8-31-2022

A DEEP LEARNING APPROACH FOR AIRPORT RUNWAY IDENTIFICATION FROM SATELLITE IMAGERY

Mahmut Gemici
Rowan University

A DEEP LEARNING APPROACH FOR AIRPORT RUNWAY IDENTIFICATION FROM SATELLITE IMAGERY

by

Mahmut Gemici

A Thesis

Submitted to the
Department of Electrical and Computer Engineering
College of Engineering
In partial fulfillment of the requirement
For the degree of
Master of Science in Electrical and Computer Engineering
at
Rowan University
August 31, 2022

Thesis Chair:
Nidhal Bouaynaya, Ph.D., Professor, and Associate Dean for Research and Graduate Studies,
Electrical and Computer Engineering

Committee Members:
Charles C. Johnson, Rotorcraft Safety Research Team Lead FAA Williams J. Hughes Technical Center, FAA
Gregory Ditzler, Ph.D., Associate Professor, Electrical and Computer Engineering

Dedications

I dedicate this work to my parents Mehmet Gemici, Rabiye Gemici, and my lovely sister Gizem Gemici.

Acknowledgments

This work was supported by the Federal Aviation Administration (FAA) Cooperative Agreement Number 16- G-015 and NSF Award DUE-1610911. This Thesis was also supported by a sub-award from Rutgers University, Center for Advanced Infrastructure & Transportation, under Grant no. 69A3551847102 from the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (OST-R). We would also like to thank LZControl for their guidance and assistance with this effort. Via a Cooperative Research and Development Agreement with the FAA, LZControl provided a set of data from their system and subject matter expertise, which provides landing zones for helicopters across the U.S., often complementing the FAA's 5010 database and including locations/sites not presented in the FAA's 5010 system.

Abstract

Mahmut Gemici
A DEEP LEARNING APPROACH FOR AIRPORT RUNWAY IDENTIFICATION USING SATELLITE IMAGERY
2021-2022
Nidhal C. Bouaynaya, Ph.D.
Master of Science in Electrical and Computer Engineering

The United States lacks a comprehensive national database of private Prior Permission Required (PPR) airports. The primary reason such a database does not exist is that there are no federal regulatory obligations for these facilities to have their information re-evaluated or updated by the Federal Aviation Administration (FAA) or the local state Department of Transportation (DOT) once the data has been entered into the system. The often outdated and incorrect information about landing sites presents a serious risk factor in aviation safety. In this thesis, we present a machine learning approach for detecting airport landing sites from Google Earth satellite imagery. The approach presented in this thesis plays a crucial role in confirming the FAA's current database and improving aviation safety in the United States. Specifically, we designed, implemented, and evaluated object detection and segmentation techniques for identifying and segmenting the regions of interest in image data. The in-house dataset has been thoroughly annotated that includes 400 satellite images with a total of 700 instances of runways. The images - acquired via Google Maps static API - are 3000x3000 pixels in size. The models were trained using two distinct backbones on a Mask R-CNN architecture: ResNet101, and ResneXt101, and obtained the highest average precision score @0.75 with ResNet-101 at 92% and recall at %89. We finally hosted the model in the StreamLit front-end platform, allowing users to enter any location to check and confirm the presence of a runway.

Table Of Contents

Table Of Contents (Continued)

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Federal Aviation Administration (FAA) has an up-to-date database of public airports but lacks a comprehensive national database for privately held landing areas. In the event of an emergency, pilots must be able to locate suitable landing locations to safely deal with a critical aviation situation. The precision of the landing locations including latitude and longitudinal coordinates is vitally important to the FAA. Unfortunately, there is no federal requirement for private airports to report their landing sites, leading to an inaccurate database. It is widely known that the coordinates of several locations are off by hundreds of yards to several miles. Because of these deficiencies, pilots receive inaccurate information, which may lead to the expiration of their fuel supply or even to fatalities. Therefore, it is necessary to validate the current geospatial database of airport runways. Airports, on the other hand, are facilities that are significant from both an economic and a military point of view. Considering the crucial role that these goals play in the overall strategy, automatic airport identification is of the highest concern. This thesis presents a detection approach for paved runways, which are the elements of an airport that most stand out from one another to meet the FAA's requirement of validating the current database. Object detection and image segmentation are two components that make up this approach. Thus, the issue of non-validated landing sites in the FAA's database can be addressed by confirming their coordinates by employing deep learning algorithms.

1.2 Scope of Thesis

This thesis discusses airport runway identification systems, and it aims to develop, build, and experimentally assess a neural network pipeline for identifying runways in satellite images. The relevant satellite images contain chromatic information that is visible to the human eye and no other remotely sensed data channels than RGB (red, green, and blue). The runway is typically the most prominent aspect of an airport facility when viewed from above. Compared to the remainder of the airport region, runways have an easily recognized shape, share more characteristics, and occupy the most space. This thesis makes a case for the implementation of an autonomous runway identification system as a means of resolving the issue of accurately identifying runways and validating their locations. Since there is such a massive supply of satellite data, it is now feasible to capture overhead imagery at any coordinates that are requested. The data that was collected may then be used to train a system to detect the existence of a runway at defined coordinates. Additionally, the system can be expanded to enable users to search for a runway within a specified region.

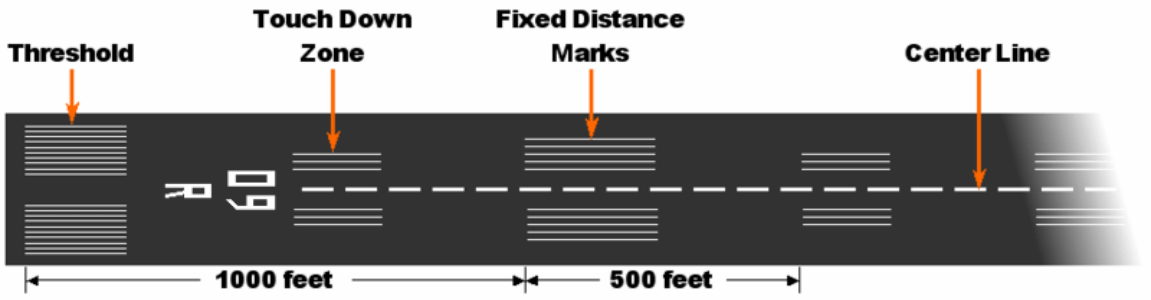
1.3 What are Airport Runways?

An airport is a facility designed for aircraft takeoff and landing. Airports may be divided into civil and military categories based on their intended use. Generally, civil airports have a runway or multiple runways, taxiways, passenger and freight terminals, a control tower, and other smaller structures such as passenger walkways or parking lots. There can be small civil airports built exclusively for gliders or flying boats that have grass runways or none. Military airports often have a runway or multiple runways, taxiways,

hangars, barracks, control towers, and defensive structures such as SAM (Surface to Air Missile) installations, as well as various smaller structures such as shelters, administration buildings, fire stations, and ammunition storage. The runways at military airports designed for jet aircraft are typically longer than those at airfields designed for transport aircraft. Some airports can serve both civil and military functions. Temporary airfields and emergency airfields are often lengthy, straight roadway portions that serve as runways. As a vital airport component, a runway is a long strip that provides the requisite distance for aircraft to attain the necessary speed for takeoff and safely land. An airport may have a single runway, or many runways arranged in parallel, crossing, non-crossing (non-parallel at the same time), or any combination of these layouts. The landing surface may be a natural surface such as grass or gravel, but man-made surfaces are more prevalent. These man-made surfaces may be asphalt (e.g., Helicopter Airfields), concrete, anti-skid concrete (e.g., Jet Airfields), or even punctured planking in the case of temporary runways [1].

Figure 1

Runway Layout [2]



1.4 Related Works

This section examines the present approaches for detecting and classifying airport runways in satellite imagery. The image processing task that is utilized to identify common objects on the surface of the Earth is becoming easier as both the quality and quantity of satellite images improve. Due to the importance of the strategic location of airports, it has been the subject of much research recently. The method proposed in [3] is based on combining texture segmentation and shape detection to identify the region of interest, followed by the detection of elongated rectangles in the region of interest to locate runways. An approach that utilizes hypothesis creation and verification was suggested by Han et al. [4] to locate airports. Using edge tracking algorithms applied to an image that has been edge detected, hypotheses may be created by obtaining parallel lines that are oriented in the opposite direction. Given that the validity of a hypothesis is determined by examining color intensity, illumination which can provide substantial difficulties for this technique. Satellite Image Classification with Deep Learning [5] uses a CNN for classifying the IARPA Functional Map of the World (fMoW) challenge. FMoW dataset consists of 63 classes and the presented method achieved an overall average accuracy of 83%. The study showed the feasibility of employing deep learning to accomplish precise object detection in satellite imagery. Most of these previous methods use traditional image processing techniques and work only for those images having a runway as a combination of antiparallel lines. To the best of our knowledge, we introduce a novel approach to runway detection by combining object detection and image segmentation techniques with the Mask R-CNN algorithm.

Chapter 2

Background

This chapter gives an overview of the sub-fields this research draws from as its foundation. Some of these are convolutional neural networks (CNN), object detection, and image segmentation.

2.1 Convolutional Neural Network

Since the subject of computer vision first emerged, researchers have been developing a variety of methods that may teach a computer program to comprehend digital images. Convolutional neural networks are a popular architecture of Artificial neural networks that are inspired by the human brain. The architecture of convolutional neural networks is a hierarchy of layers that process images in multiple steps. The key idea of CNN is to extract features from an image at every layer. The convolutional neural network is composed of multiple layers; The input layer receives the image as an input, and the first hidden layer contains filters (also known as kernels) that process the input from the previous layer. The output from this layer is a set of feature maps, which are then passed on to another set of filters in the second hidden layer. The process is repeated until the last layer, which outputs a class prediction.

Image classification, object detection, and semantic segmentation are the most common application of Computer Vision that makes use of CNNs. All three applications are part of supervised learning which means labels must be provided along with the training image set. Unlike image classification which attempts to classify the content of an image, with object detection, two things are being accomplished, one is classification and the other

one is localization. This allows us to answer “what/where and how many of this object is in the image?” questions. Finally, segmentation goes one step further by providing a pixel-level classification. In the object detection and image segmentation task, each detected object is marked by bounding boxes, and each bounding box is associated with a confidence score indicating the likelihood that the model estimates a region to contain a target object. This granular level of localization is helpful for applications such as satellite photography, the decision-making process for autonomous vehicles, and medical imaging.

2.1.1 Residual Network

The Residual neural network (ResNet) has served as the foundation for a wide variety of cutting-edge deep learning applications. It is a backbone feature extractor in many contemporary networks due to its strong capacity to represent features. As it became popular to incorporate deeper networks into deep learning architectures, models like AlexNet [6] and the VGG network [7] began to gain traction. However, increasing the number of layer cause a common problem called the Vanishing/Exploding gradient. This problem occurs when a gradient is backpropagated through numerous layers. And, as a result of repeated multiplications, the gradient might become extremely small. The authors of ResNet [8] came up with a solution to this problem by implementing what’s called "skip connection" which connects activations of a layer to the next layers by skipping one or more layers in between. The residual mapping method makes it possible to train networks with more than a hundred of layers. In this thesis, we trained our dataset using two versions of the residual network which are ResNet101 and ResneXt101 [9] backbones and compared their results.

2.2 Object Detection

Object detection has become one of the most well-known and widely explored subjects in the field of computer vision and machine learning. As a result of its popularity, which has attracted a large number of researchers in recent years, the area has experienced a surge in development. The most recent detectors based on deep learning can be broken down into one of two categories: one-stage object detectors and two-stage object detectors. The trade-off between the two approaches is speed and precision. While one-stage detectors have fast inference times, two-stage detectors provide precise localization and recognition. The initial step of two-stage detectors is called a Region Proposal Network (RPN), and its purpose is to predict potential bounding boxes. During the second stage, the features for the subsequent classification and bounding box regression task are obtained from each potential box using a pooling operation that uses the region of interest. The most common two-stage detectors are the R-CNN, Fast R-CNN, and Faster R-CNN, as well as the FPN and SPPNet [10]–[14].

A one-stage detector, on the other hand, is capable of predicting bounding boxes in a single step without making use of region proposals. It uses a grid box and anchors to narrow down the detecting area in the image and place limits on the object's form. YOLO: You Only Look Once, and SSD: Single Shot Detector, are the two of the most commonly used one-stage detectors that are optimized for making inferences from video data. When one-stage detectors are compared to two-stage detectors, the improvement in detection speed comes at the trade-off of localization accuracy, especially when dealing with relatively small objects.

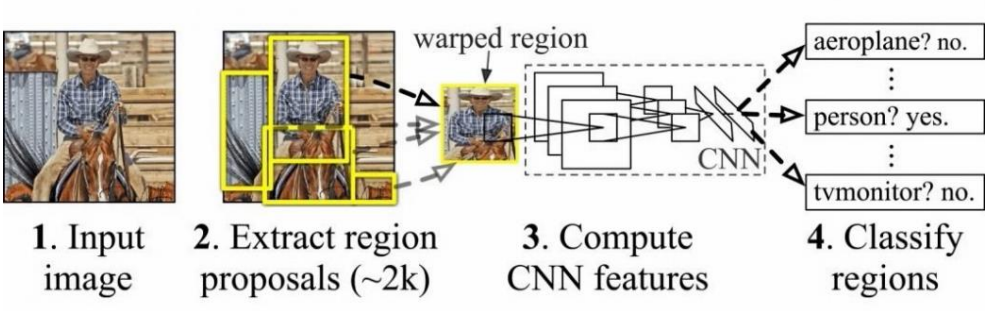
Since our goal is on developing reliable and accurate models, we are less concerned with prediction speed as it is not as important for us. We, therefore, didn't train our dataset with either YOLO or SSD framework.

2.2.1 R-CNN

R-CNN is the name given to the product since it is a collaboration between CNN and regional proposals. The R-CNN is composed of three primary modules, the first of which employs a selective search method to provide two thousand new proposals for regions. A feature vector with a length of 4096 is extracted from each suggested region by the second module, which also resizes each proposed region to a specific size that has been predefined. The third module employs a support vector machine (SVM) classification technique that has been pre-trained to assign the proposed region to either the background or one of the object classes. With the R-CNN method, we have avoided the usage of the sliding window technique, however, it still has some drawbacks. It is a multi-stage model, where each stage is an independent component. Thus, it cannot be trained end-to-end. On the other hand, each region proposal is fed independently to the CNN for feature extraction which makes it impossible to run R-CNN in real-time as it takes about 47 seconds for each test image [10].

Figure 2

Concept of Region-Based Convolutional Neural Networks[10]



2.2.2 Fast R-CNN

The same author who proposed the earlier work R-CNN came up with the idea for Fast R-CNN in 2015, which is an enhanced version of R-CNN. By rebuilding the original layers of the network, Fast R-CNN was able to overcome some of the shortcomings of R-CNN, which resulted in improved performance. The method is comparable to the R-CNN, but instead of running CNN on each of the 2000 proposals, the input image itself is processed by CNN in order to build a feature map. Despite this, the RoI is still calculated with respect to the input image utilizing selective search, and the results are then projected onto the feature map that is produced by CNN. In the past, CNN was executed on each RoI in a standalone fashion; however, this change means that convolution is now distributed evenly over all of the network's region proposals. [11]

2.2.3 Faster R-CNN

Selective search is utilized by both the R-CNN and the Fast R-CNN algorithms in order to determine region proposals. The network's performance can suffer as a result of selective search because it is a tedious operation that runs incredibly slowly. In response to this, a brand-new technique known as Faster R-CNN was developed. The image is given as an input to a CNN, which then generates a convolutional feature map in the same way as Fast R-CNN does. However, the selective search algorithm is taken out of Faster R-CNN, so the network can learn from the region proposals. In terms of inference speed and accuracy, Faster R-CNN is faster and more accurate than both previous models R-CNN and Fast R-CNN.

2.3 Image Segmentation

Image segmentation is a computer vision technique that divides an image into several specific components with unique attributes. Modern image segmentation algorithms are usually categorized as semantic (pixel-wise association with class label), instance (accurate delineation of each object in an image), and panoptic (assigning class labels to objects and stuff in images). A visual comparison of object detection, semantic segmentation, and instance segmentation can be observed in figure 3. Early image segmentation methods include threshold-based, centroid-based, density-based, graph-based, fuzzy theory-based, hierarchical, and distribution-based methods. Since the literature on image segmentation is so vast, it is not feasible to cover every algorithm and all of its variations. The goal of this section is to outline some of the most popular techniques that are well suited to solving the problems of runway extraction.

Figure 3

Object Detection, Semantic Segmentation, Instance Segmentation [15]

2.3.1 Mask R-CNN

Mask R-CNN [16] is the most advanced Convolutional Neural Network for image segmentation. This version of a Deep Neural Network recognizes items in an image and creates a segmentation mask of high quality for each instance. Mask R-CNN is built on Faster R-CNN architecture by extending a parallel branch for pixel-level segmentation. Mask R-CNN can detect and segment the objects from the background simultaneously. Its simplicity, performance, and flexibility are some of the advantages of the Mask R-CNN. The pixel-to-pixel orientation is the most crucial component of Mask R-CNN which was the primary issue with Fast/Faster R-CNN had have. Mask R-CNN is easy to set up and train because it uses the Faster R-CNN framework, which makes it possible to build a wide range of flexible architectures. Detectron2 [17] is an open-source framework that is an improvement over the original version of Detectron which was developed by Facebook. It is now capable of various types of real-time implementation such as instance and semantic segmentation, and human body key point identification are all possible with the Detectron2 platform thanks to its modular design. Detectron2's training efficiency has been

significantly enhanced, and the platform that it was built on, PyTorch, is utilized entirely. In this thesis, we trained Mask R-CNN using Detectron2 with two types of ResNet backbones for feature extraction.

Chapter 3

Methodology

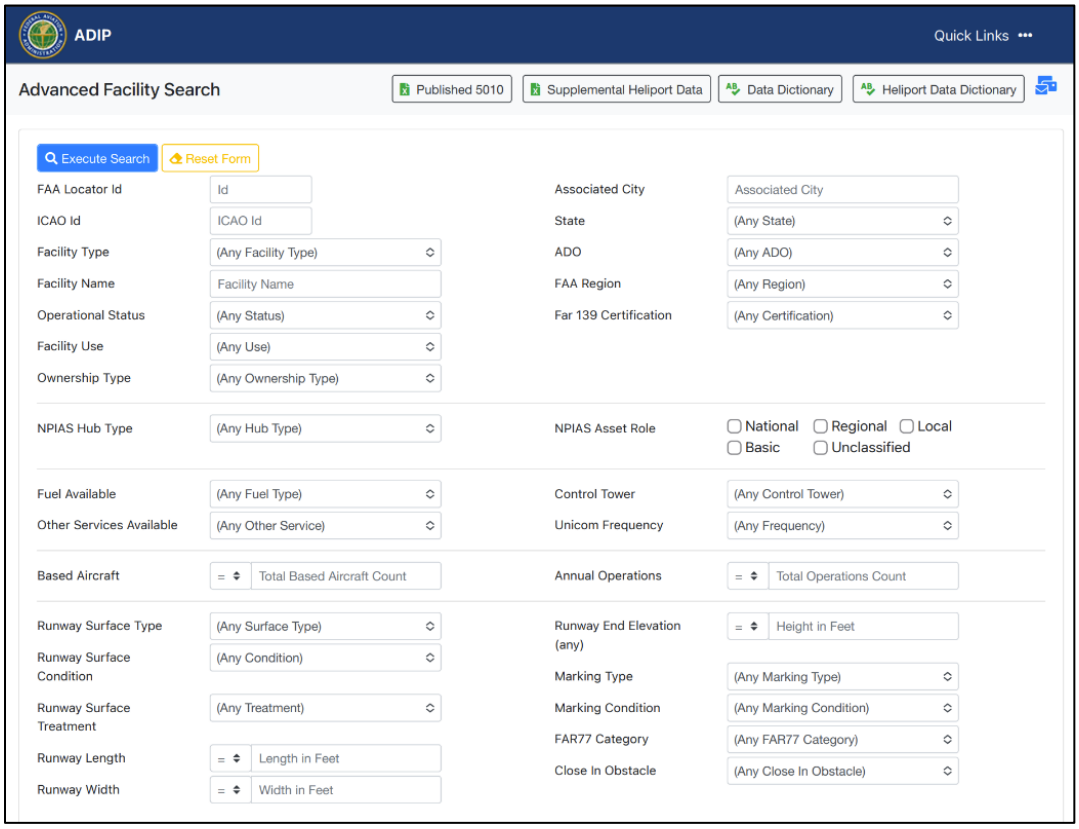
This chapter describes the data collection process, methods we used while training an object detector and how we evaluate the results.

3.1 Data Acquisition

To be able to acquire the satellite image data, it was necessary for us to collect the latitude and longitude information of airports that had previously been reported to the FAA. The FAA's Airport Data and Information Portal [18] was the source from which we obtained the information of latitude and longitude. From there, we were able to filter out any airport-related information that may have caused noise in our dataset. For this thesis, we determined that including grass or dirt airports would not add any value to the discussion; hence, we restricted the types of runways to only include paved ones.

Figure 4

FAA - Airport Data and Information Portal [18]



After selecting the facility type and other parameters, we were able to get the excel file that consists of each airport's information. The column of interests was the latitude and longitude information of each airport. In the excel sheet latitude and longitude values were entered in the format of "Degree-Minute-SecondsCardinal Direction". For example, Abbeville Municipal Airport's coordinates are, 31-36-00.8000N, 085-14-17.9100W, however, this format is not intuitive to download data from satellite image providers,

hence, we converted them into decimal degrees. For decimal degrees, we included a negative sign for south and west coordinates.

3.2 Google Static Maps API

To collect imagery, Google's static maps API were utilized, which includes Google Earth imagery. The Google Maps Statics API returns an image in a variety kind of image formats including GIF, PNG, or JPEG. All images used in our dataset were downloaded in PNG format. Accessing the service is accomplished by submitting an HTTP request along with a query that includes the requested parameters. In response, the service returns an image that is based on the parameters. Latitude, longitude, image size, and map size are the factors that are utilized here. The position that should be the image's center is determined by the latitude and longitude of that location. The pixel count along each axis is determined by the size of the image. For instance, if the image size is set to 1000, the resulting image will have a resolution of 1000 pixels across and 1000 pixels tall. The scale of the map affects how much of the underlying terrain is seen in the image. To meet the requirements of this project, the image size was consistently kept at 3000, while the map size remained at 5000. To get a fair level of resolution across all images, we believe that a map size of 5000 is the best possible figure to use as a trade-off with the image size. In Figures 5,6 and 7, it can be seen how the image size and map size impact the resolution of objects.

Figure 5

Atlantic City Airport Satellite View at Low Scale and Low Resolution

Map size = 5000, Image size = 1000

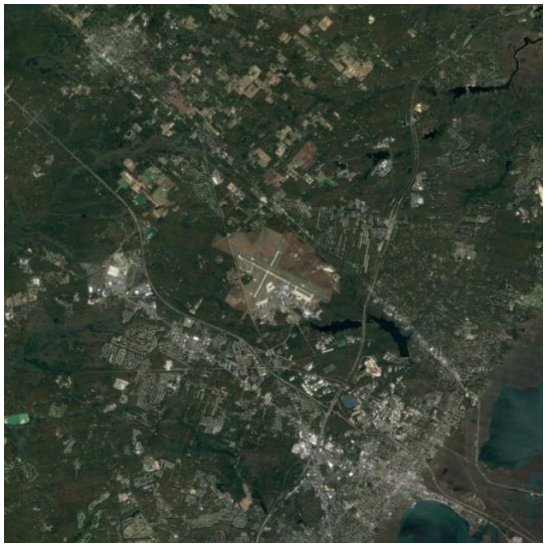
Figure 6

Atlantic City Airport Satellite View at Low Scale and High Resolution

Map size = 5000, Image Size = 5000

Figure 7

Atlantic City Airport Satellite view at High Scale and Low Resolution



Map size = 25000, Image size = 5000

3.3 Image Annotation

LabelMe is a free graphical image annotation tool designed to be utilized in computer vision applications to assist with object detection and image segmentation tasks [19]. The annotations are stored in JSON file format that is in the PASCAL VOC format, which is ImageNet's preferred format. LabelMe was used to manually draw bounding boxes and polygons around each runway. The use of bounding box labeling enables us to think about a detection problem in a more complex way than a straightforward binary classification problem (such as "Is there an airport visible in this picture?" yes or no). Bounding boxes simply indicate the area of interest for the model for each given object type. Each bounding box has its own (x, y) coordinates and object type. This information allows the model to localize each object and its type to its respective image. Figure 8 illustrates an image with a bounding box around the airport runways.

Figure 8

Labeled Airport Runways

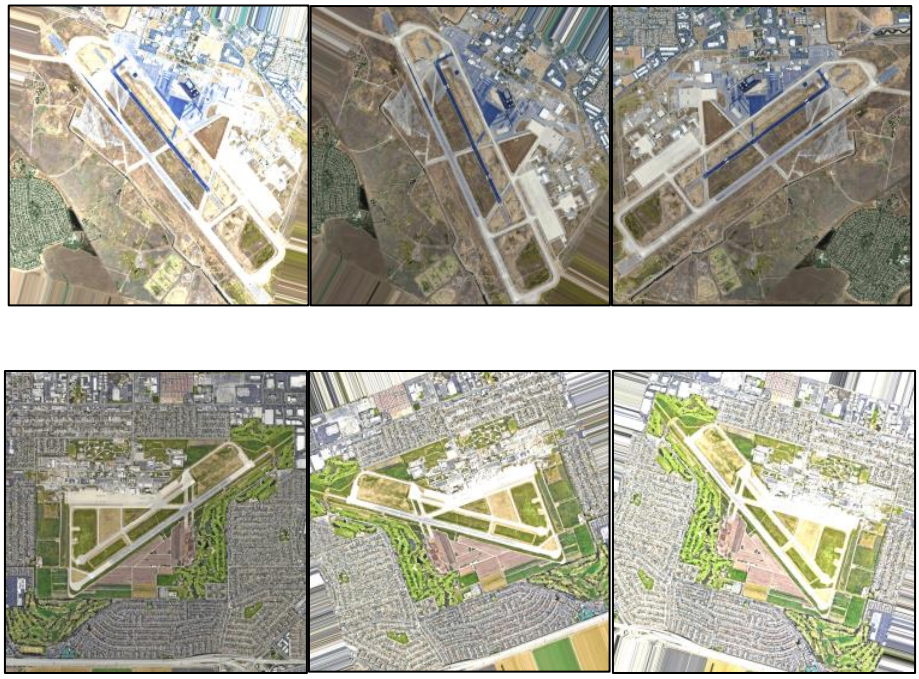
Because some airports consist of multiple runways connected to each other, we carefully labeled each of them individually. Out of 400 images, 700 runway instances were annotated. It is possible for the label file format to vary depending on the object detection API that is used to train a model. For example, the TensorFlow object detection API requires labels to be provided in XML file format, but Detectron2 demands labels to be provided in JSON file format. Luckily the LabelMe image annotation tool can provide either of these options.

3.4 Data Augmentation

The process of annotating huge datasets, such as segmentation, typically comes at a high cost and takes a lot of time. As an illustration, the COCO dataset required 22 worker hours for every one thousand instance masks that were created [20]. The act of developing a model that is more resilient and less prone to overfitting by artificially increasing the quantity of data collected is referred to as data augmentation. This method also makes the model more accurate. It is a method for analyzing data that generates duplicates of the data that have been subtly altered or brand-new copies that have been synthetically made [21]. Data augmentation is important for improving the performance of machine learning models and the results they provide. This is accomplished by adding fresh and unique instances to the dataset that is being trained. The use of data augmentation techniques enables transformations in datasets, which in turn allows for a reduction in the cost of data labeling processes and the risk of overfitting. Although there are a lot of different approaches to data enhancement, for this thesis, we decided to employ random brightness, random contrast, random saturation, and random flip.

Figure 9

Augmented Image Examples



3.5 Transfer Learning

Transfer learning is the process of taking a network that has already been trained and retraining it based on new classes. In the case of object detection, transfer learning is done by importing the weights that were optimized on another dataset. Most of the time, it's not a best practice to start from scratch when training a very large Deep neural network.[22] Because of this, object detectors that use deep learning are often pre-trained on large datasets like ImageNet or COCO [20] Some of the reasons you might want to use transfer learning are: 1) collecting data is expensive and beyond the scope of the project; 2) using a pre-trained network enhances generalization; or 3) it reduces training time

because many features have already been learned from larger datasets. It is essential that the architecture of the model that was utilized during training, for instance on MS COCO, be like the design of the model that would be utilized during transfer learning. It is not necessary to make use of all the weights. For instance, only the weights from the first ten convolutional layers can be frozen and imported into a subsequent fresh untrained CNN model [21]. In this thesis, we imported COCO weights without freezing any layer to have a better starting point instead of starting randomly initialized weights.

3.6 Performance Metrics

Unlike image classification tasks, object detection is more challenging since it is necessary to determine whether or not the given image contains the required class and if it does, is it accurately localized. As a consequence of this, we need to examine its precision, recall, and intersection over union (IoU) scores. In this thesis, precision was used to measure how many successful runways were detected out of all the detected regions by the model. In a mathematical expression, precision is calculated by dividing the number of true positives (TP) by the total number of true positives and false positives (FP):

$$Precision = \frac{TP}{TP + FP}$$

When it comes to defining each detected area whether it's a true positive or false positive, we set a certain threshold value for the bounding box that overlaps between the ground truth label and the predicted region. True positive means that the object detector's prediction matches the ground truth label more than the threshold value. In contrast, a false

positive indicates that the detector tried to guess where a runway would be, but the area it picked up was wrong.

Table 1

Confusion Matrix Truth Table

Actual	**Predicted**	**Confusion Matrix**
Positive	Positive	TP
Positive	Negative	FN
Negative	Positive	FP
Negative	Negative	TN

With missed runways taken into consideration, recall is a measure of how many runways were correctly spotted. As a mathematical quantity, recall equals the proportion of correct identifications made to the total number of correct identifications and false negatives(FN):

$$Recall = \frac{TP}{TP + FN}$$

IoU is used to figure out how accurate object detection prediction is. IoU analyzes the overlap between the regions of Ground Truth and Prediction. An IoU score of 1 means that the bounding box that was predicted matches the bounding box that's been identified. A score of 0 means that the predicted bounding box and the true bounding box do not intersect at all. The MS-COCO dataset was evaluated using an IoU threshold of 0.5 [20],

which will also be used in this thesis meaning that every prediction that has above IoU score of 0.5 is a True Positive, and the ones below are considered as False Positive.

$$IoU = \frac{BB_{overlap}}{BB_{union}}$$

Mean Average Precision(mAP) on the other hand is now the most often used evaluation metric in object detectors. In Mask-RCNN, mAP@0.5 is the primary measurement. Evaluating mAP@0.5 means measuring the Average precision at a 0.5 IoU threshold across all classes in the dataset. Since we only have one class, which is a runway, the mAP is already the same as precision.

Chapter 4

Experimental Results

In the previous chapter, I presented the implementation of the data collection, preprocessing, training, and evaluation process. This section, I present and analyze the results.

4.1 Computation Environment

Although, I performed preliminary training on local computer with a CUDA-enabled GPU, I ended up choosing a remote server to speed up the model training phase. The operation system (OS) used was Ubuntu 20.04.3 LTS generic. For the entire project, the programming language Python 3.8 and PyTorch version 1.10 was used. We implemented the runway identification model using Detectron2 which allowed us to create Mask-RCNN. The model training was performed on Rowan Artificial Intelligence Lab's lambda machines which are equipped with NVIDIA Quadro RTX 8000 GPUs for 50000 iterations each. The longest training took 13 hours and 51 minutes with the help of 48GB VRAM. The version of CUDA and cuDNN was 11.1 and 8.0.5 respectively.

4.2 Results

We trained the neural network models using the Detecton2 framework for the Mask CNN instance segmentation task with two different ResNet backbone architectures which are ResNet 101-FPN and ResNeXt-101-FPN. To have a fair comparison between the models, both models were trained in the same configuration settings, meanthe ing same learning rate and same batch size for 50000 iterations. Although Resnet-101 and ResneXt-

101 yielded similar mAP scores @0.5:0.95, we noticed Resnet-101 yields significantly higher precision score at AP0.75 compared to ResneXt-101.

Table 2

Model Performance Comparison

Architecture	**Model Size**	**Map @0.5:0.95**	**AP @0.75**	**Recall**	**Training time**
ResNet 101	480MB	80%	85%	89%	9 hr 40 mins
ResNeXt 101	817MB	81%	92%	83%	13 hr 53 mins

The confidence score is a measure of how certain the model is that there is an object in the image, and it is used to filter out weak detections when evaluating a model for object detection. The threshold is usually set to 0.5 or above, meaning only detections with a confidence score higher than 0.5 are included in the detection results. A higher confidence score means that model believes that the detected area is likely to consist of target object, and a lower confidence score means that it's less likely. However, our task is to detect only one object. Therefore, setting the confidence threshold at 0.5 will likely uncover lots of false positives due to the fact that highways look pretty much the same as the runways from a top-down view. As a consequence, we set the confidence score threshold at 0.75 to prevent our model from detecting highways as runways.

Figure 10

ResNet 101 vs ResneXt 101

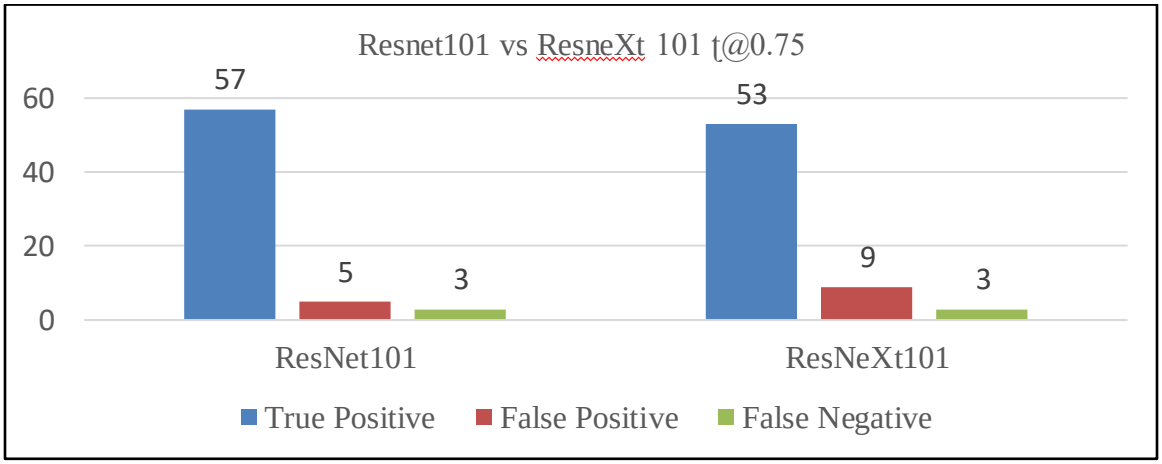


Figure 10 presents the quantitative results of true positive, false positive, and false negative predictions for both models when the threshold set to 0.75. Although both models failed detecting 3 runways (false negative), the model trained with ResNet-101 was able to successfully output more true positives and fewer false positive predictions.

Figure 11 shows an inference example of both models. Both models were able to detect true positive targets, however, ResNeXt101 outputted two other predictions which are not the runway. As it is apparent, the model trained with ResneXt101 could not differentiate between runway, taxiway, and highway.

Figure 11

Model Comparison #1

Figure 12 shows that while Resnet 101 is able to detect and segment both runways visible in an image, ResneXt 101 failed to detect the second runway which counts as a false negative.

Figure 12

Model Comparison #2

Figure 13 is an example of successful detection from both models. All detections count as true positive.

Figure 13

Model Comparison #3

Figure 14 is an example of both true and false positive predictions generated on the same image.

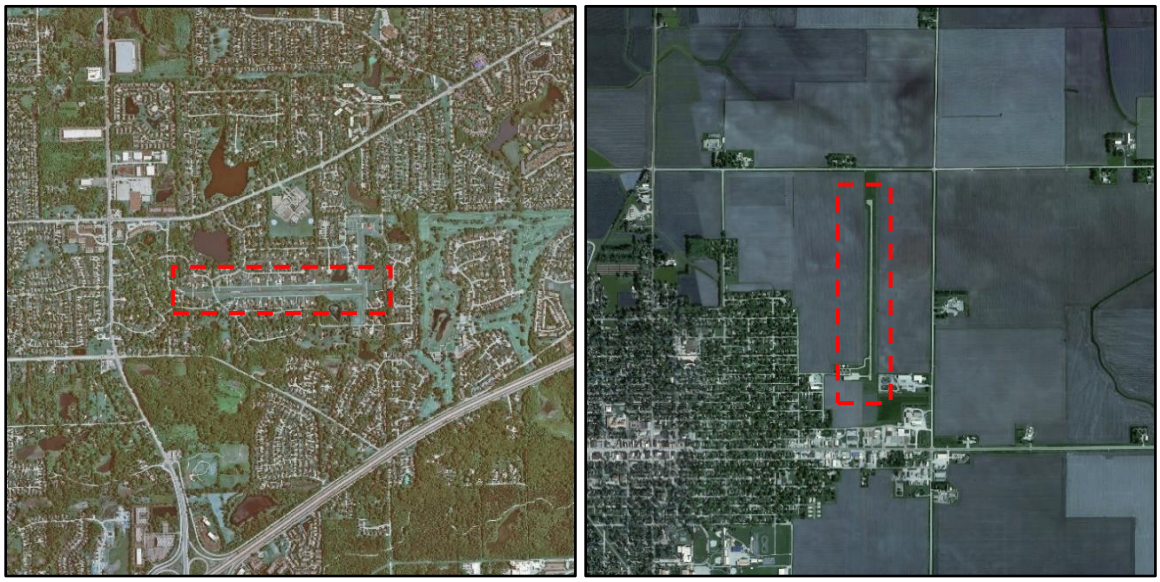
Figure 14

True Positive and False Positive Examples

Figure 15 is an example of a false negative where the model fails to identify the runway. Runway locations are denoted with red dashes on the figures.

Figure 15

False Negative Examples



4.3 Limitations

There are a few problems that still exist with the Google static maps service, despite the fact that this service makes it possible to easily access overhead imagery for the majority of coordinates. One of these problems is that the imagery for this service is not available at all zoom levels or in all geographic locations. This implies that, while you may be able to see a large-scale map of the city. The image will not change if you zoom in. In

some cases, the API sends images at a large scale, so objects in the image appear extremely small and contain relatively few features recognizable by the trained model.

Figure 16

Expected Scale vs Large Scale

Another issue with Google Maps is that certain regions have information that is out of date and does not match with actual topography of the land. For example, figure 17 represents the same region from two different satellite image provider. While Apple maps could locate the AKUTAN (AKQ) Airport which was built in 2012, google maps provides an image of bare land which was captured before airport was built. This is a big issue as it complicates our ability to locate and identify places.

Figure 17

Google Maps vs Apple Maps

4.4 Model Implementation on StreamLit

Streamlit is an open-source python library for constructing web apps for machine learning and data science projects which gives users a chance to actively interact with the dashboard. We developed a very simple front-end using StreamLit as it is compatible with most of the deep learning libraries, including Scikit-learn, Keras, PyTorch, NumPy, etc. The Runway Detector dashboard enables users to enter any coordinate on earth. Our model pulls the satellite image from Google Maps Static API and makes predictions. Finally, it returns an image with the predicted bounding box and segmented area for each runway.

Figure 18

Runway Detector – Webapp Dashboard

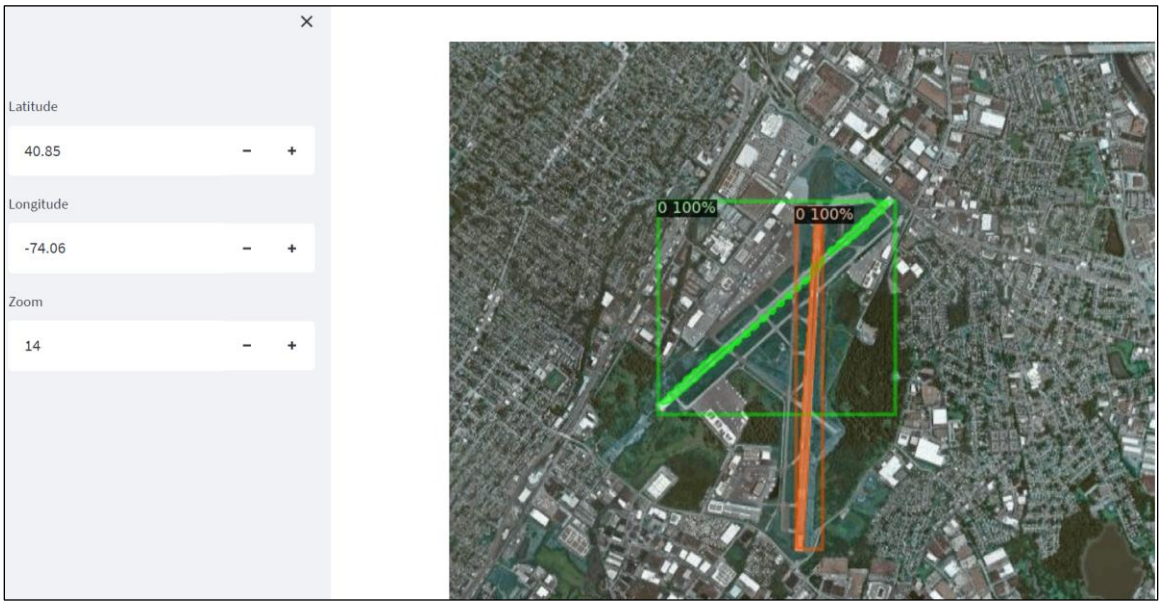


Figure 18 shows an image outputted by the model at given location.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

In this thesis, a region-based detector, Mask R-CNN was applied to detect and segment airport runways in satellite images. The aim of this project was to develop an algorithm with the ability to identify airport runways in satellite imagery. We achieved satisfactory results, although most object detection algorithms are rarely well suited to the object sizes or orientations present in satellite imagery, nor designed to handle images with hundreds of megapixels. The best model trained with ResNet-101 backbone reached 92% precision and 89% recall. Given that runways and highways are the most visually similar constructions from an aerial view, it is undeniable that the model needs improvement in order to distinguish between the two.

5.2 Future Work

In this thesis, two distinct models -ResNet101 and ResNeXt101 explored, and trained models with a relatively small dataset using Detectron2 framework. We suggest future works to extend the dataset including more runway instances and surface types to be able to verify more locations. It is also recommended to explore different satellite image data providers such as OpenStreetMap, Sentinel, Nasa Worldview, etc. as Google Maps itself comes with some drawbacks. To address the false positive predictions due to highway-runway similarity, runway detector model may be combined with a Road Detection model. This method can then be used to identify other parts of the airport facility such as taxiways, airplanes, and helicopters.

References

[1] Ö. Aytekin, U. Zongur, and U. Halici, "Texture-Based Airport Runway Detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 3, pp. 471–475, May 2013, doi: 10.1109/LGRS.2012.2210189.

[2] W. Commons, "File: RunwayDiagram.png — Wikimedia Commons, the free media repository." 2020. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:RunwayDiagram.png&oldid=454758344

[3] Dehong Liu, Lihan He, and L. Carin, "Airport detection in large aerial optical imagery," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. V-761–4. doi: 10.1109/ICASSP.2004.1327222.

[4] JunWei Han, Lei Guo, and YongSheng Bao, "A method of automatic finding airport runways in aerial images," in *6th International Conference on Signal Processing, 2002.*, pp. 731–734. doi: 10.1109/ICOSP.2002.1181160.

[5] M. Pritt and G. Chern, "Satellite Image Classification with Deep Learning," in *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Oct. 2017, pp. 1–7. doi: 10.1109/AIPR.2017.8457969.

[6] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," Apr. 2014.

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sep. 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," Nov. 2016.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.

[11] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," Dec. 2016.

[14] F. Meslet-Millet, E. Chaput, and S. Mouysset, "SPPNet: An Approach For Real-Time Encrypted Traffic Classification Using Deep Learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2021, pp. 1–6. doi: 10.1109/GLOBECOM46510.2021.9686037.

[15] A. Arnab *et al.*, "Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction," *IEEE Signal Process Mag*, vol. 35, no. 1, pp. 37–52, Jan. 2018, doi: 10.1109/MSP.2017.2762355.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Mar. 2017.

[17] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." 2019.

[18] "https://adip.faa.gov/agis/public/#/airportSearch/advanced."

[19] K. Wada, "Labelme: Image Polygonal Annotation with Python."

[20] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," May 2014.

[21] R. Medrano and J. Isaacs, "Channel Islands Feasibility of Real-Time Weed Detection in Turfgrass on an Edge Device," 2021.

[22] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*.