024.
8. Then normalize again.
9. up_block1 reduces the number of filters from 1024 to 512.
10. up_block2 reduces the number of filters from 512 to 256.
11. up_block3 reduces the number of filters from 256 to 128.
12. up_block4 reduces the number of filters from 128 to 64.
13. up_block5 reduces the number of filters from 64 to 32.
14. up_block6 reduces the number of filters from 32 to 3.
15. last_conv2 mixes the number of filters from 3 to 1, where the result is a matrix with zeros and ones.

So the main job of this neural network is to reduce the size of the image so that only the information about the location of the ships remains, and the rest of the information is discarded. Thus, as a result, we get a new object—a matrix with zeros and ones.

6. General Discussion and Evaluation of the Results

Since this system is based on the creation of a neural network, the main part of testing will be the algorithm itself. Testing neural networks is an important part of their development and use. It is the process of evaluating the quality of a training model on independent test data to confirm that it works properly.

Backpropagation testing is the process of testing the performance of a machine learning model using a dataset that the model has not seen before.

To perform testing on a deferred dataset, you first need to divide the total dataset into training, validation, and test samples. The training set is used to train the model, the validation set is used to adjust hyperparameters and evaluate the model performance on unknown data, and the test set is used to finally evaluate the model performance.

After training the model, testing on a deferred dataset is performed on a test set that consists of data that the model did not see during training. The purpose of testing is to evaluate the model's performance on new data that the model has not seen before. The results of the testing can be used to make decisions about using the model in real-world applications, such as production tasks or research.

When testing on a deferred dataset, it is important to keep in mind that the test results may be dependent on the composition of the test sample. If the test sample does not represent diverse data, you may experience a carryover problem where the model performs well on a 50-test sample but performs poorly on real data. To avoid this problem, you should use a test sample that represents as much diversity as possible.

After evaluating the results on the deferred set, you can determine how well the model can generalize its knowledge to new data. If the results on the deferred dataset are poor and the results on the training dataset are very good, then this may indicate that the model is overtrained on the training dataset.

It is also important to keep in mind that the resulting metrics on the deferred dataset may be slightly worse than on the training dataset, as the model has not had the opportunity to learn from this data and they are assigned the role of evaluating the overall performance of the model. However, if the difference between the results on the training and deferred datasets is significant, it may be a sign of the model's poor ability to generalize its knowledge to new data.

Testing on a deferred dataset is an important step in the process of developing neural networks, as it allows you to check the overall performance of the model on new, previously unseen data.

Evaluation is an important step to check whether the model is working well and whether there are moments when the model is overlearning or underlearning. It is important to look at each iteration and choose the best one.

Using the testing method described above, we checked the quality of the image model classification. Fig. 11 shows the training process.

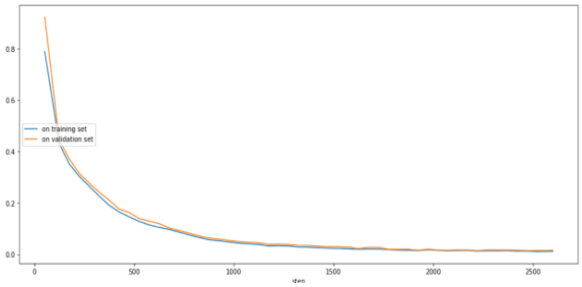

Figure 11: Metric results for each iteration of the model during training

The data selected for validation shows a good result. However, the graph shows that there is a slight overfitting. Overfitting is an unpleasant moment when the model memorizes the data it has been trained on well but performs poorly on new data. But as you can see on the graph, the overfitting is not critical and the model copes well with the result.

To better understand the model's estimates, we need to look at the table (Table 2):

Table 2
Model results and comparison with analogs

Title	51	674	1274	1974	2600/last_res
MyModel	0.922	0.102	0.039	0.029	0.015
AlexNet	-	-	-	-	0.004
U-Net	-	-	-	-	0.009

Table 2 compares 3 different models, where MyModel is the model created in the course of this work. And the other two are state-of-the-art models that are publicly available and have been trained on different data. They are also trained on 52 digital images of ships from the satellite, so it is possible to compare these models with each other.

The table shows that the model does not segment the image well at a small number of iterations. At iteration 51, the model is wrong in about 92% of cases. But with many iterations, the result improves.

It is also possible to train the model for a much higher number of iterations, but it can cause the problem of gradient decay—this is when the model in the layer works with very small values, and it takes a lot of time and power to calculate them, and calculating small gradients will almost imperceptibly improve the results.

The model performed best at iteration number 2600. The model is wrong in almost 2% of cases, which is a good result and this result will be the final one.

7. Conclusions

This paper studies modern methods of working with digital images. This gives an understanding of working with artificial intelligence, namely neural networks. What types of them are there, what they are used for, their advantages and disadvantages for maximum quality of digital image segmentation?

The architecture of the neural network was created. For its construction, convolutional neural networks in the autoencoder system, a method for encoding and decoding were used. The main function of this network is to obtain a digital image and convert it to a matrix form, where the elements have values of 0 and 1, where 0 is not a ship and 1 is a ship. We also analyzed the architecture components: advantages and disadvantages, metrics, activation functions, and the data the model receives at the input and output. Using this architecture, we created software for recognizing ships in digital images.

We went through all the main stages of working with the model. The first step was to search for data, which allowed us to choose the right methods for working with images.

The next step was to analyze the data to see the features of the dataset. It was found that there was an imbalance of classes in the set, where there were many more photos without ships than photos with ships. Then it was decided to artificially create data—digital images that had at least one ship were flipped horizontally and vertically, which made it possible to increase the number of digital images with a ship by 3 times.

Then the data was processed, and an additional number of images with ships were created. This set can also be used in other tasks.

The next step was to create a neural network architecture using convolutional methods and encoding and decoding—autoencoding. For this purpose, we used the Python programming language and the PyTorch library. 54 This library is designed to work with neural networks and makes it possible to use GPU technology.

The last stage in the program's development was model training and quality assessment. The model was trained for 2600

iterations. The trained model can be used in other tasks and programs.

To measure the quality, we chose the metric for segmentation—BCEJaccardWithLogitsLoss. To qualitatively measure the model, we chose the method of splitting the data into training, test, and validation data. In the final testing of the test data, it was found that the trained model was wrong in 1.5% of cases from the correct value, which is a good result.

A possible improvement of the architecture is to turn the task into a multi-segmentation one, where the model can show not only the location of the ship but also its type, for example, cargo, tourist, military, etc.

References

[1] O. Zinchenko, O. Zvenigorodsky T. Kisil, Convolutional Neural Networks for Solving Computer Vision Problems, Telecommunication and Information Technologies 2(75) (2022) 4–12. doi: 10.31673/2412-4338.2022.020411

[2] L. Yasenko, Y. Klyatchenko, Convolutional Neural Network Properties Based on an Autoencoder, Inf. Technol. Comput. Eng. 52(3) (2021) 77–85. doi: 10.31649/1999-9941-2021-52-3-77-85.

[3] H. Buck, et al., Ship Detection and Classification from Overhead Imagery, Proc. SPIE 6696, Applications of Digital Image Processing XXX, 66961C (2007). doi: 10.1117/12.754019.

[4] G. Santhalia, S. Singh, S. Singh, Safer Navigation of Ships by Image Processing & Neural Network, Second Asia International Conference on Modelling & Simulation (AMS) (2008) 660–665. doi: 10.1109/AMS.2008.48.

[5] B. Bebeshko, et al., Application of Game Theory, Fuzzy Logic and Neural Networks for Assessing Risks and Forecasting Rates of Digital Currency, J. Theor. Appl. Inf. Technol. 100(24) (2022) 7390–7404.

[6] K. Khorolska, et al., Application of a Convolutional Neural Network with a Module of Elementary Graphic Primitive Classifiers in the Problems of Recognition of Drawing Documentation and Transformation of 2D to 3D Models, J. Theor. Appl. Inf. Technol. 100(24) (2022) 7426–7437.

[7] V. Sokolov, P. Skladannyi, A. Platonenko, Video Channel Suppression Method of Unmanned Aerial Vehicles, in: IEEE 41st International Conference on Electronics and Nanotechnology (2022) 473–477. doi: 10.1109/ELNANO54667.2022.9927105.

[8] K. Eldhuset, An Automatic Ship and Ship Wake Detection System for Spaceborne SAR Images in Coastal Regions, IEEE Transactions on Geoscience and Remote Sensing 34(4) (1996) 1010–1019. doi: 10.1109/36.508418.

[9] A. Vecherkovska, S. Popereshniak, Review of Machine Learning Algorithms and Their Application for Forecasting Cryptocurrency Purchase Prices, Bulletin of Kherson National Technical University 4(87) (2023) 223–229. doi: 10.35546/kntu2078-4481.2023.4.26.