Aircraft detector

Enache Ana-Iulia
Enache Alexandru
Gologan Christin-Tarciziu

1. Application - functionalities

The context is to develop a system that can accurately detect and identify aircraft in satellite imagery. This involves creating a robust and efficient model that can identify the presence of an aircraft in these images.

The core functionality is to accurately detect the presence of aircraft in satellite imagery. This involves object detection. Other important functionalities of the application include integrating with external data sources, to enhance the application's capabilities, continuous learning for the model to adapt to changes in aircraft design and technology.

In order to achieve this, the user is able to upload an image in the application. The uploaded image will be opened in the application window. Then, the process button becomes enabled, and the user can trigger the processing of the image. After processing, the old image is replaced by the new processed one that detects the presence of the aircraft in the picture.

2. Domain problem

The primary task is to detect the presence of aircraft within the imagery. This involves identifying regions or objects within the image that correspond to aircraft. Satellite images can vary significantly in terms of lighting conditions, image quality, aircraft orientation, size, and position. Also, striking a balance between reducing false

positives (incorrectly identifying objects as aircraft) and avoiding false negatives (missing actual aircraft) is a critical challenge.

The domain problem is the complex and multifaceted challenge of applying artificial intelligence, computer vision, and remote sensing techniques to detect and identify aircraft within satellite imagery or aerial photographs. This problem resides at the intersection of several key domains.

Computer vision and machine learning are concerned with developing and training algorithms and models that can analyze and interpret visual data. In the context of aircraft detection, it involves creating deep learning models capable of object detection and classification.

Remote sensing is the domain of collecting and interpreting data about the Earth's surface from a distance, using satellites. It includes techniques for image acquisition, preprocessing, and analysis, all of which are essential. To enhance the accuracy of aircraft detection, integrating historical air traffic data is necessary.

Successfully addressing this problem requires integration of these domains to create a reliable and effective aircraft detection system.

3. Related work

Early approaches

Early attempts at aircraft recognition in images and satellite data often relied on traditional computer vision and image processing techniques. Some techniques used were **manual feature extraction** which involved manual identification and extraction of features relevant to aircraft characteristics, such as shape, size, and color. Another technique was based on **color classification**, for example, the color and shape of an aircraft could be differentiated from the background. **Spectral analysis** was another

interesting technique and was utilized to detect specific signatures associated with aircraft, such as the reflection of sunlight from metallic surfaces.

Nevertheless, while these early approaches laid the foundation for aircraft recognition, they had limitations in terms of robustness and scalability. The advent of **deep learning**, particularly neural networks and its derivatives, such as Convolutional Neural Networks (CNNs), marked a significant shift in the field, allowing for more automated and data-driven feature extraction, enabling improved accuracy and generalization across diverse datasets.

Due to the rapid advancement of technology and, unfortunately, the increase of conflicts in the world, there has been an increased interest in the modernization of military technology. Because of this, a lot of research is being done to gain an advantage over the competitors in this field.

State-of-the-art

Some of the state-of-the-art methods in object detection are **Region-based Convolutional Neural Networks** (RCNN) that use selective search to search for local cues, such as texture, intensity and color to generate locations of an object and propose approximately 2000 region proposals. Some better variations of this method are **Fast RCNN** and **Faster RCNN**. **YOLO** (You Only Look Once) is another approach that divides the image into a squared grid. Each grid predicts a number of bounding boxes with confidence.

Military vehicles detection using aerial imagery

One report written by Narcisse [2] has the objective to explore the potential of instance segmentation for military vehicles detection in satellite and aerial imagery. The methodology employs R-CNN (Region-based Convolutional Neural Network) and the models were trained on a dataset consisting of 200 satellite and aerial images in RGB (link images) format, which contained 10,624 annotated military vehicles. The

performance was evaluated using various quantitative measures, such as Average Precision (AP) metrics for each class and at varying Intersection over Union (IoU) thresholds using the detection evaluation metrics by COCO. The method proposed in the paper was able to achieve good performance and efficiency in detecting military vehicles, even in complex and cluttered environments.

Airplanes detection in busy airports using aerial imagery

Problem

Another paper [1] is addressing the traffic jams of the planes, especially for the busiest airports. They think that one of the most effective solutions, but quite costly, is through the use of satellite images with deep learning techniques. Instead of using expensive and complicated satellites, the paper recommends the use of drones to feed the learning model with images of airports, with the drones acting as the data set. The model will then detect whether or not an airplane is present in the analyzed images.

Dataset

The dataset initially contained 557 images, which were reduced in resolution (369 x 259 px) for faster processing. To artificially increase the size of the dataset, the images were subjected to an **augmentation process** in which duplicate versions of the images were added, but with noise, rotated, and cropped. For labeling, two supervised techniques

have been applied: **bounded box labeling** (first picture) and **semantic segmentation** (second picture).

Algorithm

For training, the authors decided that the **Mask RCNN** model and 12 **COCO metrics** were suitable for their research. The *mask_rcnn_inception_v2_coco* configuration has been applied from the TensorFlow models and it is the base configuration for the training process. 7 additional models have been added with slight adjustments on the learning rate, number of proposals and image resize. In the paper, the sixth model achieved the best performance among all other models. According to the authors, this is due to its lower learning rate.

Model	Metrics											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0.448	0.911	0.384	0.226	0.477	0.637	0.230	0.505	0.520	0.386	0.540	0.697
2	0.405	0.845	0.317	0.184	0.434	0.611	0.221	0.466	0.484	0.329	0.508	0.659
3	0.244	0.531	0.174	0.061	0.29	0.465	0.157	0.334	0.375	0.234	0.394	0.579
4	0.397	0.813	0.300	0.188	0.427	0.575	0.209	0.455	0.482	0.335	0.506	0.656
5	0.456	0.902	0.398	0.209	0.495	0.637	**0.241**	0.520	0.524	0.343	0.555	0.709
6	**0.472**	**0.932**	**0.430**	**0.260**	**0.499**	**0.687**	0.240	**0.530**	**0.538**	**0.387**	**0.560**	**0.738**
7	0.393	0.852	0.250	0.245	0.436	0.456	0.211	0.480	0.506	0.374	0.540	0.532
8	0.314	0.693	0.244	0.106	0.360	0.491	0.190	0.380	0.409	0.224	0.441	0.612

Performance

The evaluation yielded promising outcomes in terms of COCO metrics. Although satellite images are different from drone images, the accuracy values are encouragingly similar, further indicating the reliability of this approach.

Aircraft detection in airports and airfields using aerial imagery

There were also other challenges faced in detecting aircraft from modern remote sensing images. The sophistication of current remote sensing technology has resulted in images with a wealth of information, making the detection of objects, especially aircraft, a complex task.

Problem

Several challenges are highlighted in the paper [3] such as the influence **of surrounding objects**, **overlap problems** and **numerous objects in the airfield**.

The presence of smaller objects around the aircraft and unidentified objects can interfere with the detection model. This suggests that the model needs to account for and distinguish various objects in the scene to avoid false positives or missed detections.

Existing research studies on aircraft detection from remote sensing images have not adequately addressed the overlap problem. The overlap problem refers to situations where multiple objects may overlap in the image, making it difficult for the detection model to accurately identify and delineate individual objects.

The wide-open space of airfields and airports presents a challenge due to the multitude of objects that may be present. This indicates a need for a robust detection model capable of handling many objects in complex environments.

Algorithm

To address these challenges, the text introduces a proposed solution – a method using the Region-based Convolutional Neural Network (RCNN) architecture based on the YOLOv3 (You Only Look Once version 3) model. This method aims to resolve the overlap problem by utilizing the interaction of unit and ground truth data to select appropriate bounding boxes for aircraft.

First, the dataset details were discussed compared to earlier studies on detecting aircraft with the images and then followed by the neural network model adopted for the study, which is YOLOv3 and transfer learning was adopted here for customized objects. The model uses 53 convolutional layers with no pooling layers.

The architecture consists of a fully convolutional network with convolutional layers. The images were initially loaded into the convolutional layer with dimensions 416

x 416. Residual blocks were used for feature extraction, and images were downsampled to various scales.

YOLOv3 adopts a single-stage object detection approach. The image is split into grids, and each cell is analyzed to create bounding boxes. Bounding boxes are created based on the confidence parameter, which utilizes ground truth and the interaction of units from the image. **Binary cross-entropy** is used as the loss function for predicting object scores and truth object scores.

The model uses equations for predicting coordinates (bx, by, bw, bh) of the bounding box. The coordinates are determined based on the ground truth and the offset of the grid cell.

Dataset

The dataset was collected from Kaggle open-source, containing images with and without aircraft. It was pre-processed and annotated for the presence of aircraft. It was split into two sets: **8,000 images with aircraft and 24,000 images without aircraft**.

The YOLOv3 model is trained using 8,000 images with a resolution of 20 x 20 pixels. The images were scaled to 416 x 416 to fit the YOLOv3 architecture. Training details include a batch size of 120, 300 epochs.

Performance

The proposed method achieved an accuracy of 98.5%, outperforming other models. False Positive Ratio (FPR), Missing Ratio (MR), and Error Ratio (ER) for the proposed method were better than other models. Despite minor differences in detection time, the proposed model showed superior accuracy.

Compared with a conventional CNN model, the proposed RCNN model demonstrated slightly lower accuracy but better False Positive (FP) and False Negative (FN) rates. The proposed model exhibited less error in detection.

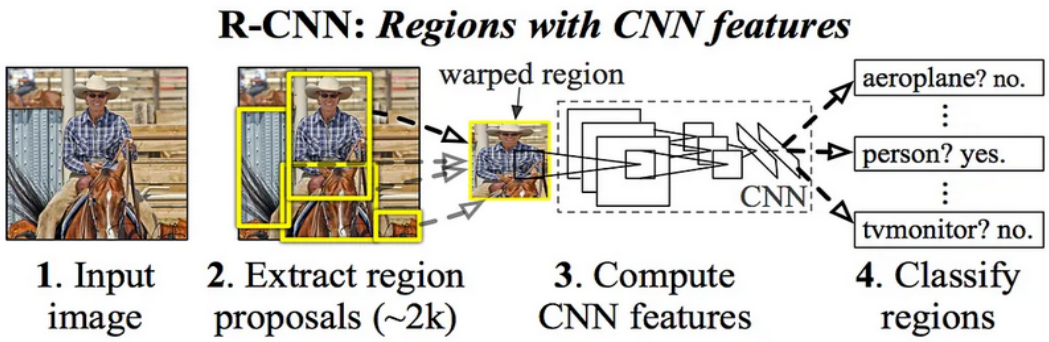
Model	Accuracy	Specificity	Sensitivity	False Positive rate	False Negative rate
CNN [2]	98.4	98.13	98.62	0.0187	0.0138
Proposed	98.5	97.72	97.45	0.0168	0.0129

Precision and Recall metrics were compared with previous studies. The proposed model showed a precision of 97.65% and a recall of 97.13%, indicating consistent and accurate results.

Method	Recall (%	Precision (%)
Method 1 [10]	59.2	67.3
Method 2 [11]	63.6	80.6
F-RCNN [12]	82.4	87.8
F-RCNN+ResNet-50 [13]	83.7	88.6
Only conv4 x+HOG [7]	84.1	89.6
Existing Method [7]	91.9	93.5
Proposed method	97.13	97.65

4. Solution logic

In R-CNN, instead of applying classification to a large number of regions in an image, here is utilized a selective search algorithm. This process involves passing the image through selective search, generating region proposals, and selecting the first 2000 proposals for further classification. By doing so, it streamlined the classification task to focus on a limited set of regions, making the algorithm significantly faster than earlier object detection techniques.



Initiate the process by passing the image through **selective search**, generating potential region proposals. Calculate the **Intersection over Union (IOU)** between the proposed regions and the ground truth data, determining how well they align. Assign labels to the proposed regions based on the IOU values, indicating whether they correspond to actual objects. Perform **transfer learning** using the proposed regions and their assigned labels. Fine-tune a pre-trained neural network on this limited set of regions, leveraging existing knowledge to enhance object feature recognition. During testing, apply selective search to the test image to generate region proposals. Pass the first 2000 proposed regions from the trained model and predict their corresponding classes.

After creating the dataset, it will be passed to a pre-trained model - the model chosen being VGG16, which due to its strong performance for computer vision applications, is a great choice for transfer learning with ImageNet weight (a very large dataset containing millions of labeled images across thousands of categories). After creating the model, the dataset is split into train and test, keeping 10% of the dataset as test and 90% as training.

Before starting the training, the dataset is also subjected to an augmentation by horizontal and vertical flipping and some rotation to artificially increase the size of the dataset. When doing the prediction, the bounding boxes around the proposed regions (that

should contain airplanes) will be created in the output image only if the confidence is above a defined threshold.

Model evaluation

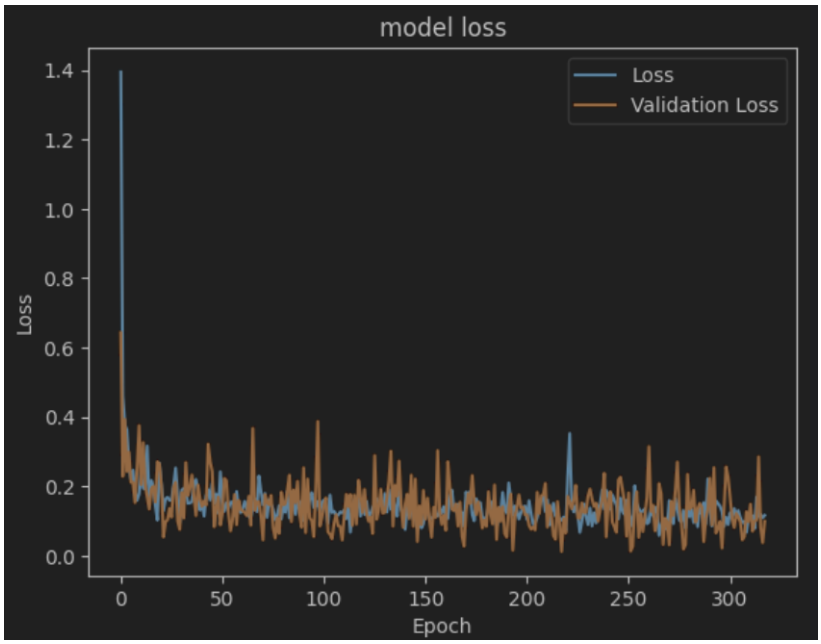
For evaluating the model we used the following metrics:

- Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.9594
- Loss = 0.1169

Where:

- TP means true positives
- TN means true negatives
- FP means false positives
- FN means false negatives

Those results can be also checked in the following image:



Implementation details

We used Keras with Python having as layers Conv2D, MaxPooling2D and Dense. The neural network will receive the following parameters:

- Optimizer: Adam
- Learning rate: 0.001
- Batch size: 70
- Number of epochs: 1000

Below, you can check the summary of the model:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
dense_1 (Dense)	(None, 2)	8194

Total params: 134,268,738
Trainable params: 126,633,474
Non-trainable params: 7,635,264

Dataset

The dataset had a pretty consistent number of images, 700 with different scenarios (no airplanes, one airplane, 2 or more airplanes, half of an airplane etc.) for training and 17 to test the trained model.

Below you can find an example of a processed image and the corresponding areas where the model identified the airplanes:

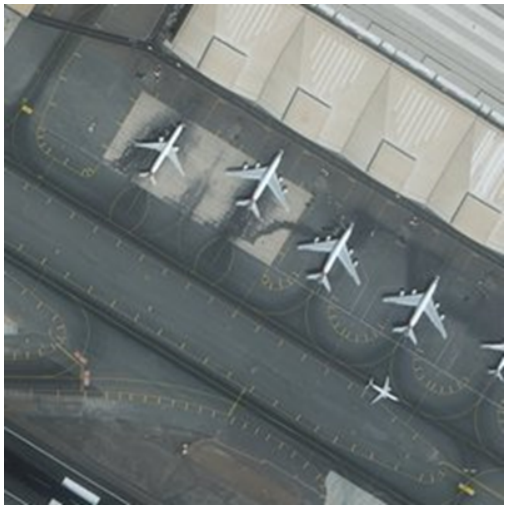


Figure 1. Input image

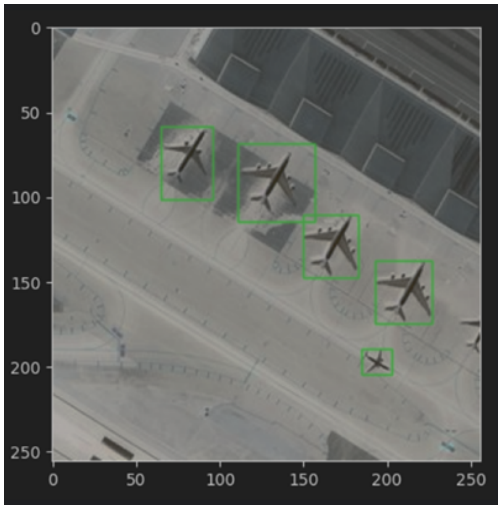


Figure 2. Output image

The boxes you can see around the airplanes are based on the model's confidence which has to exceed 0.7 accuracy of their prediction.

5. Tools and technologies

Tensorflow

Machine learning is an intricate field, but the implementation of machine learning models has become significantly more approachable, largely due to the availability of machine learning frameworks like Google's TensorFlow. These frameworks simplify the tasks of data acquisition, model training, prediction deployment, and iterative result improvement.

TensorFlow is an open-source library designed for extensive numerical computation and the facilitation of large-scale machine learning. It consolidates a wide range of machine learning and deep learning models, including neural networks, and presents them in a user-friendly manner through familiar programming paradigms. TensorFlow employs Python and JavaScript to offer a convenient front-end API for application development, while executing these applications with high performance in C++.

TensorFlow, in competition with frameworks like PyTorch and Apache MXNet, possesses the capability to train and execute deep neural networks for various applications, including handwritten digit recognition, image classification, word embeddings, recurrent neural networks, sequence-to-sequence models for tasks like machine translation, natural language processing, and simulations based on partial differential equations (PDE). An outstanding feature of TensorFlow is its support for deploying models at scale for production, utilizing the same models that were used during training.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast

experimentation. Being able to go from idea to result as fast as possible is key to doing good research. Keras is:

- Simple - but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible - Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful - Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

Tkinter

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library. Moreover, the framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of 'graphical control elements', aka widgets, for building application interfaces.

It also provides a simple way to create GUI elements using the widgets found in the Tk toolkit. Tk widgets can be used to construct buttons, menus, data fields, etc. in a

Python application. Once created, these graphical elements can be associated with or interact with features, functionality, methods, data or even other widgets.

Python

It is known that **Python** has become the dominant programming language in the field of machine learning. This is because Python is simple and easy to read, its syntax allowing to express ideas in fewer lines of code compared to other languages. Python can also be integrated with lower-level languages like C and C++, meaning that performance-critical parts of code can be optimized, especially when talking about machine learning.

These aspects made Python very appealing to use in this field and because of this, it received strong support for data manipulation and analysis, with libraries such as NumPy, Matplotlib or TensorFlow. These tools are essential during the machine learning workflow like data exploration or visualization.

Jupyter notebook

Jupyter notebook is another step forward to facilitate machine learning development. It provides an interactive computing environment that allows code execution in a step-by-step manner. This is particularly useful in machine learning where iterative development and experimentation are common. Jupyter also supports inline visualization, allowing plots and graphs creation alongside their code. This is crucial for exploring datasets, visualizing model performance, and gaining insights into the behavior of algorithms.

PyCharm

PyCharm is a great and powerful IDE that provides intelligent code completion, suggestions and automatic error highlighting for Python development. It also integrates

interactive debugging and the libraries and frameworks aforementioned, making this IDE a great choice for machine learning development.

6. Bibliography

[1] Alshaibani, W. T., Helvaci, M., Shayea, I. & Mohamad H. (2021). Airplane Detection Based on Mask Region Convolution Neural Network. *ArXiv*. https://doi.org/10.48550/arXiv.2108.12817

[2] Narcisse, N. (2023, August 2). *Military Vehicles Detection in Satellite & Aerial Imagery using Instance Segmentation*. WandB. Retrieved October 17, 2023, from https://wandb.ai/nadernarcisse/MVD-Satellite-Imagery/reports/Military-Vehicles-Detection-in-Satellite-Aerial-Imagery-using-Instance-Segmentation--VmlldzoyNzE1MTY0

[3] PANDA, Bhavani Sankar; GOPAL, Kakita Murali; SATPATHY, Rabinarayan. DETECTION OF AIRCRAFTS USING RCNN IN REMOTE SENSING IMAGES, from https://ceur-ws.org/Vol-3283/Paper105.pdf