IMAGE IDENTIFICATION AND RECOGNITION USING NOVEL HYBRID ARCHITECTURE DEVELOPMENT FOR SATELLITE IMAGES

A PROJECT REPORT

Submitted by

Elamathy S[211417104058]
Ridu Varshini.N[211417104225]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

PANIMLAR ENGINERRING COLLEGE

ANNA UNIVERSITY : CHENNAI 600 025

AUGUST 2021

IMAGE IDENTIFICATION AND RECOGNITION USING NOVEL HYBRID ARCHITECTURE DEVELOPMENT FOR SATELLITE IMAGES

A PROJECT REPORT

Submitted by

ELAMATHY S[REGISTER NO: 211417104058]

RIDU VARSHINI.N [REGISTER NO: 211417104225]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI600 025

AUGUST 2021

BONAFIDE CERTIFICATE

Certified that this project report **"IMAGE IDENTIFICATION AND RECOGNITION USING NOVEL HYBRID ARCHITECTURE DEVELOPMENT FOR SATELLITE IMAGES "** is the bonafide work of "**ELAMATHY S [REGNO:211417104058] and RIDU VARSHINI.N [REGNO:211417104225]"** who carried the project under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Ms.V.SATHIYA
Associate Professor

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on.....05.08.2021.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D** for the support extended throughout the project.We would like to thank **Dr.L.JABA SHEELA** for the support extended throughout the project.

We would like to thank my **Project Guide Ms.V.SATHIYA,M.E.** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

We would like to thank **god** for showering his blessing on us and we also thank our **parents** for their support both emotional and financial over the years.

ELAMATHY S[211417104058]

RIDU VARSHINI.N[211417104225]

ABSTRACT

Recent advances in deep learning made tasks such as Image and speech recognition possible. Deep Learning is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. The continuous innovation and development of satellite technology has brought the world closer together. At present, there are thousands of artificial satellites in the world, and the number of spacecraft working in orbit is increasing. With the constant exploration of outer space, there is inevitably a large quantity of space debris, e.g., lacquer, satellite debris. How to distinguish space targets of interested from the massive debris is a hot topic and of great significance. The effective separation of space targets and space debris can ensure the safe operation of on-orbit spacecraft. But, nowadays with the advancement of technology there is still a lag in predicting the space crafts and the related targets in an accurate manner. The project is focused on developing a novel hybrid architecture for determining the different types of satellite images, where four different kind of pre-processing techniques as well as two different optimization techniques are used to increase the accuracy of the proposed hybrid model. The concept of network surgery is used for implementing the hybrid algorithm development. The project will be able to determine any kind of satellite images given as an input to the generated model. Thus, the project provides a solution for the determination of spacecraft with the most accurate prediction by developing a novel architecture.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATION

S.No	ABBREVATION	EXPANSION
1.	ISRO	Indian Space Research Organisation
2.	PSLV	Polar Satellite Launch Vehicle
3.	GSLV	Geosynchronous Satellite Launch Vehicle
4.	IRNSS	Indian Regional Navigation Satellite System
5.	GAGAN	GEO Augmented Navigation
6.	DCNN	Deep Convolutional Neural Network
7.	ATR	Automatic Terminal Recognition
8.	SAR	Synthetic Aperture Radar
9.	ISAR	Inverse Synthetic Aperture Radar
10.	STN	Spatial Transformer Network
11.	MBRT	Minimum Bounding Rectangle with Threshold
12.	SR	Super Resolution
13.	COCO	Common Objects In Context
14.	CPVR	Conference on Computer Vision and Pattern Recognition
15.	RNN	Recurrent Neural Networks
16.	CNN	Convolutional Neural Network

CHAPTER I

INTRODUCTION

1. INTRODUCTION

1.1Overview

Indian space programme encompasses research in areas like astronomy, astrophysics, planetary and earth sciences, atmospheric sciences and theoretical physics. Balloons, sounding rockets, space platforms and ground-based facilities support these research efforts. A series of sounding rockets are available for atmospheric experiments. Several scientific instruments have been flown on satellites especially to direct celestial X-ray and gamma-ray bursts. ISRO, by successfully demonstrating its unique and cost-effective technologies, has gained place among the elite space agencies in the world over the years. The first Indian satellite, Aryabhata, was built by the ISRO and launched with the help of the Soviet Union on April 19, 1975. The year 1980 marked the launch of Rohini, which was the first satellite to be successfully placed in the orbit by SLV-3, an Indian made launch vehicle. Subsequently with more efforts, two other rockets were developed by ISRO: the PSLV (Polar Satellite Launch Vehicle) for placing satellites into polar orbits and the GSLV (Geosynchronous Satellite Launch Vehicle) for placing satellites into geostationary orbits. Both the rockets have successfully launched several earth observation and communication satellites for India as well as other countries. Indigenous satellite navigation systems like IRNSS and GAGAN have also been deployed. In January 2014, ISRO used an indigenously built cryogenic engine for a GSLV-D5 launch of the GSAT-14 satellite making it one of the only six countries in the world to develop a cryogenic technology. Most recent and remarkable space probes of ISRO include Chandrayaan-1 lunar orbiter, Mars Orbiter

Mission (Mangalyaan-1) and ASTROSAT space observatory. The success of the Mars Orbiter Mission made India only the fourth country in the world to reach the Martian orbit.

1.2Problem Definition

But, nowadays with the advancement of technology there is still a lag in predicting the space crafts and the related targets in an accurate manner. In this project we will be focusing on developing a novel hybrid architecture for determining the different types of satellite images, where four different kind of pre-processing techniques as well as two different optimization techniques will be used to increase the accuracy of the proposed hybrid model.

Recognition of the type of image becomes difficult for space scientists as well as decrease in resolution leads to misidentification of the space objects. Time consuming process when did manually. Increased error rate.

CHAPTER II

LITERATURE SURVEY

1. **TITLE: Topological Sweep for Multi-Target Detection of Geostationary Space Objects,2020**

AUTHOR : Daqi Liu, Bo Chen, Tat-Jun Chin , and Mark G. Rutten

YEAR : 2020

DESCRIPTION:

Conducting surveillance of the geocenric orbits is a key task towards achieving space situational awareness (SSA). Our work focuses on the optical detection of man-made objects (e.g., satellites, space debris) in Geostationary orbit (GEO), which is home to major space assets such as telecommunications and Earth observing satellites. GEO object detection is challenging due to the distance of the targets, which appear as small dim point-like objects among a background of streak-like objects. In this paper, we propose a novel multi-target detection technique based on topological sweep, to find GEO objects from a short sequence of optical images. Our topological sweep technique exploits the geometric duality that underpins the approximately linear trajectory of target objects across the sequence, to extract the targets from significant clutter and noise. Unlike standard multi-target methods, our algorithm deterministically solves a combinatorial problem to ensure high-recall rates without requiring accurate initializations.

MERITS:

The system provides a solution to detect the geostationary objects.

DEMERITS:

The accuracy is very poor compared to other state of the art methods.

2.TITLE: Numerically Efficient Determination of the Optimal Threshold in Natural Frequency-Based Radar Target Recognition,2019

AUTHOR : Joon-Ho Lee, Hyun-Jin Moon, and So-Hee Jeong

YEAR : 2019

DESCRIPTION:

In this communication, we address a numerical method to efficiently calculate a non-zero optimal threshold value for the performance improvement of a natural frequency-based radar target recognition. From the probability of correct classification with respect to a varying threshold value, we define a function, denoted by f(z) in this communication, so that one of the roots of f(z)=0 corresponds to the optimal threshold. The final optimal threshold value is obtained via the Newton iteration. The scheme is validated by comparing the threshold obtained from the Newton iteration with that using the probability density function.

MERITS:

The scheme is validated by comparing the threshold obtained from the Newton

iteration with that using the probability density function.

DEMERITS:

The results are based out of potential assumptions alone making it un reliable.

3. **TITLE: Robust Pol-ISAR Target Recognition Based on ST-MC-DCNN,2018**

AUTHOR : Xueru Bai , Member, IEEE, Xuening Zhou, Feng Zhang, Li Wang, and Ruihang Xue .

YEAR : 2018

DESCRIPTION:

Although the deep convolutional neural network (DCNN) has been successfully applied to automatic target recognition (ATR) of ground vehicles based on synthetic aperture radar (SAR), most of the available techniques are not suitable for inverse synthetic aperture radar (ISAR) because they cannot tackle the inherent unknown deformation (e.g., translation, scaling, and rotation) among the training and test samples. To achieve robust polarimetric-ISAR (Pol-ISAR) ATR, this paper proposes the spatial transformer-multi-channel-deep convolutional neural network, i.e., ST-MC-DCNN. In this structure, we adopt the double-layer spatial transformer network (STN) module to adjust the image deformation of each polarimetric channel and then perform a robust hierarchical feature extraction by MC-DCNN. Finally, we carry out feature fusion in the concatenation layer and output the recognition result by the softmax classifier. For the fully Pol-ISAR image database generated from electromagnetic (EM) echoes of four satellites, the proposed structure achieves higher recognition accuracy than traditional DCNN and MC-DCNN.

MERITS:

The system has shown robustness to image scaling, rotation, and combined deformation.

DEMERITS:

The system cannot tackle the inherent unknown deformation.

4. **TITLE:T-SCNN: A Two-Stage Convolutional Neural Network for Space Target Recognition,2017**

AUTHOR : Tan Wu, Xi Yang, Bin Song, Nannan Wang, Xinbo Gao, Liyang Kuang, Xiaoting Nan, Yuwen Chen, Dong Yang

YEAR : 2017

DESCRIPTION:

Space target recognition plays an important role in the field of space security and exploration. With the rapid development of artificial intelligence technique and explosive increase of image dataset, object recognition based on deep learning has achieved favorable performance. However, the recognition of deep space targets in visible spectrum images still remains in the traditional manual interpretation approach, thus leading to low efficiency and inevitable subjective errors. In this paper, we propose an artificial intelligence method for space target recognition, called Two-Stage Convolutional Neural Network (T-SCNN). Our T-SCNN is composed of two stages, i.e., target locating and target recognition. In the stage of target locating, we first detect all suspected targets from the total image dataset by presenting a minimum bounding rectangle with threshold (MBRT) approach, then cut out all regions encompassing targets to generate target images for training. In the stage of target recognition, we send target images to the well-trained recognition network for identification.

MERITS:

Extensive experiments are performed on our synthetic space target image dataset, and the result demonstrate that the proposed method achieves high accuracy within a short time.

DEMERITS:

The results are very poor in accuracy.

5. **TITLE:SAR Target Configuration Recognition via Two-Stage Sparse Structure Representation,2015**

AUTHOR : Jifang Pei, Student,Member, IEEE, Yulin Huang, Member, IEEE, Weibo Huo,Yin Zhang, Member, IEEE.

YEAR : 2017

DESCRIPTION:

A two-stage sparse structure representation algorithm which can preserve the manifold structure of the data is proposed for synthetic aperture radar target configuration recognition in this paper. Manifold structure of the data is preserved by two stages. In the training stage, taking advantage of both the sparse representation (SR) and manifold learning, local structure of the data is preserved in the reconstruction space, where SR-based recognition is realized. In the testing stage, two structure preserving factors based on the testing samples are embedded into the SR model to enhance structure preserving performance. The first one is constructed to preserve the local structure of the testing samples, which can guarantee the samples that are close to each other in the original space will also be close to each other in the sparse space. And the second one is established to preserve the distant structure of the testing samples, which can ensure the samples that are far from each other in the original space will also be far from each other in the sparse space. Manifold structure of the data is well captured and preserved by two stages.

MERITS:

Experimental results on the moving and stationary target acquisition and recognition database demonstrate the effectiveness of the proposed algorithm.

DEMERITS:

The system is too slow for processing to achieve the results.

CHAPTER III

SYSTEM ANALYSIS

3.1Existing System

A supervised nonlinear dictionary learning (DL) method, called multiscale supervised kernel DL (MSK-DL), is proposed for target recognition in synthetic aperture radar (SAR) images. We use Frost filters with different parameters to extract an SAR image's multiscale features for data augmentation and noise suppression. In order to reduce the computation cost, the dimension of each scale feature is reduced by principal component analysis (PCA). Instead of the widely used linear DL, we learn multiple nonlinear dictionaries to capture the nonlinear structure of data by introducing the dimension reduced features into the nonlinear reconstruction error terms. A classification model, which is defined as a discriminative classification error term, is learned simultaneously. Hence, the objective function contains the nonlinear reconstruction error terms and a classification error term. Two optimization algorithms, called multiscale supervised kernel K-singular value decomposition (MSK-KSVD) and multiscale supervised incremental kernel DL (MSIK-DL), are proposed to compute the multi-dictionary and the classifier. Experiments on the moving and stationary target automatic recognition (MSTAR) data set are performed to evaluate the effectiveness of the two proposed algorithms. And the experimental results demonstrate that the proposed scheme outperforms some representative common machine learning strategies, state-of-the-art convolutional neural network (CNN) models and some representative DL methods, especially in terms of its robustness against training set size and noise.

3.2 Proposed System

Recent advances in deep learning made tasks such as Image and speech recognition possible. Deep Learning is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. The continuous innovation and development of satellite technology has brought the world closer together. At present, there are thousands of artificial satellites in the world, and the number of spacecraft working in orbit is increasing. With the constant exploration of outer space, there is inevitably a large quantity of space debris, e.g., lacquer, satellite debris. How to distinguish space targets of interested from the massive debris is a hot topic and of great significance. The effective separation of space targets and space debris can ensure the safe operation of on-orbit spacecraft. But, nowadays with the advancement of technology there is still a lag in predicting the space crafts and the related targets in an accurate manner. In this project we will be focusing on developing a novel hybrid architecture for determining the different types of satellite images, where four different kind of pre-processing techniques as well as two different optimization techniques will be used to increase the accuracy of the proposed hybrid model. The concept of network surgery will be used for implementing the hybrid algorithm development. Thus by this project we will be able to determine any kind of satellite images given as an input to the generated model. Thus, we propose a solution for the determination of spacecraft with the most accurate prediction by developing a novel architecture.

3.3 Requirement Analysis and Specification

3.3.1 Input Requirements

Input requirements consists of .jpg, .png images of Synthetic Aperture Radar (SAR) images.

3.3.2 Output Requirements

Output requirements contains test image with satellites identified in it using squeeze net algorithm.

3.3.3 Functional Requirements

Functional requirements has Keras model and Json model

3.4 Technology stack

Domain -Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Front end- Python 3.9

- **OpenCV**
- **NumPy**
- **scikit-learn**
- **pandas**
- **matplotlib**
- **keras**

OpenCV

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. It makes use of Numpy, which is a highly optimized library for numerical operations with MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. It is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays. This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib. It is an open source computer vision and machine learning software library . It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

NumPy

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages which cannot be missed while learning data science , mainly because this library provides an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient. To make a numpy array, np.array() function is used.

Scikit-learn

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

The functionality that scikit-learn provides include:

- **Regression**, including Linear and Logistic Regression

- **Classification**, including K-Nearest Neighbors
- **Clustering**, including K-Means and K-Means++
- **Model selection**
- **Preprocessing**, including Min-Max Normalization

Pandas

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

We can analyze data in pandas with:

- ✓ Series
- ✓ DataFrames

Series

Series is one dimensional (1-D) array defined in pandas that can be used to store any data type.

Data Frames

Data Frames is two-dimensional (2-D) data structure defined in pandas which consists of rows and columns.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It consists of several plots like line, bar, scatter, histogram etc.

Keras

Keras is an open-source neural-network **library** written in **Python**. It is capable of running on top of Tensor Flow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It is developed using four guiding principles:

1. Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

2. Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

3. Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

4. Python: No separate model files with custom file formats. Everything is native Python.

Back end - TENSORFLOW 2.0

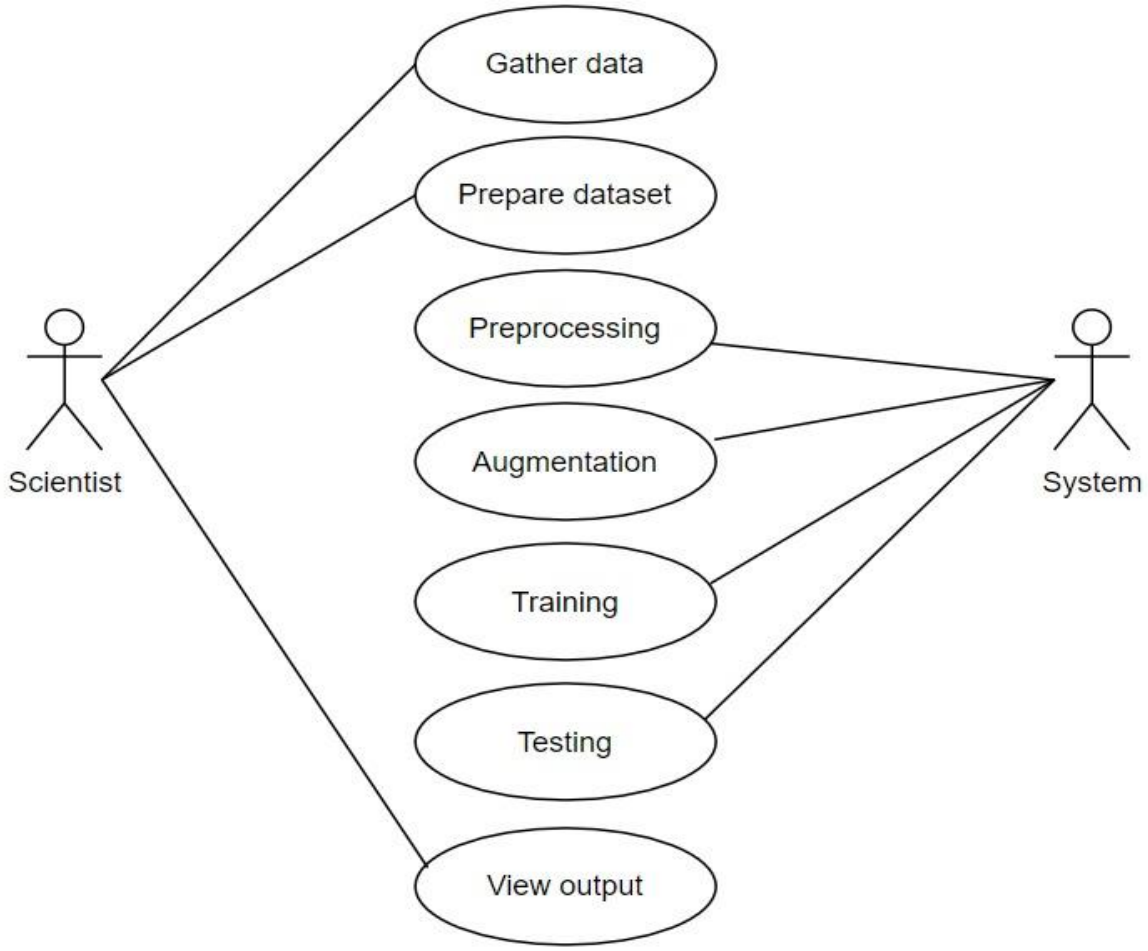
Tensor Flow is an end-to-end open source platform for deep learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in deep learning and developers easily build and deploy deep learning powered applications. It is a general-purpose high-performance computing library open-sourced by Google in 2015. Since the beginning, its main focus was to provide high-performance APIs for building Neural Networks (NNs). However, with the advance of time and interest by the Deep learning community, the lib has grown to a full deep learning ecosystem.

CHAPTER IV

SYSTEM DESIGN

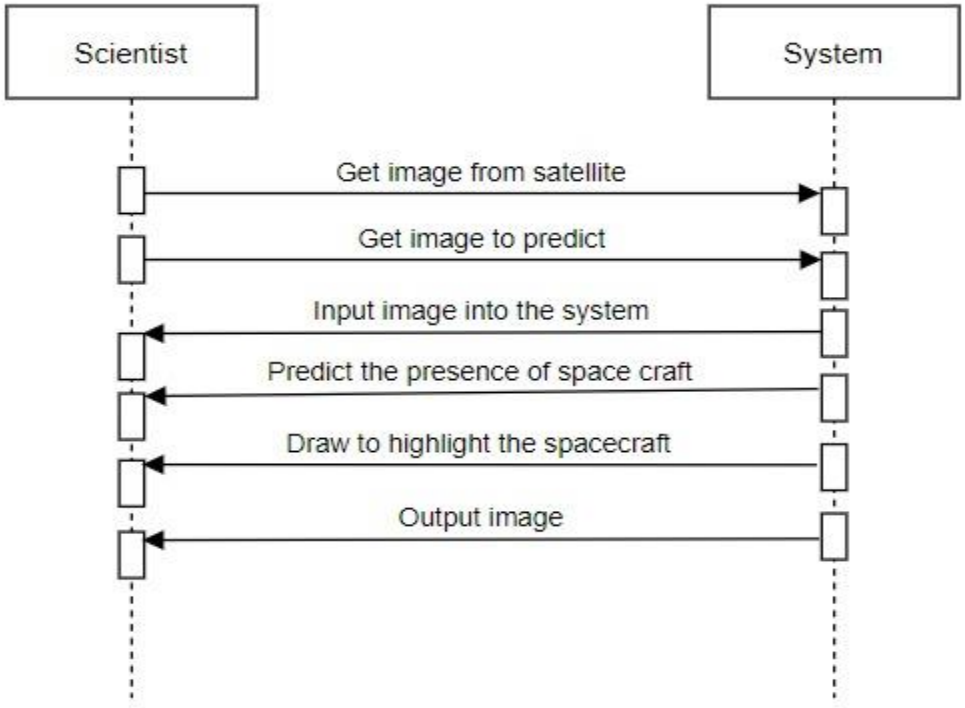
4.1Uml diagram

4.1.1Use case diagram



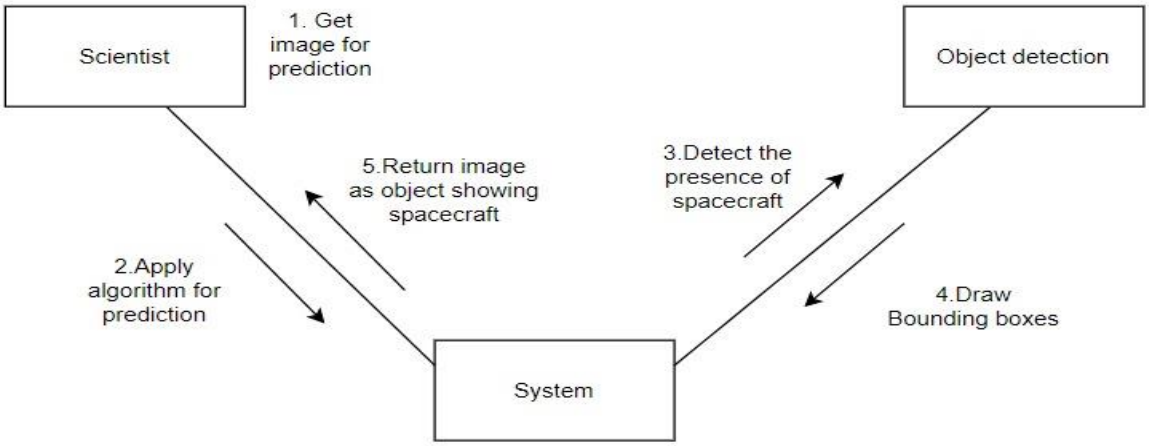
There are two users scientist and the system. The scientist will gather the data and prepare the data whereas the system will preprocess them. After preprocessing, augmentation process will takes place. Then data set will get trained and will be tested and the output can be viewed.

4.1.2 Sequence diagram



In the sequence diagram, the scientist will get the image from the satellite and send is to the system. Once the system gets the image training process will undergo which helps in predicting the presence of space craft and draw the bounding box to give the output.

4.1.3 Colaboration diagram



In the collaboration diagram, scientist get the image and give it to the system. The system will apply the algorithm and detect the presence of spacecraft and draw the bounding boxes and return the image to the scientist.

CHAPTER V

SYSTEM ARCHITECTURE

The overall system architecture of the Image identification and recognition using novel hybrid architecture development for satellite images is shown in the Figure 5.1. Data is collected from the public source. As there are limited no of dataset available, and training requires a large number of data, Data augmentation can help a lot. Number of images is increased through rotation, shifting, flipping, blurring, cropping, noising, and their combination. Thus the given data is augmented. Later in preprocessing technique, Aspect aware algorithm is used and the image is converted to array. Then this preprocessed data will enter into the novel architecture for training. In this architecture the SqeezeNet algorithm is modified. After the training, for the model generation, optimization and loss minimization is done. Once these processes are completed, evaluation and validation is done. COCO(Common Objects in Context) algorithm which is a large-scale object detection, segmentation, and captioning dataset is used. Thus by using COCO algorithm space craft will be detected. Once the space craft is detected, the bounding boxes are drawn.

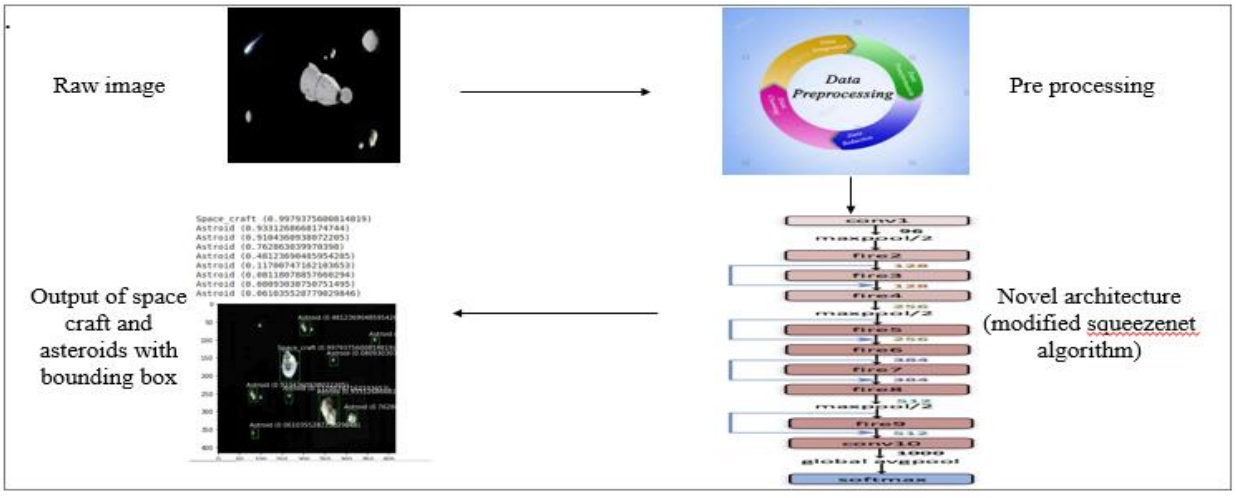


Figure 5.1 System Architecture

5.1 Architecture Overview

The project is focused on developing a novel hybrid architecture for determining the different types of satellite images, where four different kind of pre-processing techniques as well as two different optimization techniques are used to increase the accuracy of the proposed hybrid model. The concept of network surgery is used for implementing the hybrid algorithm development. Thus by the project we will be able to determine any kind of satellite images given as an input to the generated model. Thus, a solution is proposed for the determination of spacecraft with the most accurate prediction by developing a novel architecture.

5.2Module Design Specification

Dataset collection module

A data set is a collection of data. Deep Learning has become the go-to method for solving many challenging real-world problems. It's definitely by far the best performing method for computer vision tasks. The image above showcases the power of deep learning for computer vision. With enough training, a deep network can segment and identify the "key points" of every person in the image. These deep learning machines that have been working so well need fuel lots of fuel; that fuel is data. The more labelled data available, the better our model performs. The idea of more data leading to better performance has even been explored at a large-scale by Google with a dataset of 300 Million images! When deploying a Deep Learning model in a real-world application, data must be constantly fed to continue improving its performance. And, in the deep learning era, data is very well arguably the most valuable resource. There are three steps of collecting data

Scraping From the Web

Manually finding and downloading images takes a long time simply due to the amount of human work involved. The task probably has some kind of common objects are to be detected. And so that becomes the keyword for web-scraping. It also becomes the class name for that object. From the sounds of it this is of course very easy for a task such as image classification where the images annotations are quite coarse. But to do something like instance segmentation? Every *single pixel in the image is required.* To get those, it's best to use some really great image annotation tools that are already out there. The paper shows how to create a model that, given a rough set of polygon points around an object, can generate the pixel labels for segmentation. Deep extreme cut is also quite similar except they use only the four extreme points around the object. This will then give some nice bounding box and segmentation labels. Another option is to use an existing image annotation GUIs. Label someone very popular where one can draw both bounding boxes and set polygon points for segmentation maps. Amazon Mechanical Turk is also a cheap option.

Third-party

Since data has become such a valuable commodity in the deep learning era, many start-ups have started to offer their own image annotation services they'll gather and label the data. Given a description of what kind of data and annotations needed. Mighty is one that has been doing self-driving car image annotation and has become pretty big in the space were at CVPR 2018 too. Payment AI are less specialized than Mighty AI, offering image annotation for any domain. They also offer a couple more tools such as video and landmark annotations.

Dataset augmentation

The performance of deep learning neural networks often improves with the amount of data available.Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples.

Image data augmentation is perhaps the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image.Transforms include a range of operations from the field of image manipulation, such as shifts, flips, zooms, and much more.

The intent is to expand the training dataset with new, plausible examples. This means, variations of the training set images that are likely to be seen by the model. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from the left or right. A vertical flip of the photo of a cat does not make sense and would probably not be appropriate given that the model is very unlikely to see a photo of an upside-down cat.

As such, it is clear that the choice of the specific data augmentation techniques used for a training dataset must be chosen carefully and within the context of the training dataset and knowledge of the problem domain. In addition, it can be useful to experiment with data augmentation methods in isolation and in concert to see if they result in a measurable improvement to model performance, perhaps with a small prototype dataset, model, and training run.

Modern deep learning algorithms, such as the convolutional neural network, or CNN, can learn features that are invariant to their location in the image. Nevertheless, augmentation can further aid in this transform invariant approach to learning and can aid the model in learning features that are also invariant to transforms such as left-to-right to top-to-bottom ordering, light levels in photographs, and more.

Image data augmentation is typically only applied to the training dataset, and not to the validation or test dataset. This is different from data preparation such as image resizing and pixel scaling; they must be performed consistently across all datasets that interact with the model.

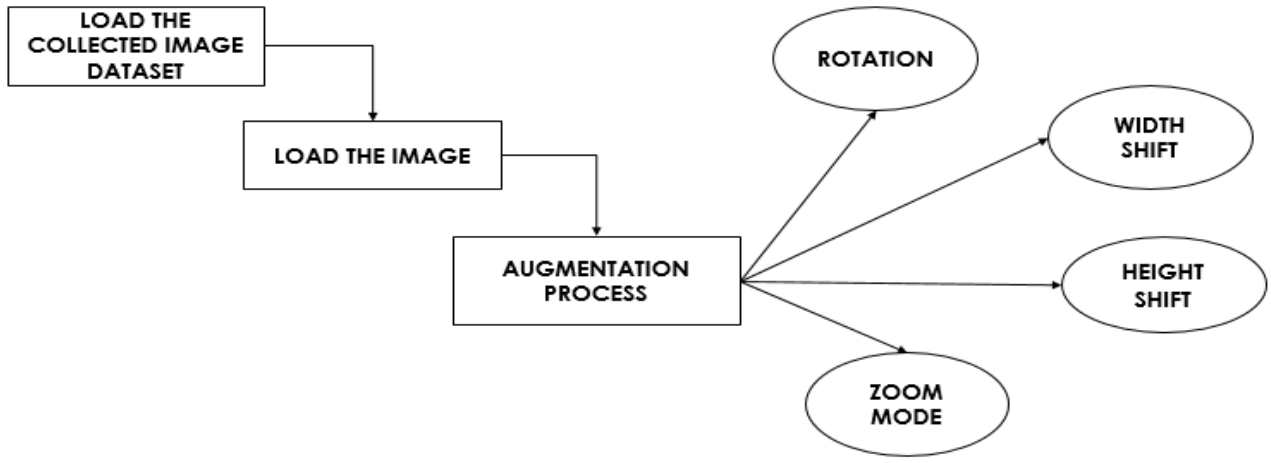


Figure 5.2 Data augmentation

Dataset preprocessing

Deep learning has truly come into the mainstream in the past few years. Deep learning uses neural nets with a lot of hidden layers (dozens in today's state of the

art) and requires large amounts of training data. These models have been particularly effective in gaining insight and approaching human-level accuracy in perceptual tasks like vision, speech, language processing. The theory and mathematical foundations were laid several decades ago. Primarily two phenomena have contributed to the rise of machine learning a) Availability of huge data-sets/training examples in multiple domains and b) Advances in raw compute power and the rise of efficient parallel hardware.

Building an effective neural network model requires careful consideration of the network architecture as well as the input data format. This article deals with the latter. The most common image data input parameters are the number of images, image height, image width, number of channels, and the number of levels per pixel.

Uniform aspect ratio: One of the first steps is to ensure that the images have the same size and aspect ratio. Most of the neural network models assume a square shape input image, which means that each image needs to be checked if it is a square or not, and cropped appropriately. Cropping can be done to select a square part of the image, as shown. While cropping, we usually care about the part in the center.

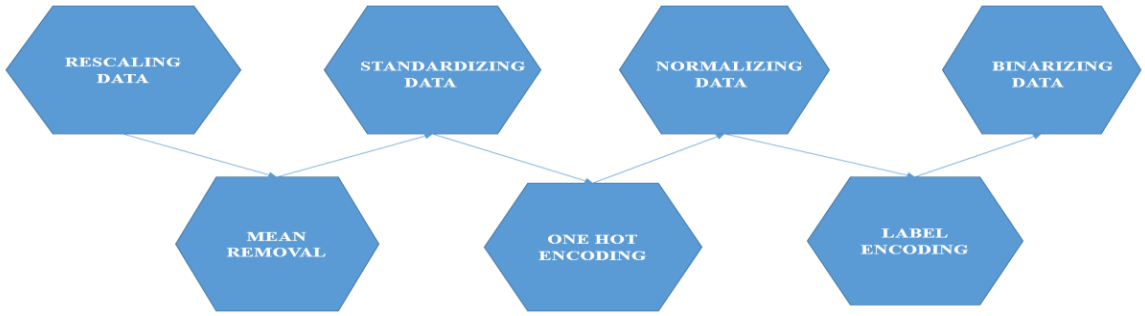


Figure 5.3 Data pre-processing

Training with novel architecture

The project is developed by using a novel architecture by effectively modifying the squeezenet architecture.

The main ideas of the novel architecture are:

1. Using 1x1(point-wise) filters to replace 3x3 filters, as the former only 1/9 of computation.
2. Using 1x1 filters as a bottleneck layer to reduce depth to reduce computation of the following 3x3 filters.
3. Downsample late to keep a big feature map.

The building brick of Novel architecture is called fire module, which contains two layers: a squeeze layer and an expand layer. A Novel architecture stackes a bunch of fire modules and a few pooling layers. The squeeze layer and expand layer keep the same feature map size, while the former reduce the depth to a smaller number, the later increase it. The squeezing (bottoleneck layer) and expansion behavior is common in neural architectures. Another common pattern is increasing depth while reducing feature map size to get high level abstract.

As shown in the below chart, the squeeze module only contains 1x1 filters, which means it works like a fully-connected layer working on feature points in the same position. In other words, it doesn't have the ability of spatial abstract. As its name says, one of its benifits is to reduce the depth of feature map. Reducing depth

means the following 3x3 filters in the expand layer has fewer computation to do. It boosts the speed as a 3x3 filter need as 9 times computation as a 1x1 filter. By intuition, too much squeezing limits information flow; too few 3x3 filters limits space resolution.

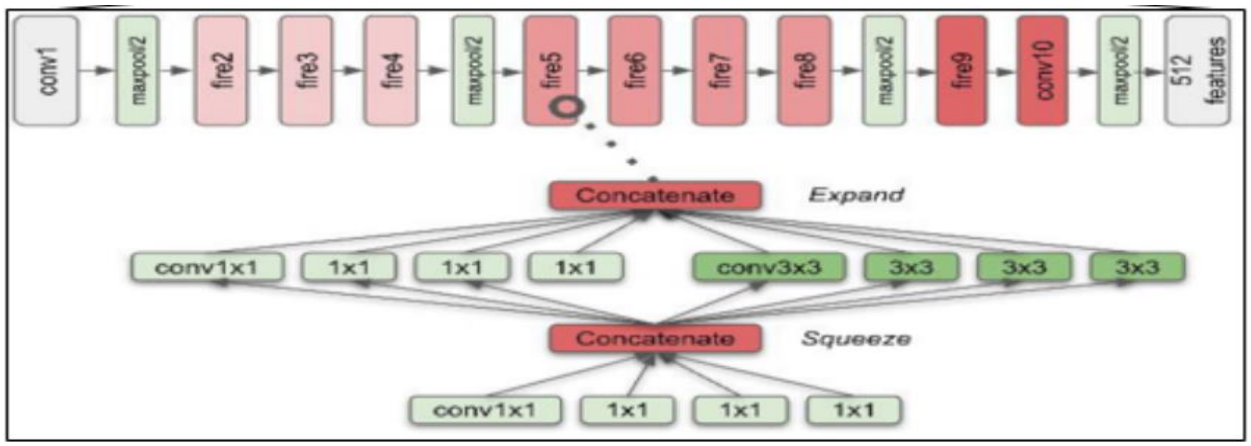


Figure 5.4: Novel architecture

Object detection:

Object detection is a process of finding all the possible instances of real-world objects, such as human faces, flowers, cars, etc. in images or videos, in real-time with utmost accuracy. The object detection technique uses derived features and learning algorithms to recognize all the occurrences of an object category.

Object detection technique helps in the recognition, detection, and localization of multiple visual instances of objects in an image or a video. It provides a much better understanding of the object as a whole, rather than just basic object classification. This method can be used to count the number of instances of unique objects and mark their precise locations, along with labelling. With time, the

performance of this process has also improved significantly, helping us with real-time use cases. All in all, it answers the question: “What object is where and how much of it is there?”

The technical evolution of object detection started in the early 2000s and the detectors at that time. They followed the low-level and mid-level vision and followed the method of ‘recognition-by-components’. This method enabled object detection as a measurement of similarity between the object components, shapes, and contours, and the features that were taken into consideration were distance transforms, shape contexts, and edgeless, etc. Things did not go well and then machine detection methods started to come into the picture to solve this problem.

Multi-scale detection of objects was to be done by taking those objects into consideration that had “different sizes” and “different aspect ratios”. This was one of the main technical challenges in object detection in the early phases. But, after 2014, with the increase in technical advancements, the problem was solved. This brought us to the second phase of object detection, where the tasks were accomplished using deep learning.

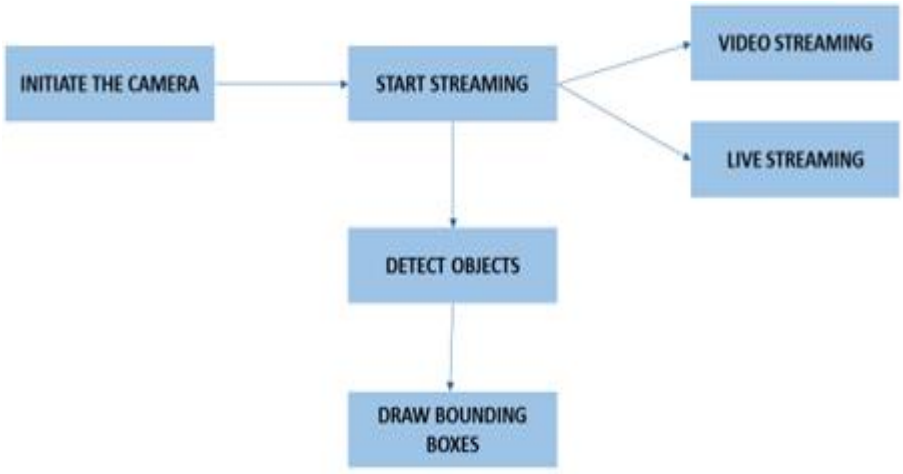


Figure 5.5 Object Detection

CHAPTER VI

SYSTEM IMPLEMENTATION

6.1Coding

Model.py

```
import torch.nn as nn
import torch
import math
from efficientnet_pytorch import EfficientNet as EffNet
from torchvision.ops.boxes import nms as nms_torch

def nms(dets, thresh):
    return nms_torch(dets[:, :4], dets[:, 4], thresh)


class ConvBlock(nn.Module):
    def __init__(self, num_channels):
        super(ConvBlock, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(num_channels, num_channels, kernel_size=3, stride=1, padding
=1, groups=num_channels),
            nn.Conv2d(num_channels, num_channels, kernel_size=1, stride=1, padding
=0),
            nn.BatchNorm2d(num_features=num_channels, momentum=0.9997, eps=4
e-5), nn.ReLU())

    def forward(self, input):
        return self.conv(input)
```

```
class BiFPN(nn.Module):
    def __init__(self, num_channels, epsilon=1e-4):
        super(BiFPN, self).__init__()
        self.epsilon = epsilon
        # Conv layers
        self.conv6_up = ConvBlock(num_channels)
        self.conv5_up = ConvBlock(num_channels)
        self.conv4_up = ConvBlock(num_channels)
        self.conv3_up = ConvBlock(num_channels)
        self.conv4_down = ConvBlock(num_channels)
        self.conv5_down = ConvBlock(num_channels)
        self.conv6_down = ConvBlock(num_channels)
        self.conv7_down = ConvBlock(num_channels)

        # Feature scaling layers
        self.p6_upsample = nn.Upsample(scale_factor=2, mode='nearest')
        self.p5_upsample = nn.Upsample(scale_factor=2, mode='nearest')
        self.p4_upsample = nn.Upsample(scale_factor=2, mode='nearest')
        self.p3_upsample = nn.Upsample(scale_factor=2, mode='nearest')

        self.p4_downsample = nn.MaxPool2d(kernel_size=2)
        self.p5_downsample = nn.MaxPool2d(kernel_size=2)
        self.p6_downsample = nn.MaxPool2d(kernel_size=2)
        self.p7_downsample = nn.MaxPool2d(kernel_size=2)

        # Weight
```

```
    self.p6_w1 = nn.Parameter(torch.ones(2))
    self.p6_w1_relu = nn.ReLU()
    self.p5_w1 = nn.Parameter(torch.ones(2))
    self.p5_w1_relu = nn.ReLU()
    self.p4_w1 = nn.Parameter(torch.ones(2))
    self.p4_w1_relu = nn.ReLU()
    self.p3_w1 = nn.Parameter(torch.ones(2))
    self.p3_w1_relu = nn.ReLU()

    self.p4_w2 = nn.Parameter(torch.ones(3))
    self.p4_w2_relu = nn.ReLU()
    self.p5_w2 = nn.Parameter(torch.ones(3))
    self.p5_w2_relu = nn.ReLU()
    self.p6_w2 = nn.Parameter(torch.ones(3))
    self.p6_w2_relu = nn.ReLU()
    self.p7_w2 = nn.Parameter(torch.ones(2))
    self.p7_w2_relu = nn.ReLU()

def forward(self, inputs):
    """
        P7_0 -------------------------- P7_2 -------->

        P6_0 ---------- P6_1 ---------- P6_2 -------->

        P5_0 ---------- P5_1 ---------- P5_2 -------->
```

```
        P4_0 ---------- P4_1 ---------- P4_2 -------->

        P3_0 -------------------------- P3_2 -------->
        """

        # P3_0, P4_0, P5_0, P6_0 and P7_0
        p3_in, p4_in, p5_in, p6_in, p7_in = inputs
        # P7_0 to P7_2
        # Weights for P6_0 and P7_0 to P6_1
        p6_w1 = self.p6_w1_relu(self.p6_w1)
        weight = p6_w1 / (torch.sum(p6_w1, dim=0) + self.epsilon)
        # Connections for P6_0 and P7_0 to P6_1 respectively
        p6_up = self.conv6_up(weight[0] * p6_in + weight[1] * self.p6_upsample(p7
_in))
        # Weights for P5_0 and P6_0 to P5_1
        p5_w1 = self.p5_w1_relu(self.p5_w1)
        weight = p5_w1 / (torch.sum(p5_w1, dim=0) + self.epsilon)
        # Connections for P5_0 and P6_0 to P5_1 respectively
        p5_up = self.conv5_up(weight[0] * p5_in + weight[1] * self.p5_upsample(p6
_up))
        # Weights for P4_0 and P5_0 to P4_1
        p4_w1 = self.p4_w1_relu(self.p4_w1)
        weight = p4_w1 / (torch.sum(p4_w1, dim=0) + self.epsilon)
        # Connections for P4_0 and P5_0 to P4_1 respectively
        p4_up = self.conv4_up(weight[0] * p4_in + weight[1] * self.p4_upsample(p5
_up))
```

```
        # Weights for P3_0 and P4_1 to P3_2
        p3_w1 = self.p3_w1_relu(self.p3_w1)
        weight = p3_w1 / (torch.sum(p3_w1, dim=0) + self.epsilon)
        # Connections for P3_0 and P4_1 to P3_2 respectively
        p3_out = self.conv3_up(weight[0] * p3_in + weight[1] * self.p3_upsample(p4
_up))

        # Weights for P4_0, P4_1 and P3_2 to P4_2
        p4_w2 = self.p4_w2_relu(self.p4_w2)
        weight = p4_w2 / (torch.sum(p4_w2, dim=0) + self.epsilon)
        # Connections for P4_0, P4_1 and P3_2 to P4_2 respectively
        p4_out = self.conv4_down(
            weight[0] * p4_in + weight[1] * p4_up + weight[2] * self.p4_downsample(
p3_out))
        # Weights for P5_0, P5_1 and P4_2 to P5_2
        p5_w2 = self.p5_w2_relu(self.p5_w2)
        weight = p5_w2 / (torch.sum(p5_w2, dim=0) + self.epsilon)
        # Connections for P5_0, P5_1 and P4_2 to P5_2 respectively
        p5_out = self.conv5_down(
            weight[0] * p5_in + weight[1] * p5_up + weight[2] * self.p5_downsample(
p4_out))
        # Weights for P6_0, P6_1 and P5_2 to P6_2
        p6_w2 = self.p6_w2_relu(self.p6_w2)
        weight = p6_w2 / (torch.sum(p6_w2, dim=0) + self.epsilon)
        # Connections for P6_0, P6_1 and P5_2 to P6_2 respectively
        p6_out = self.conv6_down(
```

```
            weight[0] * p6_in + weight[1] * p6_up + weight[2] * self.p6_downsample(
p5_out))
        # Weights for P7_0 and P6_2 to P7_2
        p7_w2 = self.p7_w2_relu(self.p7_w2)
        weight = p7_w2 / (torch.sum(p7_w2, dim=0) + self.epsilon)
        # Connections for P7_0 and P6_2 to P7_2
        p7_out = self.conv7_down(weight[0] * p7_in + weight[1] * self.p7_downsam
ple(p6_out))

        return p3_out, p4_out, p5_out, p6_out, p7_out


class Regressor(nn.Module):
    def __init__(self, in_channels, num_anchors, num_layers):
        super(Regressor, self).__init__()
        layers = []
        for _ in range(num_layers):
            layers.append(nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=
1, padding=1))
            layers.append(nn.ReLU(True))
        self.layers = nn.Sequential(*layers)
        self.header = nn.Conv2d(in_channels, num_anchors * 4, kernel_size=3, stride
=1, padding=1)

    def forward(self, inputs):
        inputs = self.layers(inputs)
```

```
inputs = self.header(inputs)
        output = inputs.permute(0, 2, 3, 1)
        return output.contiguous().view(output.shape[0], -1, 4)


class Classifier(nn.Module):
    def __init__(self, in_channels, num_anchors, num_classes, num_layers):
        super(Classifier, self).__init__()
        self.num_anchors = num_anchors
        self.num_classes = num_classes
        layers = []
        for _ in range(num_layers):
            layers.append(nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=
1, padding=1))
            layers.append(nn.ReLU(True))
        self.layers = nn.Sequential(*layers)
        self.header = nn.Conv2d(in_channels, num_anchors * num_classes, kernel_si
ze=3, stride=1, padding=1)
        self.act = nn.Sigmoid()

    def forward(self, inputs):
        inputs = self.layers(inputs)
        inputs = self.header(inputs)
        inputs = self.act(inputs)
        inputs = inputs.permute(0, 2, 3, 1)
```

```
output = inputs.contiguous().view(inputs.shape[0], inputs.shape[1], inputs.shape[2]
, self.num_anchors,
                                  self.num_classes)
        return output.contiguous().view(output.shape[0], -1, self.num_classes)


class EfficientNet(nn.Module):
    def __init__(self, ):
        super(EfficientNet, self).__init__()
        model = EffNet.from_pretrained('efficientnet-b0')
        del model._conv_head
        del model._bn1
        del model._avg_pooling
        del model._dropout
        del model._fc
        self.model = model

    def forward(self, x):
        x = self.model._swish(self.model._bn0(self.model._conv_stem(x)))
        feature_maps = []
        for idx, block in enumerate(self.model._blocks):
            drop_connect_rate = self.model._global_params.drop_connect_rate
            if drop_connect_rate:
                drop_connect_rate *= float(idx) / len(self.model._blocks)
            x = block(x, drop_connect_rate=drop_connect_rate)
            if block._depthwise_conv.stride == [2, 2]:
```

```
            feature_maps.append(x)

        return feature_maps[1:]


class EfficientDet(nn.Module):
    def __init__(self, num_anchors=9, num_classes=20, compound_coef=0):
        super(EfficientDet, self).__init__()
        self.compound_coef = compound_coef

        self.num_channels = [64, 88, 112, 160, 224, 288, 384, 384][self.compound_co
ef]

        self.conv3 = nn.Conv2d(40, self.num_channels, kernel_size=1, stride=1, padd
ing=0)
        self.conv4 = nn.Conv2d(80, self.num_channels, kernel_size=1, stride=1, padd
ing=0)
        self.conv5 = nn.Conv2d(192, self.num_channels, kernel_size=1, stride=1, pad
ding=0)
        self.conv6 = nn.Conv2d(192, self.num_channels, kernel_size=3, stride=2, pad
ding=1)
        self.conv7 = nn.Sequential(nn.ReLU(),
                                   nn.Conv2d(self.num_channels, self.num_channels, kernel_si
ze=3, stride=2, padding=1))

        self.bifpn = nn.Sequential(*[BiFPN(self.num_channels) for _ in range(min(2
```

```
+ self.compound_coef, 8))])

        self.num_classes = num_classes
        self.regressor = Regressor(in_channels=self.num_channels, num_anchors=nu
m_anchors,
                                   num_layers=3 + self.compound_coef // 3)
        self.classifier = Classifier(in_channels=self.num_channels, num_anchors=nu
m_anchors, num_classes=num_classes,
                                     num_layers=3 + self.compound_coef // 3)

        self.anchors = Anchors()
        self.regressBoxes = BBoxTransform()
        self.clipBoxes = ClipBoxes()
        self.focalLoss = FocalLoss()

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                n = m.kernel_size[0] * m.kernel_size[1] * m.out_channels
                m.weight.data.normal_(0, math.sqrt(2. / n))
            elif isinstance(m, nn.BatchNorm2d):
                m.weight.data.fill_(1)
                m.bias.data.zero_()

        prior = 0.01

        self.classifier.header.weight.data.fill_(0)
```

```
        self.classifier.header.bias.data.fill_(-math.log((1.0 - prior) / prior))

        self.regressor.header.weight.data.fill_(0)
        self.regressor.header.bias.data.fill_(0)

        self.backbone_net = EfficientNet()

    def freeze_bn(self):
        for m in self.modules():
            if isinstance(m, nn.BatchNorm2d):
                m.eval()

    def forward(self, inputs):
        if len(inputs) == 2:
            is_training = True
            img_batch, annotations = inputs
        else:
            is_training = False
            img_batch = inputs

        c3, c4, c5 = self.backbone_net(img_batch)
        p3 = self.conv3(c3)
        p4 = self.conv4(c4)
        p5 = self.conv5(c5)
        p6 = self.conv6(c5)
        p7 = self.conv7(p6)
```

```
        features = [p3, p4, p5, p6, p7]
        features = self.bifpn(features)

        regression = torch.cat([self.regressor(feature) for feature in features], dim=1)
        classification = torch.cat([self.classifier(feature) for feature in features], dim=
1)
        anchors = self.anchors(img_batch)

        if is_training:
            return self.focalLoss(classification, regression, anchors, annotations)
        else:
            transformed_anchors = self.regressBoxes(anchors, regression)
            transformed_anchors = self.clipBoxes(transformed_anchors, img_batch)

            scores = torch.max(classification, dim=2, keepdim=True)[0]

            scores_over_thresh = (scores > 0.05)[0, :, 0]

            if scores_over_thresh.sum() == 0:
                return [torch.zeros(0), torch.zeros(0), torch.zeros(0, 4)]

            classification = classification[:, scores_over_thresh, :]
            transformed_anchors = transformed_anchors[:, scores_over_thresh, :]
            scores = scores[:, scores_over_thresh, :]

            anchors_nms_idx = nms(torch.cat([transformed_anchors, scores], dim=2)
```

```
        [0, :, :], 0.5)

            nms_scores, nms_class = classification[0, anchors_nms_idx, :].max(dim=1)

            return [nms_scores, nms_class, transformed_anchors[0, anchors_nms_idx, :
]]


if __name__ == '__main__':
    from tensorboardX import SummaryWriter
    def count_parameters(model):
        return sum(p.numel() for p in model.parameters() if p.requires_grad)

    model = EfficientDet(num_classes=80)
    print (count_parameters(model))
```

Inference.py

```
import os
import argparse
import torch
import torch.nn as nn
from torch.utils.data import DataLoader
from torchvision import transforms
from tensorboardX import SummaryWriter
import shutil
import numpy as np
```

```
from tqdm.autonotebook import tqdm
import cv2
import time as time

class Infer():
    def __init__(self, verbose=1):
        self.system_dict = {};
        self.system_dict["verbose"] = verbose;
        self.system_dict["local"] = {};
        self.system_dict["local"]["common_size"] = 512;
        self.system_dict["local"]["mean"] = np.array([[[0.485, 0.456, 0.406]]])
        self.system_dict["local"]["std"] = np.array([[[0.229, 0.224, 0.225]]])

    def Model(self, model_dir="drive/MyDrive/trained/"):
        self.system_dict["local"]["model"] = torch.load(model_dir + "/signatrix_efficientdet_coco.pth").module
        if torch.cuda.is_available():
            self.system_dict["local"]["model"] = self.system_dict["local"]["model"].cuda();

    def Predict(self, img_path, class_list, vis_threshold = 0.4,output_folder = 'Inference'):

        if not os.path.exists(output_folder):
            os.makedirs(output_folder)
```

```
 image_filename = os.path.basename(img_path)
        img = cv2.imread(img_path);
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB);
        image = img.astype(np.float32) / 255.;
        image = (image.astype(np.float32) - self.system_dict["local"]["mean"]) / self.s
ystem_dict["local"]["std"]
        height, width, _ = image.shape
        if height > width:
            scale = self.system_dict["local"]["common_size"] / height
            resized_height = self.system_dict["local"]["common_size"]
            resized_width = int(width * scale)
        else:
            scale = self.system_dict["local"]["common_size"] / width
            resized_height = int(height * scale)
            resized_width = self.system_dict["local"]["common_size"]

        image = cv2.resize(image, (resized_width, resized_height))

        new_image = np.zeros((self.system_dict["local"]["common_size"], self.syste
m_dict["local"]["common_size"], 3))
        new_image[0:resized_height, 0:resized_width] = image

        img = torch.from_numpy(new_image)

        t0 = time.time()
        with torch.no_grad():
```

```
            if torch.cuda.is_available():
                scores, labels, boxes = self.system_dict["local"]["model"](img.cuda().per
mute(2, 0, 1).float().unsqueeze(dim=0))
            else:
                scores, labels, boxes = self.system_dict["local"]["model"](img.permute(2
, 0, 1).float().unsqueeze(dim=0))
            boxes /= scale;
        duration = time.time() - t0
        print('Done. (%.3fs)' % (time.time() - t0))


        try:
            if boxes.shape[0] > 0:
                output_image = cv2.imread(img_path)

                for box_id in range(boxes.shape[0]):
                    pred_prob = float(scores[box_id])
                    if pred_prob < vis_threshold:
                        break
                    pred_label = int(labels[box_id])
                    xmin, ymin, xmax, ymax = boxes[box_id, :]
                    color = colors[pred_label]
                    cv2.rectangle(output_image, (xmin, ymin), (xmax, ymax), color, 2)
                    text_size = cv2.getTextSize(class_list[pred_label] + ' : %.2f' % pred_p
rob, cv2.FONT_HERSHEY_PLAIN, 1, 1)[0]
```

```
                cv2.rectangle(output_image, (xmin, ymin), (xmin + text_size[0] + 3, y
min + text_size[1] + 4), color, -1)
                cv2.putText(
                    output_image, class_list[pred_label] + ' : %.2f' % pred_prob,
                    (xmin, ymin + text_size[1] + 4), cv2.FONT_HERSHEY_PLAIN, 1,
                    (255, 255, 255), 1)

            #cv2.imwrite(os.path.join(output_folder, image_filename), output_image)
            #cv2.imwrite("output.jpg", output_image)
            return duration, scores, labels, boxes

        except:
            print("NO Object Detected")
            return None

    def predict_batch_of_images(self, img_folder, class_list, vis_threshold = 0.4, ou
tput_folder='Inference'):

        all_filenames = os.listdir(img_folder)
        all_filenames.sort()
        generated_count = 0
        for filename in all_filenames:
            img_path = "{}/{}".format(img_folder, filename)
            try:
                self.Predict(img_path , class_list, vis_threshold ,output_folder)
                generated_count += 1
```

```
            except:
                continue
        print("Objects detected  for {} images".format(generated_count))

gtf = Infer()
gtf.Model(model_dir="drive/MyDrive/trained_weight/")

#extract class list from our annotations
import json
with open('space/train/_annotations.coco.json') as json_file:
    data = json.load(json_file)
class_list = []
for category in data['categories']:
  class_list.append(category['name'])

test_images = [f for f in os.listdir('space/test') if f.endswith('.jpg')]
import random
img_path = "space/test/" + random.choice(test_images);
#img_path = "space/test/" +"trial.jpg";

#img_path="Downloads/trial.jpg";
duration, scores, labels, boxes = gtf.Predict(img_path, class_list, vis_threshold=0.2
)
pred = {}
pred['boxes'] = boxes
pred['scores'] = scores
```

```
pred['labels'] = labels

import torchvision

def apply_nms(orig_prediction, iou_thresh=0.3):

    # torchvision returns the indices of the bboxes to keep
    keep = torchvision.ops.nms(orig_prediction['boxes'], orig_prediction['scores'], io
u_thresh)

    final_prediction = orig_prediction
    final_prediction['boxes'] = final_prediction['boxes'][keep]
    final_prediction['scores'] = final_prediction['scores'][keep]
    final_prediction['labels'] = final_prediction['labels'][keep]

    return final_prediction

pred = apply_nms(pred, iou_thresh=0.01)
import matplotlib.pyplot as plt
import matplotlib.patches as patches
plt.rcParams['text.color'] = 'white'
img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
fig, a = plt.subplots(1,1)
fig.set_size_inches(5,5)
a.imshow(img)
```

```
for (box, label, score) in zip(pred['boxes'], pred['labels'], pred['scores']):
    x, y, width, height  = box[0], box[1], box[2]-box[0], box[3]-box[1]
    rect = patches.Rectangle((x, y),
                             width, height,
                             linewidth = 1,
                             edgecolor = 'g',
                             facecolor = 'none')

    s = class_list[label.cpu().numpy()] + ' ({})'.format(score.item())
    print(s)
    a.text(x-5, y-5, s)

    # Draw the bounding box on top of the image
    a.add_patch(rect)
plt.show()
```

CHAPTER VII

Performance Evaluation

The error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function. In the project, the loss is reduced in each epoch during the training. The following graph depicts the loss during the training

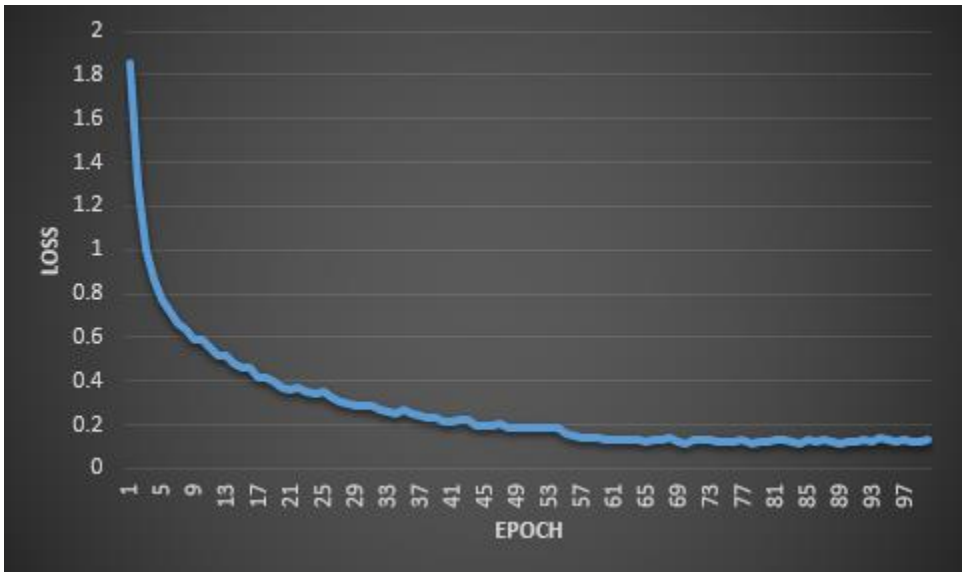


Figure 7.1 Loss-graph

The following graph shows the training accuracy and training loss of the project during each epoch. From this graph it is observed that the accuracy comes to stable after first few epoch. An epoch is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed.

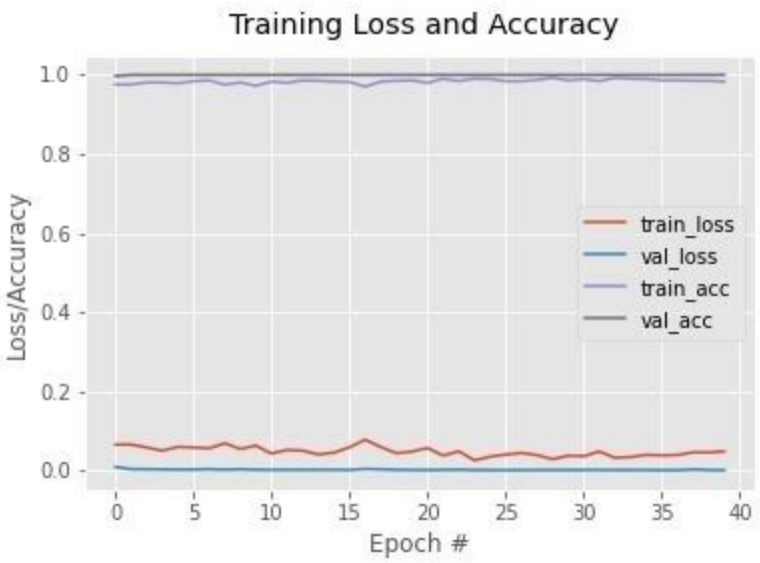


Figure 7.2 Training Loss and Accuracy

The following table has the comparison of our architecture with other architectures

Architecture	**Mean Average Precision**	**Frames**
Mobilenet SSD	76.9% mAP	22 FPS
Faster RCNN	73.2% mAP	7 FPS
Novel Hybrid	81%mAP	15FPS

CHAPTER VIII

CONCLUSION & FUTURE WORK

8.1 CONCLUSION

The project is successfully implemented for effectively identifying the spacecrafts using the available deep learning approach. The project is very helpful in providing a cheap yet effective solution to detecting space crafts and space objects.

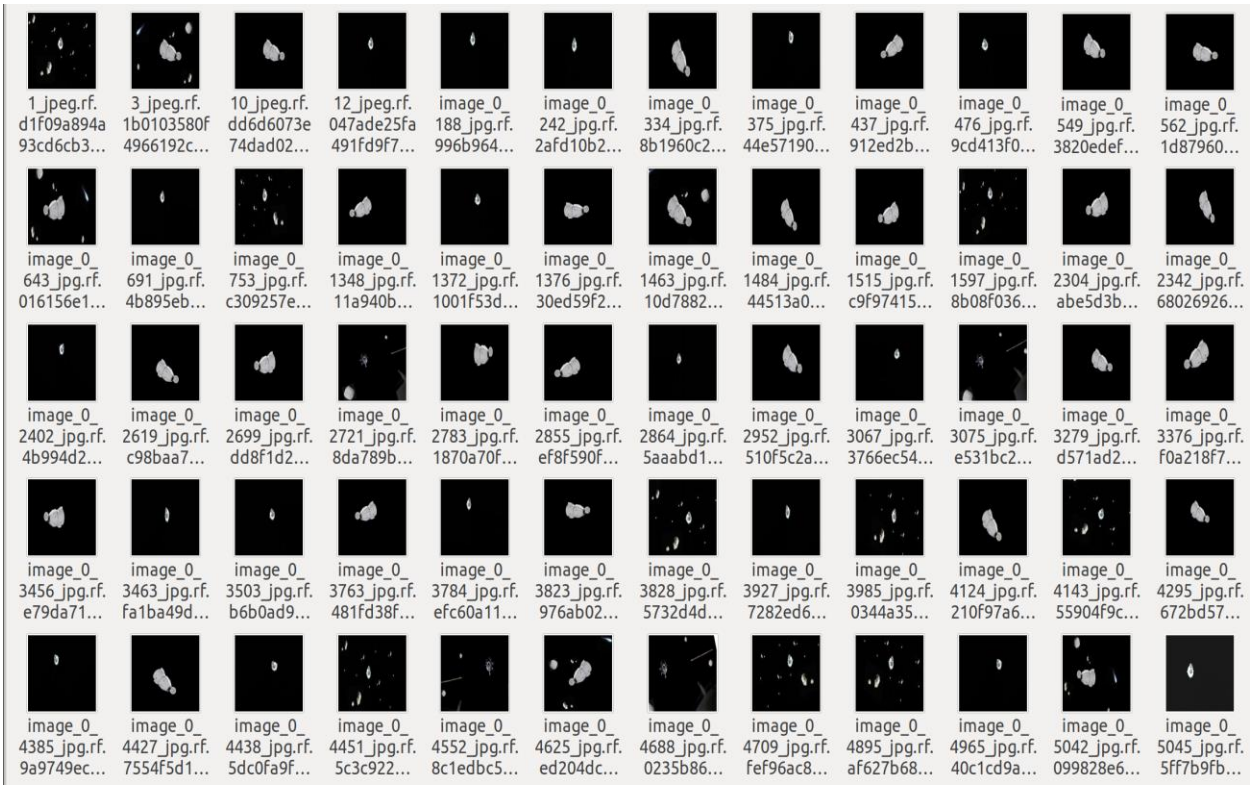
8.2 FUTURE WORK

In the coming future, we review the application of the space detection technology in the space exploration field and it can promote for effective advancing the present technological methods. In this field, there are more chance to develop or convert this project in many ways. Thus, the project has an efficient scope in coming future where this idea can be converted to computerized space detection in a cheap way.

APPENDICES

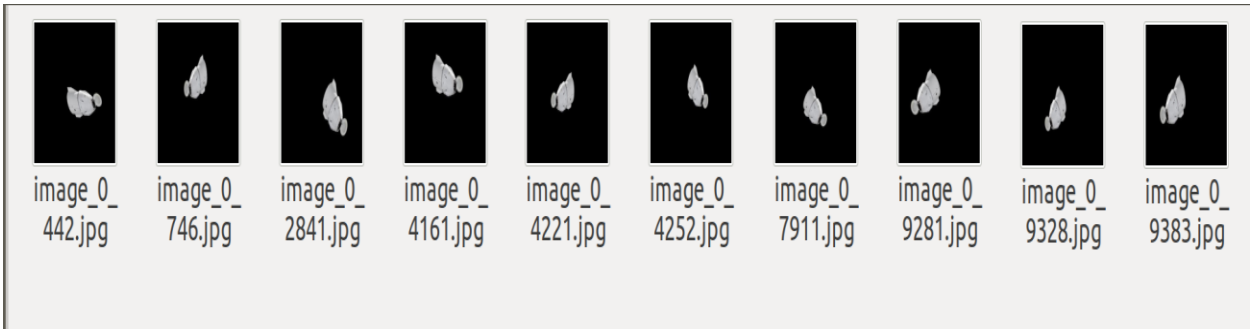
A.1 Sample Screens

1) Dataset Collection

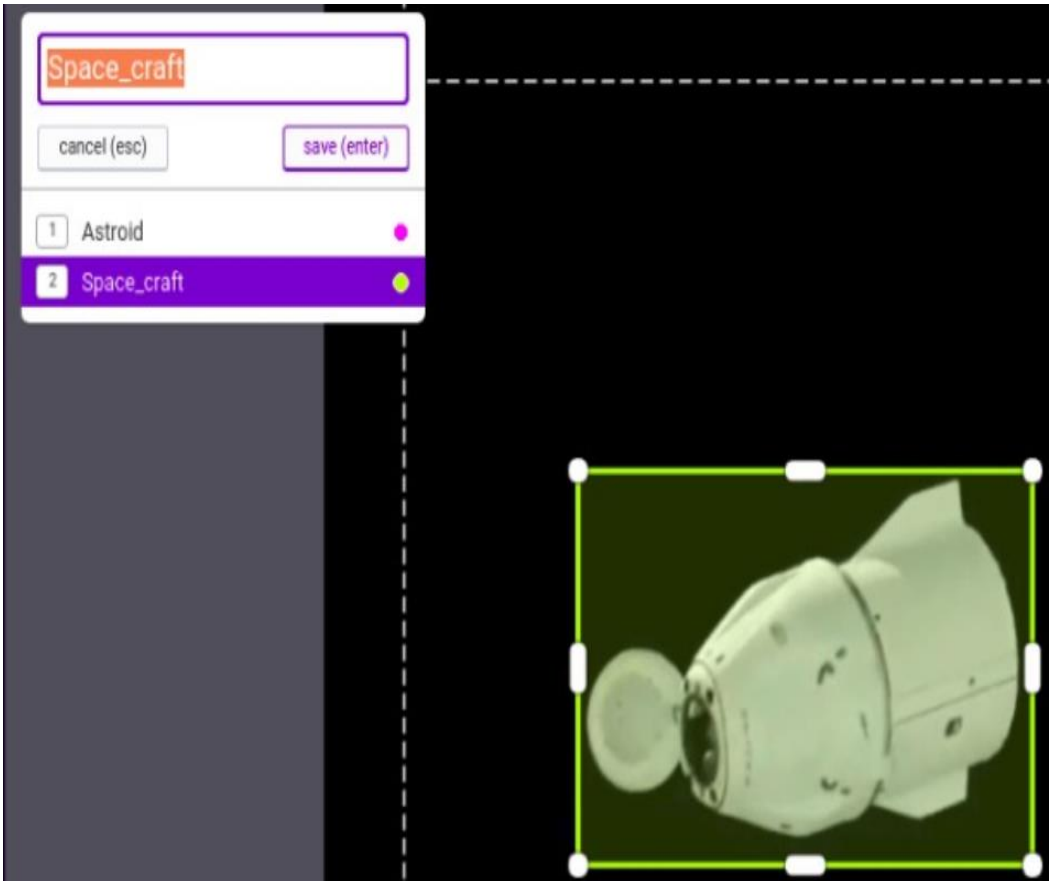


2) Image after Preprocessing

3)Image after Augumentation



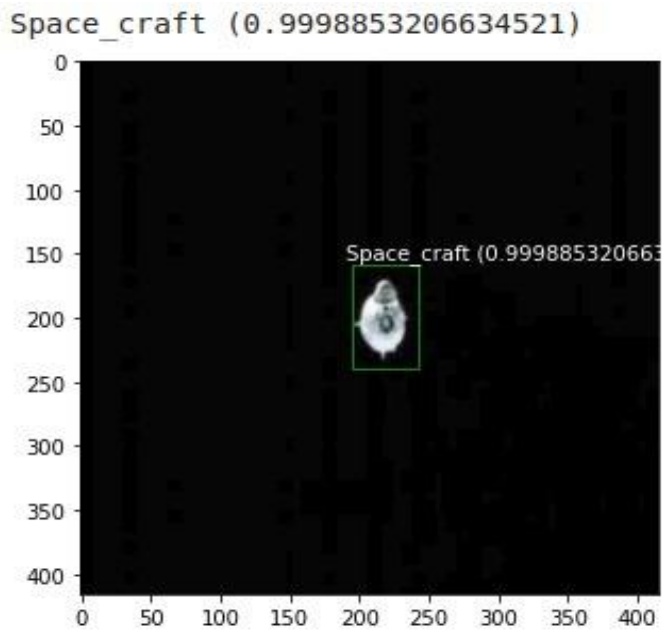
4) Labelling of dataset



5) Training process

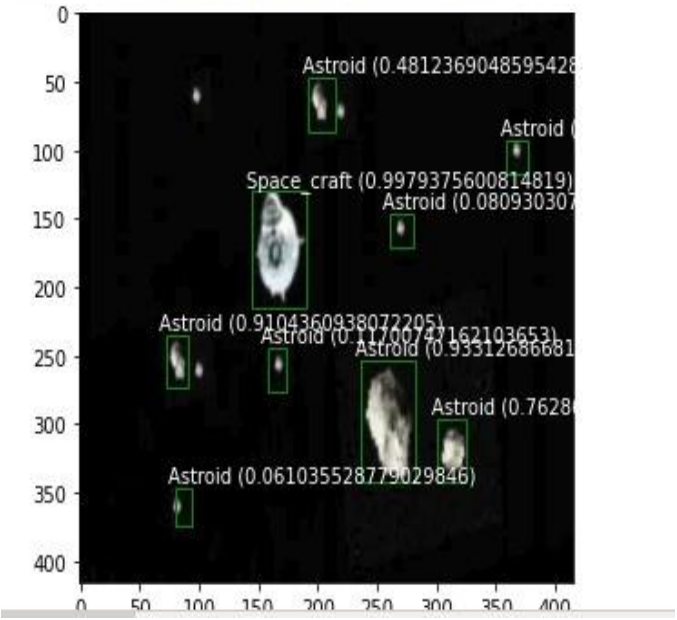
```
Epoch: 94/100. Iteration: 16/16. Cls loss: 0.02082. Reg loss: 0.13735. Batch loss: 0.15818 100% 16/16 [00:07<00:00, 2.65it/s]
Total loss: 0.08828

Epoch: 95/100. Iteration: 16/16. Cls loss: 0.00802. Reg loss: 0.06241. Batch loss: 0.07043 100% 16/16 [00:07<00:00, 2.63it/s]
Total loss: 0.07767

Epoch: 96/100. Iteration: 16/16. Cls loss: 0.00671. Reg loss: 0.05355. Batch loss: 0.06026 100% 16/16 [00:07<00:00, 2.64it/s]
Total loss: 0.08409

Epoch: 97/100. Iteration: 16/16. Cls loss: 0.00846. Reg loss: 0.05337. Batch loss: 0.06183 100% 16/16 [00:07<00:00, 2.64it/s]
Total loss: 0.08002

Epoch: 98/100. Iteration: 16/16. Cls loss: 0.00915. Reg loss: 0.09728. Batch loss: 0.10644 100% 16/16 [00:07<00:00, 2.64it/s]
Total loss: 0.08200

Epoch: 99/100. Iteration: 16/16. Cls loss: 0.01366. Reg loss: 0.07073. Batch loss: 0.08439 100% 16/16 [00:07<00:00, 2.64it/s]
Total loss: 0.08084

Epoch    99: reducing learning rate of group 0 to 1.0000e-06.
Epoch: 100/100. Iteration: 16/16. Cls loss: 0.00806. Reg loss: 0.04589. Batch loss: 0.05395 Total loss: 0.07791 100% 16/16 [00:07<00:00, 2.66it/s]

CPU times: user 19min 41s, sys: 2min 46s, total: 22min 28s
Wall time: 24min 27s
```

6) Space craft detection



7) Multiple space objects detection

```
Space_craft (0.9979375600814819)
Astroid (0.9331268668174744)
Astroid (0.9104360938072205)
Astroid (0.762863039970398)
Astroid (0.48123690485954285)
Astroid (0.11700747162103653)
Astroid (0.08118078857660294)
Astroid (0.08093030750751495)
Astroid (0.061035528779029846)
```



A.2 Publications

Journal name - International Research Journal of Engineering and Technology (IRJET)

Paper title - Image Identification and Recognition using Novel Hybrid Architecture Development for Satellite Images

Publication issue - Volume 8, Issue 4, April 2021

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

IMAGE IDENTIFICATION AND RECOGNITION USING NOVEL HYBRID ARCHITECTURE DEVELOPMENT FOR SATELLITE IMAGES

Ridu Varshini.N[1], Elamathy S[2]

[1,2]*UG Scholar, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai*

Abstract - *Recent advances in deep learning made tasks such as Image and speech recognition possible. Deep Learning is a subset of Machine Learning that is very good at recognizing patterns but typically requires a large number of data. The continuous innovation and development of satellite technology has brought the world closer together. At present, there are thousands of artificial satellites in the world, and the number of spacecraft working in orbit is increasing. With the constant exploration of outer space, there is inevitably a large quantity of space debris, e.g., lacquer, satellite debris. How to distinguish space targets of interested from the massive debris is a hot topic and of great significance. The effective separation of space targets and space debris can ensure the safe operation of on-orbit spacecraft. But, nowadays with the advancement of technology there is still a lag in predicting the space crafts and the related targets in an accurate manner. In this work we will be focusing on developing a novel hybrid architecture for determining the different types of satellite images, where four different kind of pre-processing techniques as well as two different optimization techniques will be used to increase the accuracy of the proposed hybrid model. The concept of network surgery will be used for implementing the hybrid algorithm development. Thus by this work we will be able to determine any kind of satellite images given as an input to the generated model. Thus, we propose a solution for the determination of spacecraft with the most accurate prediction by developing a novel architecture.*

***Key Words*: Space situational awareness, novel hybrid architecture, recognition of spacecraft,**

1. INTRODUCTION

Modern life depends on space technology, including communications, media, commerce, and navigation. Enabling space technology are the thousands of space assets (satellites, space station, etc.) currently in orbit, which amount to trillions of dollars of investment. With space usage moded to increase rapidly, in part due to the participation of new state and private operators, the number of space assets will also grow quickly.

Residents space objects (RSOs) such as space assets that directly benefit the intended applications, as well as orbital debris that occurs as a by-product of similar space activities, inevitably trigger more "crowding" of geocentric orbits.. The increase in RSOs raises the potential of collision between space assets and debris and this has been identified as a pressing issue.

Achieving SSA is crucial for reducing the chances of collisions destroying space assets. The recognition of space targets utilizes various techniques to obtain their characteristics and information, determining the attributes, types, locations of the targets. However, most of them are based on artificial features and their reliability is restricted by various conditions. For example, when the shape of the fragment is similar to the partial shape of the space target, they would not be distinguished easilydue to thelack of deep semantic digging.

1.1 Existing model for the recognition of spacecraft

The existing system focuses on effective optical detection of multiple man-made objects in the geostationary orbits. The system performs the optical detection from the optical images. Topological sweep is used for the multiple geostationary objects detection. It detects the geostationary objects from only the optical images. The accuracy is very poor compared to other state of the art methods.

1.2 Our contribution

In our work, we will be focusing on developing a novel architecture by effectively modifying the squeeze net for determining the different types of satellite images, where pre-processing technique as well as different optimization technique will be used to increase the accuracy of the proposed model. Thus, by this work we will be able to recognize any kind of satellite images given as an input to the generated model. A low weighted efficient model is been generated for use in the real time application.

2. System architecture

Data is collected from the public source. As there are limited no of dataset available, and training requires a large number of data, data augmentation can help a lot. we increased the number of images through rotation, shifting, flipping, blurring, cropping, noising, and their combination. Thus the given data is augmented.

Later in preprocessing technique, Aspect aware algorithm is used and the image is converted to array. Then this preprocessed data will enter into the novel architecture for

training. In this architecture the SqeezeNet algorithm is modified. SqueezeNet is the name of a deep neural network for computer vision and **it** is a convolutional neural network that employs design strategies to reduce the number of parameters, particularly with the use of fire modules that use 1x1 convolutions to "squeeze" parameters.

After the training, for the model generation, optimization and loss minimization is done. Once these processes are completed, evaluation and validation is done. We use COCO (Common Objects in Context) algorithm which is a large-scale object detection, segmentation, and captioning dataset.

Thus by using COCO algorithm space craft will be detected. Once the space craft is detected, the bounding boxes are drawn. Fig.1 shows the architecture diagram for image identification and recognition using the novel hybrid architecture.

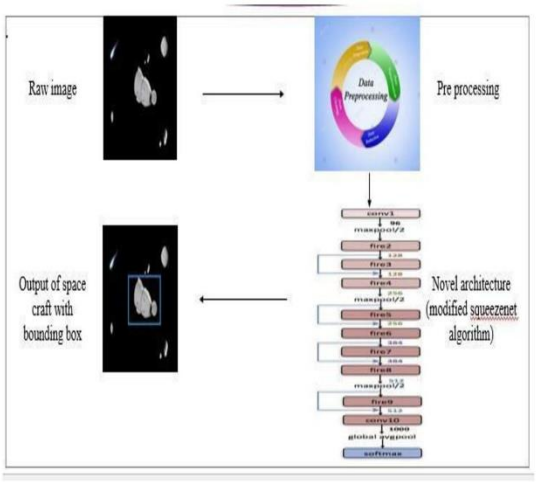


Fig -1: System architecture

2.1 Dataset collection

A data set is a collection of data. Deep Learning has emerged as the preferred approach for addressing a wide range of complex real-world issues. It is without a doubt the most effective approach for computer vision tasks.

A deep network can segment and classify the "key points" of any individual in an image with enough training. These deep learning machines, which have been performing admirably, need a lot of fuel, and that fuel is data. The more labelled data available, the better our model performs. Google has also experimented with the concept of more data contributing to better results on a wide scale, with a dataset of 300 million images! When deploying a Deep Learning model in a real-world application, data must be constantly fed to continue improving its performance. And, in the deep learning era, data is very well arguably the most valuable resource

2.2 Preprocessing data

In this work the preprocessing data module is used to resize the images. In deep Learning module the quality of the training data determines the quality of your model. In most instances, the data you will find in practice will not be clean. It means the data will contain non-uniform data formats, missing values, outliers, and features with very different ranges. The data would not suitable to be used as training data for your model. For those reason, the data must be preprocessed in various ways.

Fig -2: Preprocessing

2.3 Data augmentation

The most well-known method of data augmentation is image data augmentation, which entails transforming images in the training dataset into transformed versions that belong to the same class as the original image. Shifts, turns, zooms, and other operations from the field of image processing are included in transforms. The aim is to add new, plausible examples to the training dataset. This refers to variants of the training set images that the model is likely to see. Modern deep learning algorithms, such as the convolutional neural network (CNN), can learn features that are independent of where they appear in the picture. However, augmentation can help with this transform invariant approach to learning by assisting the model in learning features that are also transform invariant, such as left-to-right to top-to-bottom ordering, light levels in images, and so on. Usually, image data augmentation is only used on the training dataset, not the validation or evaluation datasets.

2.4 Training with novel architecture

In this work, we will be developing a novel architecture by effectively modifying the squeezenet architecture.

The main ideas of the novel architecture are:

1. Using 1x1 (point-wise) filters to replace 3x3 filters, as the former only 1/9 of computation.
2. Using 1x1 filters as a bottleneck layer to reduce depth to reduce computation of the following 3x3 filters.
3. Downsample late to keep a big feature map.

The building brick of Novel architecture is called fire module, which contains two layers: a squeeze layer and an expand layer. A Novel architecture stackes a bunch of fire modules and a few pooling layers. The function map size is kept the same by the squeeze and expand layers, but the former reduces the depth while the latter increases it. In neural architectures, squeezing (bottleneck layer) and expansion behaviour is normal. To get a high level abstract, another popular trend is to increase depth while reducing feature mapsize.

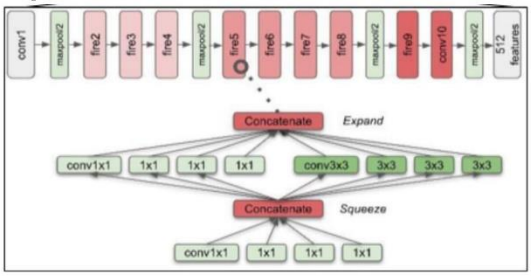


Fig -3: SqueezeNet

The squeeze module only has 1x1 filters, as seen in the diagram above, which means it acts like a fully-connected layer on feature points in the same position. In other words, it doesn't have the ability of spatial abstract. One of its advantages, as the name implies, is that it reduces the depth of the function map. The following 3x3 filters in the expand layer would have less computation to do as the depth is reduced. It improves speed since a 3x3 filter needs 9 times the computation of a 1x1 filter. Too much squeezing, intuitively, limits information flow; too few 3x3 filters, on the other hand, limits space resolution.

2.5 Object detection

Object detection is the method of accurately locating all potential instances of real-world objects in photographs or videos, such as human faces, flowers, vehicles, and so on. To identify all instances of an object type, the object detection technique employs derived features and learning algorithms.

Object detection is a technique for identifying, detecting, and localising several visual instances of objects in an image or video. Rather than just simple object classification, it offers a much clearer interpretation of the object as a whole.

This approach can be used to count the number of instances of unique objects and mark them as well as count the number of instances of unique objects. This process' efficiency has also improved over time.

To begin with, testing of the trained model, we can split our study into modules of implementation that is done. Dataset collection involves the process of collecting different space craft images for training. Various datasets were collected and one example among the collected dataset can be found below

Fig -4: Dataset collection

The next step is data pre-processing, in this phase the spacecraft images collected are set to standard resolution format. The below screenshot shows the image before preprocessing:

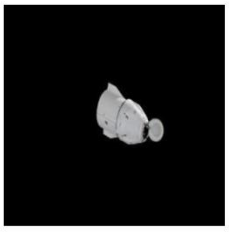


Fig -5: Before pre-processing

The below screenshot shows the image after preprocessing:

Fig -6: After pre-processing

The next step is data augmentation which duplicates the collected data into many images so that the prediction percentage can be increased. A sample augmentation of an image can be seen in the below figure.The below screenshot shows the image before data augmentation:

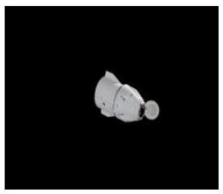


Fig -7: Image before data augmentation

The below screenshot shows the image after augmentation:

Fig -8: Image after data augmentation

The next step is labelling of data where the spacecraft images collected are labelled according to their nature and the images are annotated and saved in xml format. The below screenshot shows the labelling of the dataset.

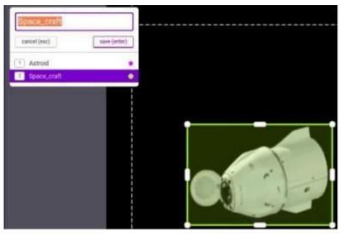


Fig -9: Labelling of dataset

The below image shows the training process of the work using the available deep learning architectures:

Fig -10: Training process

In the training process, we can see the loss has got reduced in each step. The below image shows the reduction of losses.

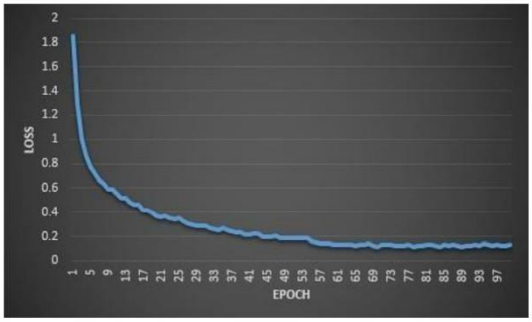


Chart -1: Reduction of losses

The below image shows the space craft detection of the work after training:

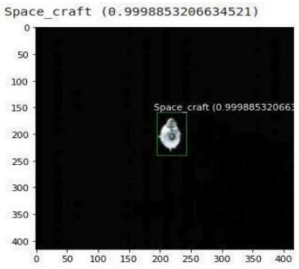


Fig -11: Space craft detection

The below image shows the multiple space objects detection of the work after training:

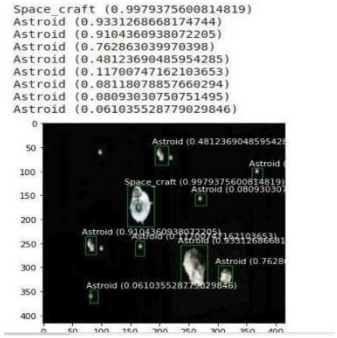


Fig -12: Multiple space objects detection

Thus, from the above results and discussions it is clear that the work to effectively detect and identify space objects has been effectively implemented.

3. CONCLUSION

This work is successfully implemented for effectively identifying the spacecrafts using the available deep learning approach. This work is very helpful in providing a cheap yet effective solution to detecting space crafts and space objects.

REFERENCES

[1] Lei Tao , Student Member, IEEE, Xue Jiang , Senior Member, IEEE, Xingzhao Liu , Member, IEEE, Zhou Li, and Zhixin Zhou," Multiscale Supervised Kernel Dictionary Learning for SAR Target Recognition",2020

[2] Joon-Ho Lee, Hyun-Jin Moon, and So-Hee Jeong," Numerically Efficient Determination of the Optimal Threshold in Natural Frequency-Based Radar Target Recognition",2019

[3] Jifang Pei , Student Member, IEEE, Yulin Huang, Member, IEEE, Weibo Huo, Yin Zhang , Member, IEEE, Jianyu Yang, Member, IEEE, and Tat-Soon Yeo, Fellow, IEEE," SAR Automatic Target Recognition Based on Multiview Deep Learning Framework",2017

[4] Chunlei Huo, Member IEEE, Zhixin Zhou, Kun Ding, Chunhong Pan," Online Target Recognition for Time-sensitive Space Information Networks",2016

[5] Xueru Bai , Member, IEEE, Xuening Zhou, Feng Zhang, Li Wang, Ruihang Xue , and Feng Zhou , Member, IEEE," Robust Pol-ISAR Target Recognition Based on ST-MC-DCNN",2018

[6] Tan Wu, Xi Yang, Bin Song, Nannan Wang, Xinbo Gao, Liyang Kuang, Xiaoting Nan, Yuwen Chen, Dong Yang," T-SCNN: A Two-Stage Convolutional Neural Network for Space Target Recognition",2017

[7] Lin Chen , Student Member, IEEE, Xue Jiang , Senior Member, IEEE, Zhou Li, Xingzhao Liu , Member, IEEE, and Zhixin Zhou," Feature-Enhanced Speckle Reduction via Low-Rank and Space-Angle Continuity for Circular SAR Target Recognition",2015

[8] Jiemin Hu, Wei Wang, Qinglin Zhai, Jianping Ou, Ronghui Zhan, and Jun Zhang," Global Scattering Center Extraction for Radar Targets using a Modified RANSAC Method",201

[9] Zhiqiang He , Huaitie Xiao, Chao Gao, Zhuangzhuang Tian, and Si-Wei Chen , Senior Member, IEEE," Fusion of Sparse Model Based on Randomly Erased Image for SAR Occluded Target Recognition",2016

[10] Nasser M. Nasrabadi, Fellow, IEEE," DeepTarget: An Automatic Target Recognition using Deep Convolutional Neural Networks",2017

[11] Haibin Duan, Senior Member, IEEE, and Lu Gan," Elitist Chemical Reaction Optimization for Contour-Based Target Recognition in Aerial Images",2018

[12] Ming Liu , Shichao Chen , Jie Wu , Fugang Lu, Xili Wang, and Mengdao Xing, Member, IEEE," SAR Target Configuration Recognition via Two-Stage Sparse Structure Representation",2015

[13] Lan Du, Senior Member, IEEE, Jian Chen, Jing Hu, Yang Li, Hua He," Statistical Modeling with Label Constraint for Radar Target Recognition",2016

[14] Baiyuan Ding and Gongjian Wen," Target Reconstruction Based on 3-D Scattering Center Model for Robust SAR ATR",2019

[15] Yao-tung tsou1 , yung-li hu2 , yennun huang3 , (fellow, ieee), and sy-yen kuo2 , (Fellow, IEEE)," SFTopk: Secure Functional Top-k Query via Untrusted Data Storage",2017

REFERENCES

[1] Lei Tao , Student Member, IEEE, Xue Jiang , Senior Member, IEEE, Xingzhao Liu , Member, IEEE, Zhou Li, and Zhixin Zhou," Multiscale Supervised Kernel Dictionary Learning for SAR Target Recognition",2020

[2] Joon-Ho Lee, Hyun-Jin Moon, and So-Hee Jeong," Numerically Efficient Determination of the Optimal Threshold in Natural Frequency-Based Radar Target Recognition",2019

[3] Jifang Pei , Student Member, IEEE, Yulin Huang, Member, IEEE, Weibo Huo, Yin Zhang , Member, IEEE, Jianyu Yang, Member, IEEE, and Tat-Soon Yeo, Fellow, IEEE," SAR Automatic Target Recognition Based on Multiview Deep Learning Framework",2017

[4] Chunlei Huo, Member IEEE, Zhixin Zhou, Kun Ding, Chunhong Pan," Online Target Recognition for Time-sensitive Space Information Networks",2016

[5] Xueru Bai , Member, IEEE, Xuening Zhou, Feng Zhang, Li Wang, Ruihang Xue , and Feng Zhou , Member, IEEE," Robust Pol-ISAR Target Recognition Based on ST-MC-DCNN",2018

[6] Tan Wu, Xi Yang, Bin Song, Nannan Wang, Xinbo Gao, Liyang Kuang, Xiaoting Nan, Yuwen Chen, Dong Yang," T-SCNN: A Two-Stage Convolutional Neural Network for Space Target Recognition",2017

[7] Lin Chen , Student Member, IEEE, Xue Jiang , Senior Member, IEEE, Zhou Li, Xingzhao Liu , Member, IEEE, and Zhixin Zhou," Feature-Enhanced Speckle Reduction via Low-Rank and Space-Angle Continuity for Circular SAR Target Recognition",2015

[8] Jiemin Hu, Wei Wang, Qinglin Zhai, Jianping Ou, Ronghui Zhan, and Jun Zhang," Global Scattering Center Extraction for Radar Targets using a Modified RANSAC Method",201

[9] Zhiqiang He , Huaitie Xiao, Chao Gao, Zhuangzhuang Tian, and Si-Wei Chen , Senior Member, IEEE," Fusion of Sparse Model Based on Randomly Erased Image for SAR Occluded Target Recognition",2016

[10] Nasser M. Nasrabadi, Fellow, IEEE," DeepTarget: An Automatic Target Recognition using Deep Convolutional Neural Networks",2017

[11] Haibin Duan, Senior Member, IEEE, and Lu Gan," Elitist Chemical Reaction Optimization for Contour-Based Target Recognition in Aerial Images",2018

[12] Ming Liu , Shichao Chen , Jie Wu , Fugang Lu, Xili Wang, and Mengdao Xing, Member, IEEE," SAR Target Configuration Recognition via Two-Stage Sparse Structure Representation",2015

[13] Lan Du, Senior Member, IEEE, Jian Chen, Jing Hu, Yang Li, Hua He," Statistical Modeling with Label Constraint for Radar Target Recognition",2016

[14] Baiyuan Ding and Gongjian Wen," Target Reconstruction Based on 3-D Scattering Center Model for Robust SAR ATR",2019

[15] Yao-tung tsou1 , yung-li hu2 , yennun huang3 , (fellow, ieee), and sy-yen kuo2 , (Fellow, IEEE), " SFTopk: Secure Functional Top-k Query via Untrusted Data Storage",2017