МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА АВІОНІКИ ТА СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____Олена ТАЧИНІНА

«_____» _____2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: «Розпізнавання та класифікація літальних апаратів за супутниковими зображеннями з використанням штучного інтелекту»

Виконавець: студент групи Ба-151-21-2-СУ _____ Станіслав ГОНЧАРЕНКО

Керівник: к.т.н., доцент _____ Антоніна КЛІПА

Нормоконтролер: _____ Микола ДИВНИЧ

Київ 2025

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
STATE UNIVERSITY “KYIV AVIATION INSTITUTE”
FACULTY OF AIR NAVIGATION, ELECTRONICS AND TELECOMMUNICATIONS
DEPARTMENT OF AVIONICS AND CONTROL SYSTEMS

APPROVED FOR DEFENCE

Head of the Department

_____ Olena TACHYNINA

“____” _____ 2025

QUALIFICATION PAPER

(EXPLANATORY NOTE)

FOR THE ACADEMIC DEGREE OF BACHELOR

Title: **“**Detection and Classification of Aircraft from Satellite Images using Artificial Intelligence**”**

Submitted by: student of group Ба-151-21-2-СУ _____Stanislav HONCHARENKO

Supervisor: PhD, associate professor _____Antonina KLIPA

Standards inspector: _____ Mykola DYVNYCH

Kyiv 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра авіоніки та систем управління

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олена ТАЧИНІНА

«_____»_____2025 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Гончаренка Станіслава Євгеновича

1. Тема кваліфікаційної роботи «Розпізнавання та класифікація літальних апаратів за супутниковими зображеннями з використанням штучного інтелекту» затверджена наказом ректора від «20» березня 2025 р. № 429/ст.

2. Термін виконання роботи: з 19.05.2025 по 14.06.2025.

3. Вхідні дані роботи:

Нейронна мережа для класифікації зображень; набір даних, підготовлений з використанням методів доповнення

4. Зміст пояснювальної записки:

Огляд методів з розпізнавання об'єктів та класифікації зображень; Моделювання системи з розпізнавання та класифікації літальних апаратів на супутникових зображеннях з використанням штучного інтелекту

5. Перелік обов’язкового ілюстративного матеріалу:

1. Приклад сегментації зображення
2. Структура Faster R-CNN
3. Архітектура EfficientNet
4. Повний цикл отримання інформації про літальний апарат із супутникового зображення

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Аналіз існуючих моделей з розпізнавання об’єктів та класифікації зображень	19.05.2025 – 24.05.2025	
2	Вибір моделей та побудова структури системи з розпізнавання літальних апаратів	25.05.2025 – 29.05.2025	
3	Вивчення процесу підготовки датасету	29.05.2025 – 01.06.2025	
4	Дослідження методів покращення точності розпізнавання об’єктів	02.06.2025 – 04.06.2025	
5	Аналіз програмного та апаратного забезпечення, необхідного для тренування моделей	04.06.2025 – 07.06.2025	

7. Дата видачі завдання: «05» травня 2025 р.

Керівник кваліфікаційної роботи _____ Антоніна КЛІПА

(підпис керівника)

Завдання прийняв до виконання _____ Станіслав ГОНЧАРЕНКО

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
STATE UNIVERSITY “KYIV AVIATION INSTITUTE”

Faculty of Air Navigation, Electronics and Telecommunications

Department of Avionics and Control Systems

Specialty: 151 “Automation and Computer-integrated Technologies”

APPROVED BY

Head of the Department

_____ Olena TACHYNINA

“__–__”_____2025

Qualification Paper Assignment for Graduate Student

Honcharenko Stanislav Yevhenovich

1. The qualification paper title “Detection and Classification of Aircraft from Satellite Images using Artificial Intelligence” was approved by the Rector’s order of March 20, 2025 № 429/ст.

2. The paper to be completed between: 19.05.2025 and 14.06.2025

3. Initial data for the paper:

Neural network for image classification; dataset prepared using augmentation techinques

4. The content of the explanatory note:

Overview of object recognition and image classification methods, modeling of a system for recognizing and classifying aircraft on satellite images using artificial intelligence

5. The list of mandatory illustrations:

1. Example of image segmentation
2. Structure of Faster R-CNN
3. EfficientNet architecture
4. Full process of retrieving information about aircrafts from satellite image

6. Timetable

№	Assignment	Dates of completion	Completion mark
1	Analysis of existing object recognition and image classification models	19.05.2025 – 24.05.2025	
2	Model selection and structure construction of an aircraft recognition system	25.05.2025 – 29.05.2025	
3	Studying the dataset preparation process	29.05.2025 – 01.06.2025	
4	Research on methods to improve object recognition accuracy	02.06.2025 – 04.06.2025	
5	Analysis of software and hardware required for training models	04.06.2025 – 07.06.2025	

7. Assignment issue date: “05” May 2025

Qualification paper supervisor _____ Antonina KLIPA

(the supervisor’s signature)

Issued task accepted _____ Stanislav HONCHARENKO

(the graduate student’s signature)

ABSTRACT

Text part of the work: 61 pages, 25 figures, 2 tables.

Object of the study – the process of recognizing and classifying aircraft at home airfields by analyzing satellite images with artificial intelligence models

Subject of the study – software for recognizing aircraft by analyzing satellite images with artificial intelligence models

Purpose of the work – development of a procedure for recognizing and classifying aircraft at airfields on satellite images using artificial intelligence.

Research methods – image processing, object recognition in images, image classification, neural networks, fine-tune of artificial intelligence models.

In this qualification work, a study of existing methods of recognizing objects in satellite images and classifying images was conducted. Based on the data obtained, a system for recognizing aircraft at airfields was modeled. The main criterion for building such a system was high accuracy in recognizing and classifying aircraft, even those of small size or with poor image quality due to low resolution and surrounding elements.

SOFTWARE, ARTIFICIAL INTELLIGENCE, NEURAL NETWORK, AIRCRAFT, MODELLING, OBJECT DETECTION, IMAGE CLASSIFICATION, FINE-TUNE, SATELITTE IMAGERY.

CONTENT

LIST OF ABBREVIATION

CNN – Convolution Neural Network

R-CNN – Region-based Convolutional Neural Network

RPN – Region Proposal Network

OAN – Objectness Activation Network

mAP – mean Average Precision

INTRODUCTION

In the current conditions of the development of automatic image analysis technologies, the need for object recognition in satellite images is growing. In particular, aircraft identification plays an important role in the areas of security, airspace monitoring, and military analytics. Existing image processing methods have limitations in terms of the accuracy of object detection at different angles and in conditions of low image quality. The proposed approach, which combines object recognition using Oriented R-CNN or Faster R-CNN and their subsequent classification, allows you to increase the accuracy and speed of data analysis.

Among the challenges in this area, the following ones are crucial in designing the aircraft detection and classification system:

- The large size of satellite images due to which it takes many computational powers to detect and classify objects. It is not possible to analyze the full image without splitting it into smaller parts because existing hardware solutions for training AI models are very expensive or limited on resources.
- Wide variety of images due to different shooting conditions (level of lighting, weather conditions, viewing angle).
- Overlapping of objects. In complex scenes with many objects there is a chance that part of them will overlap, causing the model to predict less precise.
- Lack of a sufficiently large and high-quality open dataset for training models.

Before neural networks became popular, techniques such as contour analysis and histogram approach were used to recognize objects in images. However, due to their poor performance in processing large data sets and recognizing small objects, these approaches have been replaced by newer ones. With the rapid development of deep learning, it has become possible to use neural networks for highly accurate image analysis.

In modern time, the main accent in developing such systems is to combine machine learning methods with deep neural networks to increase the accuracy and speed of recognition [1]. Among the main trends in this area are:

- Using deep convolutional neural networks (CNN). Faster R-CNN, YOLO, RetinaNet are actively used to detect objects in satellite images. Oriented R-CNN allows you to work with objects that have an arbitrary slope, which is especially important for aircraft.
- Optimization of calculations and speeding up detection in large images using methods for dividing large images into smaller overlapping patches. Introducing mechanisms for pre-filtering "empty" image fragments, for example, using Objectness Activation Network (OAN).
- Combined models for recognition and classification. After detecting objects with bounding boxes, ResNet, EfficientNet, Vision Transformers and other models are used to identify different classes of objects.
- Enriching datasets through augmentation (rotations, contrast changes, reflections) to increase the number of training examples.

Purpose of this qualification work is to develop a system for automatic recognition of aircraft in satellite images using deep learning methods.

Main tasks:

- To prepare high-resolution satellite images for model training and validation.
- Create a dataset for training and testing models, including image augmentation. Extract sufficient data from images, such as coordinates of center of an object on the image, its height and width, angle of rotation relative to coordinate X.
- To implement a model for object recognition in images using Oriented R-CNN.
- Apply Objectness Activation Network on the model to filter out patches that do not contain objects.
- Develop a classification model to identify aircraft types.

- Fine-tune models to improve recognition accuracy.

The process of detecting and classifying objects in satellite images involves multiple steps that must be carefully designed. Each step impacts on the overall effectiveness of the system, from the initial preparation of the data to the final fine-tuning of deep learning models. In this work the key components involved in this process are outlined, with a focus on practical and widely adopted methodologies in remote sensing and artificial intelligence.

The foundation of any object detection and classification pipeline is the quantity and quality of data. There are a lot of public datasets available in the internet for free that can be used for object detection tasks or other purposes. Examples of satellite imagery datasets include DOTA 2.0 (Dataset for Object Detection in Aerial Images), MAR20 (Maritime Object Detection Dataset), and xView. These datasets contain thousands of high-resolution satellite images annotated with bounding boxes and class labels for various object types such as aircraft, ships, vehicles, and buildings.

After choosing the matching set, data augmentation techniques are applied to increase variability and robustness. Among techniques the most popular and simple are rotation, flipping, scaling, contrast adjustments, and cropping. Because of big variety of orientation and sizes of objects, augmentation helps models generalize better across diverse scenarios. Generalization in the context of object detection is the ability of a trained model to accurately detect and classify objects on new, unseen data, and not only on the data it was trained on.

Labeling is another important step in data preparation. Datasets come with already prepared annotations, but when you are creating your own dataset or fine-tuning existing, manual or semi-automated labeling may be required. To mark object locations and assign class labels, tools like LabelImg, VGG Image Annotator, or custom scripts can be used.

Satellite images often come in high resolution and cover large areas, so it is often necessary to divide each image into smaller patches. A common approach is to segment images into fixed-size tiles, for example, 1024x1024 pixels with overlaps of 200 pixels. Overlapping is important to ensure that objects near patch borders are not lost. This

segmentation not only improves memory efficiency during model training but also allows for parallel processing. Such technique improves memory efficiency during training of the model and lets analyzing of images in parallel.

To further enhance efficiency and accuracy, filtering of images is used. Many patches may contain no objects of interest, so there will be unnecessary computational overhead. Therefore, before feeding all patches into the detection model, empty or low-relevance patches are filtered out.

The next step is to design a model capable of detecting objects in segmented image patches. Among popular architectures are Faster R-CNN, Oriented R-CNN, RetinaNet, and YOLOv5, all of which can be adapted to the unique requirements of data to be extracted from the images.

The principle of work of object detection model is that takes the segmented patches as input and outputs bounding boxes along with confidence scores for each detected object. The model must be trained on annotated patches, using a loss function that combines localization and classification objectives. During training, performance metrics such as mean Average Precision (mAP) are used to evaluate the detection quality.

To avoid wasting computational resources on empty patches and to improve detection accuracy, an Objectness Activation Network (OAN) is used. This lightweight neural network acts as a gatekeeper by evaluating whether a given patch contains any objects of interest or not. The OAN is trained on binary labels, distinguishing between object-containing and empty patches.

The OAN processes each patch and outputs an objectness score. Only those patches exceeding a predefined threshold are passed to the main detection model. This selective approach reduces false positives and speeds up the overall system by minimizing unnecessary executions.

After detection of the objects, a classification model is used to assign each object to a specific category, such as cargo aircraft, fighter jet, or helicopter, and determine its model, like MIG-29 and SU-27M.

The classification model typically uses cropped regions corresponding to the detected bounding boxes. These crops are normalized and resized before being input into a convolutional neural network (CNN) such as ResNet, R-CNN, or YOLOv8. The model is trained using labeled crops and optimized using cross-entropy loss.

In cases where a high degree of visual similarity exists between object classes, additional features such as texture, context, or orientation may be incorporated to improve classification accuracy. Data augmentation is very important at this step because it prevents from overfitting.

After initial training, both detection and classification models, developer must manually and with the help of performance metrics check the precision of predictions and fine-tune the models. This involves retraining the models on a smaller dataset that closely matches the target application domain. Fine-tuning helps adapt the model to specific image characteristics, object types, or environmental conditions such as lighting and weather. Also this technique is very useful in cases when new object representations are introduced. It can be new model of an aircraft, or masking technique, which prevents from detecting the airplane.

Fine-tuning often includes adjusting learning rates, unfreezing deeper layers of NN, and performing additional data augmentation. Then, Validation metrics are monitored to prevent overfitting and to identify the optimal training epoch.

CHAPTER 1
PREPARATION OF DATASET FOR TRAINING

1.1. What is Dataset

Datasets are very important in process of creating artificial intelligence models for object detection and classification in satellite imagery because the success of model training depends heavily on the quality and relevance of the dataset used for training. Dataset serves as the foundation upon which machine learning algorithms learn how to recognize patterns, localize objects, and distinguish between different classes. If data is poorly-structured and is not representative, even the most advanced models would fail to perform precisely in real-world scenarios. In terms of satellite imagery, where on the appearance of objects impact factors such as scale, orientation, lighting, and environmental conditions, the role of a comprehensive dataset becomes even more critical. This not only guides the learning process, but also determines how well the model will adapt to new, unknown data. So, a properly selected and prepared dataset is a key component of the entire model development pipeline.

A dataset is a structured collection of labeled data that is used for training and evaluating processes in machine learning models. In terms of object detection and classification in satellite imagery, a dataset typically consists of high-resolution aerial or satellite images with annotations that store information about objects that are presented in those images. Most common information is object's size, location, and class, to which it belongs. Such annotations represented mostly in the form of bounding boxes or polygons associated with specific labels, such as "aircraft", "ship", or "vehicle". The quality, composition, and relevance of the dataset to data, that will

be processed, directly impact the performance and generalization capability of the models trained on it.

The selection of a suitable dataset is very crucial step in the development of AI models. A properly selected dataset ensures that the model will learn features and detect patterns that reflect the actual data it will encounter during operation. On the other hand, poorly-selected data set can lead to models that perform well during development but cannot be generalized to real-world scenarios.

Accordance of dataset with the target application is one of the primary criteria in choosing a dataset. The more images in dataset resemble the type of satellite imagery that will be processed by the final system, the more accurately model will be trained. It consists of factors such as image resolution, geographic coverage, object scale, and the environment in which objects appear. For example, final application involves recognizing military aircraft, so the dataset should ideally contain groups of similar aircrafts captured under similar environmental conditions, but also have cases that are not so common but can describe some objects that cannot be detected based on similar data. It consists of airplanes that are covered with mask, painted, or partially covered with other objects, so are not visible fully. Examples of such cases are presented in Figure 1.1. Datasets that differ significantly in scene content or object characteristics can result in poor model performance due to domain mismatch. So, dataset should cover different cases but for each of them there should be multiple data samples.

Fig. 1.1. Samples of different environmental conditions. Aircraft on the left upper corner is masked, on the lower left is disassembled, on upper right there are many objects, on lower right image quality is poor

Image quality plays essential role in selection of dataset. High-resolution images allow models to more effectively detect fine details and small-scale objects. Low resolution, compression artifacts, or noise can hide critical features and reduce detection accuracy. Parameters, such as lightning conditions, consistency, and clarity are as important, as the nominal resolution of images.

Another critical parameter is the volume of dataset. Model will learn more generalized representations and reduce overfitting if the dataset is large enough and consist of diverse examples. A dataset should contain enough images to capture variations in object appearance, orientation, size, and background. However, the usefulness of a dataset is not solely determined by its size; the quality and diversity of annotations are equally important. Balanced representation of all target classes and inclusion of challenging examples such as occluded or partially visible objects contribute to robust model training.

Diversity of objects within the dataset enhances the model's ability to recognize objects under different scenarios. This includes variations in object types, shapes, colors, and environments. For instance, a model trained only on side-view images of aircraft may fail to detect them from top-down views unless the dataset includes such perspectives. Seasonal, lighting, and weather variations are also valuable for ensuring the model can handle real-world conditions.

Another important consideration is the ability to expand the dataset with custom data. In many practical applications, publicly available datasets may not fully represent the operational environment or specific object types of interest. Therefore, a flexible dataset that can be augmented with user-collected imagery and annotations is preferable. The dataset format should support standard labeling schemes, and the tools used for annotation should be compatible with common training frameworks.

Additionally, the presence of metadata can add value to a dataset. Information such as image timestamps, sensor details, geolocation, and altitude can be leveraged for advanced modeling techniques or for preprocessing steps like image normalization or alignment. While not always required, such metadata can enhance the contextual understanding of the scene and support more sophisticated classification schemes.

Finally, the availability of a clearly defined evaluation protocol is an asset. Datasets that provide predefined training, validation, and test splits, along with evaluation scripts and baseline results, enable fair comparisons and reproducible experiments. This is particularly important in research settings where performance metrics like mean Average Precision (mAP) or F1-score are used to benchmark different approaches.

In summary, the choice of dataset fundamentally influences the outcome of object detection and classification systems in satellite imagery. Key factors to consider include the visual similarity between training and target data, image quality, dataset size, object diversity, compatibility with custom data augmentation, and the presence of useful metadata. A carefully selected and prepared dataset lays the groundwork for successful model development and reliable deployment.

1.2. Choice of Dataset

Among the publicly available datasets, DOTA 2.0, xView and MAR20 can be distinguished. Below is a comparative description of these three datasets.

DOTA 2.0 (Dataset for Object Detection in Aerial Images)

It contains more than 11 thousand large images with 18 classes of objects, including airplanes, ships, vehicles, etc. All images have a high size (from 800×800 to 4000×4000 pixels), which allows for detailed processing of scenes. The marking is done using bounding boxes and oriented polygons, which allows for accurate localization of objects regardless of their orientation.

The advantage of DOTA is a large number of diverse scenes, including urban and rural areas, seaports, airports, etc (see Fig. 1.2). However, despite the presence of the "aircraft" class, this class covers both civil and military aircraft without a clear separation, which makes it difficult to use the dataset in specific task related only to military aircrafts.

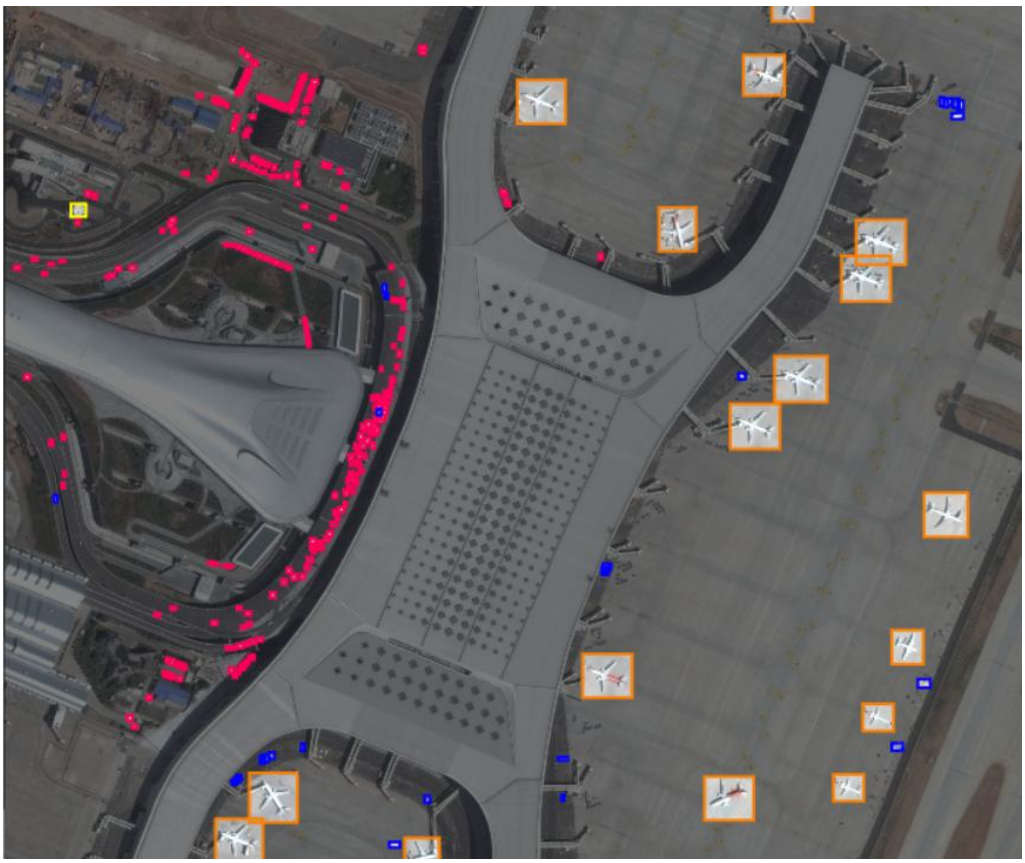


Fig. 1.2. Image sample from DOTA 2.0 Dataset

xView

This dataset is also a major initiative in the field of satellite monitoring, designed for multi-class object detection. It contains over 1 million annotated objects in 60 classes, including airplanes, helicopters, buildings, trucks, and others. The images are georeferenced, allowing for contextual information.

xView is useful for general detection tasks and demonstrates good coverage of object types and geographic distribution (presented on Fig. 1.3). However, as with DOTA, the object classes are too general for highly specialized tasks. The "aircraft" class is presented here without any type or purpose details, so it is not suitable as the basis for a model that should distinguish, for example, fighters from transport aircraft.

Fig. 1.3. Image sample from xView Dataset

MAR20 (Military Aircraft Recognition 2020)

MAR20 is a specialized dataset focused exclusively on military aviation (see Fig. 1.4). It contains a large number of high-quality satellite images with detailed markings of military aircraft. Objects are not only localized, but also classified by aircraft type, which is especially important for a task that combines detection and subsequent classification.

The advantage of MAR20 is its full compliance with the research topic. All objects are military aircraft, the images have sufficient resolution, and the number of examples provides the possibility of full-fledged training. In addition, the structure of the dataset allows its expansion by adding your own images and markings, which allows you to adapt the model to specific conditions.

Fig. 1.4. Image sample from MAR20 Dataset

Another important criterion when choosing a dataset was the annotation format, that is, the way in which objects in the images are described. This affects both the complexity of further data processing and the convenience of expanding the set with your own examples.

In the MAR20 and DOTA 2.0 datasets, metadata about objects is presented in the form of bounding boxes — rectangular bounding frames specified by coordinates. This format is a standard in object detection tasks and is supported by most modern frameworks (for example, YOLO, Faster R-CNN, RetinaNet). Working with bounding boxes is much easier to implement, as it does not require processing complex geometric shapes or vectorizing contours. In addition, adding new objects or creating annotations for your own images takes less time, since for each object you only need to specify the coordinates of its upper left and lower right corners.

Example of annotations, stored in JSON format where each object is in the form of bounding box:


```
{
  "boxes": [
    {
      "label": "A4",
      "x": 347.5,
      "y": 171.5,
      "width": 155,
      "height": 163
    },
    {
      "label": "A4",
      "x": 163,
      "y": 503,
      "width": 140,
      "height": 158
    },
    {
      "label": "A4",
      "x": 311.5,
      "y": 537,
      "width": 139,
      "height": 160
    }
  ]
}
```


In this example, each aircraft is given its class (label), position in the image (x, y), and the dimensions of the bounding box (width, height). This approach makes it easy to read information about the objects, pass it to a neural network, perform visualization, or convert to other formats (e.g., Pascal VOC, COCO, or YOLO).

The main advantage of the bounding box is its minimalist structure, which does not require complex geometric descriptions. This allows you to quickly create, edit, and process annotations even without specialized tools. In cases where the exact outline of the object is not critical — such as when detecting aircraft in satellite images — using rectangles is the optimal choice.

In contrast, in the xView dataset, objects are annotated using polygons. This format allows for a more accurate representation of the shape of an object, especially if it has complex contours or irregular geometry. However, in tasks where the orientation or exact shape of the object is not critical, the use of polygons complicates data preparation and increases the amount of computation. In particular, generating new

annotations in the form of polygons requires significantly more time and effort, especially when manually labeling.

Example of annotations, presented in form of polygons:


```
{
  "objects": [
    {
      "label": "A4",
      "polygon": [
        [340, 160],
        [400, 150],
        [450, 200],
        [430, 260],
        [370, 270],
        [330, 220]
      ]
    },
    {
      "label": "A4",
      "polygon": [
        [150, 490],
        [210, 480],
        [250, 530],
        [230, 580],
        [170, 590],
        [130, 540]
      ]
    }
  ]
}
```


Among the three datasets considered, the best one for the task of detecting and classifying military aircraft is MAR20. It provides high relevance to the target problem, contains high-quality and detailed object labeling, and allows for further expansion and adaptation. While DOTA 2.0 and xView are powerful general datasets, their lack of specialization for military aircraft classes limits their use as a basis for this specific task. Thus, MAR20 can serve as the main source of data for training, as well as be supplemented with our own images to increase the accuracy and generalizability of the model.

1.3. Dataset Enrichment

To improve recognition accuracy, one of the key steps is to expand the existing dataset by adding new images. Even if the base dataset is of high quality and thematically relevant, as in the case of MAR20, the real conditions of the task may

differ from those presented in the original set. Therefore, supplementing the dataset with new examples allows you to adapt the model to the specifics of the target images and significantly improve its generalization ability.

New images can come from open sources, such as satellite services (for example, Google Maps or Bing Maps), where a large number of detailed aerial photographs are available. They can contain military aviation objects located at airfields, military bases or in open areas. Closed or internal sources can also be used, for example, images from corporate satellites, drones or archives of organizations that have access to specialized materials. In such cases, it is important to comply with confidentiality and copyright rules when collecting and processing data.

New images should contain objects that are as similar as possible to those that the model will recognize in the future. This applies to both the type of objects themselves (for example, a certain model of a fighter or transport aircraft), and the shooting conditions - resolution, scale, viewing angle, lighting, etc.

The process of expanding the dataset involves not only collecting new images, but also marking them in a format compatible with the base set. If MAR20 uses bounding boxes, then new examples must be annotated in the same format to maintain the uniformity of the data structure. This stage can be performed manually using marking tools, or partially automated, using already trained models for preliminary detection of objects with subsequent refinement of the results.

Thus, expanding the dataset with new images is a necessary condition for achieving high recognition accuracy. It allows you to take into account real operating conditions, ensure that input data meets the model's expectations, and significantly increase its reliability in practical application.

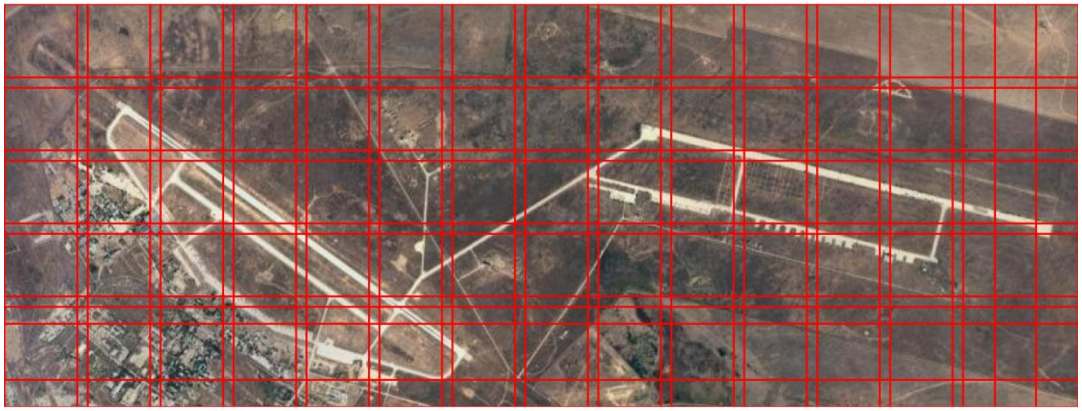
Satellite images usually have extremely high resolution — for example, 30,000 × 30,000 pixels or even more. Such large images cover a significant area, but their direct use in neural networks is impractical, since most modern models are not designed to process images of this size due to memory and resource limitations.

Therefore, it is necessary to perform a preliminary segmentation of large images into smaller blocks that can already be used for training models. In this work, a block

size of 1024 × 1024 pixels was chosen. This size is a balanced option that allows preserving the details of objects in the images and at the same time is compatible with most standard deep learning architectures.

However, when simply dividing a large image into uniform squares, the problem of "borderline" objects arises. These are objects that are located on the boundaries between two adjacent blocks - in such cases there is a risk that the object will fall into two blocks partially, or will be clipped so much that the model will not be able to recognize it. To avoid this problem, the segmentation process uses a technique that involves overlapping between blocks (see Fig, 1.5). In particular, each block overlaps with the neighboring ones by 200 pixels horizontally and vertically. This ensures that objects located near the boundaries completely fall into at least one of the blocks. This achieves more complete coverage and improves the quality of future annotation and model training.

a) Original image



b) Segmented image

Fig. 1.5. Example of image segmentation

1.4. Data Augmentation

Even after expanding the dataset by adding new images, its volume may still be insufficient for full-fledged training of a deep neural network. In such cases, the process of data augmentation is used - artificially increasing the amount of data by modifying existing images. This is one of the key stages in data preparation, which allows to significantly increase the generalization ability of the model, reduce the risk of overtraining and make the model more resistant to changes in observation conditions.

The essence of data augmentation is to create new variants of existing images, while maintaining the correctness of the markup. The main methods of such transformation include [6]:

The Cutout technique consists of randomly closing a separate area in an image to artificially complicate object recognition during model training (example on Fig. 1.6).

Fig. 1.6. On the left – original image, on the right – with cutout applied

The Mosaic augmentation technique involves combining multiple fragments from different images into one (Fig. 1.7). The result is a new image that is a mosaic of areas from several different sources.

Fig. 1.7. Mosaic augmentation applied

In the **random blur augmentation** method, training images are randomly blurred using a convolution with a specific blur kernel (Fig. 1.8). This blurring allows the model to learn to recognize objects even in reduced quality, which contributes to better generalization to new, previously unknown data.

Fig. 1.8. Random blur applied

In the **random rotation augmentation** method (see Fig. 1.9), the training images are randomly rotated by a certain angle, usually within a given range, such as -10 to 10 degrees. This rotation allows the model to see objects in different orientations. Since objects can have any orientation in satellite images, it is desirable to perform rotations in a wider range, from -180 to 180 degrees.

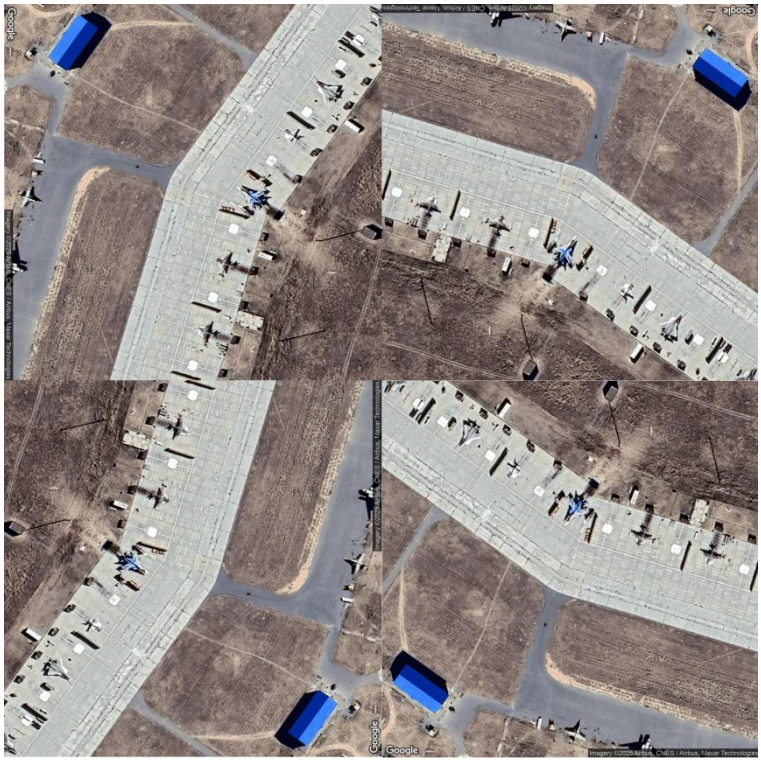


Fig. 1.9. Random rotation applied

Random Noise augmentation involves adding speckle-like noise to randomly selected pixels in training images (see Fig. 1.10). Similar to the blurring method, this processing allows the model to be trained on images with degraded quality, which contributes to the formation of a more generalized and robust model.

Fig. 1.10. Random noise applied

Image flipping is a simple but effective data augmentation technique that inverts an image about either the vertical or horizontal axis (as presented in Fig. 1.11). In the case of horizontal flipping, the left and right sides of the image are swapped, and in the case of vertical flipping, the top and bottom are swapped. This transformation allows the model to learn to recognize objects regardless of their symmetry or position in the frame.

This technique is particularly useful for satellite imagery, as objects such as airplanes can occur in any orientation. Flipping increases the diversity of the training set without changing the content of the objects, improving the model's generalization ability and reducing the risk of overtraining.

Fig. 1.11. Horizontal flipping applied

1.5. Data Labeling

After completing the stages of preparation and expansion of the dataset by adding new images and applying data augmentation methods, the next critically important step in building an object recognition system is to create annotations for each image. This process is a mandatory component of effective model training, since it is from the annotations that the model receives information about which objects should be detected and classified, where they are located in the image and which class they belong to.

Annotations are a source of “knowledge” that the model formalizes during training. If the annotation is created incorrectly, with an error in localization or classification, this directly affects the quality of training — the model either does not recognize the object, or mistakenly assigns it to another category. Therefore, accuracy, precision and consistency when creating annotations are crucial for achieving high model accuracy in real-world conditions.

In this qualification work, the objects to be detected and classified are military aircraft models — airplanes, helicopters, and other aircraft that are present in satellite images of airfields, bases, or field conditions. The variety of aircraft models, their orientation, size, color, and contextual environment complicate the task for both the model and the marker, so high-quality direction recognition depends on the ability to accurately identify and label objects at the annotation stage.

To ensure maximum consistency, the annotation format used in the MAR20 dataset was chosen. This format is simple, convenient, and widely supported by modern frameworks for training deep learning models.

Fig. 1.12. Bounding box of an object

After the bounding box creation stage is completed (Figure 1.12), corresponding annotations in JSON format are generated for the objects in the image. This file stores both general information about the image and data about each object that was selected on it. In particular, the JSON structure specifies the dimensions of the image — its width and height, which allows for accurate scaling of the coordinates during further data processing. Next, the array of objects describes each unit of detected equipment. For each of them, the class (i.e., aircraft model), the coordinates of the center (x, y), as

well as the width and height of the rectangle that encompasses the corresponding object are specified. And the resulting JSON file with annotations for the image is the following:


```
{
  "boxes": [
    {
      "label": "TU22M3",
      "x": 202.2,
      "y": 364,
      "width": 158.4,
      "height": 168.5
    }
  ],
  "height": 800,
  "key": "2485.jpg",
  "width": 800
}
```

CHAPTER 2
CREATION OF AI MODELS FOR OBJECT DETECTION AND CLASSIFICATION

2.1. Overview of available models

Object recognition and classification in images are two separate but closely related tasks in computer vision, which underlie most automatic image analysis systems. In the context of satellite imagery, particularly when working with military aircraft and helicopters, both processes play a key role: the first is responsible for localizing objects in the image, the second for determining their type.

Recognition (detection) is the process by which an artificial intelligence system determines where exactly in the image the objects of interest are located. It outputs the coordinates for each object found in the form of a rectangular area (bounding box) that surrounds it. In the case of satellite imagery, detection allows you to determine where exactly in the large area of the image potential targets are located - airplanes, helicopters or other equipment. This task is extremely complex due to the variety of angles, object orientations, background obstacles and differences in image resolution.

Classification is the next step after an object has been found. The AI model analyzes the content of each selected image fragment and tries to determine which class the object belongs to. For example, it must distinguish an F-16 fighter from an A-10 attack aircraft or an Apache helicopter from an Mi-24 helicopter. Classification is based on studying the characteristic features of each type of equipment, such as wing shape, fuselage size, number of engines, characteristic protrusions, etc.

Artificial intelligence models solve these problems by using deep convolutional neural networks (CNN), which are able to automatically select and analyze visual features of images. For recognition tasks, architectures such as Faster R-CNN, YOLO, SSD are usually used, which combine localization and primary classification functions. They are able to simultaneously scan the image, detect objects in it, and immediately pre-indicate their class.

However, in this qualification work, the detection and classification processes are separated to achieve higher accuracy. First, a model is used to detect all potential aircraft objects in the image, after which each selected object is passed as input to a separate classification model, which more accurately determines its type. This two-stage approach allows for better performance, especially in cases where the classes are similar to each other or require more complex analysis of details.

Neural networks for object detection and recognition are divided into two main types by architecture - one-stage and two-stage. This division is based on how the network processes the image to detect objects and determine their classes.

One-stage models perform object recognition and classification simultaneously, within a single pass through the neural network. They scan the entire image, automatically determine the probable areas where objects can be located, and simultaneously assign appropriate classes to these objects. The most famous representatives of this approach are YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector). The main advantage of such models is high speed. Due to their simplicity, they are well suited for tasks where real-time processing is critical, for example, in monitoring systems or on drones.

However, these networks also have disadvantages. Single-stage architectures typically suffer from poor accuracy compared to two-stage ones, especially when working with complex images or when recognizing small, partially overlapping, or similar objects. In addition, it is more difficult to achieve high adaptation to specific types of objects in single-stage approaches, which limits their application in specialized tasks.

Two-stage models work sequentially in two steps. First, they select regions with potential objects (this process is called region proposal generation), and then analyze each of these regions separately, determining the exact location of the object and its class. One of the most popular architectures is Faster R-CNN, which combines a regional proposal network (RPN) and a classifier. The main advantage of the two-stage approach is high accuracy, especially when working with detailed or densely located objects. This makes it the best choice for scientific and military tasks, where maximum reliability of the results is important.

The disadvantage is higher computational complexity and correspondingly longer processing time. Such models are more difficult to implement, require more resources and time for training, but in return provide better quality results.

In this study, the most important criterion is to achieve maximum accuracy in object recognition in satellite images. Due to the high similarity between different aircraft models, significant variability in angles, scales and shooting conditions, it was decided to use a two-stage neural network architecture. It is models such as Faster R-CNN that provide better quality of localization and classification of objects, due to separate processing of each of them.

2.2. Choice of backbone for object detection neural network

To build modern object recognition systems, it is not at all necessary to create model architecture "from scratch". In the field of computer vision, there are already a large number of pre-trained models that can be adapted to specific tasks. These models have been trained on large publicly available datasets, such as ImageNet or COCO, and contain an already formed idea of the main visual features characteristic of various types of objects. This approach allows you to significantly reduce training time and increase efficiency in cases where your own dataset is relatively small or specialized.

The main element of each neural network that performs image processing is the so-called backbone - a basic convolutional network that is responsible for extracting

features from the input image [15]. There are several popular models that are commonly used in this role. Among them:

VGG (Visual Geometry Group) - a simple in structure, but a three-dimensional model that requires a lot of computing resources

MobileNet - a lightweight and fast model focused on mobile devices, but with reduced accuracy

EfficientNet - an architecture that combines high accuracy and good speed optimization

ResNet (Residual Network) - a deep neural network with residual connections, which has shown high efficiency on many tasks

In this study, ResNet was chosen as the backbone because it provides the optimal balance between accuracy and performance. Thanks to residual connections, ResNet avoids the problem of gradient decay, which allows you to build deeper and more powerful models without losing efficiency. In addition, this architecture has proven itself well in combination with two-stage recognition models, in particular Faster R-CNN.

The selected ResNet model will be used as the basis for feature extraction from the image, after which these features will be passed to a regional propositional network (RPN) for object detection, and subsequently to a classification module to determine the object type.

The choice of ResNet as the main architecture (backbone) in this study is based not only on its popularity, but also on the results of numerous scientific publications and tests that demonstrate its high efficiency in object recognition tasks, in particular on satellite images. ResNet is one of the most stable and best adapted architectures for deep learning tasks with high model depth, which does not suffer from the problem of gradient decay due to residual connections.

In various comparative studies conducted on datasets (for example, DOTA and xView), ResNet showed consistently high accuracy results with a reasonable ratio of training time and number of parameters. It turned out to be especially effective in combination with the Faster R-CNN architecture.

Table 2.1

Comparison of the characteristics of the most popular models used as a backbone in computer vision tasks

Architecture	**Accuracy (Top-1, ImageNet)**	**Number of parameters**	**Advantages**	**Disadvantages**
VGG-16	~71.5%	~138 million	Simple structure	High memory consumption
ResNet-50	~76.0%	~25 million	Good depth without losing accuracy	Greater complexity of implementation
MobileNetV2	~71.8%	~3.4 million	Fast, lightweight for mobile devices	Reduced accuracy
EfficientNet-B0	~77.1%	~5.3 million	Good balance between accuracy and weight	Scaling difficulty
DenseNet-121	~74.9%	~8 million	High quality of features	Increasing costs with depth

Comparing to other popular architectures (see Table 2.1) such as VGG-16, MobileNetV2, EfficientNet-B0 and DenseNet-121, ResNet-50 provides better accuracy at moderate model complexity. This makes it particularly attractive for tasks where both accuracy and computational efficiency are important. In addition, ResNet-50 is widely used as the basis for two-stage models such as Faster R-CNN, which confirms its suitability for the task of object detection and classification in high-resolution images.

Therefore, considering the above advantages, ResNet-50 is selected as the base architecture for building a model for recognition of military aircraft in satellite images in this study.

2.3. Principle of work of Faster R-CNN

The basic principle of Faster R-CNN is to step by step pass the image through several blocks, each of which performs a separate task in the recognition process. If ResNet is used as the backbone, the Faster R-CNN architecture includes the following stages:

1. Feature Extraction:

The first stage of the Faster R-CNN model is feature extraction from the input image. This process is performed using a convolutional neural network (backbone). The goal of this stage is to transform the original image into a set of features that contain compressed but meaningful information about the structure, contours, texture, and other visual characteristics of objects in the image. The resulting feature map is the basis for further detection and classification of objects.

The ResNet architecture consists of several layers (blocks), which include normalization, convolutional layers, and subsampling layers (max pooling layers).

Data normalization is a mandatory preprocessing step for input data before it is fed into a neural network. It involves bringing the data to a specific numerical range or statistical distribution, which facilitates the learning process, improves model convergence, and avoids numerical instability in calculations. The main goal of normalization is to reduce variations in the scale of input feature values. If the data have different ranges (for example, one feature is in the range 0-1, and the other is 0-10000), this can lead to the model favoring stronger numerical signals, neglecting smaller, but important features. In addition, with a large spread of values, the network converges more slowly and less stably, which complicates the training process.

Input images typically consist of pixels, each with an intensity value ranging from 0 to 255 (for images with a depth of 8 bits per channel). However, such values are not suitable for direct input to deep models, since their scale is significantly different from the values that the internal mechanisms of the neural network work with. This can slow down the learning process and worsen convergence. To solve this problem, Min-Max normalization is widely used - a method that brings pixel values to a unified range,

usually [0, 1]. This allows you to level the scale of features and facilitate further processing. Formula for min-max normalization is:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}},$$

where x - value of specific pixel on the image;

$x_{min} = 0$;

$x_{max} = 255$.

So, formula can be simplified to:

$$x' = \frac{x}{255}.$$

The next block is convolutional layer. It is the main building block of convolutional neural networks. It performs the convolution operation between the input image (or an intermediate feature map) and the convolution kernel (filter or kernel). The kernel goes through the entire image, multiplying its values by the corresponding pixels of the image, and creates a new map that captures certain characteristics, such as horizontal or vertical contours, texture, angles, etc. Each filter is automatically trained and is responsible for a certain type of feature. Main types of convolution layers are:

• Standard convolution

This is the classic type of convolution that is most often used. Each kernel filters the entire depth of the input tensor, for example, all RGB channels, and creates a single feature map (example of convolutional layers presented in Fig. 2.1). The result is a set of new channels, each of which is responsible for a specific visual feature.

In convolutional neural networks, filters are the main tool for extracting features from an image. Each filter is a small matrix of weights (e.g., 3×3, 5×5) that is passed over the image and at each step performs a dot product with the pixel values in the corresponding region. The result is a new feature map that “highlights” those areas of the image where a certain visual characteristic is detected.

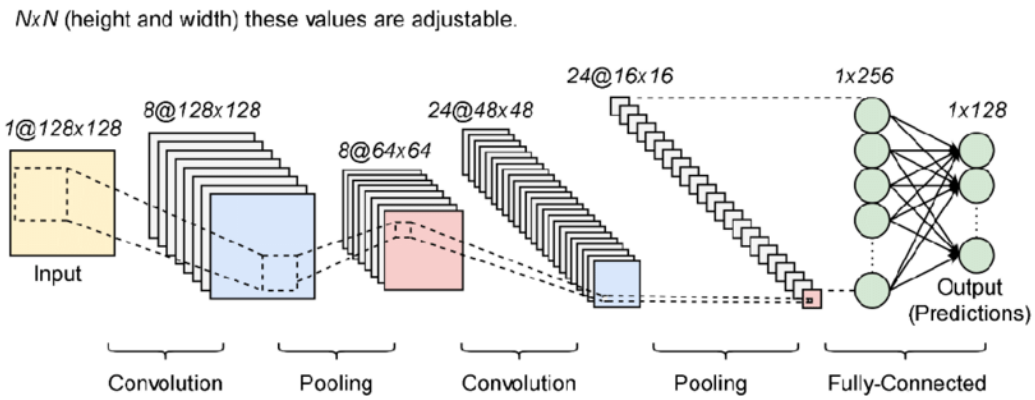


Fig. 2.1. Example of standard convolutional layer. Here, input image of size 128x128 is taken and 8 conv layers with different filters are applied on the first stage of processing

Filters in the convolutional layers of a neural network allow you to detect features in images, such as edges, corners, textures, or shapes (see Fig. 2.2). Thanks to them, the network learns to recognize characteristic elements of objects, which is critical for accurate detection and classification.

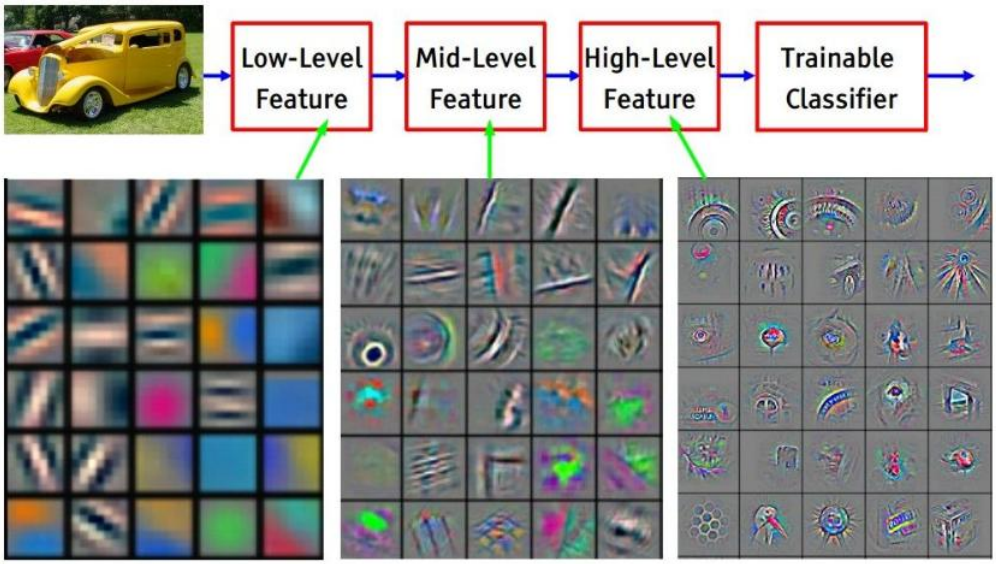


Fig. 2.2. Different filters

• Strided convolution:

Instead of the filter moving across the image with a step of 1, in this case the step is larger - for example 2 or 3. This allows you to quickly reduce the size of the feature map (i.e., perform "compression"), similar to pooling, but while preserving important features. Graphical representation of such convolution is in Fig. 2.3.

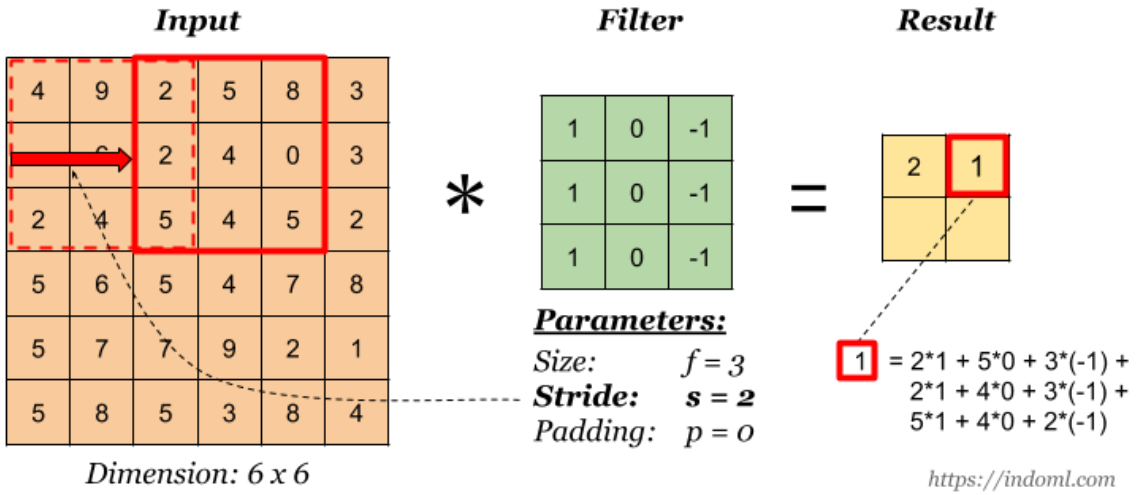


Fig. 2.3. Example of strided convolution

• Zero-padding:

In order not to lose information at the edges of the image during convolution, additional "empty" pixels are added along the contour (example Fig. 2.4). This allows you to keep the size of the input and output feature maps unchanged.

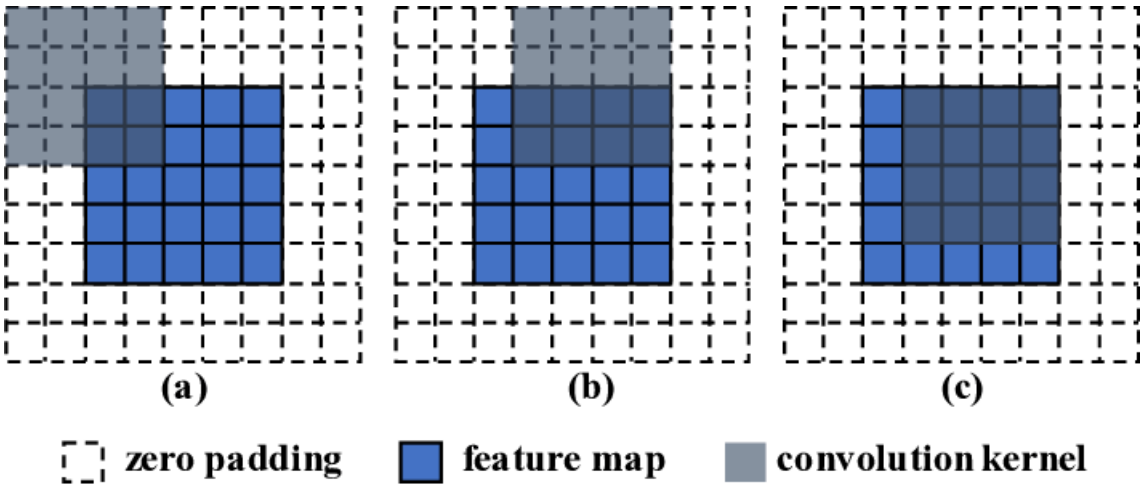


Fig. 2.4. Example of zero-padding convolution

• Dilated (dilated or atrous) convolution:

The kernel of the convolution has gaps between the elements (like presented in Fig. 2.5), allowing for an increase in the receptive field without increasing the number of parameters. Used for image analysis, where information on larger scales is important.

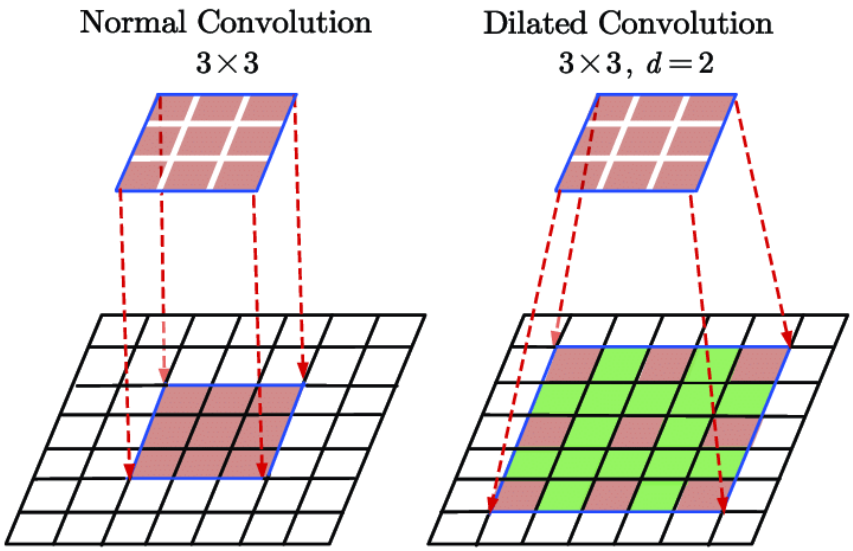


Fig. 2.5. Example of dilated convolution

Max pooling is an operation that is performed after several convolutional layers in order to reduce the dimensionality of the feature map. For each block of pixels (for example, 2×2), the maximum value is selected (like in Fig. 2.6). This operation allows you to reduce computational costs, increase the generalization ability of the network, and preserve the most important information. At the same time, the image is compressed, and small details may be lost, but the overall structure is preserved.

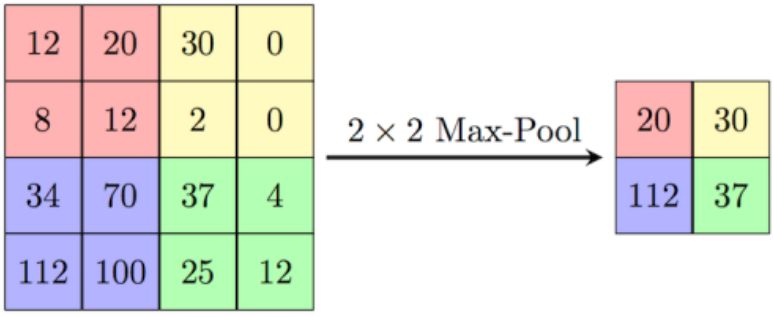


Fig. 2.6. Example of max pooling

After the image passes through several convolutional and max pooling layers, the data may have different scales of values. This makes it difficult to further train the neural network, since each subsequent layer receives input with a different distribution of values, which slows down and destabilizes training. To solve this problem, deep neural networks use Batch Normalization, a technique that allows you to stabilize and speed up the learning process.

Batch Normalization performs a smoothing of the distribution of the output values of each layer for each mini-batch, bringing them to a mean close to zero and a

variance close to one. These values are then scaled and shifted using two parameters that are trained together with the rest of the model. Thus, the network retains the ability to adapt to the specifics of the input data, but without the abrupt changes that can occur without normalization.

In ResNet, batch normalization is applied after each convolution and before activation function. This approach allows not only to stabilize the data distribution, but also to improve the generalization ability of the model.

The next stage in image processing is neural layer. The neural layer is the basic structural element of an artificial neural network. It consists of a set of artificial neurons — mathematical models that simulate the work of biological neurons. Each neuron takes numerical values as input, weights them using weighting coefficients, sums them, and passes them through an activation function, which forms the final output of the neuron.

The principle of operation of one neuron can be described by the formula:

$$y = \varphi(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b),$$

where $x_1, x_2, \ldots, x_n$ - input values (e.g. image pixels or outputs from the previous layer);

$w_1, w_2, \ldots, w_n$ - weight coefficients;

b – bias;

φ - activation function;

y - output of neuron.

The role of the activation function is to introduce nonlinearity into the operation of the neural layer. If the model did not use activation functions, then regardless of the number of layers, it would remain a linear transformation of the input. This would mean that complex dependencies between the input and output data could not be modeled.

The nonlinearity provided by activation allows the network to "learn" to recognize patterns, features and contexts specific to a particular task - for example, to detect the contours of aircraft in satellite images, to distinguish between models of equipment, or to classify objects by shape, size and color.

In convolutional neural networks used for image processing, an activation function is usually added after each convolutional layer or after a group of convolutional layers. This allows the selected features (features) obtained as a result of convolution to pass through a "filter" of nonlinearity, which increases the depth of image analysis.

There are several types of activation function:

• Sigmoid

The sigmoid function is one of the first activation functions used in artificial neural networks. It is defined as:

$$f(x) = \frac{1}{1+e^{-x}}.$$

The output of this function is always in the range 0 to 1, which is convenient for tasks where you need to interpret the result as a probability. It is well suited for the output layer in binary classification tasks, but for large or small values, the derivative of the function becomes very small, which slows down the learning process.

• Tanh

The $tanh(x)$ function is similar to the sigmoid, but its output lies in the range from -1 to 1:

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}.$$

This function preserves the properties of the sigmoid, but has a zero mean, which improves the convergence of the learning. Also, like the sigmoid, it suffers from gradient decay.

• ReLu (Rectified Linear Unit)

The most popular function today, which is defined as:

$$f(x) = max(0, x).$$

This is a very simple and efficient function that returns 0 for all negative values and the value x itself for all positive values. Among its advantages are fast computation and elimination of the problem of gradient decay for positive values. The disadvantage is the problem of "dead neurons": if the weights are set so that the input is always less than zero, the gradient is zero and the neuron stops learning.

• Softmax

This is a function that converts a vector of values into a vector of probabilities. It is used in the output layer of a multiclass classification:

$$softmax(x_i) = \frac{e^{x_i}}{\sum e^{x_j}}.$$

This function gives an interpreted result in the form of probabilities for each class, but can only be used on the last layer.

In this work, it is most appropriate to use ReLU, since this function: is optimal for deep convolutional networks, provides high learning speed, and does not complicate the model architecture.

In addition, ResNet uses residual connections, which allow you to avoid the problem of gradient vanishing in very deep networks. A residual connection is an architectural element that allows the input data of a layer to be passed unchanged, adding it to the output of several subsequent layers. This design looks like a short connection that “jumps” through one or more layers. The main idea is that the neural network learns not on the transformation function itself, but on the difference between the input and the desired output. This greatly facilitates the optimization process, especially in deep networks, where the problem of gradient decay often arises. Thanks to residual connections, the ResNet architecture can have hundreds of layers and at the same time maintain stability and training efficiency.

As a result of passing the image through ResNet, a feature map is formed - a tensor in which each element is responsible for the presence of a certain feature in the corresponding part of the image. This map is passed to the next stage - region of interest generation (RPN), where potential objects in the image are searched.

2. Region Proposal Network (RPN):

A separate neural network, the regional proposal network (RPN), works on the basis of the feature map. It passes through the entire feature map and generates a set of regions of interest (region proposals) - that is, rectangles in which objects are likely to be located. RPN evaluates whether each proposed region contains an object and determines the approximate coordinates of these regions.

At each position of the feature map, several rectangles of different sizes and aspect ratios are placed - these are the so-called "anchors". They cover the image with possible variants of object shapes (see Fig. 2.7). For each anchor, the RPN determines the probability that there is an object inside it. This is called the objectness score. If the probability is high, the region is considered promising. In parallel with the objectness assessment, the network also refines the anchor coordinates - that is, it slightly shifts it and scales it so that it better fits the object in the image.

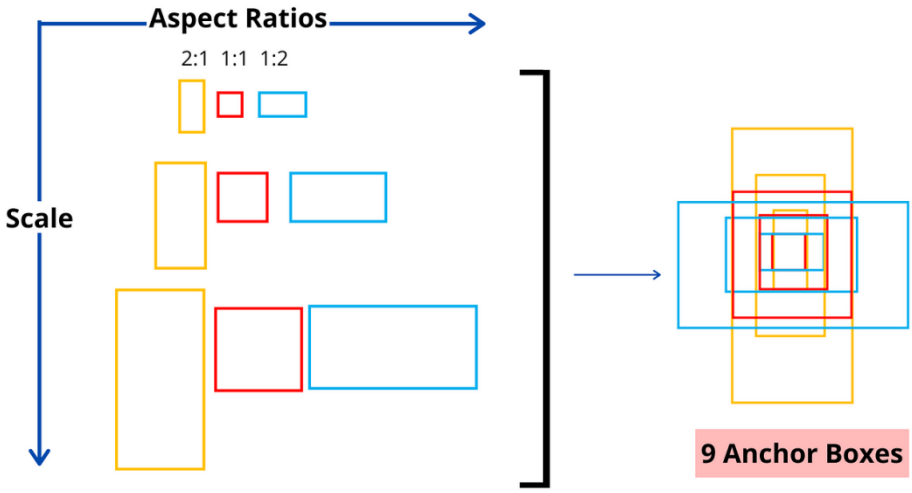


Fig. 2.7. How anchors are constructed

In the process of generating regions of interest in the Faster R-CNN model, the Intersection over Union (IoU) metric plays an important role. This indicator is used to assess how well a rectangle (anchor or predicted region) matches the real position of the object in the image. IoU is calculated as the ratio of the area of intersection of two rectangles to the area of their union (graphically it can be shown as in Fig. 2.8). One rectangle is the proposed region, the other is the ground truth (the real position of the object, given in the annotations).

$$IoU = \frac{Intersection}{Union}.$$

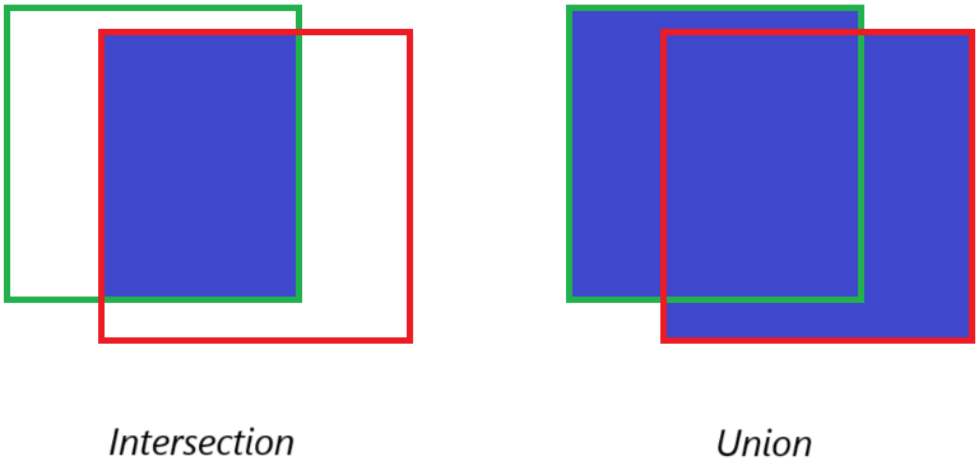


Fig. 2.8. Calculation of IoU

The IoU value ranges from 0 to 1. The closer the value is to 1, the more accurately the region matches the real object. In the Region Proposal Network (RPN) training process, the IoU metric is used to:

• Identify positive anchors: If the IoU between an anchor and the real object exceeds a certain threshold (e.g., 0.7), it is considered positive and is used as an object example for training.

• Identify negative anchors: If the IoU is less than a certain lower threshold (e.g., 0.3), the anchor is considered negative and is used as a background example.

• Ignore ambiguous examples: Anchors with IoU between these two thresholds are often excluded from training to avoid introducing ambiguity.

Another important metric in the object recognition process is the confidence score. This is a numerical value that indicates how confident the model is that an object of a certain class is actually within the predicted rectangle. Its value usually ranges from 0 to 1. The higher this score, the more confident the model is.

At the RPN stage, the model forms preliminary regions of interest (anchors) and for each estimates whether there is an object there (regardless of class). This prediction is called the objectness score, and it is a preliminary estimate of the presence of an object. At the ROI classification level (the next stage), the ROI regions formed from features pass through fully connected layers that classify the object within each region (choose a class from all possible ones) and predict the refined coordinates of the object (BBox regression). In the classification process, the last layer is usually Softmax, which

gives the probability of the object belonging to each class. The highest value among them is the confidence score for this class. For example, if softmax gives [0.1, 0.3, 0.6] for three classes, then the model believes that with a probability of 0.6 in this rectangle there is an object of the third class.

Some implementations also take into account the objectness score from RPN — in which case the final confidence score may be the product of both values:

$$confidence = objectness\ score * class\ score.$$

3. ROI classification & bounding box regression:

Each region of interest undergoes a Pooling process (more precisely, ROI Align), due to which its dimensions are unified [13]. Then these fragments are again passed through a convolutional network, after which they are classified into a specific object class (for example, aircraft type) or as a background. At the same time, the model refines the bounding box coordinates for each object, ensuring accurate localization.

For further image processing, each of these regions of interest need to be aligned in size so that they can be fed to the input of a fully connected network for classification and coordinate regression. At this stage, the Faster R-CNN architecture uses the ROI Align approach. Its task is to accurately "cut" from the feature map those fragments that correspond to the regions of interest and bring them to a single fixed size (for example, 7x7 or 14x14), while maintaining spatial accuracy. This is critical for cases where the objects in the image are small in size or when positioning accuracy is key - as in the case of military aircraft recognition.

Unlike the previous ROI Pooling method, which used coarse rounding of coordinates and caused the loss of spatial information, ROI Align allows you to get a more accurate representation of each region. This is achieved by interpolating the values in the feature map without rounding the boundaries. As a result, the model receives more correct and complete information about each region, which directly affects the quality of classification and the accuracy of object localization.

After ROI Align has extracted fixed regions from the feature map, the model moves on to the stage of accurately determining the position of the object in the image. This stage is called Bounding Box Regression [13]. The main goal of this process is to

refine the coordinates of the proposed region of interest. The regions formed by the RPN network are only initial guesses, and although they are close to the real position of the object, their coordinates may be inaccurate. BBox regression allows the model to learn to calculate corrective offsets - that is, how much the center of the rectangle needs to be shifted and its width and height changed so that it more accurately covers the object. This process is implemented as a regression problem - the neural network assumes four numerical values: x-offset, y-offset, change in width and change in height. Formally, this looks like calculating deltas:

$\Delta x \; and \; \Delta y$ — center offset

$\Delta w \; and \; \Delta h$ — width and height scaling

During training, the model optimizes these parameters by comparing the predicted coordinates with the real (ground truth) coordinates using a loss function such as L1 Loss or Smooth L1 Loss. As a result, after regression, a more accurate bounding box position is obtained, which allows for more correct localization of objects in the image.

4. Post-processing (NMS and Confidence thresholding):

At the final stage, non-maximum suppression (NMS) is applied, which removes duplicate detections of the same object. Only the most probable and accurate predictions that meet the conditions of the confidence threshold remain.

After the model classifies each region of interest and performs bounding box regression, it generates a list of bounding boxes, each of which has the coordinates of the box (x, y, width, height), the object class, and a confidence score that the region contains an object. The problem is that for the same object, the model can produce dozens of bounding boxes with very similar, but not identical, coordinates and scores. This is where Non-Maximum Suppression comes into play to keep only one, best box.

The NMS process consists of the following steps:

1) For each object class, all predicted boxes are selected.

2) All these boxes are sorted in decreasing order of their confidence values.

3) The top box in the list (the one with the highest confidence score) is designated as the final box.

4) Next, the Intersection over Union (IoU) of this box with all the others is calculated: if the degree of overlap exceeds a certain threshold (e.g. 0.5), these other boxes are discarded.

5) The procedure is repeated for the remaining boxes in the list.

Thus, NMS allows us to retain one most likely rectangle for each object, removing unnecessary overlaps. This method significantly improves the quality of the final result and simplifies further analysis [13].

Full sctructure of Faster R-CNN is shown in Fig. 2.9.

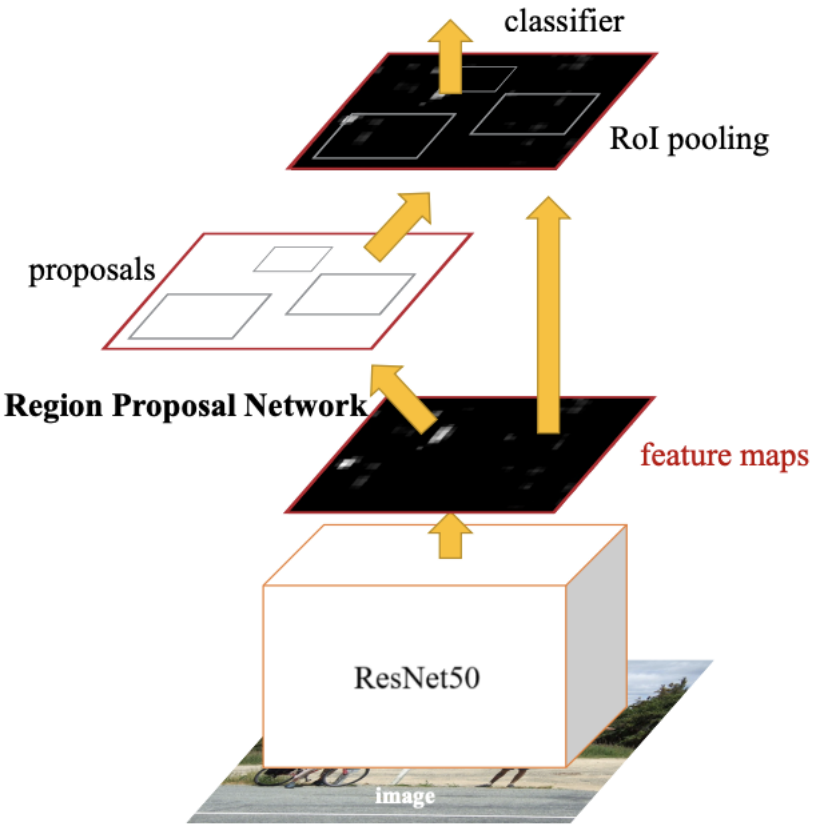


Fig. 2.9. Structure of Faster R-CNN

2.4. Choice of NN model for object classification

After completing the development stage of the object recognition model, the task arose of choosing an effective architecture for image classification, in particular, individual objects that had already been detected by the Faster R-CNN model. The main requirement for the classifier model is high accuracy when working with small image

fragments corresponding to individual aircraft in satellite photos. These fragments may contain little visual information, be distorted due to atmospheric conditions, shadows, or scaling artifacts, so the classification model should demonstrate high resistance to such factors.

Among a number of modern architectures that are actively used for image classification tasks, such as VGG, ResNet, MobileNet, DenseNet, and EfficientNet, special attention was paid to the last one — EfficientNet. The choice was due to the combination of several key advantages of this architecture, which best met the needs of the project.

EfficientNet is the result of searching for the optimal structure of a neural network using AutoML (automated machine learning). The basic version of the model — EfficientNet-B0 — was formed as a compromise between accuracy, size and processing speed. The main feature of this architecture is the use of the compound scaling mechanism, which allows for a balanced expansion of three key network parameters: depth, width and resolution of the input image [10]. Unlike traditional approaches, where scaling occurs only in one of the parameters, EfficientNet offers a balanced change in all three components using a single scaling factor φ.

This approach is described by the following equations:

$$d = \alpha\hat{}\varphi,$$
$$w = \beta\hat{}\varphi,$$
$$r = \gamma\hat{}\varphi,$$

where d - the depth of the model;

w - the width;

r - the image resolution.

The constants α, β, γ are chosen so that with each increase in φ the computational complexity doubles, i.e. the condition is satisfied:

$$\alpha * \beta^2 * \gamma^2 \approx 2.$$

This strategy allows the architecture to be scaled efficiently depending on available resources. As a result, several variants of EfficientNet were built, from B0 to B7, with each successive version being more complex and accurate, but requiring more

computational power. Another feature of EfficientNet is the use of modified MBConv (Mobile Inverted Bottleneck Convolution) blocks, which also include the Squeeze-and-Excitation mechanism [10]. These components allow the network to focus on the most important spatial features of the image without losing the global context.

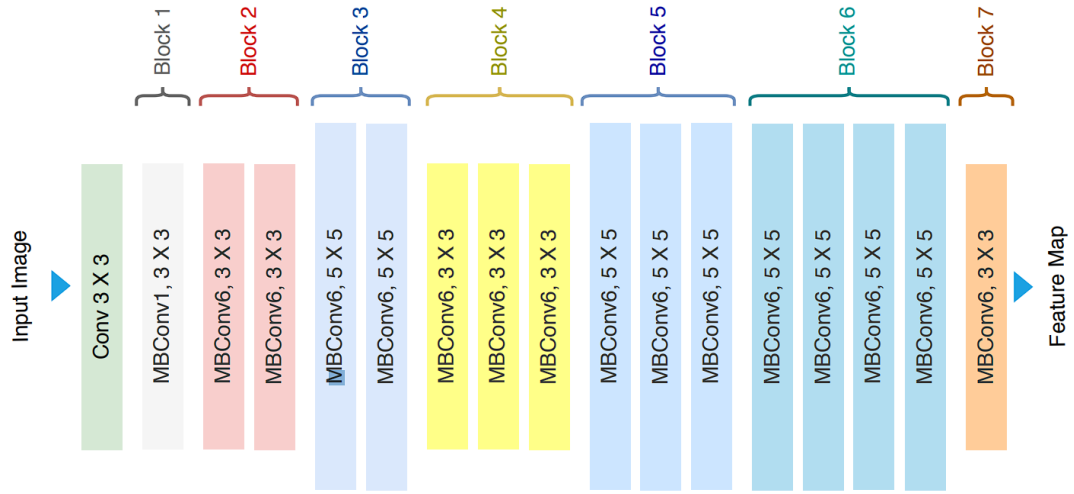


Fig. 2.10. EfficientNet architecture

Structure of EfficientNet (see Fig. 2.10) consists of such blocks:

• Conv 3x3. The initial convolution layer with a 3×3 kernel that is responsible for the initial extraction of low-level features (e.g., edges, contours) from the input image.

• Mobile Inverted Bottleneck Convolution Blocks. All subsequent layers are variations of MBConv blocks, combining deep convolution, an inverted bottleneck structure, and a Squeeze-and-Excitation module to focus attention on the most important channels. Each MBConv has the form:

$$\text{Expand} -> \text{Depthwise Conv} -> \text{Squeeze \& Excitation} -> \text{Project}$$

• Feature map. The final stage is to obtain a generalized representation of the image, which will be used for classification or other purposes (e.g., passing to a classifier or detector).

The EfficientNet model uses the Swish activation function to obtain the output feature map. This is a modern nonlinear activation function that has the form:

$$Swish(x) = x \cdot \sigma(x) = \frac{x}{1+e^{-x}}.$$

Swish combines the properties of linear and nonlinear functions, providing smooth transitions between values and preserving information even in negative ranges

(as seen in Fig. 2.11). Unlike ReLU, which trims all negative values to zero, Swish allows the model to better learn complex relationships.

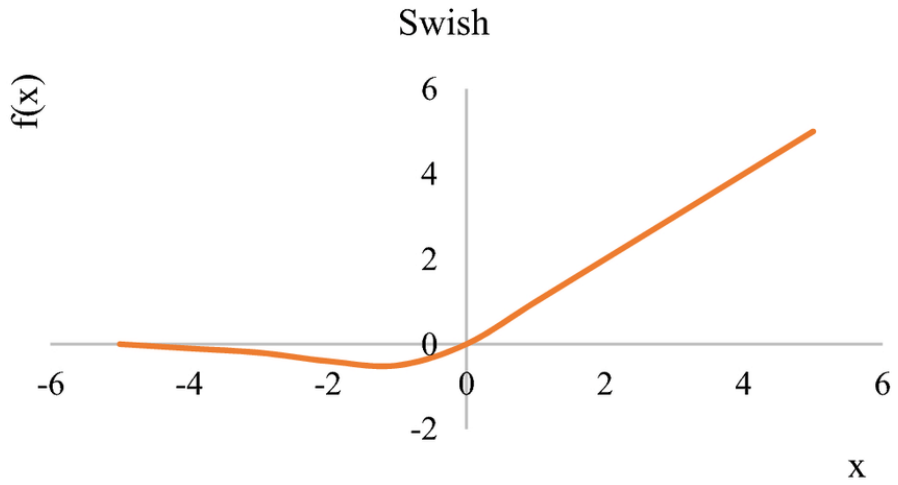


Fig. 2.11. Swish activation function diagram

CHAPTER 3
OBJECTNESS ACTIVATION NETWORK (OAN)

3.1. Definition of OAN

In the process of building an aircraft recognition system, one of the common problems is a large number of images or patches that do not contain any objects of interest. Despite this, such images are fed to the input of object detectors, which leads to unreasonable costs of computing resources, memory and time. With large amounts of data, this factor can critically affect the efficiency of the entire system, especially when it comes to processing satellite data in real time or building a model on a large dataset. To reduce the load on the main model and increase overall performance, the study uses the Objectness Activation Network. Its main goal is to quickly pre-estimate the presence of objects in the image. OAN acts as a filter: it analyzes each image and returns the probability that it contains at least one object that can potentially be recognized by the main mode [3]. If the probability value is lower than the specified threshold, the image is considered empty and is not passed on for further processing by detectors or classifiers.

OAN is usually a lightweight neural network that contains a small number of convolutional layers and works on the principle of binary classification: "object is present / object is absent" [3]. It is important that it is trained separately from the main model, on a specially formed data set that includes both positive examples (with objects present) and negative ones (where there are no aircraft).

Thus, the use of OAN allows you to significantly reduce the number of unnecessary calculations, narrow the search space for the main detector and improve

the overall efficiency of the model. This is especially relevant in satellite monitoring tasks, where a significant part of the images may not contain target objects, but must be pre-analyzed.

In the OAN process, the input image of 1024×1024 pixels is first divided into a grid of size $S \times S$, where S =16. Thus, each grid cell corresponds to a fragment of size 64×64 pixels in the original image. To obtain features, the last layer of the ResNet50 network (the so-called C5 layer) is used, which has a stride of 32 relative to the input image. Next, at the C5 layer, a 3×3 convolutional layer with a step of 2 and 256 filters is applied, which allows reducing the dimensionality of the features and forming a map of size 16×16×256. After that, for further processing, a 1×1 convolution with 512 filters is used, and at the end — another 1×1 convolution, but with one filter. Thus, an objectness activation map M of size 16×16×1 is generated. Each value in this map corresponds to an objectness score in the corresponding image region—that is, each grid element.

The simplicity of this architecture allows for easy integration of OAN into existing systems. Although more sophisticated approaches, such as multi-scale feature fusion, are possible, this work focuses on the performance of the basic model.

3.2. Structure of OAN

Its architecture typically includes a sequence of convolutional layers that extract features from the input image, activation layers (ReLU), and a final classification layer that outputs a single scalar value in the interval $[0, 1]$. This value is interpreted as the probability that the image contains an object.

Structure of processing the image has the following steps:

1. The input image or patch (usually 1024×1024 pixels) is fed to the input of the network.

2. Convolutional layers (e.g. 3×3 or 5×5) are used to extract spatial features. This allows you to detect local structures - contours, contrasts, shapes.

3. An activation function provides the model with nonlinearity, thanks to which it is able to describe complex dependencies between pixels.

4. Global Average Pooling or Flatten Layer reduces the spatial dimensions and converts the features into a vector.

5. A fully connected (Dense) layer using the Sigmoid activation function forms the final output - the probability of at least one object in the image.

6. If the resulting value exceeds a certain threshold (e.g. 0.2), the image is passed on to the main recognition model. If the value is lower, the fragment is ignored.

3.3. Loss Function

Let us denote the set of real (reference) objects in the image $\boldsymbol{I}$ as *{GT}*, where each object is represented as a rectangle with coordinates. Let ***M(i,j)*** be the objectness score for a grid cell ***I(i,j)***, where the $0 \leq i, j \leq S-1$ are the grid indexes.

Based on these definitions, a rule for assigning labels for training is formed. For each grid cell ***I(i,j)***, a binary label $\widehat{\boldsymbol{P}}(\mathrm{i,j})$ is determined. In particular, if the center of *l-th* ground-truth box $\boldsymbol{GT_l}$ falls within the boundaries of this cell, then it is marked as positive (contains an object) (see Fig 3.1). Otherwise, the cell is considered negative (no object). It is important that this rule depends solely on the geometric arrangement of objects and does not take into account their class affiliation [3]. So, the process of sample assignment can be formulated as follows:

$$\widehat{\boldsymbol{P}}(\mathrm{i,j}) = \begin{cases} 1 & if\ GT_l(x,y) \in \boldsymbol{I}(i,j) \\ 0 & otherwise \end{cases}.$$

where $GT_l(x,y)$ - the center of *l-th* ground-truth box.

Loss function can be calculated in the following form:

$$L_{OAN} = \frac{1}{S^2 \sum_{i=0}^{S-1} \sum_{j=0}^{S-1} FL\left(\widehat{\boldsymbol{P}}(\mathrm{i,j}), \boldsymbol{M}(\mathrm{i,j})\right)},$$

Where $FL(\bullet)$ - the conventional focal loss which balances negative and positive samples.

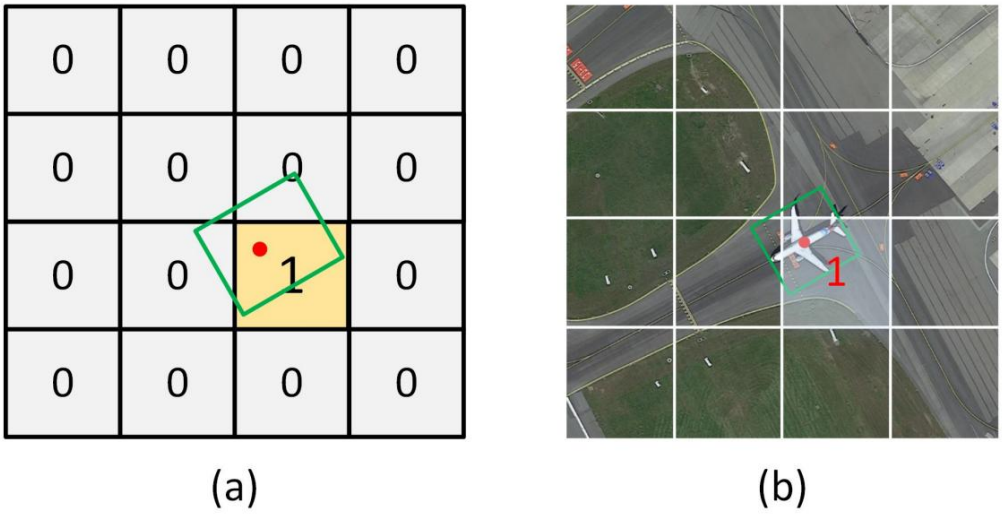


Fig. 3.1. Green box is the ground truth. 1 stands for positive samples and 0 – for negative

3.4. Training OAN model

To train the Objectness Activation Network, a prepared dataset is used, in which each image has a binary label:

• 1 — object is present (there is at least one bounding box),

• 0 — object is absent.

Example of segmenting the image and specifying a binary label is presented in Fig. 3.1.

The goal of training is to minimize the loss function which describes how much the predicted probability differs from the true class. Since filtering tasks are often characterized by class imbalance (much more empty images than filled ones), during training the following can be used: weights that compensate for the imbalance, and oversampling of the positive class [3].

The overall loss L is the summation of OAN loss, bounding-box loss, and classification loss:

$$L = \lambda L_{OAN} + L_{box} + L_{class}.$$

CHAPTER 4
TRAINING OF OBJECT DETECTION AND CLASSIFICATION

4.1. Process of models training

In the process of creating a system for automatic analysis of satellite images, the stage of training artificial intelligence models plays an important role. For this, a single, carefully prepared dataset containing images with appropriate annotations in the bounding box format is used. The full process of preparing the dataset is described in chapter 1. Thanks to such preparation, it was possible to ensure high quality input data for training neural networks.

Model training will be carried out in two stages, since within the framework of this work, the use of two separate architectures is envisaged to solve different subtasks. The first stage will be training an object recognition model that will determine the coordinates of aircraft on satellite images. In this part, a two-stage Faster R-CNN architecture with ResNet as a feature extractor is used, which ensures high accuracy of object localization.

The second stage will be training a separate classification model, the task of which is to determine the type of aircraft based on individual image fragments cut out according to the bounding box obtained in the previous step. For this purpose, the EfficientNet model is used, which demonstrates excellent results in classifying small objects and allows achieving high accuracy while maintaining a moderate amount of computational resources.

Thus, the step-by-step training of individual models allows dividing a complex task into two independent parts and optimizing the training process for each of them

separately. This increases the flexibility of the system and allows achieving better results in both detection and classification of objects.

4.2. Training of object detection model

In the process of training the object recognition model, the input annotations to the images that specify the positions of objects in satellite images play a key role. For this task, the input annotations in the bounding box format are sufficient, while the object class is not taken into account, i.e. all objects are marked as "aircraft object", regardless of whether it is an airplane or a helicopter. This allows you to focus exclusively on the problem of object localization, which significantly simplifies the architecture and the training process.

Another essential element in model training are hyperparameters. Hyperparameters are settings that are set before a model is trained and do not change automatically during training. They directly affect how quickly and efficiently the model learns, as well as its final accuracy. Unlike neural network parameters (such as layer weights), which are updated automatically during error backpropagation, hyperparameters are tuned by the user manually or using special optimization methods (such as grid search or Bayesian optimization).

The following hyperparameters were chosen to train the Faster R-CNN model:

• Number of epochs: 50. An epoch is a complete pass of the model over the entire training dataset. A value of 50 epochs allows the model to approach the optimal solution well enough without causing overtraining. Previous studies with similar problems have found that increasing the number of epochs beyond 50 does not provide a significant increase in accuracy, but increases the computation time.

• Batch size: 8. A batch is the number of images that are processed simultaneously in one step of updating the model weights. Smaller batch sizes reduce memory consumption, which is important when working with large images (for example, 1024×1024 pixels), and also contribute to greater variability of gradients, which has a positive effect on the generalization of the model.

• Optimizer: Adam (Adaptive Moment Estimation). The optimizer is responsible for how the model updates its weights during the training process. Adam's feature is the combination of two other optimization approaches - Momentum and RMSProp. It uses exponentially weighted averages of the first order (gradients) and the second order (squared gradients), which allows adjusting the learning rate separately for each parameter depending on the update history.

At each training step, the optimizer performs the following actions:

- calculates the gradient of the loss function for each parameter;
- estimates the average of the gradients m_t and the average of the squares of the gradients v_t;
- performs bias correction for the initial iterations;
- updates the network parameters using the formula:

$$\theta t + 1 = \theta t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

where α - the learning rate;

$\hat{m}_t$ - the smoothed value of the first moment (gradient);

$\hat{v}_t$ - the smoothed value of the second moment (square of the gradient);

ϵ - a small number to avoid division by zero.

One of the key advantages of Adam is its adaptability, which allows it to train models efficiently even with non-uniform gradients or in cases where the data is noisy. In addition, Adam often provides faster convergence than classical stochastic gradient descent (SGD), making it a convenient choice for tasks with large amounts of data or complex network structures.

After the model training process is complete, it is very important to evaluate the quality of its training. This allows you to understand how well the model has generalized the training data and whether it is ready to be used on new, previously unknown images.

Special quality assessment metrics are used for this. They allow you to quantitatively measure the effectiveness of the model on validation or test dataю The following accuracy metrics were used to assess the quality of model training:

• IoU (Intersection over Union): This metric determines how well the predicted position of an object matches the true one. It is calculated as the ratio of the area of intersection of the predicted and true bounding boxes to the area of their union. The IoU value is used as a criterion for deciding whether an object is considered correctly recognized (for example, when IoU ≥ 0.5).

• Precision and Recall: Precision shows what proportions of predicted objects are correct, and Recall shows what proportion of true objects were found by the model. Both metrics are important for the recognition task, especially in the context of object detection in satellite images, where false non-detection (low recall) can be critical.

• mAP (mean Average Precision): This is a generalized metric that takes into account the average precision value at different recall levels. It allows you to objectively assess the quality of the entire model, taking into account its ability to both accurately and completely recognize objects.

4.3. Training of image classification model

Process of training model for image classification is very similar to previously described mechanism because in both cases, convolutional neural networks are used. The following hyperparameters were chosen to train the EfficientNet model:

• Number of epochs: 50. Choosing 50 epochs allows the model to “see” enough data to learn to recognize patterns without going into overtraining mode.

• Optimizer: Adam. Initial value for learning rate is 0.01 but it can be dynamically decreased if the value of loss function stops decreasing.

• Batch size: 32. As the images for classification are cropped out from original images, they are smaller in size, so bigger amount of them at the same time can be used during training process.

• Dropout rate: 0.3. To prevent overtraining in fully connected layers, a Dropout method with a probability of 0.3 is used. This means that at each training step, 30% of the neurons are temporarily "disconnected", which helps the model not to depend on individual features and improves generalization.

• Loss function: Categorical crossentropy. Since the classification task is multi-class (classes of airplanes, helicopters, etc.), Categorical Crossentropy is used — a loss function that measures the distance between the model's prediction probability distribution and the true class label.

• Metrics: accuracy, precision, recall, and F1-score

4.4. Fine-tuning of models

Even after the main training phase is complete, when the model has achieved high accuracy, there may be situations when new types of images or objects that the model has not previously seen (such situations consist of new aircrafts or usage of masking techniques). In such cases, completely retraining the model is resource-consuming and irrational. Instead, a fine-tuning approach is used.

Fine-tuning is the process of additional training of an existing model on a new data set that contains new or refined samples. This approach allows the model to adapt to changes in the environment or to additional information without losing previously acquired knowledge. It is important to emphasize that during retraining, existing weights obtained during the main training are used, and only a part of the parameters are updated, usually in the last layers of the network.

CHAPTER 5
TECHNICAL REALIZATION

5.1. Choosing software

The effective development and training of artificial intelligence models largely depends on the chosen environment in which the software implementation is carried out. The development environment includes not only the programming language, but also frameworks, libraries, hardware resource management tools, as well as platforms for scaling, testing and deploying models. The quality, stability and scalability of the problem being solved are largely determined by these components.

For computer vision tasks involving deep convolutional neural networks (such as Faster R-CNN and EfficientNet), the development environment must meet the following requirements:

• GPU support for computational acceleration;

• Ability to work efficiently with large image sets;

• Advanced API for building, training, and evaluating models;

• Support for a modular structure for further scaling;

• Compatibility with libraries for visualization, data augmentation, and metric evaluation.

Today, there are several main frameworks for developing deep learning models (see Table 5.1).

Table 5.1

Comparison of frameworks

Framework	**Language**	**Features**
TensorFlow	Python, C++	High performance, good integration with Google Cloud
Keras	Python	Easy to use, works as a wrapper over TensorFlow
PyTorch	Python	Flexibility, dynamic graph computation, research-friendly
MXNet	Python, Scala	Efficient work with clouds, multi-language support
JAX	Python	Latest scientific developments, functional approach

In this qualification work, a development environment based on the PyTorch framework was chosen to implement the object recognition and classification model. This choice is primarily due to the fact that this platform provides flexible capabilities for building and modifying the architecture of neural networks, which is especially important when experimenting with new approaches or combining different components (for example, combining Faster R-CNN with Objectness Activation Network). Thanks to the dynamic computation graph, PyTorch allows you to change the structure of the model on the fly without the need to completely redefine all its components, which significantly speeds up the testing and debugging stages.

In addition, PyTorch is widely used in the scientific community, so for many models, in particular Faster R-CNN and EfficientNet, there are already ready-made implementations and pre-trained weights available in the torchvision library. This allows you to focus not on implementing the basic logic, but on adapting and retraining the model to the specifics of the task. PyTorch also has a convenient interface for working with graphics processing units (GPUs), which is critically important when processing large amounts of high-resolution satellite imagery.

In addition to the main framework, additional tools and libraries were used in the work:

• NumPy — for processing arrays and basic numerical operations;
• Pandas — for working with table annotations and statistics;
• OpenCV — for image preprocessing;
• Matplotlib / Seaborn — for visualizing training results and metrics;
• Albumentations — for image augmentation during training;
• Weights & Biases / TensorBoard — for tracking the training process.

5.2. Choosing hardware

The efficiency and speed of calculations directly depend on the characteristics of the hardware. The greatest load when training neural networks falls on the graphics processing unit (GPU), so it is the key component of the system. Unlike the central processing unit (CPU), the GPU is able to process thousands of parallel calculations simultaneously, which is ideal for operations performed when passing data through convolutional layers, normalization, backpropagation, etc.

For efficient operation of the GPU, a large amount of video memory (VRAM) is required, which is used to store tensors, weights, gradients and intermediate calculation results. In the case of processing satellite images with a size of 1024×1024 pixels and above, the need for video memory increases, so the recommended minimum is 12–16 GB of VRAM, and the optimal is 24 GB or more. Models such as NVIDIA RTX 3090, RTX 4090, or server cards such as A6000 are well suited for this.

Also important is support for CUDA (Compute Unified Device Architecture) technology — this is a parallel computing platform and API developed by NVIDIA. CUDA allows you to directly use the GPU for general programming, in particular for deep learning. The PyTorch and TensorFlow libraries have deep integration with CUDA, which allows you to effectively use the resources of the video card for calculations. Without CUDA support, data processing will be carried out only on the CPU, which will significantly slow down the learning process — in some cases by dozens of times. As for the central processor, it is advisable to choose an AMD Ryzen 9 7950X or a similar model with a large number of cores (12 or 16), which allows you to

quickly process auxiliary tasks related to data preparation, logging, and training management. Random access memory (RAM) is also a critical component — when working with large datasets, for example, augmentation or preprocessing, the system can use a lot of resources. For comfortable operation, it is recommended to have at least 64 GB of RAM.

5.3. Full workflow of building the system for aircraft detection and classification

The first stage of the implementation of the system for automatic recognition and classification of aircraft on satellite images is the preparation of a dataset based on the open MAR20 set. In order to improve the overall quality of the model and its ability to generalize, the initial dataset was augmented. This is achieved by adding new images obtained from open satellite sources, as well as using augmentation methods.

Based on the prepared data, using the Python programming language and specialized libraries for machine learning, in particular PyTorch, a model for object recognition was built. The architectural basis of this model was the Faster R-CNN algorithm, in which ResNet50 was chosen as the feature extractor (backbone). This choice is due to the high accuracy and stability of the results, confirmed by many studies in the field of computer vision.

One of the key features of the system is the use of the Objectness Activation Network before feeding the image to the input of the recognition model. OAN allows filtering out those images or segments that do not contain any objects, thereby significantly reducing the computational cost and improving the overall accuracy of the system. After the recognition stage, a separate model for aircraft classification was built, which works on the basis of EfficientNet. This model was chosen due to its high accuracy even when classifying small objects.

The final stage is model training. Although the same dataset is used for both models, the training structure is adapted to the specifics of each task: for recognition, the coordinate marking (bounding boxes) is important, while for classification, the

object’s belonging to a certain class is important. All calculations and model training are performed on a computer with the following characteristics: an AMD Ryzen 9 7950X processor, 64 GB of RAM, and an NVIDIA RTX 4090 graphics accelerator with CUDA support.

As a result of all stages, a software system was built that is capable of automatically detecting and classifying aircraft on satellite images with high accuracy. The system can be integrated into analytical complexes for airspace monitoring, security tasks, or research purposes. Full pipeline of processing satellite image by detecting aircrafts from it and classifying found objects is represented in Fig. 5.1.

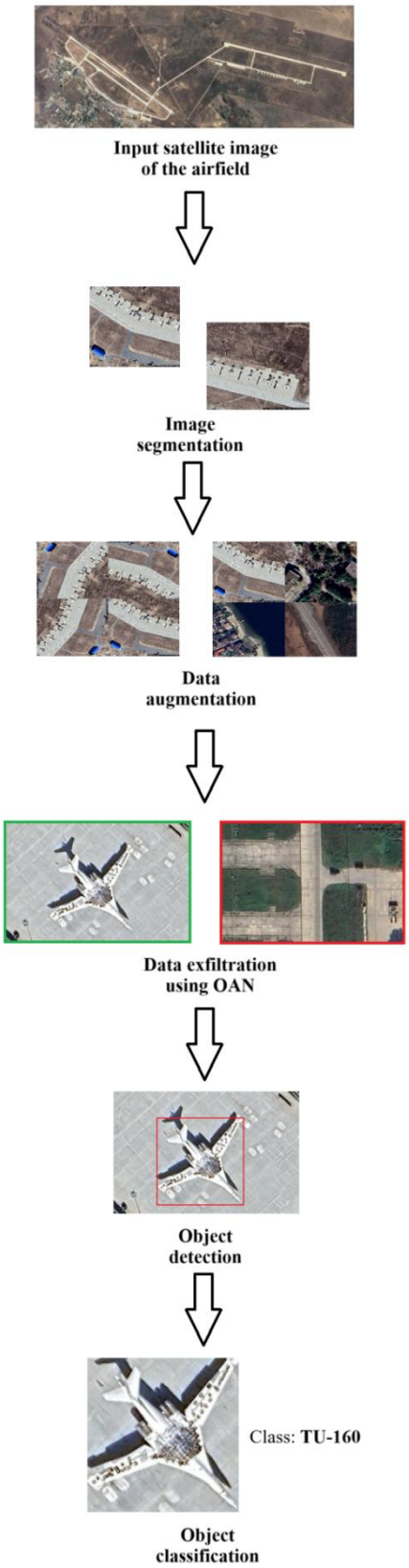


Fig. 5.1. Full process of retrieving information about aircrafts from satellite image

CONCLUSION

In terms of this qualification work, a system for automatic recognition and classification of aircraft on satellite images using artificial intelligence methods was developed. The main goal of the research was to model an effective and accurate model capable of working with large volumes of satellite data. Given the complexity of the task, a comprehensive approach was applied, which involves not only the development and training of deep neural networks, but also thorough preliminary data preparation, optimization of the model structure and provision of an appropriate technical environment.

The first stage of the qualification work was the formation and processing of the dataset. Given that satellite images can have extremely high resolution, an algorithm for segmenting images into smaller fragments of 1024×1024 pixels with an overlap of 200 pixels was implemented. This approach ensured complete coverage of the area of interest and avoided the loss of objects that may be located on the boundaries of blocks. In parallel with the image segmentation, annotations were generated that contained only object coordinates without the need to specify classes — since recognition and classification in the system are divided into separate stages.

For aircraft recognition, a two-stage Faster R-CNN model with ResNet as the basic feature network was chosen. This choice was due to the high accuracy that this approach provides due to the step-by-step analysis of the input image — first using the feature extraction and region of interest generation (RPN) block, and then — by classifying and regressing object coordinates. ResNet provides efficient gradient transfer thanks to the residual connections mechanism, which allows training deeper architectures without degrading the results.

At the second stage, an image classification model was implemented. Taking into account the task of classifying very small objects, where the preservation of details plays a key role, the EfficientNet architecture was chosen, which provides the optimal balance between accuracy, processing speed, and the number of parameters. Thanks to the technology of scaling in three dimensions — depth, width and resolution — EfficientNet allows to achieve high accuracy with limited resources.

To increase the efficiency of the entire system, the OAN was integrated — an additional network that performs the function of filtering input images at the stage before processing by the main model. This allowed to significantly reduce computational costs, since a significant part of the fragments does not contain any objects and, accordingly, does not require further analysis.

Special attention was paid to the process of training the models. For each of them, the appropriate hyperparameters were selected, in particular the number of epochs, batch size and optimizer. In the case of the recognition task, the emphasis was placed on the accuracy of localization of bounding boxes, and in the classification task — on the correct determination of the type of aircraft. To assess the quality of training, metrics such as mAP (mean Average Precision) and F-1 Score were used.

At the final stage, technical preparation for the implementation of the system was carried out. Given the requirements for computing resources, CUDA-enabled GPUs, including NVIDIA RTX series graphics cards, were selected for training. The development was carried out in the PyTorch environment, which provides flexibility in designing architectures, convenience in processing tensors, and compatibility with most libraries for working with data.

Thus, as a result of the research, a multi-component system was created that covers the full cycle - from data preprocessing to model construction, training, and optimization. The resulting system is capable of effectively recognizing and classifying aircraft in satellite images with a high level of accuracy, which makes it promising for practical application in the defense sector, airspace monitoring, or security.

LIST OF BIBLIOGRAPHICAL REFERENCES OF USED SOURCES

1. https://viso.ai/deep-learning/object-detection/
2. https://youtu.be/nJzQDpppFj0?si=rBh4GPm-n0jn1N7f
3. https://arxiv.org/pdf/2212.13136
4. https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf
5. http://www.huppelen.nl/publications/selectiveSearchDraft.pdf
6. https://uu.diva-portal.org/smash/get/diva2%3A1766080/FULLTEXT01.pdf
7. https://arxiv.org/pdf/2006.02963
8. https://universe.roboflow.com/
9. https://www.digitalocean.com/community/tutorials/faster-r-cnn-explained-object-detection
10. https://arxiv.org/pdf/1905.11946
11. https://www.researchgate.net/figure/The-architecture-of-a-standard-convolutional-neural-network_fig1_348915715
12. https://stackoverflow.com/questions/38450104/how-to-visualize-filters-after-the-1st-layer-trained-by-cnns
13. https://arxiv.org/pdf/1512.03385
14. https://viso.ai/deep-learning/efficientnet/
15. https://arxiv.org/pdf/2206.08016