3-2022

Automated Aircraft Visual Inspection with Artificial Data Generation Enabled Deep Learning

Nathan J. Gaul

AUTOMATED AIRCRAFT VISUAL INSPECTION WITH ARTIFICIAL DATA GENERATION ENABLED DEEP LEARNING

THESIS

Nathan J Gaul, B.S.E.E., Captain, USAF

AFIT-ENG-MS-20-M-028

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-028

AUTOMATED AIRCRAFT VISUAL INSPECTION WITH ARTIFICIAL DATA GENERATION ENABLED DEEP LEARNING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Nathan J Gaul, B.S.E.E., B.S.E.E.

Captain, USAF

March 24, 2022

AFIT-ENG-MS-20-M-028

AUTOMATED AIRCRAFT VISUAL INSPECTION WITH ARTIFICIAL DATA GENERATION ENABLED DEEP LEARNING

THESIS

Nathan J Gaul, B.S.E.E., B.S.E.E.
Captain, USAF

Committee Membership:

Robert C Leishman, Ph.D
Chair

Scott L Nykl, Ph.D
Member

Joseph A Curro, Ph.D
Member

AFIT-ENG-MS-20-M-028

Abstract

Aircraft visual inspection, which is essential to daily maintenance of an aircraft, is expensive and time-consuming to perform. Augmenting trained maintenance technicians with automated unmanned aerial vehicles (UAVs) to collect and analyze images for aircraft inspection is an active research topic and a potential application of convolutional neural networks (CNNs). Training datasets for niche research topics such as aircraft visual inspection are small and challenging to produce, and the manual process of labeling these datasets often produces subjective annotations. Recently, researchers have produced several successful applications of artificially generated datasets with domain randomization for training CNNs for real-world computer vision problems. The research outlined herein builds upon this idea to create an artificial data generation pipeline inside Blender and generate an artificial dataset to train an instance-segmentation CNN model for car damage detection. This research then evaluates the real-world performance of several models, each pre-trained on the COCO dataset and fine-tuned on custom generated artificial dataset. We found that fine-tuning a model on the artificial dataset with domain randomization provided poor segmentation performance on the real-world car damage dataset. Despite the poor real-world performance of our model previous research suggest refinement of the artificial data generation process should provide better real-world performance.

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

AUTOMATED AIRCRAFT VISUAL INSPECTION WITH ARTIFICIAL DATA GENERATION ENABLED DEEP LEARNING

I. Introduction

1.1 Overview

This thesis documents work on generating an artificial machine learning dataset for training instance-segmentation models for the application of automated aircraft inspection. The lack of real-world public aircraft defect segmentation datasets and the large amount of data that convolutional neural networks require to perform instance segmentation fuels the need for a way to generate artificial data for such a niche application.

1.2 Problem Statement

The Air Force's requirements on the continued inspection of aircraft exteriors constitute a significant and continuous contribution to the maintenance costs of those aircraft. Automating the visual inspection process could help reduce the cost and time required to perform these inspections. In addition, incorporating an automated quadrotor to help perform visual inspections would allow maintenance personnel to avoid the difficult and dangerous positions required to inspect some areas of an aircraft. The research community has shown recent interest in automating visual aircraft inspection with the development and increasing accuracy of machine learning-based computer vision algorithms. Although, for niche problems such as visual aircraft inspection, datasets are difficult to create and are not shared publicly like other machine learning

datasets. This research proposes to address this lack of visual aircraft inspection training data by generating artificial data and annotations through an automated process.

Although the focus of the overall project associated with this research is on aircraft inspection, a suitable real-world aircraft damage dataset was not completed in time for the completion of this research, so we use a publicly available car-damage dataset as a stand-in.

1.2.1 Aircraft Visual Inspection

According to the Federal Aviation Administration (FAA) [7], the visual inspection of an aircraft is "the process of using the eye, alone or in conjunction with various aids, as the sensing mechanism from which judgments may be made about the condition of a unit..." Visual inspections are an essential aspect of aircraft maintenance and makeup over 80% of the inspections performed on large transport category aircraft [7]. During a visual inspection, maintenance personnel assess the overall condition of the aircraft, provide early detection of defects, assess errors in the manufacturing process, and obtain further information on parts showing evidence of damage. The Air Force is particularly interested in stealth aircraft due to the integrity of stealth materials being essential to their function. Inspections of these aircraft require personnel to examine the top, bottom, and sides of an aircraft quite closely. While the sides and underside of a stealth aircraft is relatively easy to investigate, inspecting the top is a different story. Studying the top of an aircraft at a close-enough range, while not touching or walking on the aircraft, which might cause further damage, is a challenging maintenance task. Automating the visual inspection with a quad-rotor would reduce the strain placed on maintenance personnel in such difficult situations.

1.3 Research Objectives

This research aims to (1) develop a synthetic aircraft defect dataset generation pipeline for use in training image segmentation models and (2) evaluate a computer vision model trained on artificial data for use in real-world aircraft inspection. The data generation pipeline will employ the open-source 3D creation suite Blender and leverage the Python scripting interface for automation. To evaluate the artificial training data generated by the pipeline, state-of-the-art convolutional neural network (CNN) models will train on different combinations of artificial and real-world data. Finally, we will compare the performance of each model on a set of real-world test data.

1.4 Methodology

This project will use the Blender 3D Creation Suite to generate realistic-looking data. Blender is a mature and capable open-source software tool that can produce realistic rendered images. Blender comes with an extensive and powerful Python scripting interface that allows for automating the data generation process. Once a large dataset of artificial data is generated and the collection of a small set of real-world data is achieved, evaluating the artificial dataset's effectiveness will be performed by comparing several models trained on different combinations of each dataset. Comparisons of these models will be performed on a subset of the real-world data, set apart in advance for testing.

1.5 Research Contributions

The following are contributions made in this thesis:

1. A customizable artificial dataset generation pipeline with domain randomiza-

tion, packaged as a Blender add-on.

2. An artificial car damage instance segmentation dataset.

3. Evaluation of our artificial car damage dataset for pre-training a sate-of-the-art instance segmentation model against a real-world dataset.

1.6 Thesis Organization

This document is organized in the scholarly article thesis format. Chapter II provides detailed explanations of relevant information, including aircraft surface defects, object detection within images, deep neural networks, previous works on artificial data generation, and previous works on automated aircraft inspection. This information is condensed into abbreviated versions in the two scientific articles included in this thesis to support independent publication. Chapter III is the first scholarly article detailing the artificial data generation pipeline and preliminary aircraft dataset created for this research. Chapter IV is the second scholarly article that details the generation and evaluation of an artificial car damage dataset for pre-training a real-world instance segmentation model. Finally, chapter V details our conclusions drawn from the results.

II. Background and Literature Review

This chapter presents the fundamental background information and previous literature used to support the development of the research contributions made within this thesis. Section 2.1 details the aircraft surface defects and their proper documentation markings. Section 2.2 summarizes different tasks associated computer vision object detection and the metrics used to evaluate their performance. Section 2.3 deatails the basics and background of convolutional neural networks and their use in image segmentation. Section 2.4 details previous works that use artificial data generation for training image segmentation models. Section 2.5 details previous works exploring the automation of aircraft visual inspections.

2.1 Aircraft Surface Defects

Information on low observable aircraft surface defects in the public domain is limited or nonexistent; therefore, we will focus on defects observed on aircraft in general.

Properly marking aircraft defects provides a clear indication of the existence of previously observed damage and the progression of that damage over time. Any damage not marked can be considered newly formed and should be marked accordingly. The marking guides detailed below provide a standard way to mark different types of defects and inform the output of the segmentation algorithm developed in Chapter III.

2.1.1 Missing or Damaged Material

Figure 1 shows two examples of damage where the material is missing in the first column, proper markings in the second column, and incorrect markings in the third

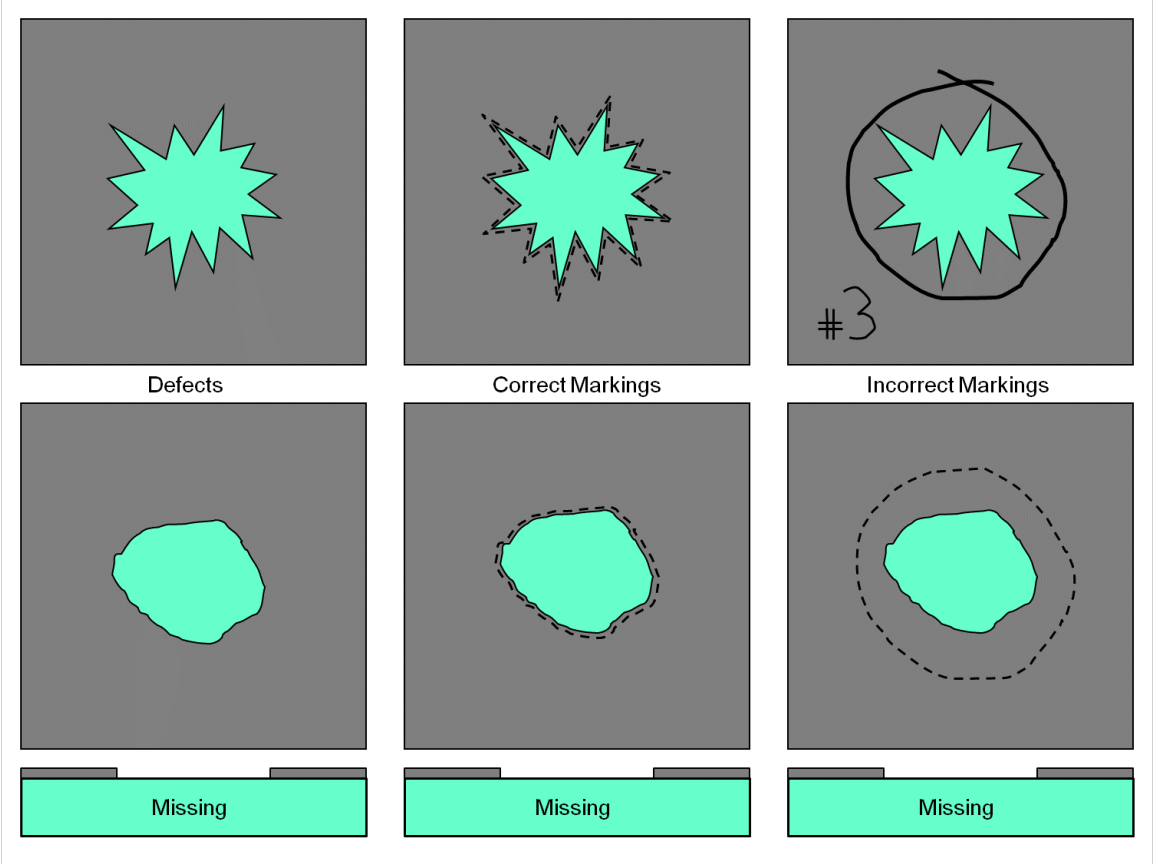

Figure 1: Markings for Missing Material

column. Markings for this sort of damage should be a dashed line at the boundary of the defect and should not include any other numbering or labels.

2.1.2 Dented or Disbonded Material

Figure 2, in the same manner as Figure 1, shows the correct and incorrect ways to mark a dent or disbonded area on an aircraft. Markings should be dashed line at the boundary of the dent or affected area.

2.1.3 Defect Progression

Figure 3 exhibits the correct marking technique to indicate defect progression for either type of defect. The dashed marking is expanded to the newly affected area

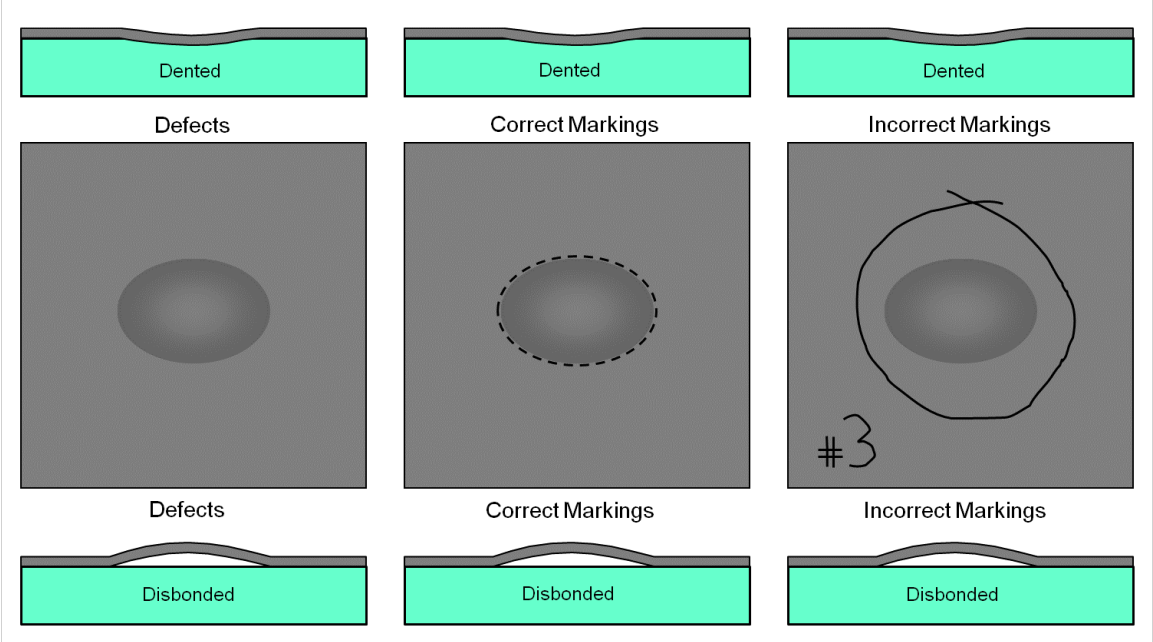

Figure 2: Markings for Dented and Disbonded Material

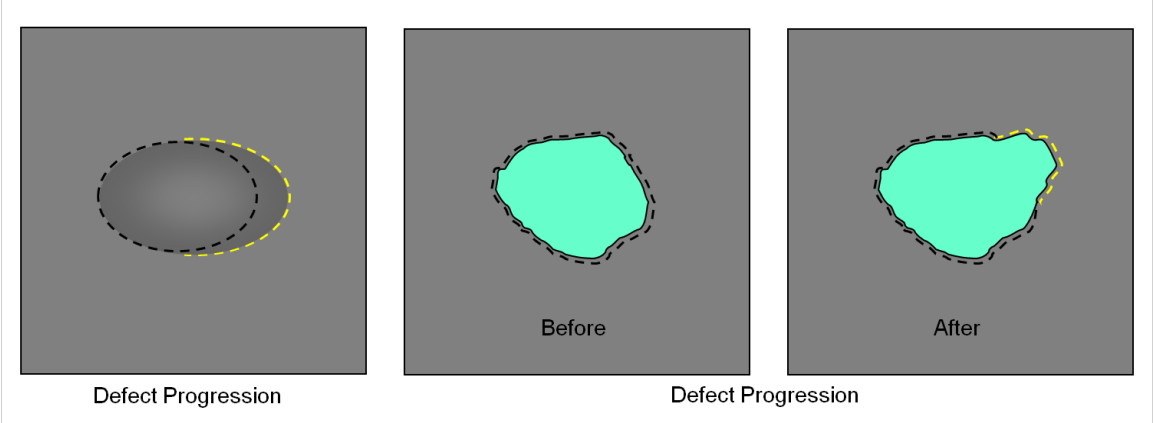

Figure 3: Markings for Progressive Damage. The black dashed line represents the previously documented defect, while yellow shows the recently progressed defect area.

of a previously existing defect for any advancement. In Figure 3, the black dashed line represents the previously documented defect, while yellow shows the recently progressed defect area.

2.2 Detecting Objects within Images

There are four different objectives for which an algorithm might detect objects within an image in the computer vision field. In order of increasing fidelity, these objectives are image recognition, object detection, semantic segmentation, and instance segmentation. Image recognition is the binary determination of whether an image contains a class of objects or not. Depending on the problem, image recognition may also determine if an image contains one or more of a specified set of classes or not. Object detection is the location and identification, indicated by a bounding box, of individual objects of interest within an image. Semantic segmentation further specifies the pixels identified as a particular class within an image. Finally, instance segmentation distinguishes between instances of an object class when segmenting the pixels of an image. As the fidelity of each objective increases, so does the difficulty of performing that task. This research will take on the task of instance segmentation of aircraft damage.

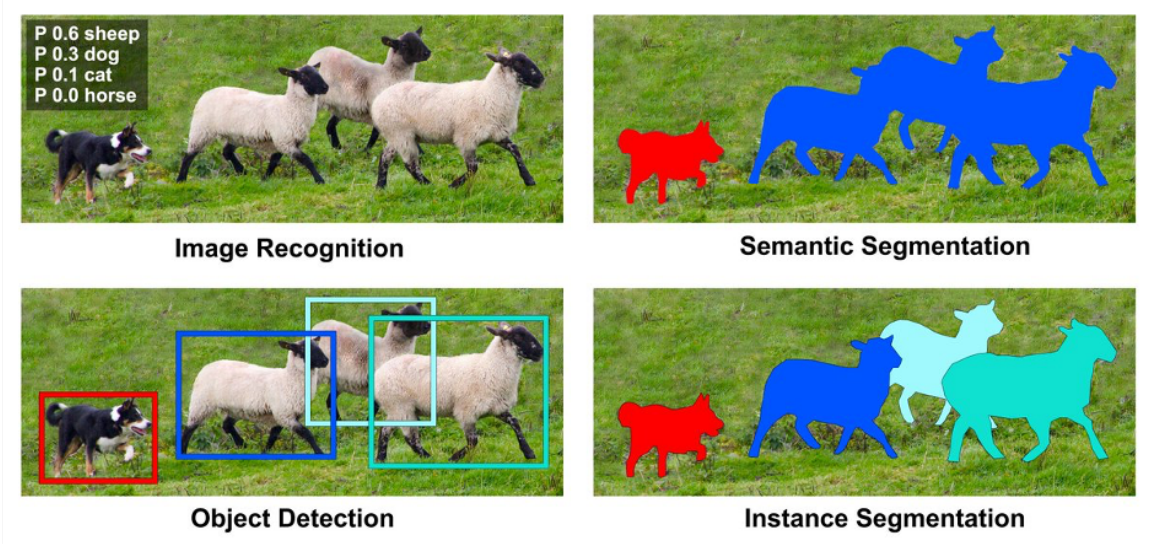

Figure 4: Visual representation of image recognition, object detection, semantic segmentation, and instance segmentation. From [5]

2.2.1 Instance Segmentation Evaluation Metrics

Described below are the metrics commonly used to evaluate the performance of instance segmentation algorithms.

A true positive (TP) is when a model correctly predicts a pixel or region as belonging to a class. A false positive (FP) is when a model incorrectly predicts a pixel or region as belonging to a class and it does not belong. A true negative (TN) is when a model correctly predicts a pixel or region as not belonging to a class. A false negative (FN) is when a model incorrectly predicts a pixel or region as not belonging to a class and it does belong.

Recall, as calculated in

$$Recall = \frac{TP}{TP + FN}, \tag{1}$$

is the number of correctly predicted positives divided by the number of ground truths. Precision, as calculated in

$$Precision = \frac{TP}{TP + FP}, \tag{2}$$

is the number of correctly predicted positives divided by the total number of predictions. Recall can be seen as a measure of the quantity of TPs, while precision can be seen as a measure of the TPs. Higher recall means the predictions contain most of the relevant results, while higher precision means the predictions contain more relevant results than irrelevant ones.

Intersection over union (IoU), as calculated in

$$IoU = \frac{target \cap prediction}{target \cup prediction}, \tag{3}$$

is a metric calculated for each prediction and its associated ground truth and is used

to determine whether a prediction is correct or not, i.e. its truthfulness. IoU, equation (3), is the intersection between the prediction and the ground truth (target) divided by their union (see Figure 7). For a prediction to be considered a TP, its IoU with its associated ground truth has to be equal to or above some threshold; otherwise, it is considered a FP (see Figure 8).

Another factor when determining the truthfulness of a prediction is the confidence level, a value from 0.0 to 1.0, the model reports for every prediction. By varying what we consider to be the cutoff confidence level, we can change whether a prediction is deemed a TP or FP.

The average precision (AP), the metric used to evaluate the models in this work, is an attempt by researchers to summarize the overall performance of a model and gives a single value that can be used to compare the performance of various models.

To calculate the AP metric of a set of predictions over a dataset, we must create a precision-recall (PR) curve. For a single class of predictions, we calculate the recall and precision values given a threshold IoU score and a chosen confidence level; this produces a single PR pair. To create the PR curve, PR pairs are calculated across the range of possible confidence levels, i.e., 0.0 to 1.0, which determine the value pair's rank. The PR curve is created by plotting these PR value pairs by descending rank, with precision on the y-axis and recall on the x-axis, and connecting line segments between subsequent points to create a piecewise function, $p(r)$, (see Figure 9). The AP of this PR curve, equation (4), is then calculated by averaging the precision values at 101 equally spaced recall values along the piecewise curve, as calculated in

$$AP = \frac{1}{101} \sum_{r \in \{0.00, 0.01, \ldots, 0.99, 1.00\}} p(r). \tag{4}$$

For multiclass datasets, an AP is calculated for each class, and the mean is found, called the mean average precision (mAP). Following the COCO researcher's example,

most research assumes AP to mean mAP. AP and mAP are equivalent for our research as our evaluation dataset has a single class of objects.

Each AP or mAP is calculated given some threshold IoU which may introduce some bias in the evaluation metric related to the correctness of a prediction. We take the average AP across a range of IoU threshold values to address this bias. Following the COCO [8] evaluation metric, we average the AP at IoU thresholds from 0.5 to 0.95 at an interval of 0.05, represented in this research's results as AP[0.5:0.05:0.95].

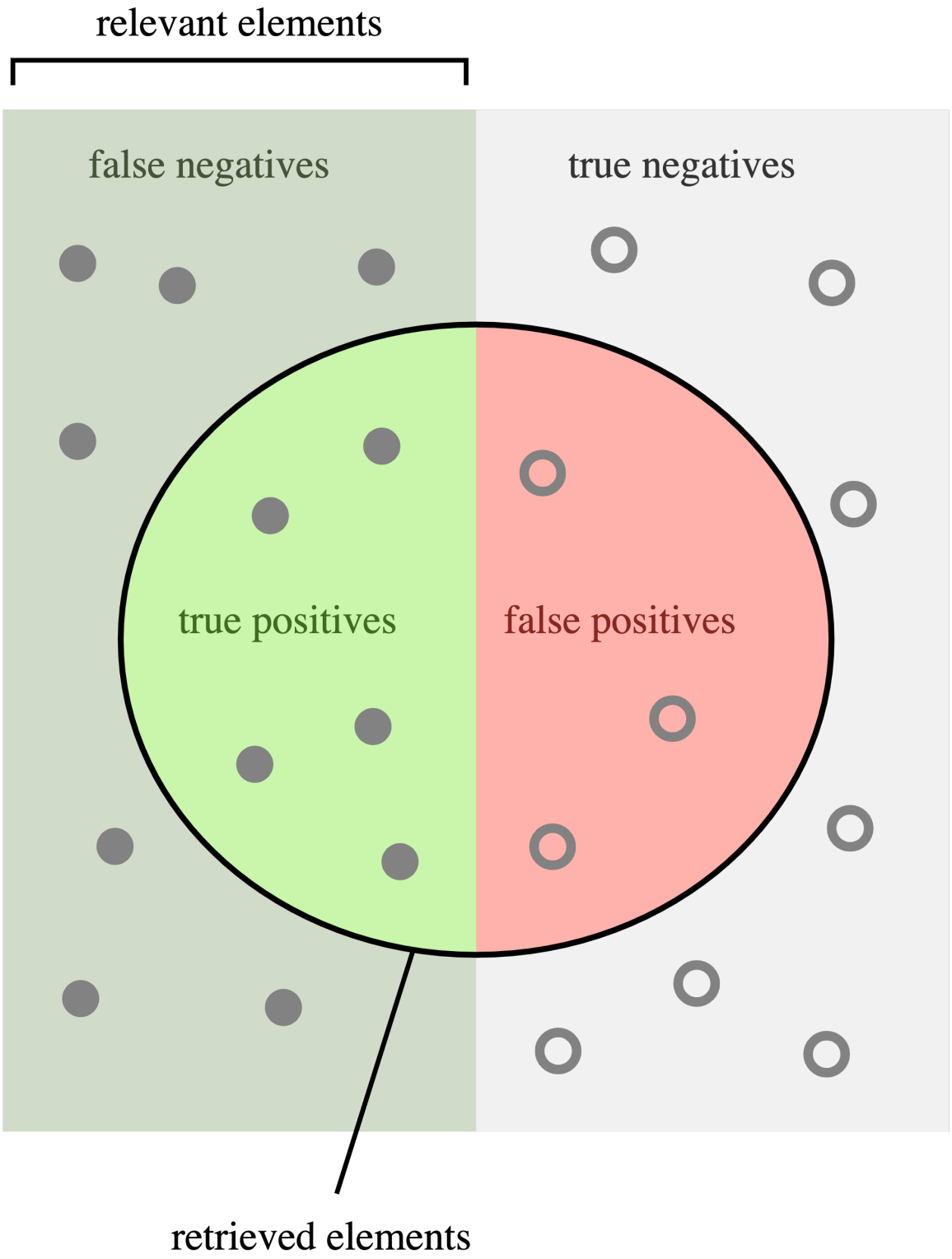

Figure 5: Visual representation of true positives, false positives, true negatives, and false negatives. Modified from [6]

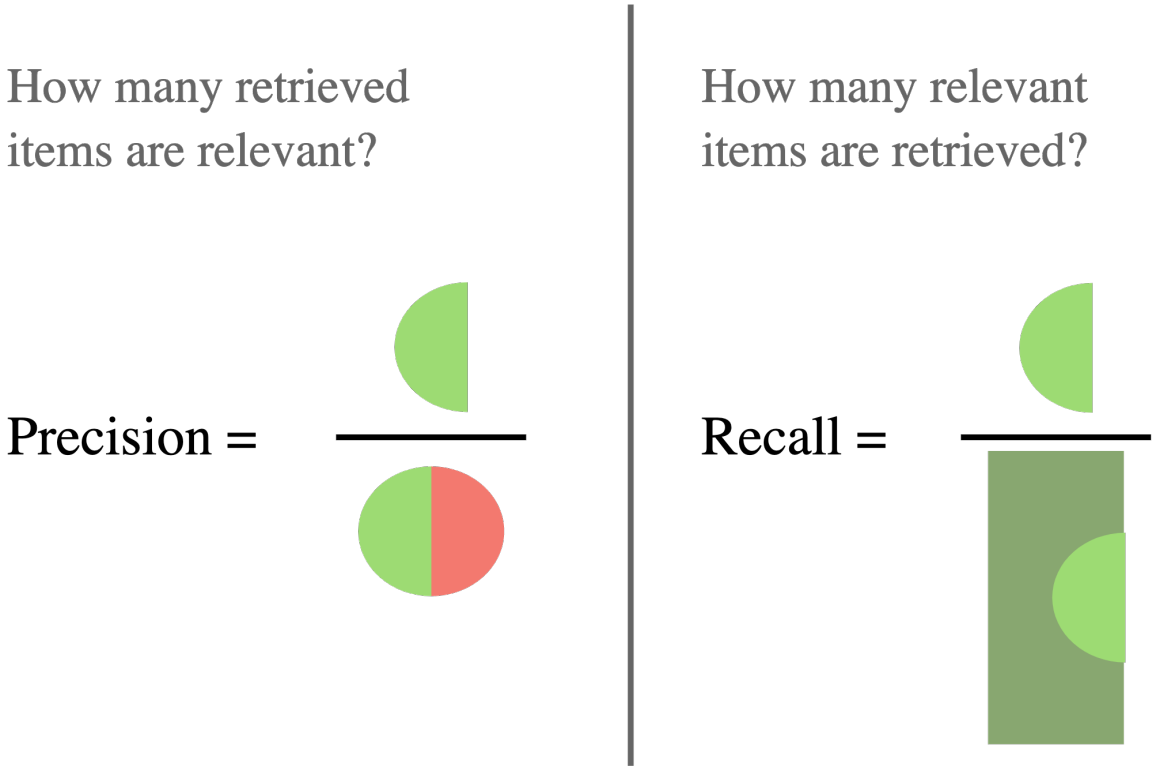

Figure 6: Visual representation of precision and recall. Modified from [6]

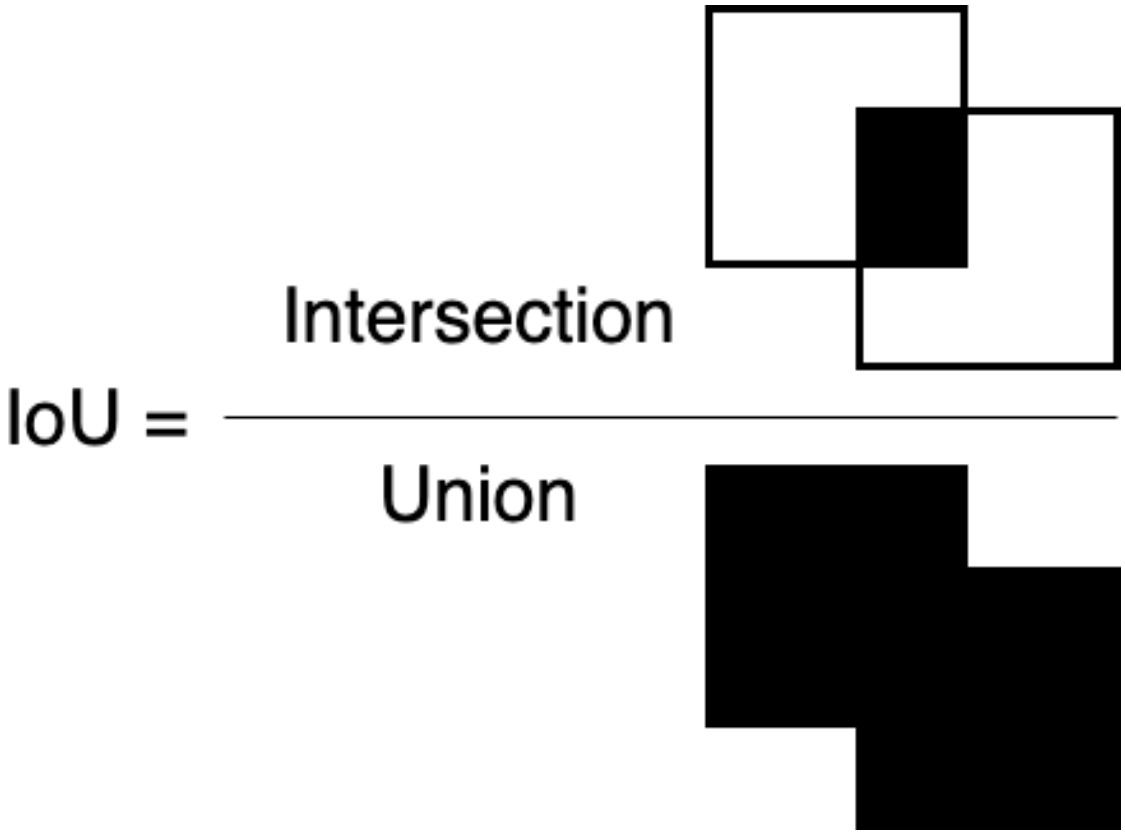

Figure 7: Visual representation of Intersection over Union (IoU)

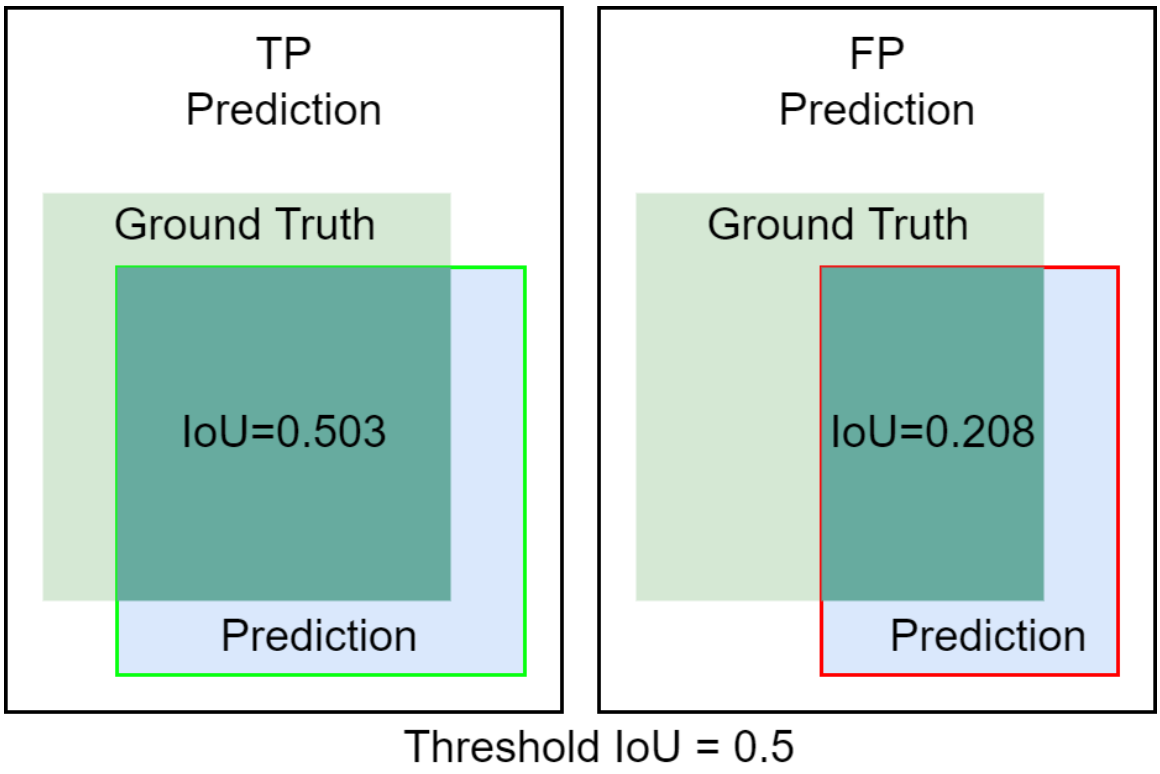

Figure 8: Visual representation of Intersection over Union A visual example of threshold IoU and how it factors into the truthfulness of a prediction

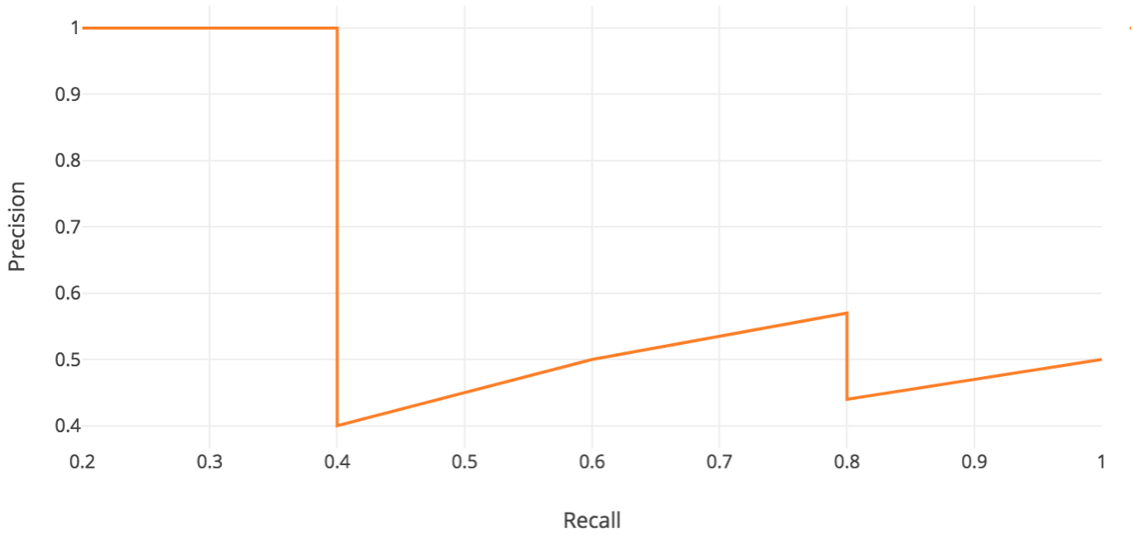

Figure 9: An example PR curve which is a piecewise function connecting precision-recall pairs in order of rank

2.3 Deep Neural Networks

Deep Learning is a subset of machine learning and is based on artificial neural networks (ANNs), which automatically learn representations for feature extraction through optimizing numerical weights utilizing large amounts of data. The structure of ANNs is inspired by the connections between neurons in biological systems [9]. An ANN is composed of artificial neurons connected in a network, similar to synapses in a biological brain. The artificial neurons in an ANN, otherwise known as nodes, are organized into layers where each layer's nodes are connected to the nodes in the adjacent layers. Like biological neural networks, different configurations of nodes and layers are better suited to specific tasks, like understanding vision or language. The

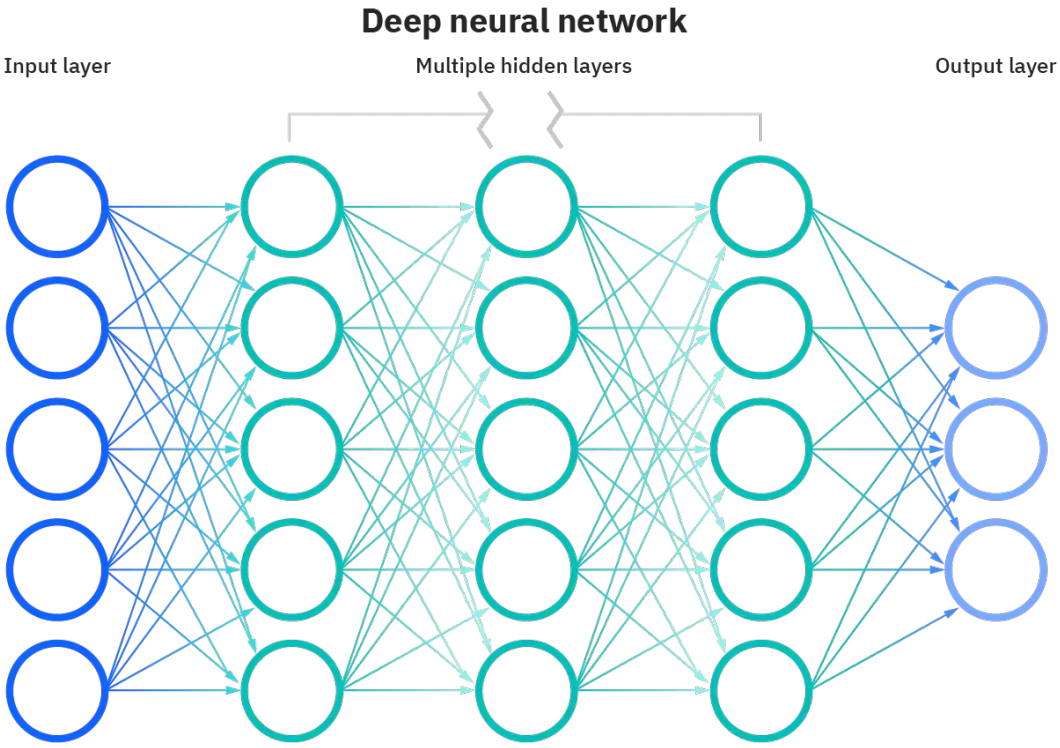

Figure 10: A high level visual representation of a Deep Neural Network. The nodes of the input, hidden, and output layers are represented. Sources from [1]

"deep" in deep learning refers to an ANN with multiple hidden layers connecting the input and output layers (see Figure 10). The depth of a network allows it to model complex non-linear relationships such as many computer vision problems including object detection and segmentation within images [10].

Convolutional neural networks (CNNs) are a class of deep learning neural networks that excel at specific problems in computer vision, and in this case [10, 11, 12, 13, 14]. Convolution, the critical process of the CNN, is particularly equivariant to the translation of features within an input datum, meaning a translation of a feature in the input of a CNN results in the same translation at the CNN's output. The CNN's ability to recognize features independent of location makes it particularly effective at processing visual data. For example, a CNN cat or dog classifier can identify whether a cat or dog is within an image independent of the animal's location within that image [15].

Within the context of CNNs, convolution is the dot product between two matrices. The first matrix is a set of learnable parameters called a kernel. The other is a restricted portion of the input, called the receptive field. The repeated application of convolution with a single kernel over an entire input creates an activation map that represents the kernel's response at every position within the input [15].

This repeated use of a kernel for multiple convolutions across the image takes advantage of parameter sharing, leading to the CNN's property of equivariance of translation.

In general, a CNN is composed of layers that perform one of three operations: convolution, non-linear function, or pooling. After a convolutional layer, the resulting activation map is run through a non-linear function, such as rectified linear unit (ReLU) (used earliest in [16, 17]). The outputs of the non-linear function layer are then further processed by a pooling layer. A pooling layer replaces the value at

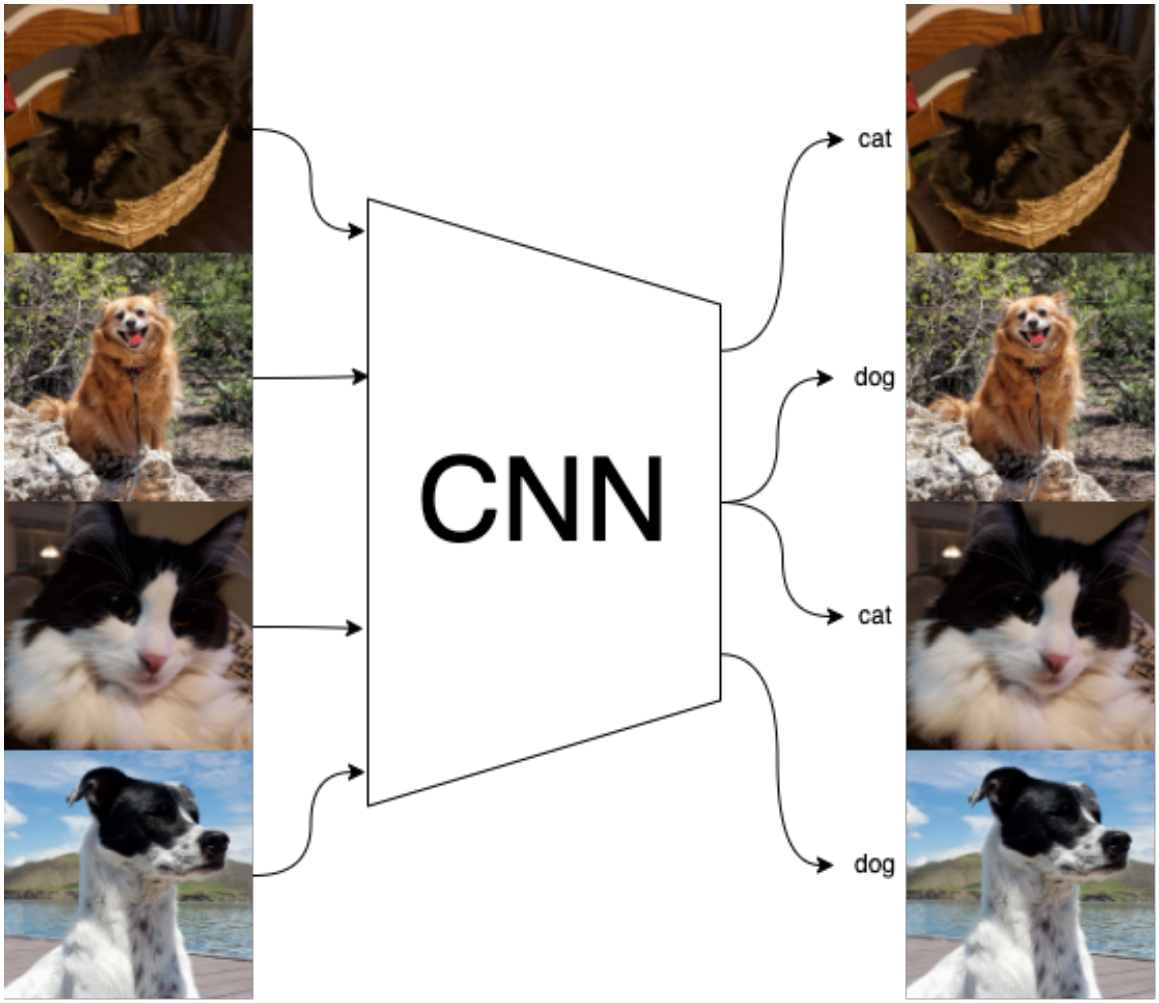

Figure 11: High level representation of a CNN cat or dog classifier

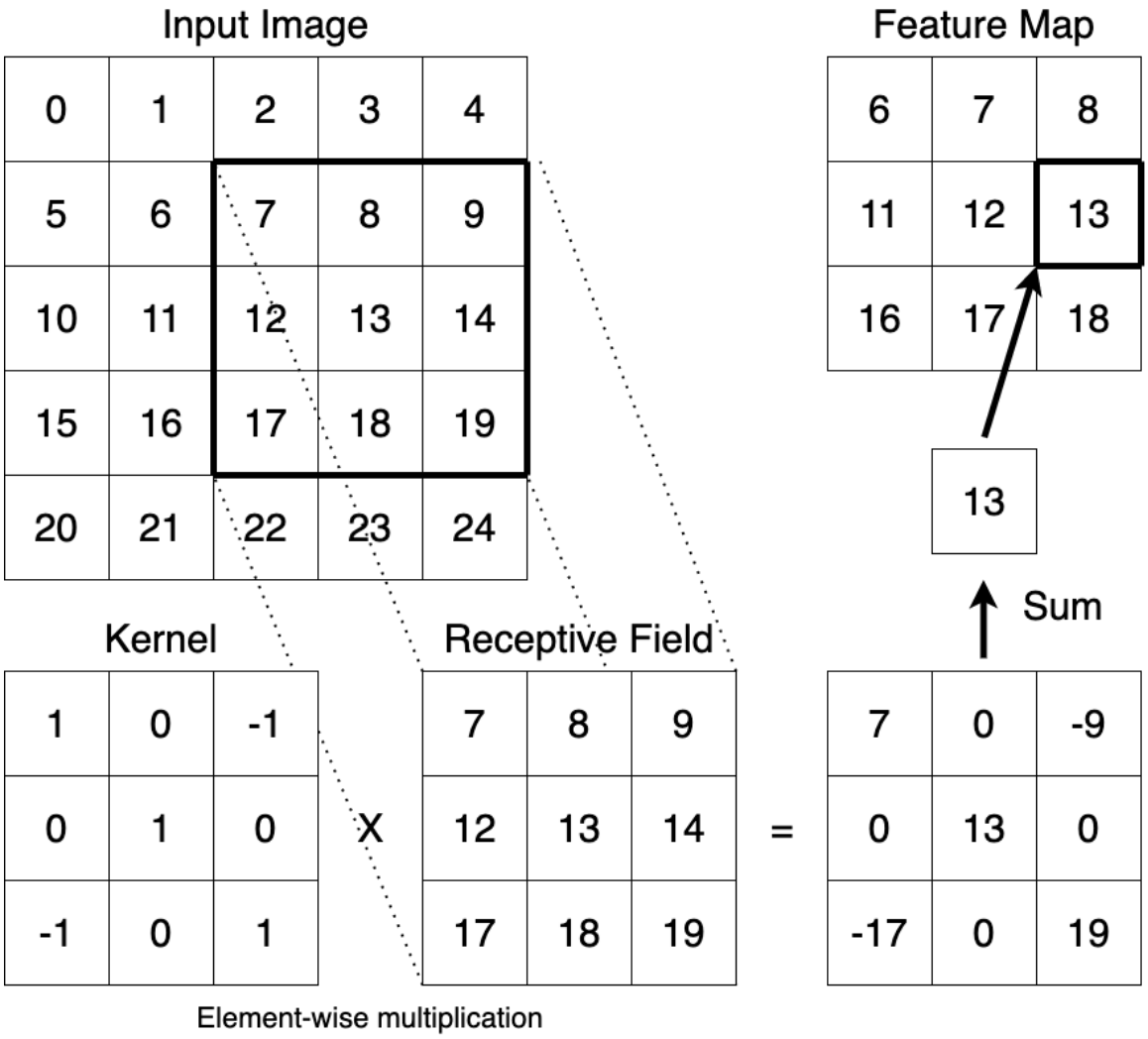

Figure 12: An illustration of a 3x3 convolution of a 5x5 input image

every location with a statistic of the surrounding values. Pooling helps make the representation invariant to small translations in the data. A network invariant to small translations in its input means that instead of recognizing a feature when it is in the exact right location, the network can detect a feature in an approximate location.

2.3.1 Deep Learning Image Segmentation Architectures

Early approaches to image segmentation included TextonForest [18] and Random Forest-based classifiers [19] but were quickly outpaced by deep learning approaches, which allowed more precise and faster segmentation [20]. AlexNet's development in 2017 [21] enabled the advancement of deep learning classification and paved the way for more advanced CNN architectures for classification, object detection, and segmentation. Since then, the development of image segmentation architectures has been a highly active area of research, with Lateef and Ruichek surveying over 100 different architectures in 2019 [22]. Of these architectures, this research will be evaluating our dataset with a Mask-RCNN [2] based network for its high performance in the COCO 2016 [8] instance-segmentation challenge and ease of use in the Detectron2 object detection framework [23].

Mask R-CNN is an extension of Faster R-CNN [24], an architecture for object classification and localization, that adds pixel-level object instance-segmentation. Mask R-CNN comprises several sub-networks: a backbone, a detector, and several region of interest (ROI) heads. As Mask R-CNN is a meta-architecture, an architecture that can be instantiated with different building blocks [25], the exact composition of Mask R-CNN depends on the designer's exact implementation. The implementation of the Mask R-CNN architecture used for this research is the GeneralizedRCNN architecture from Facebook AI Research's Detectron2 [23] object detection platform.

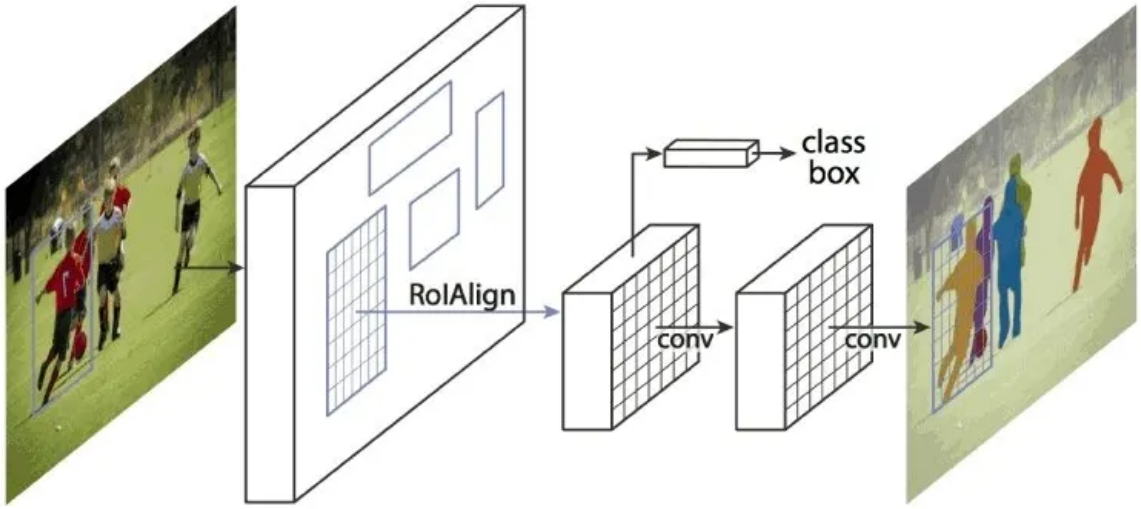

Figure 13: The Mask R-CNN instance segmentation framework [2]

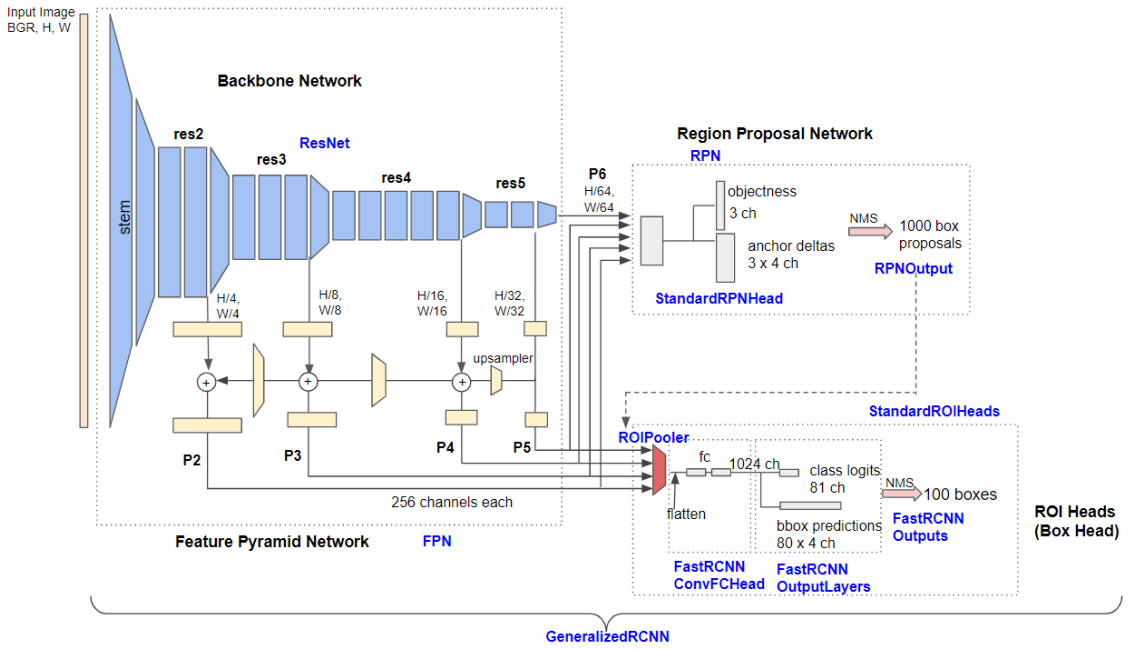

Figure 14: System diagram of Detectron2's GeneralizedRCNN architecture [3]. Note: the system diagram does not include the ROI mask head.

The GeneralizedRCNN architecture (see Figure 14) is configured with a ResNet-50 [26] based Feature Pyramidal Network (FPN) [27] as the network backbone. This backbone acts as a feature extractor that feeds feature and spatial information as feature maps to the Region Proposal Network (RPN) detector and ROI heads. The RPN applies a sliding window over the FPN feature maps and predicts the 'objectness', probability of containing a target object, and the boundary box at each window location [27]. The RPN produces several proposed object locations indicated as boxes and feeds them for consideration to the individual ROI heads [27]. The ROI heads, a box head, and a mask head, in the case of GeneralizedRCNN, take these proposed object locations and the feature maps produced from the FPN and produce the final outputs from the network. The ROI box head produces object bounding box and their associated object classification predictions. The ROI mask head produces an instance segmentation mask associated with each predicted object [24].

2.3.1.1 Mask R-CNN's Loss Function

The loss function for Mask R-CNN is the sum of the following individual losses:

- Classification loss in the RPN (see [28])
- Localisation loss in the RPN (see [28])
- Classification loss in the ROI Box head (see [24])
- Localisation loss in the ROI Box head (see [24])
- Mask loss in the ROI Mask head (see [2])

2.3.1.2 Transfer Learning with Mask R-CNN

Training the Mask R-CNN network from scratch would require significant computational resources and an extensive training dataset. Thankfully, a technique termed

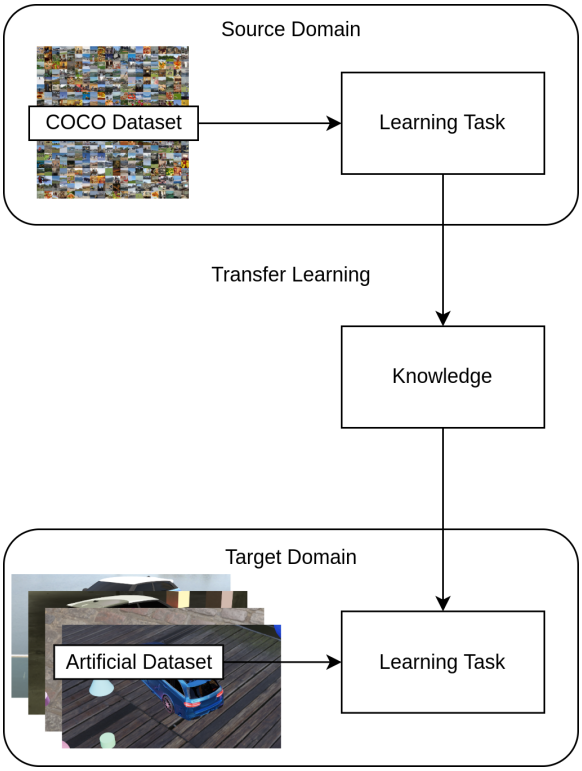

Figure 15: Visual description of transfer learning for fine-tuning a CNN model. Modified from [4]

transfer learning (TL) can help reduce the resources necessary to train the model on the artificially generated dataset. TL is the reuse of a pre-trained model for a new task requiring less computing resources and less training data than training the model from scratch. First, a model is trained on a large object detection dataset, the COCO dataset in the case of this research. This initial training is performed in what is called the source domain. Second, a portion of the same model's trained parameters are frozen (prevented from changing) and trained again on the dataset of interest. This second training is performed in what is called the target domain. See Figure 15 for a visual description. The freezing of a portion of the model's pre-trained parameters between the two training runs transfers some knowledge from the source domain to the target domain, hence the term transfer learning (TL). Training fewer parameters in the target domain compared to the source domain required fewer computing resources.

TL has been leveraged with Mask R-CNN by many researchers in recent years for various applications. Rehman et al. [4] used Mask R-CNN with a ResNet-50 and ResNet-101 [26] based FPN backbones to create an apple leaf disease detector by pre-training the ResNets on the ImageNet dataset [29] and fine-tuning on the PlantVillage dataset [30], an open access dataset of images on plant health. Ullo et al. [31] used Mask R-CNN with a ResNet-50 and ResNet-101 based backbones to create an aerial image landslide detector by pre-training on the COCO dataset [8] and fine-tuning on a small custom dataset of landslide images. Similarly, Mask R-CNN and TL have been used for solder joint detection [32], lesion detection [33], lung segmentation [34], among many other applications.

2.4 Previous Works on Artificial Data Generation for Image Segmentation

Broggi et al. [35] present an early example of generating a synthetic dataset for training a model for pedestrian image classification and detection. The author's pre-generated images of a 3D model of a pedestrian for feature matching against real-world images using a hand-coded algorithm. The authors found their implementation inaccurate when matching with bounding boxes containing real candidates but believed further development using a dynamically animated model was worth investigating. Although not showing complete success, this early work on synthetic datasets showed the further study of such dataset generation might prove fruitful for similar computer vision problems.

Marín et al. [36] created a synthetic dataset for pedestrian image segmentation using the Half-Life 2 video game to generate a dataset of images of realistic-looking city scenes. Using a game mod, the authors generated the labeled ground truth annotations associated with the pedestrians in each image. The classifier trained on the virtual dataset showed a similar performance to the classifier trained on real-world data. This work is an early demonstration of using realistic modern computer graphics and game engines to generate an image segmentation dataset for computer vision.

Jaderberg et al. [37] trained a 90k word classifying CNN for real-world scenes with synthetically generated data only. The synthetic data engine created for the work allowed for unlimited training data and required no human labeling. This engine used a five-step process to create realistic images that match the distribution of words found in real-world scenes. The data generation process described in this work may be helpful when designing the data generation process developed in our work, particularly for inserting and blending synthetic features into background images for producing

realistic images.

Ritcher et al. [38] generated a synthetic city-scene segmentation dataset consisting of 25,000 images and semantic ground-truth label maps by using the closed-source video game Grand Theft Auto V. By creating a software wrapper around the video game's rendering engine the authors were able to capture their dataset with relatively low effort when compared to gathering real-world segmentation datasets.

To address the need for large datasets required for training scene semantic models, Handa et al. [39] generated a synthetic per-pixel depth dataset from computer-generated 3D scenes. Using a public repository of 3D models for everyday household items, the researchers automatically generate 3D scenes based on object occurrence statistics in real scenes. The researcher applied post-processing to the generated depth maps to match the real world, so they were consistent with the noise of a Microsoft Kinect, using a Kinect noise model. This research is an example of using a post-processing technique to bridge the domain gap between the virtual and real worlds, often experienced when using synthetically generated data.

As an alternative to real-world and synthetic datasets, Abu Alhaija et al. [40] augment real-world images with virtual objects to generate an object instance segmentation dataset. By augmenting the KITTI 2015 [41] dataset with 3D modeled cars from [42], the authors found the model trained on their dataset was able to generalize better than the models trained on either synthetic data or real-world data. This research provides an alternative to the two methods being evaluated in our research and maybe a fruitful direction for further research in our niche.

Hinterstoisser et al. [43] evaluated the approach of using the Faster-RCNN architecture [24] with frozen feature extractor layers and the remaining layers trained on synthetic data to perform object detection. The authors placed 3D CAD models in cluttered images with random noise and lighting to generate their synthetic data.

They concluded that freezing the feature extraction network layers always performed better when compared to an unfrozen network trained on synthetic data alone. Thus, this research presents a generally applicable technique that may improve the performance of a model when paired with synthetic data.

2.5 Previous Works on Automated Visual Inspection in Aircraft Maintenance

Many researchers are currently exploring automated vision systems for improving and streamlining inspections for aircraft maintenance.

Rice et al. [44] developed both a drone and an overhead camera system to perform automated inspections of commercial aircraft. They produced two vision algorithms to detect and segment anomalies on painted areas of an aircraft and another for detecting missing screws. The authors did not include any implementation details of the vision algorithms in this work.

Miranda et al. [45], as part of the Donecle [46] project, create a combination of a classical and a CNN-based approach to detect missing and loose screws on the outside of an aircraft within images taken from a UAV. A CNN, with an unspecified architecture, performs screw detection. The detected screws are then compared to a computer-aided design (CAD) model to determine if any are missing. Given there are markings around the screws indicating the proper alignment of the screw head, the authors also implemented a classical machine vision algorithm to detect if a screw is loose.

Miranda et al. [47] analyses several different machine learning approaches for aircraft defect classification on an extremely class imbalanced dataset. This work is essential to automated aircraft inspection due to certain classes of damage, like lightning burns, being both a rare occurrence and critical to the health of an aircraft.

The authors explored data-level techniques such as oversampling and undersampling; algorithm-level methods such as deep neural networks, support vector machines, and few-shot learning algorithms [48]; and various hybrids of the previously mentioned methods. From their analysis, the authors found a hybrid approach where the output of a deep neural network is used to classify the most represented classes in the dataset while the output of a few-shot prototypical network is used for infrequent classes of damage. This work was a part of the Donecle project [46], an automated unmanned aerial vehicle (UAV) for aircraft inspection image collection.

Dogru et al. [49] expanded on the work of Bouarfa et al. [50] of applying the MASK R-CNN [2] architecture to automated aircraft inspection by balancing, homogenizing, and augmenting the original dataset. This work also adds a pre-classifier before the MASK R-CNN network to avoid false positives in images that likely do not contain any dents. This research is necessary because it shows several methods for improving the performance of previous methods on such a niche problem as aircraft inspection.

Bouarfa et al. [50] demonstrate the potential of using a MASK R-CNN architecture [2] deep-learning model to classify and segment aircraft dents within an image. Although architectures like MASK R-CNN require a higher computation cost when compared to popular object detection algorithms like You Only Look Once (YOLO) [51], MASK R-CNN provides pixel-level resolution. In contrast, YOLO only provides a bounding box around the detected object. Pixel level classification for aircraft dent detection is helpful because dents often have irregular shapes, and documenting the change in aircraft dents is essential for traceability.

III. Scholarly Article: Artificial Dataset Generation for Automated Aircraft Visual Inspection

Abstract

Aircraft visual inspection is both essential to the maintenance of an aircraft, and expensive and time-consuming to perform. Augmenting trained maintenance professionals with automated unmanned aerial vehicles (UAVs) to collect and analyze images for aircraft inspection is an active research topic and a potential application of convolutional neural networks (CNNs). Training datasets for niche research topics such as aircraft visual inspection are small and challenging to produce, and the manual labeling process of these datasets produces subjective annotations. Self-driving car researchers have experimented with generating artificial datasets with modern computer graphics that can train for real-world driving scenarios. Our research borrows this idea and proposes a work-in-progress artificial data generation pipeline to create 3D rendered automatically annotated images for training CNNs for automated visual aircraft inspection.

3.1 Introduction

Aircraft visual inspection is an essential step in the maintenance and safety of every aircraft. These inspections are expensive, time-consuming, and require highly trained maintenance personnel in potentially dangerous positions around an aircraft [52]. During a visual inspection, maintenance personnel are trained to locate, classify, and document the state of any damaged sites visible on the outside of the aircraft. These damage sites may include dents, lightning strike damage, aircraft markings, and deteriorated paint quality [50], to name a few. Objective and accurate detection and evaluation of each instance of damage are crucial for the aircraft's safe operation.

Although maintenance personnel are highly trained, there can often be discrepancies between different inspectors, and sometimes the same inspector can disagree with a previous assessment they have made. The quality and accuracy of these inspections can be affected by lighting conditions, time pressures and inspector fatigue. With these many challenges of the inspection process, automated aircraft visual inspection may be an inexpensive and objective tool for maintenance personnel.

Several companies and research groups are exploring the application of computer automation to the domain of aircraft pre-flight inspection. ROBAIR [53] is a project that has developed a climbing robot with a suite of sensors designed to inspect the wings and fuselage of aircraft for various issues. Air-Cobot [54] is an automated ground drone that captures images and 3D scans for automated pre-flight inspections by leveraging several hand selection image-processing approaches. Donecle [55] has created an automated quadrotor to collect photographs of the outside of an aircraft and feed those images to software for analysis, reporting, and tracing; they claim to reduce traditional inspection times by a factor of ten. Donecle does not reveal their method for automatically detecting damage. Mainblades [56] has developed a similar UAV and software solution that utilizes machine learning to identify and analyze damage in complex real-world environments. Machine learning algorithms, such as CNNs, have shown promise in this field for their accuracy and flexibility of input data.

CNNs are a type of deep neural network, a subset of machine learning, that are useful for analyzing images [10]. CNNs have been researched extensively in the medical field because of their outstanding performance in medical imaging classification and reduced handcrafted feature requirements [57, 58, 59, 60, 61]. For CNNs to perform well at a task, they need to be trained on a sizeable, labeled dataset that captures the variability and statistics of the real-world problem at hand. Collecting and

manually labeling CNN datasets proves to be both expensive and time-consuming [62, 40, 36, 63], and because humans perform the labeling process, the annotation can be subjective. Researchers have explored creating labeled artificial images by rendering scenes from 3D virtual worlds to help solve the problems associated with adequately producing a labeled dataset for training a CNN [64, 65, 38, 37]. The purpose of this research is to do just that with a virtually damaged 3D aircraft model.

3.2 Related Work

Due to recent advancements in graphical computing, creating realistic-looking computer-generated synthetic datasets for real-world problems is a popular thread among artificial intelligence (AI) researchers. Creating datasets in this way allows researchers to automatically generate a potentially unlimited number of images with annotations provided for free. Since the annotations are determined from a particular view by the software itself, they will always be accurate and objective. Broggi et al. [35] generated a dataset of images for training a pedestrian detecting algorithm by manipulating the pose of a simplified 3D model of a human being. Marin et al. [36] generated a pedestrian detection image dataset by 'driving' a car inside a virtual city by leveraging video games. Abu Alhaija et al. [40] places realistic 3D models of cars into images from the real-world KITTI dataset [66] in realistic positions using Blender [67], an open-source 3D creation suite, to increase the number of images available. Gaidon et al. [62] recreated a portion of the KITTI dataset [66] inside of Unity [68], a popular 3D game engine, for use in evaluating self-driving car algorithms. Our work proposes a similar synthetic dataset generation pipeline for use in training automated visual aircraft inspection CNNs.

3.3 Proposed Research

This section describes the proposed approach to data generated through the automated rendering of realistic damage on the virtual aircraft. Our proposed pipeline is composed of four essential components: (i) a detailed, high-quality 3D model of the aircraft of interest, (ii) accurate visual representations of aircraft skin damage, (iii) multiple realistic scenes and environments to position around the aircraft model, and (iv) a scriptable 3D rendering software to generate the images.

Unlike the 3D car model repository used in [40], we did not find a central repository of high-quality aircraft models to use for this research. Aircraft inspection only focuses on a single aircraft, so multiple aircraft models are not required for every scene, as in the driving scenes in [40]. Creating these models by hand would be time-consuming and are outside the scope of this research; therefore, we will use a pre-made 3D model from a marketplace such as [69, 70, 71].

Figure 16: A realistic render of a 3D aircraft model

To properly train a CNN on this synthetic dataset, the 3D modeled aircraft needs

to show signs of damage. 'Inflicting' realistic damage onto our selected 3D model is considered to be a challenging part of this research and will require further study, experimentation, and input from expert aircraft maintainers. However, the general idea is to draw the damage onto the texture of a 3D model. If a particular class of data has some volumetric effect, such as a dent, apply that effect to the model's normal map. The details of each instance of damage will be randomized but based on the appearance of actual aircraft damage created through consultations with maintenance experts.

Creating a realistic scene around the aircraft model matches what a real-world dataset may show is important. Unfortunately, at the time of writing, the real-world dataset is still being produced for this research and will not be released in time for this research. This means that we do not have a real-world scene to directly reproduce our virtual environment, unlike the Virtual KITTI [66] dataset. Until the real-world dataset is received, we will rely on expert feedback and the standard practices of aircraft maintainers to create a realistic virtual scene. We plan to place the virtual aircraft inside a well-lit hangar and adjust the scene based on the feedback from our expert contacts.

This research uses Blender, similar to [40], to create the virtual scene and automate the rendering of the synthetic dataset. Blender is a powerful open-source 3D creation suite with a built-in Python automation application programming interface (API) that can provide the functionality and flexibility required for this research project. Importing UAV paths generated with the coverage path planning algorithm developed in prior work [72] into Blender will allow for automated realistic camera views outside the aircraft model. Automating the generation of the image annotations will be accomplished through changes in the rendering engine's settings.

We believe combining these components will create an artificial data generation

pipeline to produce a training dataset for a CNN able to detect and segment aircraft damage on a real-world dataset.

3.4 Artificial Data Generation Pipeline

3.4.1 Blender

3.4.1.1 Aircraft Models

Due to the time and expertise required to create realistic aircraft 3D models, this research utilizes assets available from services such as CG Trader [69], SketchFab [70], and TurboSquid [71]. These services allow professional designers and artists to list their 3D models for purchase and use by others. Thanks to these services and Blender's ability to import a wide range of 3D file formats, numerous aircraft models, free and otherwise, were utilized in the generation of the aircraft damage dataset.

3.4.1.2 Python API

Blender comes bundled with a Python interpreter, which is used internally to draw the user interface and for some built-in tools. Addons and custom scripts can also use this Python environment to interact with Blender's data, classes, and functions. Blender provides API access primarily through the 'bpy' Python module. A brief description of the submodules available in 'bpy' is in Table 1 below.

This research extensively utilizes the Blender Python API to automate the data generation process and provide a user interface inside Blender for configuration.

3.4.1.3 Materials

Materials in Blender control the color, texture, and how light interacts with an object, i.e., the appearance of an object. Using Blender's flexible shading node system,

Table 1: Blender Python Submodules

Name	**Submodule**	**Description**
Context Access	bpy.context	Provides read-only access to the current state of Blender.
Data Access	bpy.data	Provides access to Blender's internal data.
Message Bus	bpy.msgbus	Provides a publisher/subscriber interface to changes initiated via the Python API or sliders, fields, and buttons in Blender's user interface.
Operators	bpy.ops	Provides access to calling operators built into Blender.
Types	bpy.types	A "directory" of Blender's built-in Python classes.
Utilities	bpy.utils	A collection of useful utility functions not associated with Blender's internal data.
Path Utilities	bpy.path	A collection of useful functions for dealing with paths in Blender.
Application Data	bpy.app	Contains application values that remain unchanged during runtime.
Property Definitions	bpy.props	Defines properties to extend Blender's internal data.

a large variety of different materials, including plastic, glass, metal, cloth, skin, hair, smoke, and fire, are open to designers.

Materials are composed of three basic types of shaders; surface, volume, and displacement. The surface shader controls the visual texture and light interaction at the surface of an object. For example, a material would employ a surface shader with near-perfect light reflection to create a mirror. The volume shader controls the appearance of the interior of an object. Modeling smoke or fire would only require the use of a volume shader since they are voluminous substances. A displacement shader adds detail to the shape of an object by altering its geometry through the displacement of points along its surface.

We use various materials to fulfill three objectives in this project; aircraft realism, randomized artificial aircraft damage generation, and generation of segmentation label masks. We chose the aircraft models used for this project due to their high realism, including the quality of their included materials. Blender's node-based shader system makes mixing materials easy, a feature heavily used in adding randomized artificial damage over the existing aircraft model materials. To generate the image segmentation masks, we used a shader that emits a single light color in the annotation object's materials.

3.4.1.4 Rendering

A rendering engine turns a 3D scene into a 2D image by processing the scene's objects, cameras, lights, and materials. Blender comes pre-installed with three rendering engines; Eevee, Cycles, and Workbench. Eevee and Cycles are both physics based rendering engines, while Eevee is designed for real-time rendering and Cycles for light path tracing. Workbench, however, is designed for modeling and animation preview and is not intended for rendering final images. This research used both Eevee

and Cycles for generating data, Eevee for generating most training data, and Cycles for generating a smaller portion of high detailed data.

Figure 17: A render created with the Eevee Rendering Engine.

3.4.1.5 View Generation

For each data image and label pair generated, the pipeline creates a unique scene before rendering. This strategy enables a large and generalized dataset appropriate for training a successful semantic segmentation deep-learning model. The three aspects of the scene that are randomized include the artificial damage pattern, lighting placement, and camera view.

Random Damage The base material created to represent material removed from the skin of an aircraft (i.e., damage) is composed of a surface shader with color and metallic properties and a displacement shader for removing or indenting the material. The specifics of such a material are airframe dependent and have been informed by carefully inspecting real-world images and in-hand samples. This material is then

Figure 18: A render created with the Cycles Rendering Engine.

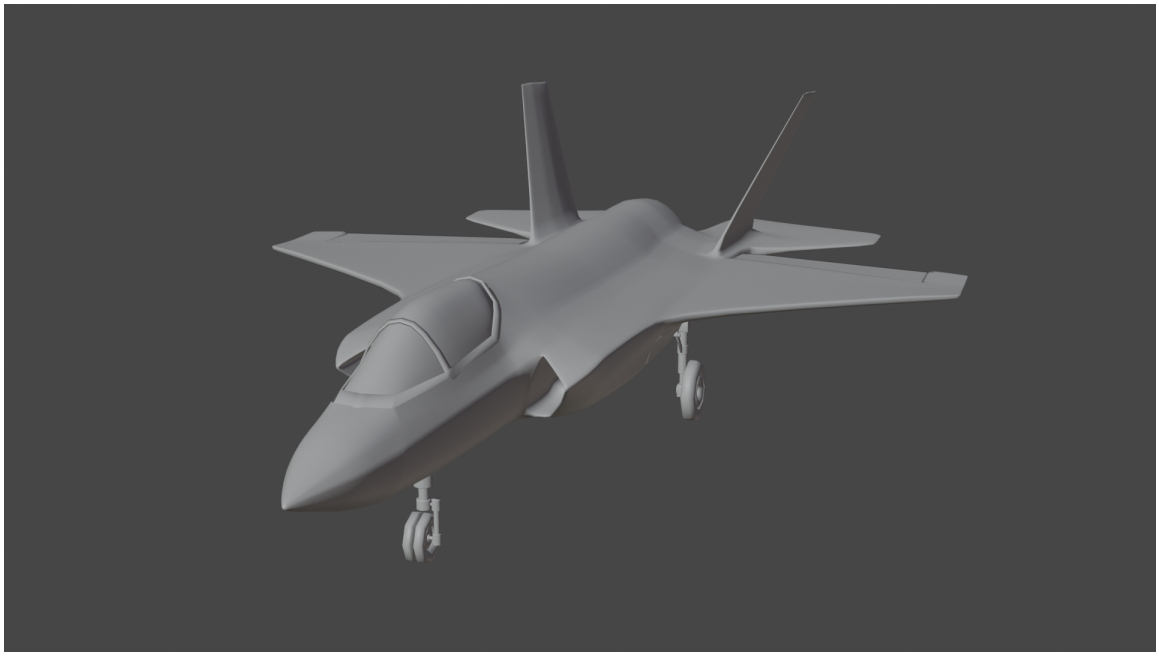

Figure 19: A render created with the Workbench Rendering Engine.

randomly patterned over the aircraft 3D models with an adjustable parameter to vary the pattern for each new view.

Lighting Placement Randomly lighting the aircraft scene is essential for generalizing the artificial model training dataset. If the data was generated with a constant light source at the same position throughout all views, the resulting model will likely not perform well on real-world data where lighting is not consistent. For this research, we chose a simple random lighting scheme where a 'Sun' light source is placed randomly in 3D space a constant distance away and above from the geometric center of the aircraft. For every new data render, the light source's position is updated according to this scheme. This strategy allows for quickly generating a large number of lighting conditions, allowing for a large and generalized artificial dataset.

Camera Placement The camera's position in 3D space is randomized in the same manner as the scene's lighting. A new camera position is chosen randomly at a specified distance away from and above the 3D model for each rendered image while the camera's focus is anchored to the model.

3.4.2 Label Post Processing

The labels rendered from Blender for each view of the aircraft are composed of masks of solid color. Each unique color in the mask classifies a pixel in the image. By default, the labels are multiclass with black representing the background, red the aircraft, and blue the damaged areas of the aircraft. Multiclass masks are used primarily for flexibility. Although this research is interested in binary pixel classification, future use may require multiclass image segmentation.

Due to the multiclass nature of the labels and the shading inherent in Blender's rendering engines, the labels need to be post-processed with the following procedure

to prepare them for processing with a deep-learning model; 1) color quantization, 2) class isolation, and 3) black and white conversion.The colors produced in the raw rendered labels are not precisely the distinct colors used when setting up the render due to shadows and processing the renders. Color quantization reduces the colors in an image to a select set of color values; this process allows for precise distinction between the pixel classes by color value. With the color space of the label reduced, the classes are separated into separate images to transform the single multiclass image into separate binary images. With the classes separated into several different images, the labels are then reduced to black and white, allowing easier use as an input to a CNN. Although these operations could be performed within a Python environment with a module such as Pillow, ImageMagick, a C-based command-line tool, performs these operations much faster and can be called from a Python instance.

3.4.2.1 ImageMagick convert and mogrify

ImageMagick is a command-line-based multi-threaded image editing suite. Using ImageMagick's 'convert' tool one can perform image format conversions, resizing, blurring, cropping, and dithering, among many other operations. 'mogrify' can perform these same operations but on an entire directory of images, which is must faster than calling 'convert' on a single image at a time. Using Python's standard 'subprocess' module, ImageMagick can be called from within the data generation pipeline, given ImageMagick is properly installed on the system. This research uses the 'mogrify' command to post-process the data labels after rendering.

3.5 Dataset Examples

Figure 20 shows four examples of artificially generated data sets created with the pipeline detailed in this research. Each series of three images consists of a realistic

image of aircraft damage, its associated multiclass (colored) label, and binary (black and white) label. These examples represent what the images in the final dataset(s) may look like as we may change parameters to improve the accuracy of the trained models.

Figure 20: Examples of data, multiclass labels, and binary labels generated by the Pipeline that would be used as inputs to a CNN for training

3.6 Future Work

This research will progress with the training of a model of the U-Net architecture [61] on a large set of artificially generated data with the evaluation of the model performed against a limited set of real-world data. Studies to be considered for further research include a comparison of data rendered with Eevee and Cycles and variations on scene lighting, camera placement, and artificial damage parameters. Artificial datasets generated with variations of these different parameters may prove to improve model performance on the available set of real-world data.

IV. Scholarly Article: Artificial Dataset Generation and its Effect on the Performance of Car Damage Instance Segmentation Models

Abstract

Aircraft visual inspection is essential to the maintenance of an aircraft, yet is expensive and time-consuming to perform. Augmenting trained maintenance professionals with automated unmanned aerial vehicles (UAVs) to collect and analyze images for aircraft inspection is an active research topic and a potential application of convolutional neural networks (CNNs). Training datasets for niche research topics such as aircraft visual inspection are small and challenging to produce, and the manual labeling process of these datasets produces subjective annotations. Recently, researchers have produced several successful applications of artificially generated datasets with domain randomization for training CNNs for real-world computer vision problems. Our research borrows this idea to create an artificial data generation pipeline and dataset to pre-train an instance segmentation CNN for damage detection. Although the original intention was to test this process on a real-world aircraft damage segmentation dataset, such a dataset was not available for this research. Instead, we evaluate the performance of several models, each pre-trained with various combinations of the newly created artificial dataset and the COCO dataset, on a small publicly available car damage dataset. We found that pre-training a model on the artificial dataset with domain randomization showed better performance than the model trained from scratch on the real-world dataset. However, when comparing the performance of two models already trained on the COCO dataset, training on our artificial dataset degraded the model's performance on the real dataset.

4.1 Introduction

The visual inspection of aircraft is essential for their continued operation and maintenance [73, 74]. Aircraft inspection is vital to the United States Air Force, with its ever-expanding fleet of military aircraft. Requiring several highly trained maintenance personnel, these lengthy inspections throughout the life of each aircraft are a costly and time-consuming element of aircraft maintenance [75]. Often requiring an inspector to explore the airframe manually, visual inspections also pose a considerable safety risk to these personnel [73].

Because of the importance and challenges associated with aircraft visual inspection, the Autonomy and Navigation Technology (ANT) Center at Air Force Institute of Technology (AFIT) is developing an UAV system that is capable of aiding in the inspection process by autonomously examining the outside of an aircraft. The goals of this system are to:

1. Determine an efficient flight path around an aircraft.
2. Take photographs of the entire exterior of the aircraft while in flight.
3. Detect any aircraft defects found in the images for review by trained maintenance personnel.
4. Localize the any defects found to the appropriate panel.

This particular research aims to develop a computer vision algorithm to fulfill the third goal of the overall system: detection of aircraft defects captured in the images collected by the UAV.

Current state-of-the-art automated object detection systems are predominately convolutional neural networks (CNNs) because of their high performance and applicability to many object detection tasks [76]. However, developing a CNN requires a

large amount of ground-truth labeled images. When done manually, the cost and manpower required to collect and annotate such a dataset can be too great for such niche problems, such as aircraft inspection. Researchers have recently explored computer-generated graphics to generate artificial datasets where there is a lack of real-world ground truth data [37, 65, 64, 39, 77, 43, 78]. The research described herein takes the approach of evaluating a synthetic dataset for use in training a CNN for real-world aircraft inspection. Unfortunately, an adequate real-world test dataset of aircraft damage was not available in time for this research, so we evaluated the proposed methods on a small dataset that contains real-world car damage [8].

4.2 Background

4.2.1 Artificial Dataset Generation

When applying a CNN to a machine vision problem, one essential requirement is to obtain a large dataset with which to train and evaluate a model. The size of a dataset required for training a CNN depends primarily on the complexity of the problem. Obtaining a large dataset is trivial for applications such as automated driving, where public datasets are plentiful (e.g. [66, 40, 79, 80]). For niche applications, such as aircraft damage segmentation, finding or collecting a dataset of significant size is difficult as fewer real-world examples exist. Creating a well-designed image segmentation dataset usually requires manual collection and mapping of hundreds or thousands of images, a time-consuming and monotonous task for humans to perform. Researchers have explored computer graphics to generate artificial datasets for training models for various real-world applications to address this challenge.

When designing a synthetic dataset for training a machine learning model, the critical challenge is bridging the performance gap when comparing a model's performance on the artificial dataset to those on the real-world dataset. This issue is

termed ‘domain shift’ and is the model over-fitting to the synthetic dataset and not generalizing well to real-world data. Researchers have explored several techniques collectively known as domain randomization to help bridge the gap resulting from ‘domain shift’ [64, 81, 78, 82].

Domain randomization is a data-generation phase solution to domain shift that adds variations to the image space of the synthetic data to avoid model overfitting. Adding domain randomization to an artificial dataset consists of randomizing the aspects of the data superfluous to the problem at hand. This research employs domain randomization techniques that fall into either content or style variation.

4.2.2 Instance Segmentation

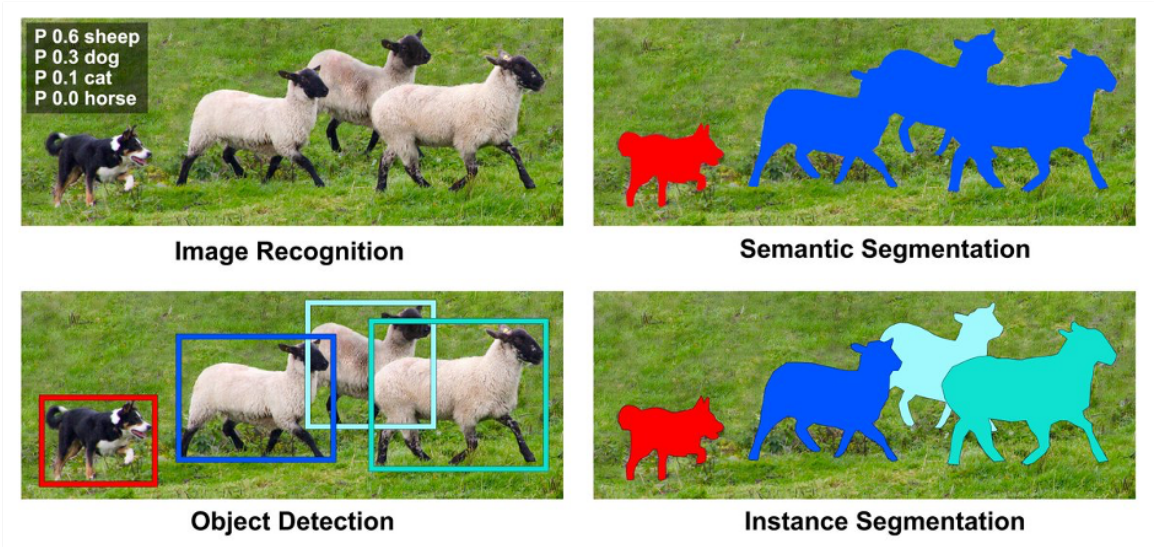

Figure 21: Visual example of the four different object-detection tasks [5]

Instance segmentation of an image is when individual objects in an image are detected and delineated. For example, processing a photograph of several sheep with an instance segmentation algorithm would produce a mask labeling the pixels making up each sheep individually.

Instance segmentation is one of several object detection problems in image processing, including image classification, object detection, semantic segmentation, and instance segmentation. A high-level description of each of these tasks is below.

- **Image Recognition:** assigning a categorical label or labels to an image depending on its contents
- **Object Detection:** locating individual instances of a particular object class within an image, where locations are indicated by bounding box
- **Semantic segmentation:** locating groups or instances of a particular object class within an image pixel by pixel
- **Instance Segmentation:** locating individual instances of a particular object class within an image pixel by pixel

The product of an instance segmentation method is a mask indicating the location, class, and instance of the objects of interest in a particular image. Compared to the ground truth, the correctness of a mask is measured in several ways, including precision, recall, and a more complicated metric called average precision.

To understand these performance metrics, we must first understand true positives, false positives, true negatives, and false negatives. A true positive (TP) is when a model correctly predicts a pixel or region as belonging to a class. A false positive (FP) is when a model incorrectly predicts a pixel or region as belonging to a class and it does not belong. A true negative (TN) is when a model correctly predicts a pixel or region as not belonging to a class. A false negative (FN) is when a model incorrectly predicts a pixel or region as not belonging to a class and it does belong.

Recall, as calculated in

$$Recall = \frac{TP}{TP + FN}, \tag{5}$$

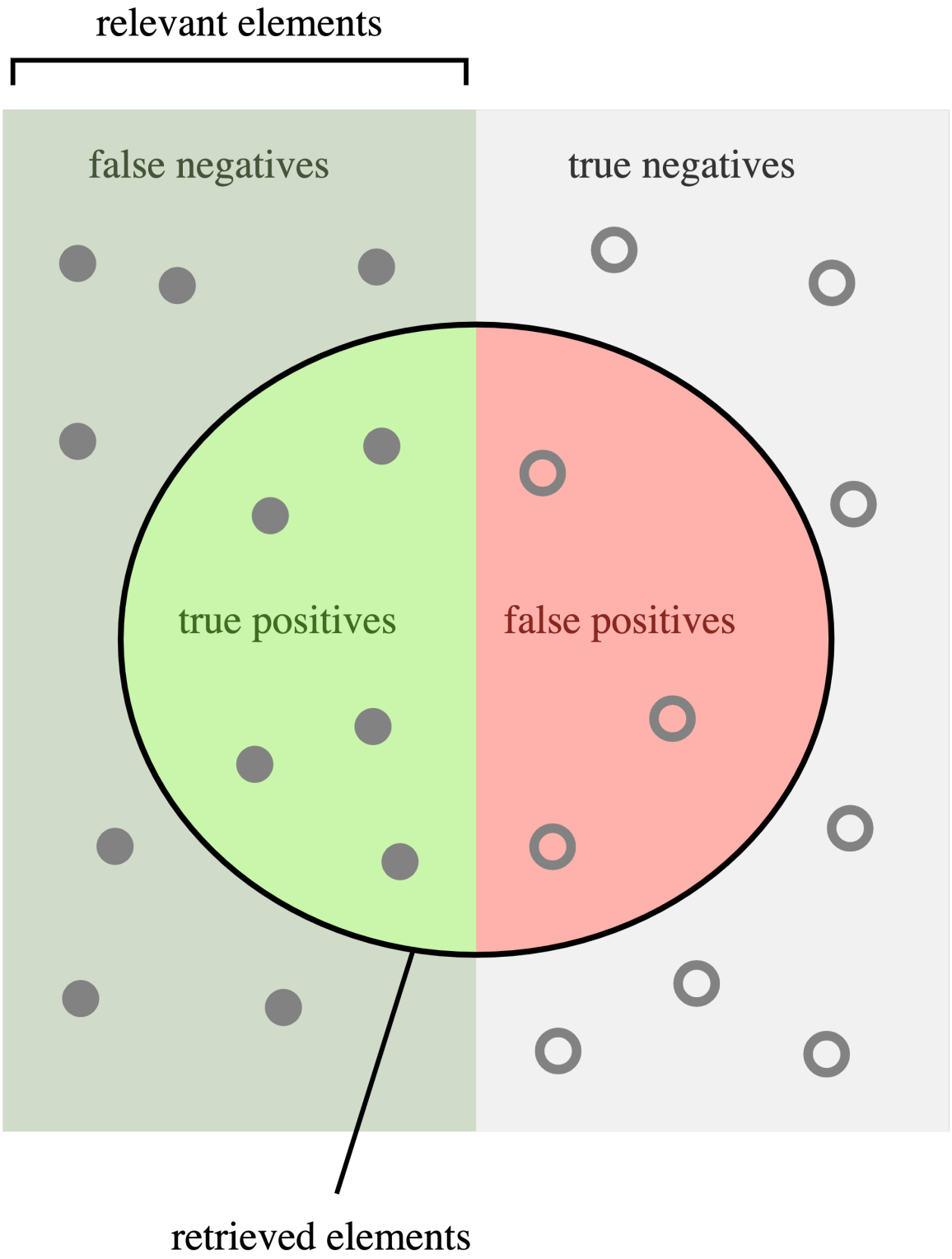

Figure 22: Visual representation of true positives, false positives, true negatives, and false negatives. Modified from [6]

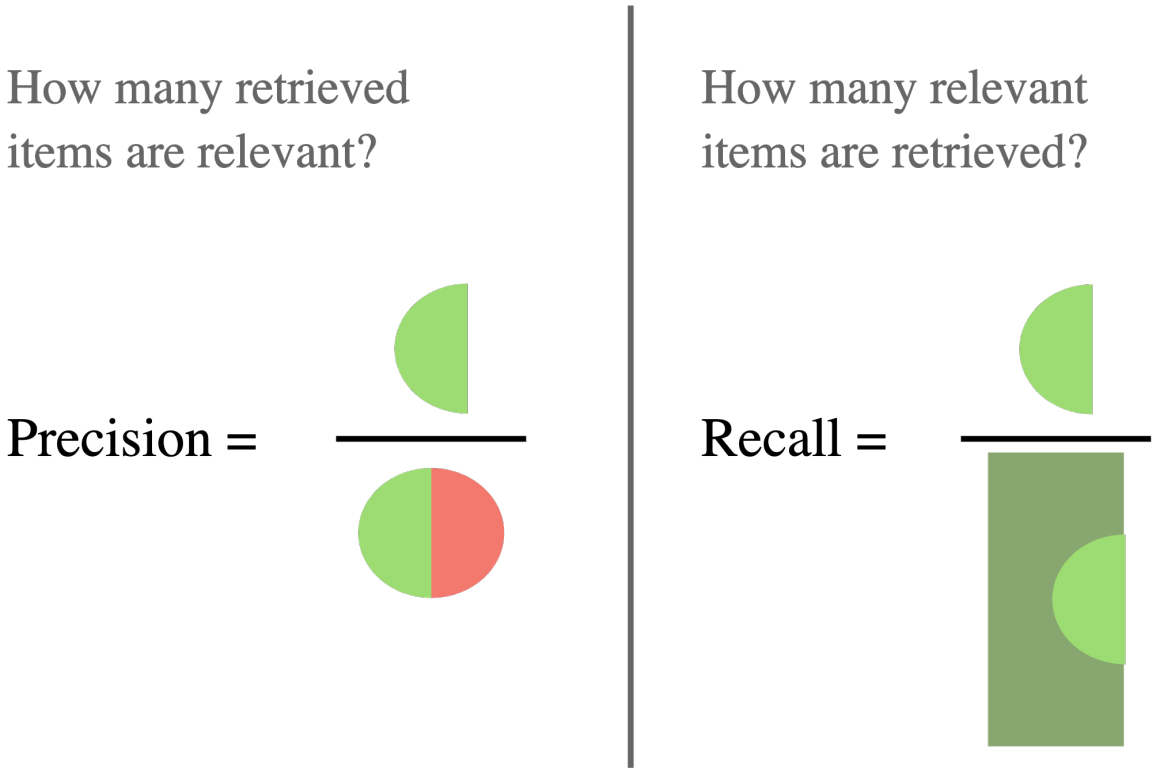

Figure 23: Visual representation of precision and recall. Modified from [6]

is the number of correctly predicted positives divided by the number of ground truths. Precision, as calculated in

$$Precision = \frac{TP}{TP + FP}, \tag{6}$$

is the number of correctly predicted positives divided by the total number of predictions. Recall can be seen as a measure of the quantity of TPs, while precision can be seen as a measure of the TPs. Higher recall means the predictions contain most of the relevant results, while higher precision means the predictions contain more relevant results than irrelevant ones.

Intersection over union (IoU), as calculated in

$$IoU = \frac{target \cap prediction}{target \cup prediction}, \tag{7}$$

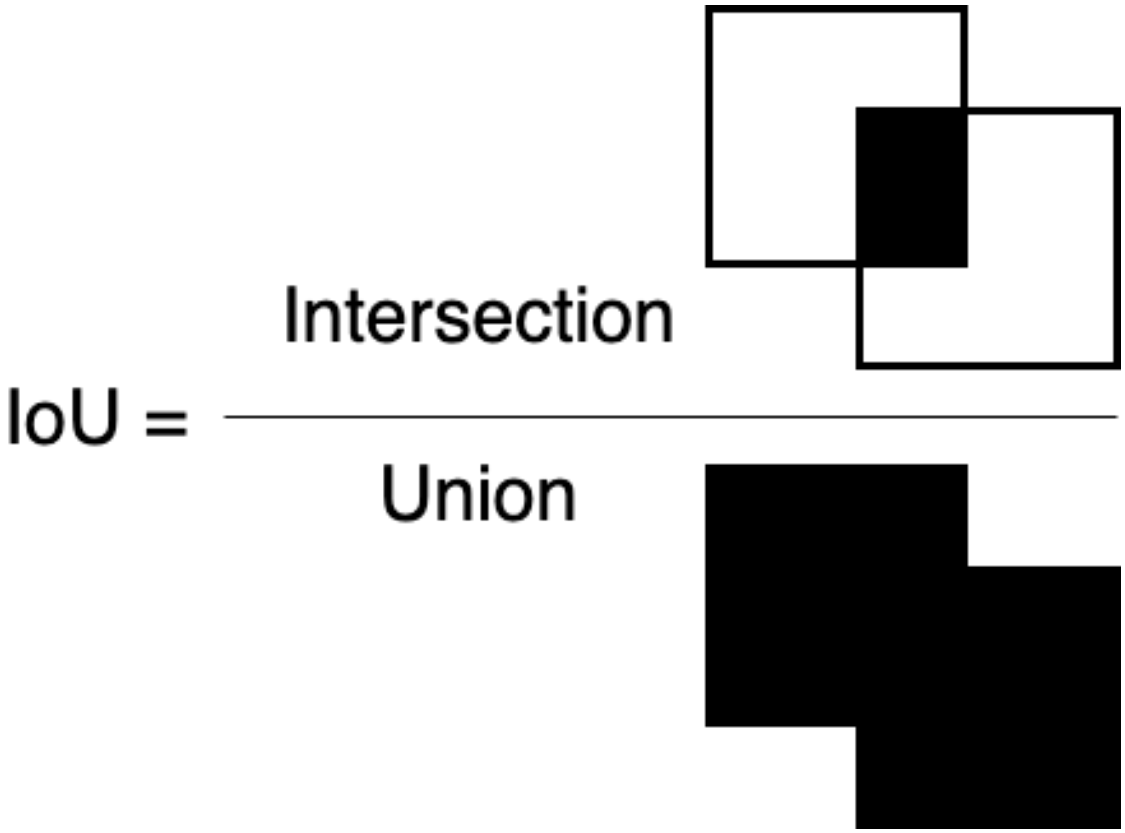

Figure 24: Visual representation of Intersection over Union (IoU)

is a metric calculated for each prediction and its associated ground truth and is used to determine whether a prediction is correct or not, i.e. its truthfulness. IoU, equation (7), is the intersection between the prediction and the ground truth (target) divided by their union (see Figure 24). For a prediction to be considered a TP, its IoU with its associated ground truth has to be equal to or above some threshold; otherwise, it is considered a FP (see Figure 25).

Another factor when determining the truthfulness of a prediction is the confidence level, a value from 0.0 to 1.0, the model reports for every prediction. By varying what we consider to be the cutoff confidence level, we can change whether a prediction is deemed a TP or FP.

The average precision (AP), the metric used to evaluate the models in this work,

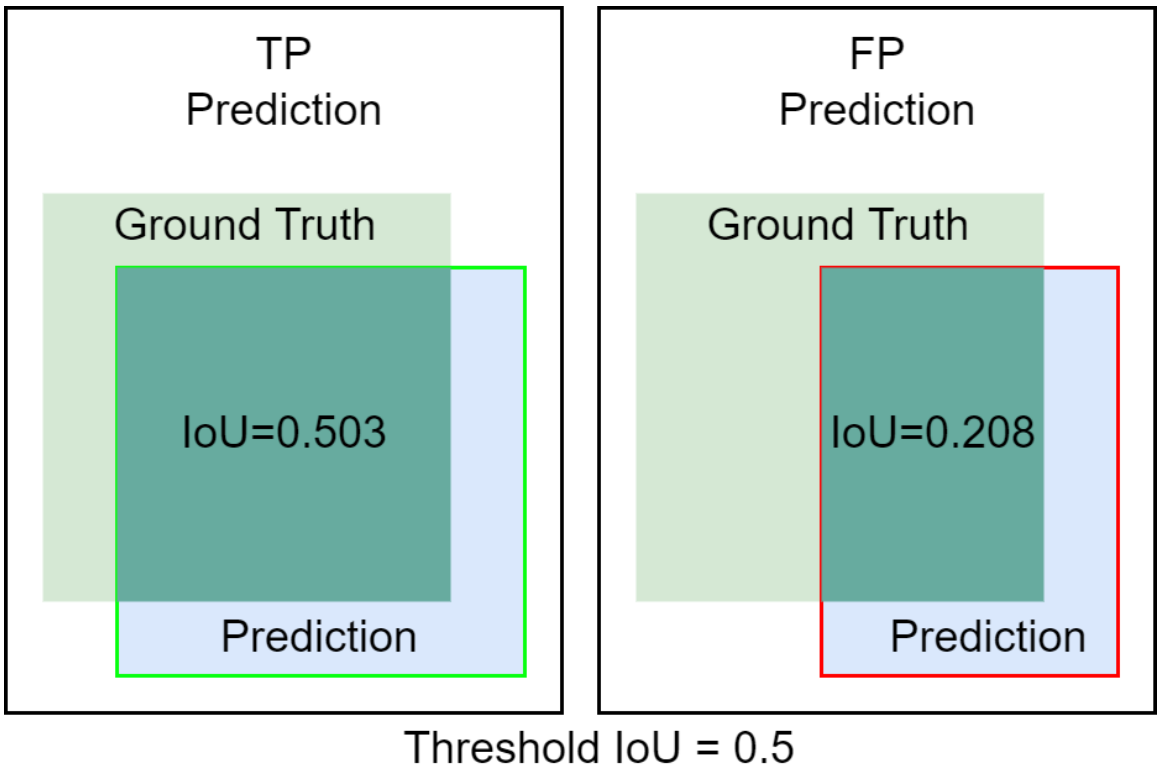

Figure 25: Visual representation of Intersection over Union A visual example of threshold IoU and how it factors into the truthfulness of a prediction

is an attempt by researchers to summarize the overall performance of a model and gives a single value that can be used to compare the performance of various models.

To calculate the AP metric of a set of predictions over a dataset, we must create a precision-recall (PR) curve. For a single class of predictions, we calculate the recall and precision values given a threshold IoU score and a threshold confidence level; this produces a single PR pair. To create the PR curve, PR pairs are calculated across the range of possible confidence levels, i.e., 0.0 to 1.0, which determine the value pair's rank. The PR curve is created by plotting these PR value pairs by descending rank, with precision on the y-axis and recall on the x-axis, and connecting line segments between subsequent points to create a piecewise function, $p(r)$, (see Figure 26). The AP of this PR curve, equation (8), is then calculated by averaging the precision values

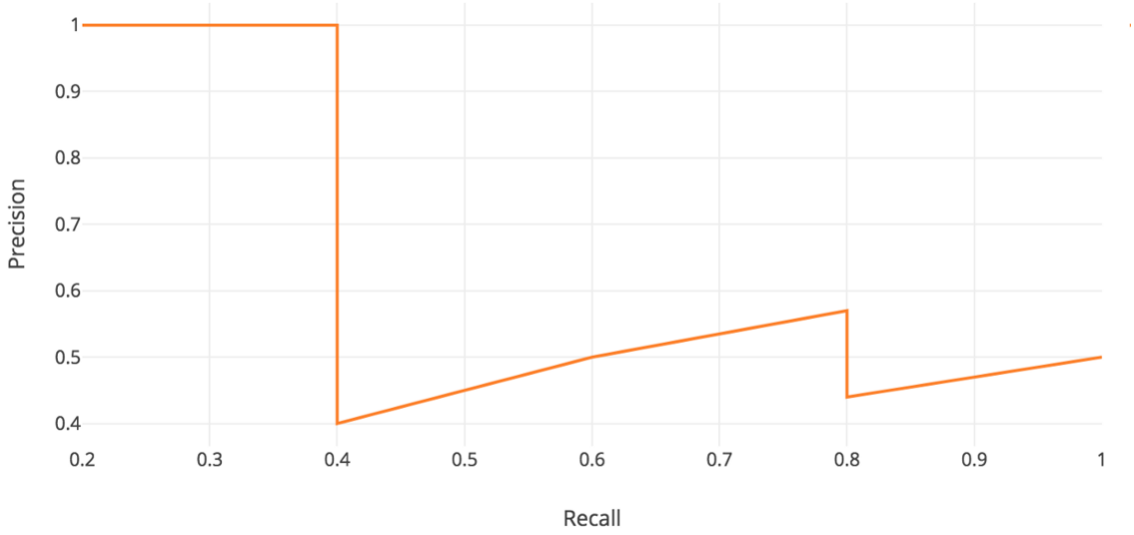

Figure 26: An example PR curve which is a piecewise function connecting precision-recall pairs in order of rank

at 101 equally spaced recall values along the piecewise curve, as calculated in

$$AP = \frac{1}{101} \sum_{r \in \{0.00, 0.01, \ldots, 0.99, 1.00\}} p(r). \tag{8}$$

For multiclass datasets, an AP is calculated for each class, and the mean is found, called the mean average precision (mAP). Following the COCO researcher's example, most research assumes AP to mean mAP. AP and mAP are equivalent for our research as our evaluation dataset has a single class of objects.

Each AP or mAP is calculated given some threshold IoU which may introduce some bias in the evaluation metric related to the correctness of a prediction. We take the average AP across a range of IoU threshold values to address this bias. Following the COCO [8] evaluation metric, we average the AP at IoU thresholds from 0.5 to 0.95 at an interval of 0.05, represented in this research's results as AP[0.5:0.05:0.95].

4.2.3 Convolutional Neural Networks

CNNs are a class of deep learning neural networks that excel at specific problems in computer vision, and in this case [10, 11, 12, 13, 14]. Convolution, the critical process of the CNN, is particularly equivariant to the translation of features within an input datum, meaning a translation of a feature in the input of a CNN results in the same translation at the CNN's output. The CNN's ability to recognize features independent of location makes it particularly effective at processing visual data. For example, a CNN cat or dog classifier can identify whether a cat or dog is within an image independent of the animal's location within that image [15].

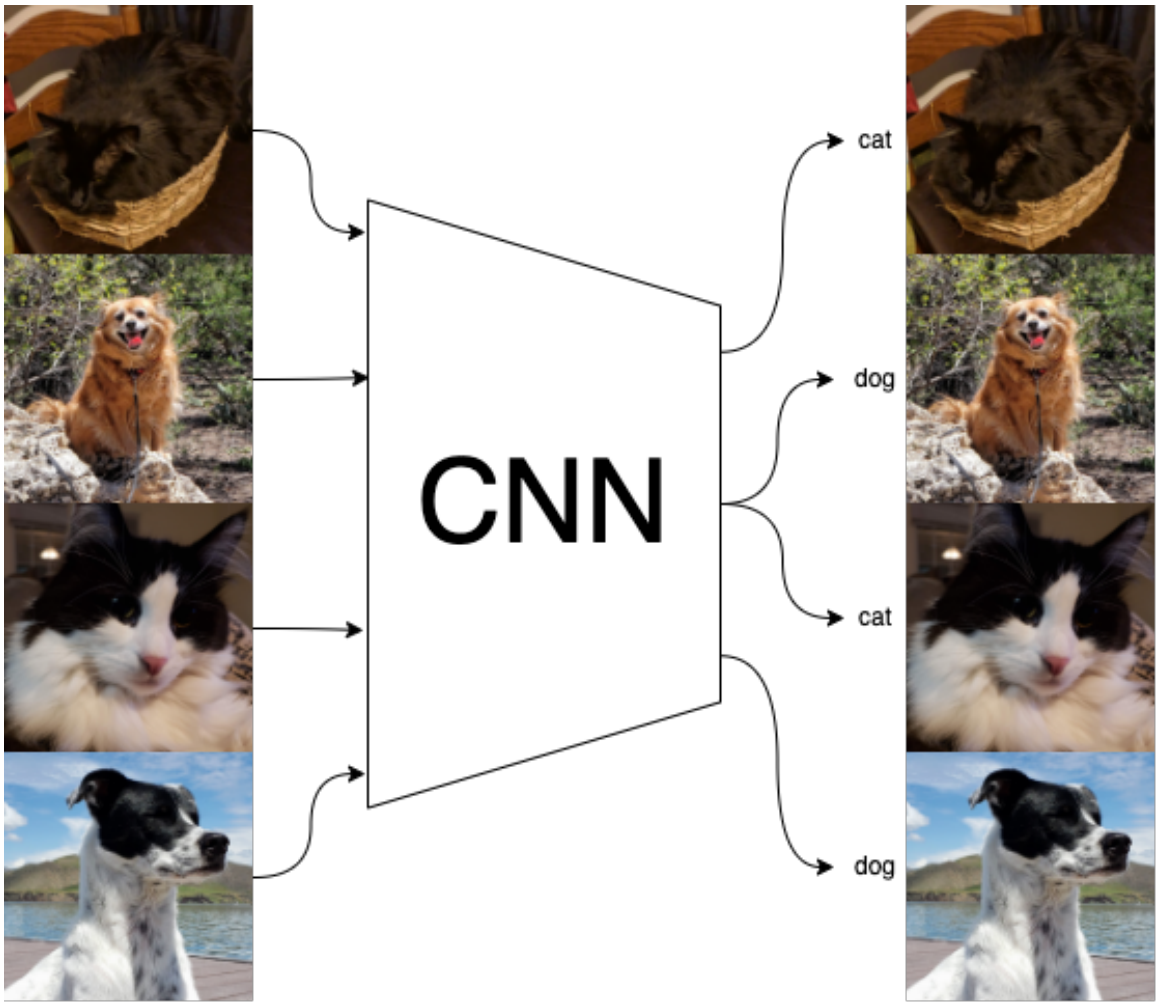

Figure 27: High level representation of a CNN cat or dog classifier

Within the context of CNNs, convolution is the dot product between two matrices. The first matrix is a set of learnable parameters called a kernel. The other is a restricted portion of the input, called the receptive field. The repeated application of convolution with a single kernel over an entire input creates an activation map that represents the kernel's response at every position within the input [15].

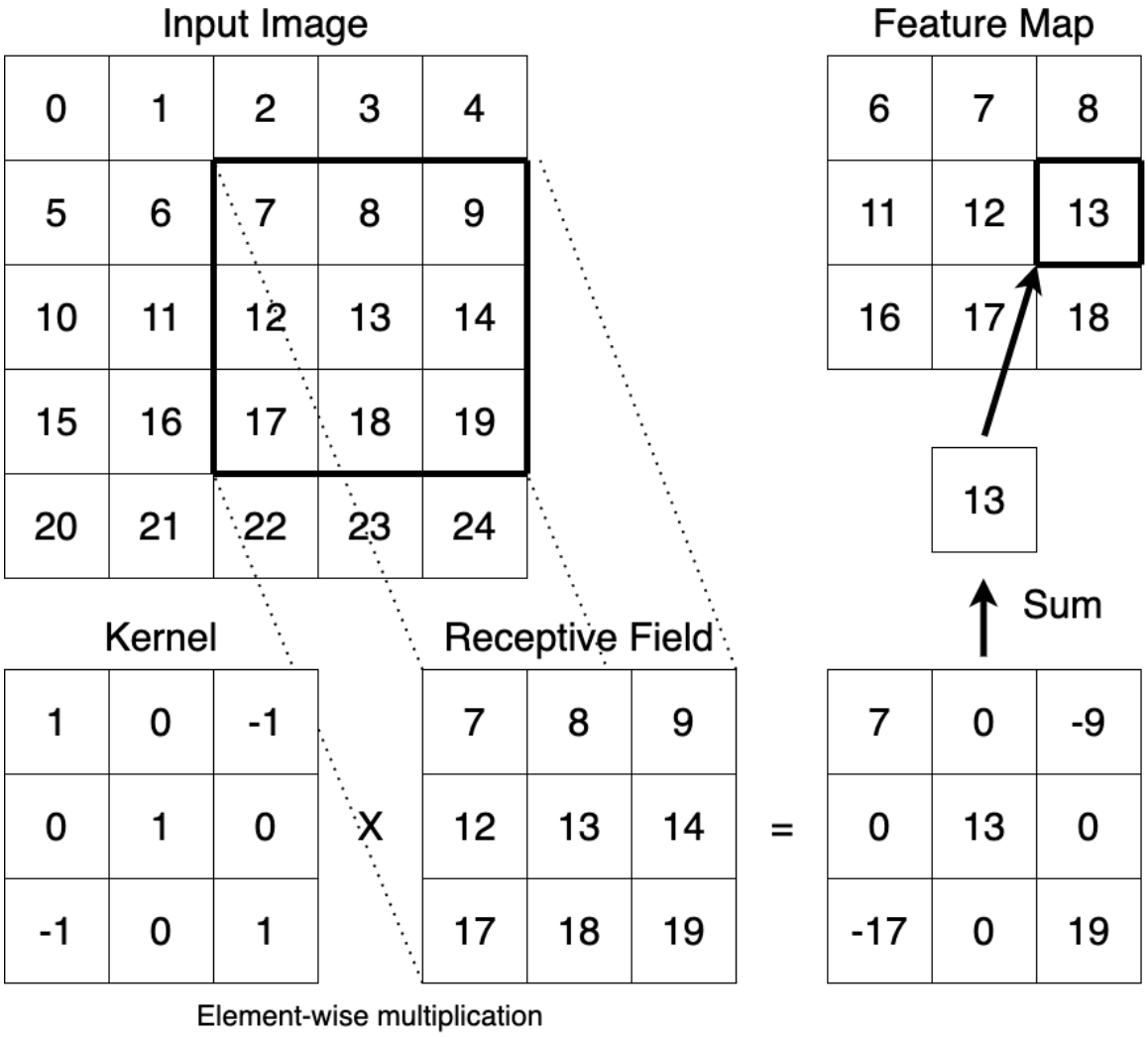

Figure 28: An illustration of a 3x3 convolution of a 5x5 input image

This repeated use of a kernel for multiple convolutions across the image takes advantage of parameter sharing, leading to the CNN's property of equivariance of translation.

In general, a CNN is composed of layers that perform one of three operations: convolution, non-linear function, or pooling. After a convolutional layer, the resulting activation map is run through a non-linear function, such as rectified linear unit (ReLU) (used earliest in [16, 17]). The outputs of the non-linear function layer are then further processed by a pooling layer. A pooling layer replaces the value at every location with a statistic of the surrounding values. Pooling helps make the representation invariant to small translations in the data. A network invariant to small translations in its input means that instead of recognizing a feature when it is in the exact right location, the network can detect a feature in an approximate location.

4.3 CNN Architecture for Instance Segmentation

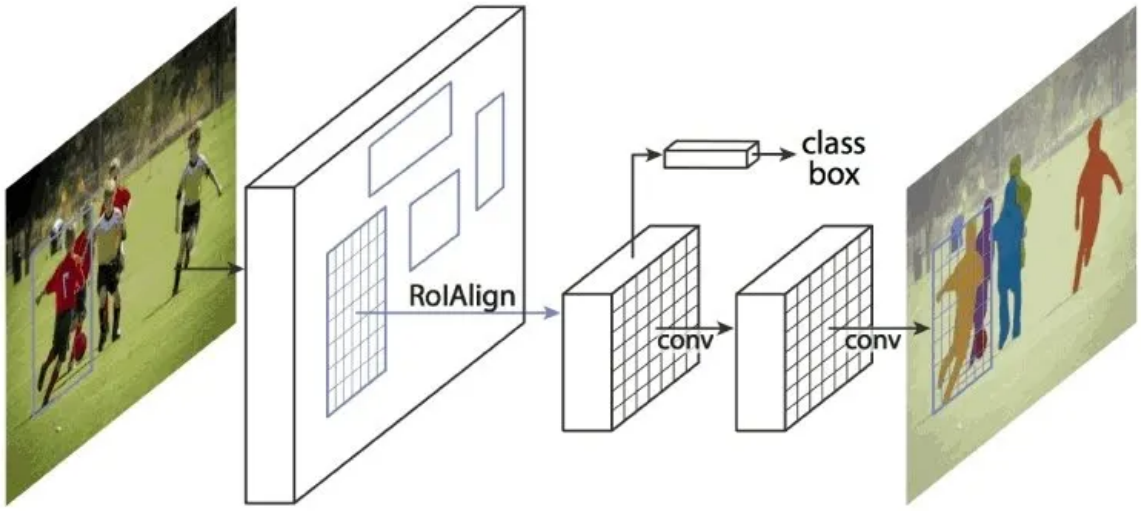

Figure 29: A high-level system view of Mask R-CNN instance segmentation framework [2]

Created by He et al. [2] at Facebook AI Research (FAIR) in 2018, Mask R-CNN is a state-of-the-art architecture for instance segmentation. With minimal additions

onto Faster R-CNN [24], Mask R-CNN surpasses all other previous single-model competitors on the COCO instance segmentation task [8] while maintaining most of the performance increases achieved by Faster R-CNN. FAIR has open-sourced a Mask R-CNN implementation in their Detectron2 [23] object detection platform built with paszkePyTorch2021 [83]. Mask R-CNN comprises several sub-networks: a backbone, a detector, and several region of interest (ROI) heads. As Mask R-CNN is a meta-architecture, an architecture that can be instantiated with different building blocks [25], the exact composition of Mask R-CNN depends on the designer's exact implementation. The implementation of the Mask R-CNN architecture used for this research is the GeneralizedRCNN architecture from Facebook AI Research's Detectron2 [23] object detection platform.

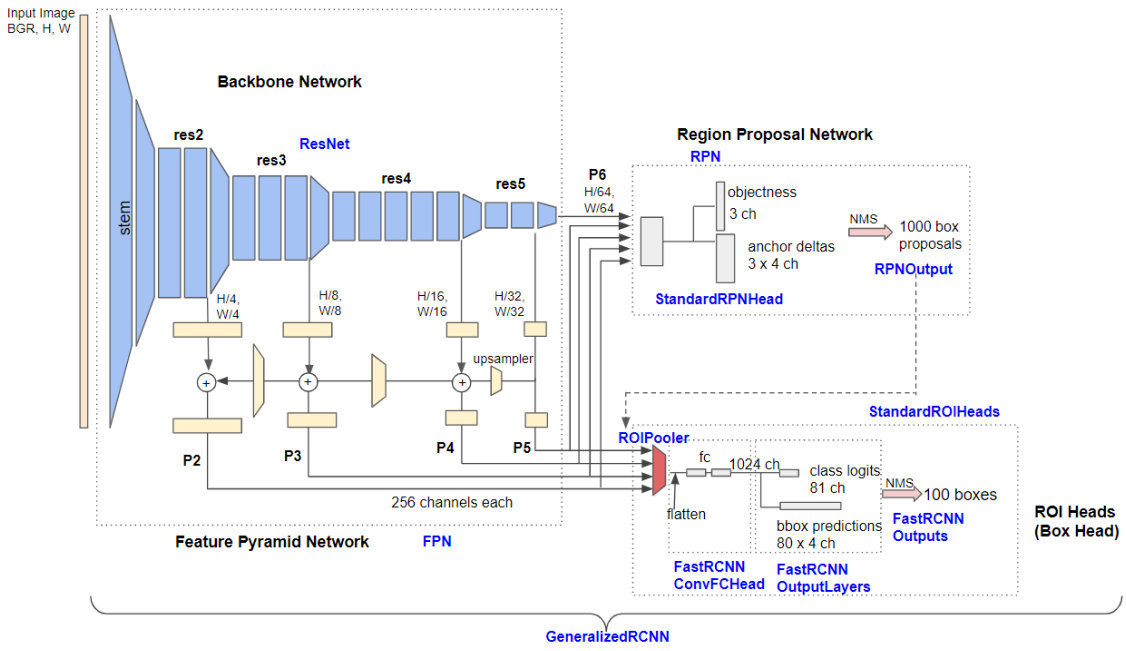

Figure 30: System diagram of Detectron2's GeneralizedRCNN architecture [3]. Note: the system diagram does not include the ROI mask head.

The GeneralizedRCNN architecture (see Figure 30) is configured with a ResNet-50 [26] based Feature Pyramidal Network (FPN) [27] as the network backbone. The ResNet-50 backbone sub-network comprises five stages, where each subsequent stage

extracts further feature information from the previous stages [26]. This backbone acts as a feature extractor that feeds feature and spatial information as feature maps to the Region Proposal Network (RPN) detector and ROI heads. The RPN applies a sliding window over the FPN feature maps and predicts the 'objectness', probability of containing a target object, and the boundary box at each window location [27]. The RPN produces several proposed object locations indicated as boxes and feeds them for consideration to the individual ROI heads [27]. The ROI heads, a box head, and a mask head, in the case of GeneralizedRCNN, take these proposed object locations and the feature maps produced from the FPN and produce the final outputs from the network. The ROI box head produces object bounding box and their associated object classification predictions. The ROI mask head produces an instance segmentation mask associated with each predicted object [24].

4.3.1 Mask R-CNN's Loss Function

The loss function for Mask R-CNN is the sum of the following individual losses:

- Classification loss in the RPN (see [28])
- Localisation loss in the RPN (see [28])
- Classification loss in the ROI Box head (see [24])
- Localisation loss in the ROI Box head (see [24])
- Mask loss in the ROI Mask head (see [2])

4.3.2 Transfer Learning with Mask R-CNN

Training the Mask R-CNN network from scratch would require significant computational resources and an extensive training dataset. Thankfully, a technique termed transfer learning (TL) can help reduce the resources necessary to train the model on

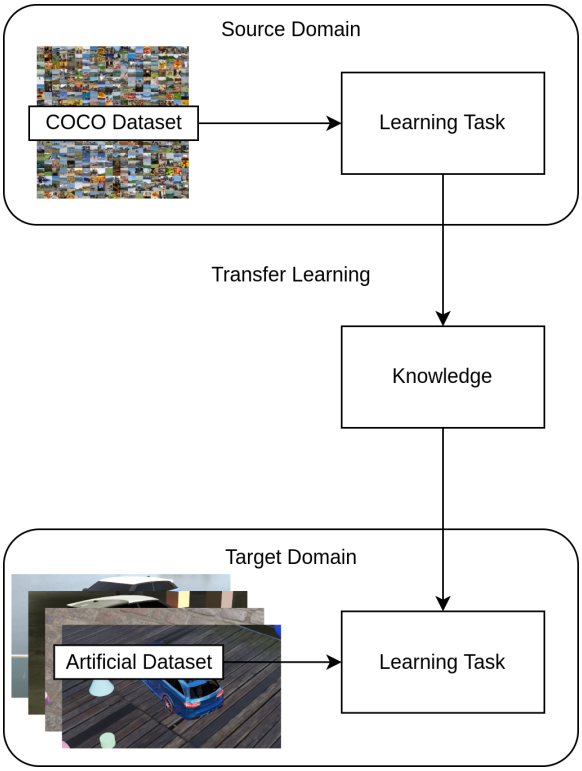

Figure 31: Visual description of transfer learning for fine-tuning a CNN model. Modified from [4]

the artificially generated dataset. TL is the reuse of a pre-trained model for a new task requiring less computing resources and less training data than training the model from scratch. First, a model is trained on a large object detection dataset, the COCO dataset in the case of this research. This initial training is performed in what is called the source domain. Second, a portion of the same model's trained parameters are frozen (prevented from changing) and trained again on the dataset of interest. This second training is performed in what is called the target domain. See Figure 31 for a visual description. The freezing of a portion of the model's pre-trained parameters between the two training runs transfers some knowledge from the source domain to the target domain, hence the term transfer learning (TL). Training fewer parameters in the target domain compared to the source domain required fewer computing resources.

TL has been leveraged with Mask R-CNN by many researchers in recent years for various applications. Rehman et al. [4] used Mask R-CNN with a ResNet-50 and ResNet-101 [26] based FPN backbones to create an apple leaf disease detector by pre-training the ResNets on the ImageNet dataset [29] and fine-tuning on the PlantVillage dataset [30], an open access dataset of images on plant health. Ullo et al. [31] used Mask R-CNN with a ResNet-50 and ResNet-101 based backbones to create an aerial image landslide detector by pre-training on the COCO dataset [8] and fine-tuning on a small custom dataset of landslide images. Similarly, Mask R-CNN and TL have been used for solder joint detection [32], lesion detection [33], lung segmentation [34], among many other applications.

4.4 Related Work

This work build upon previous work in automated visual inspection, synthetic data for computer vision, and domain randomization. The related work will be outlined

in the subsections below.

4.4.1 Automated Visual Inspection

Many researchers are exploring automated vision systems for aircraft inspection purposes. Aust et al. [84] used image processing techniques to detect defects on the edges of aircraft engine blades. As for the exterior inspection of aircraft, Ramalingam et al. [73], Miranda et al. [47], Almadhoun et al. [85], Rice et al. [44], Bouarfa et al. [50], Dogru et al. [86], Tzitzilonis et al. [87], and Jovancevic et al. [54] all present automated visual defect image collection and detection systems. Bouafa et al. [50] offer a defect detection method most similar to this work. They used a network based on the Mask R-CNN architecture pre-trained on the COCO dataset and fine-tuned on a small, but private dataset of aircraft dents.

4.4.2 Synthetic Data for Computer Vision

Synthetic datasets provide detailed ground-truth annotation and are a less-expensive and scalable alternative to manually annotating images. Barth et al. [64] developed a data synthesis method for training a DeepLab [88] VGG-16 [89] based model for semantic segmentation of agricultural scenes. Georgakis et al. [90] created a synthetic images dataset to train a CNN for object detection by placing 3D modeled household objects into real-world images of everyday household scenes. Similar to Georgakis et al., Gupta et al. [77] combined computer-generated text with real-world images to generate an artificial dataset to train a Fully-Convolutional Regression Network (FCRN) for robust real-world text detection and end-to-end spotting. Handa et al. [39] synthesized a dataset called SceneNet for training a CNN for household scene semantic segmentation using 3D computer-aided design (CAD) models to generate completely synthetic household scenes.

4.4.3 Domain Randomization

Tobin et al. [81] proposed domain randomization (DR) as an alternative to high-fidelity synthetic imaging. They introduced DR to bridge the reality gap experienced when using a synthetic dataset on real-world problems by creating synthetic data with enough variations to allow the network to treat the differences between artificial data and real-world data as another variant to be ignored. Khirodkar et al. [82] and Tremblay et al. [78] used DR to generate synthetic datasets for training a CNN for car detection and pose estimation.

4.5 System Implementation

4.5.1 Data Generation Pipeline

The goal of this research is to detect and locate damage on cars in real-world images with a CNN. To compensate for the lack of a sufficiently sized real-world dataset, we use 3D CAD models and a procedural damage shader to generate annotated synthetic data. We leveraged Blender [67], an open-source 3D graphics creation suite, and its Python [91] scripting interface [92] to automate the data generation process. The main challenge for this research is generating synthetic data for training a model that generalizes well to real-world data. As mentioned earlier, the application to car damage is a substitute for aircraft damage, since we were not able to receive real imagery. This section describes the synthetic data generation process and specifics about the domain randomization strategies used during data generation.

4.5.1.1 Data Generation

Although Blender's Python interface allows one to automate the majority of the data generation, some initial manual setup is required due to the complexity of

Blender and virtual 3D environments in general. Setup requires the manual initialization of two Blender objects (see Figure 32), one that is rendered while generating the image, labeled the 'Data Object,' and the other is rendered while generating the segmentation mask, labeled the 'Annotation Object.' These are depicted in Figure 33.

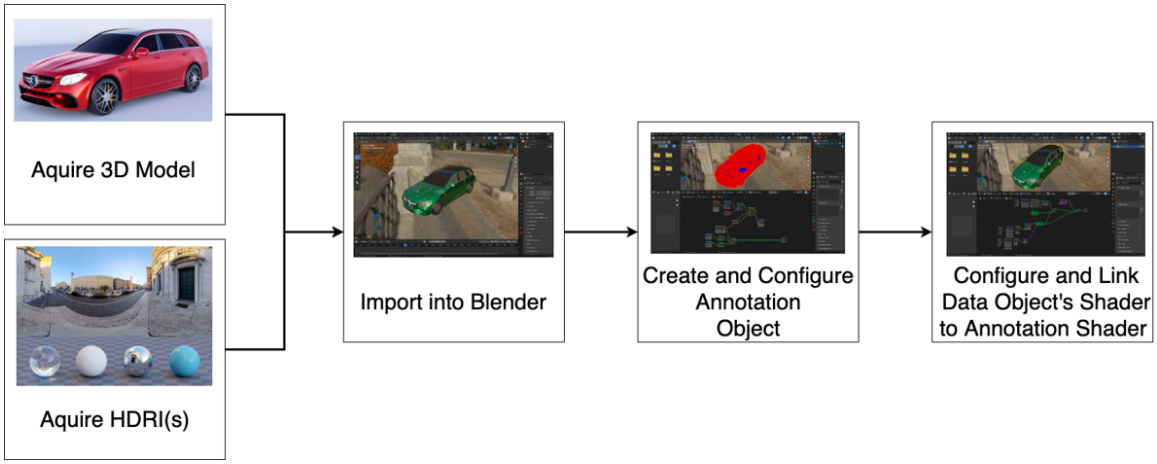

Figure 32: Blender models and scene initialization steps required before data generation

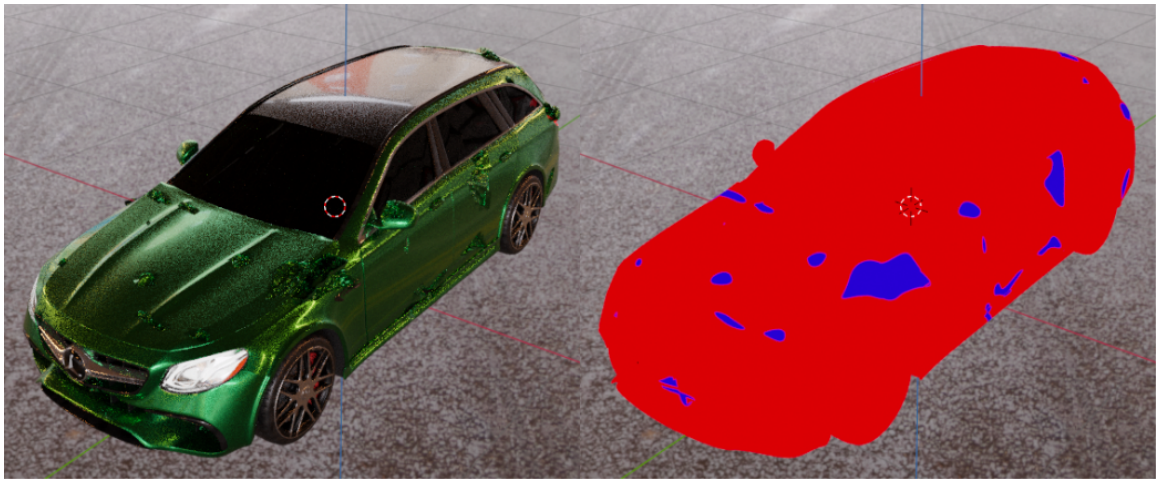

Figure 33: Example of a 'Data Object' (left) and the 'Annotation Object' (right) within Blender

The 'Data Object' is a 3D CAD model of a car with a procedural texture that mimics the appearance of damage on small sections of the model's geometry. The author created this texture by hand with real-world images of car damage for refer-

ence. Although the texture created may not reproduce all types of car damage, the author believes it is sufficient for this research. The 'Annotation Object' is created by duplicating the geometry of the 'Data Object' and applying a multi-color emission texture that labels the areas where the damage does and does not occur on the 'Data Object.' The procedural data of these two objects' textures are linked via Blender drivers so that they always match each other perfectly. After the creation of these objects, the initial setup is complete.

The data generation script is configurable through a custom-made Blender add-on and user interface, made specifically for this research. Figure 34 shows the add-on interface and Table 2 gives a description of each parameter.

4.5.1.2 Domain Randomization

Several methods of domain randomization are used in an attempt to add enough variation to avoid model overfitting to the synthetic data. These methods fall into two categories: content and style variation. Examples of these randomizations are found in Figure 35.

- **Content Variation:** For every new synthetic datum generated, the script randomizes several forms of content in the scene, including scene background, distractors, and 3D CAD model. The scene's background is randomly selected from various 360-degree spherical panoramic images, called HDRIs. A random assortment of 3D shapes, called distractors, is added to the scene to add uninteresting information. Finally, the resulting dataset uses several freely available 3D CAD car models to add variation to the object of interest.

- **Style Variation:** The stylistic variations used include randomizing the color of each distractor and car. In addition, the position of the scene camera and light source are changed to add variation in the scene's lighting conditions.

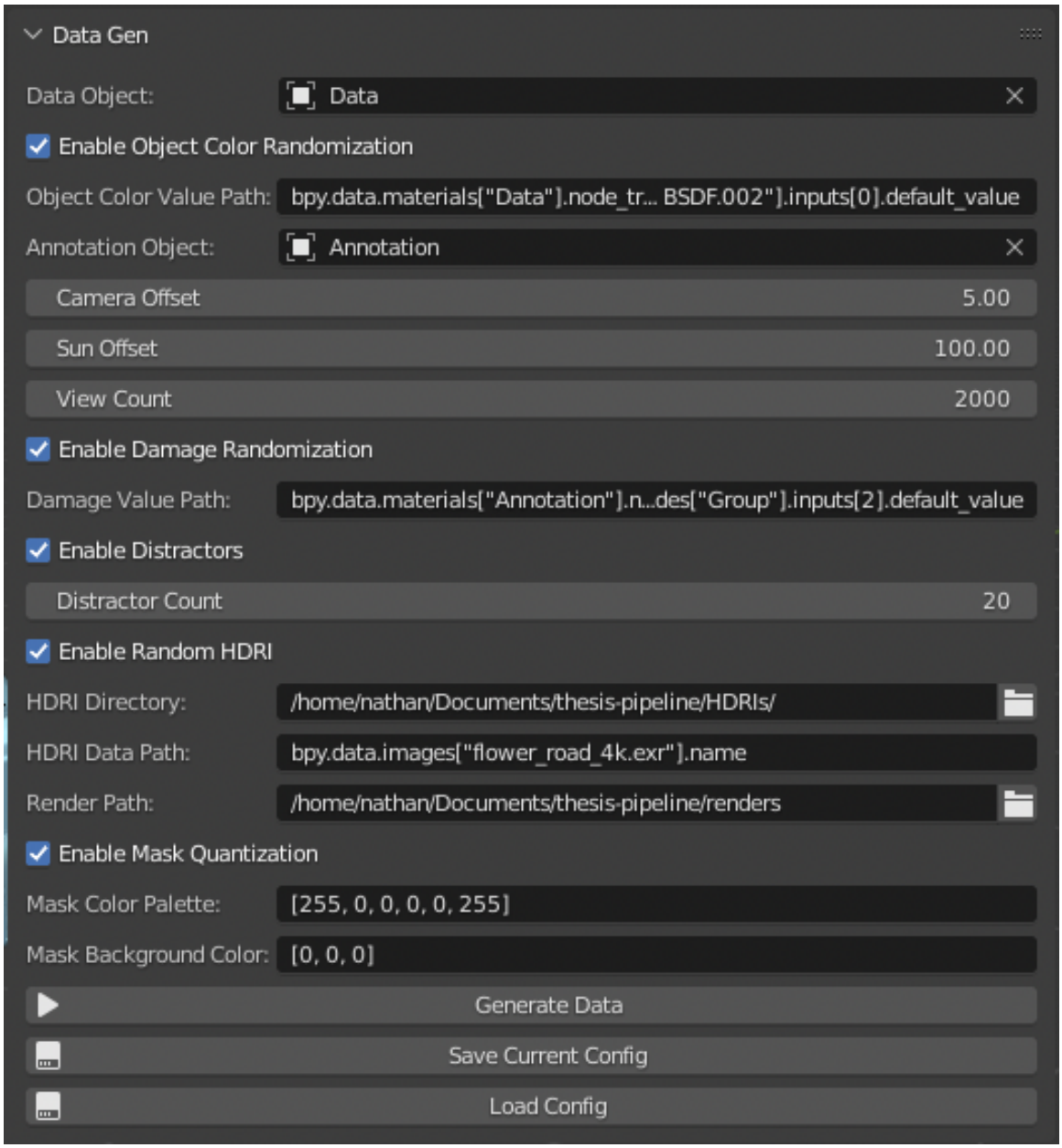

Figure 34: Data generation configuration graphical user interface inside Blender

Table 2: Data Generation Blender Add-on Parameters

Name	**Description**
Data Object	Select the Blender object to render during the data generation step
Enable Object Color Randomization	If checked, it enables randomization of the Data Object's color between views
Object Color Value Path	Blender data path to Data Object shader color
Annotation Object	Select the Blender object to render during the annotation generation step
Camera Offset	The distance from the Data Object to the rendering camera
Sun Offset	The distance from the Data Object to the rendering scene light source
View Count	The number of different views to render into data
Enable Damage Randomization	If checked, it enables randomization of the damage generation between views
Damage Value Path	Blender data path to the value controlling the object's damage randomization
Enable Distractors	If checked, it enables the addition of various random distractor objects into the scene
Distractors Count	The number of distractors to add to each scene during view randomization
Enable Random HDRI	If checked, it enables the randomization of the scene's 3D background between views
HDRI Directory	File path to a directory containing HDRI's on the user's computer
Render Path	File path to a directory for which the renders will be saved to
Enable Mask Quantization	If checked, it enables the quantization of each multi-class annotation mask into several single-class masks
Mask Color Palatte	An array of integer values between 0 and 255 (inclusive) where every three integers represent a red-green-blue color foreground color value for each of the quantized masks
Mask Background Color	An array of three integer values between 0 and 255 (inclusive) that represents a red-green-blue color of the background of all masks generated by the add-on

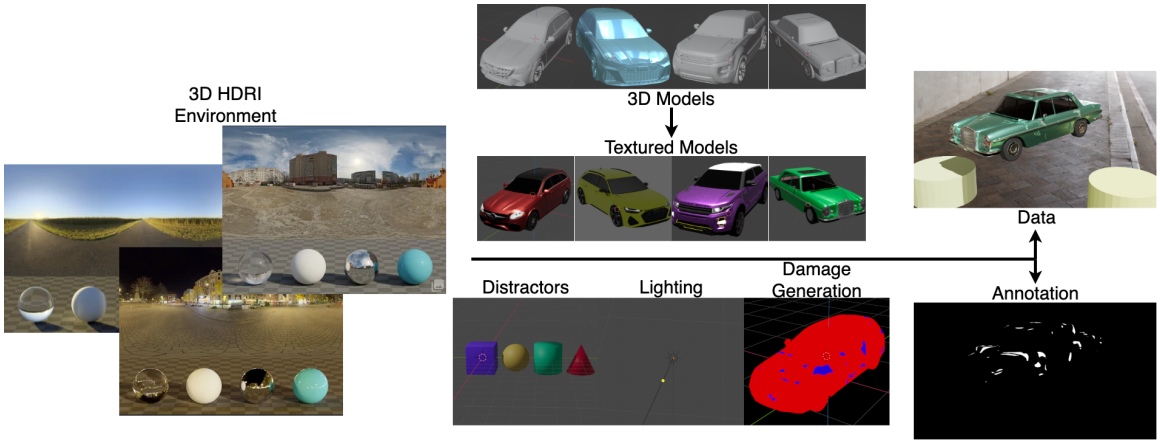

Figure 35: Synthetic data generation with domain randomization: randomization of HDRI environment, 3D model, model texture, distractors, lighting, damage generation

4.6 Experiment

In this research we propose that given a Mask R-CNN (Detectron2 GeneralizedRCNN) model pre-trained on the COCO dataset, there is an optimal amount of unfrozen backbone stages, from Stage 1 through Stage 5, we can then train on our artificial dataset for fine-tuning and testing on the real-world damage dataset. We hypothesize that the minimal unfreezing of the backbone stages before training on our artificially generated dataset will provide the best segmentation performance on the real-world car damage dataset. Essentially, the weights from pre-training on the COCO dataset contain the vast majority of what our model needs to know to perform well.

4.6.1 Datasets

We benchmark our methodology on a single small dataset composed of real-world images of car damage and their instance masks. This real-world dataset was sourced from Kaggle [93] as "Coco Car Damage Detection Dataset". This real-world car damage dataset is not created by the COCO [8] dataset team, but formatted into

the same format as the COCO dataset. This real-world car dataset contains only 70 labeled images. Each image in the real-world dataset contains a car with one or more areas of damage recorded at 1024 x 1024 pixels and the images vary in car model, background scene, camera angle and lighting. Several examples of the real-world dataset can be seen in Figure 36.

The "Coco Car Damage Detection Dataset" will only be used for evaluating our final models, as it is far too small to train Mask R-CNN with.

The artificial dataset comprises 8,000 images, each with a ground truth label, randomly split 80:20 into training and validation sets; 6400 training and 1,600 validation images. Each image is rendered at 1920 x 1080 pixels to make up the artificial dataset has randomized lighting, 3D car model, model coloring, model damage texturing, camera location, distractors (location and coloring), and background scene.

4.6.2 Models

For our experiment, we compare the performance of five Detectron2 GeneralizedRCNN models, all pre-trained on COCO, fine-tuned on the artificially generated car damage dataset, and evaluated on the real-world car damage dataset:

- **Stage 1**: FPN+ResNet-50 Backbone with stage 1 frozen before fine-tuning
- **Stage 2**: FPN+ResNet-50 Backbone with stages 1 and 2 frozen before fine-tuning
- **Stage 3**: FPN+ResNet-50 Backbone with stages 1 through 3 frozen before fine-tuning
- **Stage 4**: FPN+ResNet-50 Backbone with stages 1 through 4 frozen before fine-tuning

Figure 36: Several examples from the “Coco Car Damage Detection Dataset”

Figure 37: Several examples from the artificially generated car damage dataset

- **Stage 5**: FPN+ResNet-50 Backbone with stage 1 through 5 frozen before fine-tuning

4.6.3 Training

When training GeneralizedRCNN with Detectron2, the training is configured to run for a number of iterations instead of specifying the number of epochs. An iteration is a training pass over a batch of training images, while an epoch is a training pass over all training images. The relationship between epochs and iterations is

$$n_{epochs} = \frac{n_{iterations} * n_{batch}}{n_{training}}. \tag{9}$$

According to the Detectron2 Model Zoo code repository [23], the models with COCO pre-trained weights were trained on approximately 37 epochs of 118 thousand images from the COCO 2017 dataset [8], which equates to approximately 270,000 iterations with a batch size of 16 images. Thankfully, the weights after this training

on the COCO dataset are available on the Detectron2 Model Zoo [23] and were used for the pre-training in this research.

Fine-tuning the models on the artificially generated dataset was performed over 10,000 iterations at two images per batch or approximately 4.7 epochs.

4.6.4 Training Data Augmentations

During training, the following data augmentation operations, available from the Detectron2 [23] code base, were used:

- **ResizeShortestEdge**: Resize the image by shortest edge while keeping the aspect ratio unchanged
- **RandomFlip**: Flip the image horizontally or vertically with the given probability
- **RandomSaturation**: Randomly transforms saturation of an RGB image
- **RandomContrast**: Randomly transforms image contrast
- **RandomBrightness**: Randomly transforms image brightness
- **RandomRotation**: Rotate the image given number of degrees counter clockwise around the given center

4.6.5 Results

4.6.5.1 Fine-tuning on Artificial Data

Figure 38 shows the total training loss on the 6400 image artificial training dataset while fine-tuning each of the five models. Zooming into the final 2000 iterations (See Figure 39) shows a rough order of the models in terms of minimum training loss: Stage 1 and 2 models have the lowest losses, Stage 3 has middle performing loss,

and Stage 4 and 5 have the highest losses. This order is also reflected in each of the models' segmentation performance shown in Figure 40 and Figure 41. This order implies that freezing the earlier Stages of the ResNet-50 backbone before fine-tuning on the artificial car damage dataset results in the best performance on the artificially generated car damage dataset.

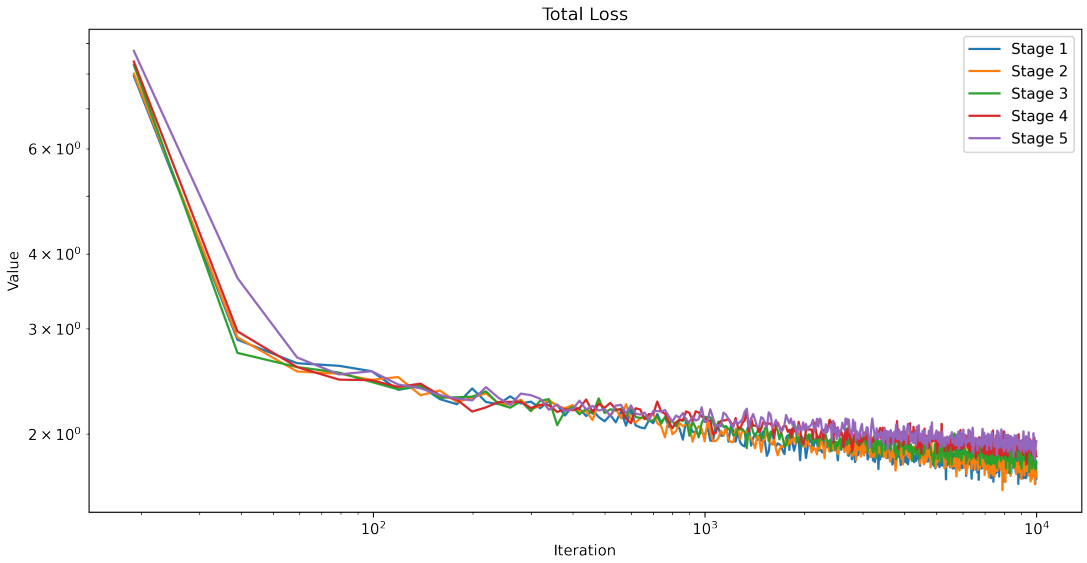

Figure 38: Mask R-CNN model's total loss throughout training on the artificial car damage dataset

4.6.5.2 Segmentation Performance on Real-World Data

Figure 42 each model's segmentation performance on the real-world car damage dataset at a period of 500 iterations. Interestingly, Stage 2 and Stage 3, the highest performing models, show peek performance at iteration 1500 while their performance drastically decrease in later iterations. Compared to the continued growth of each model's performance on the artificial validation data (Figure 40) the models seem to be overfitting to the artificial data and not fitting will to the real-world car damage data.

Comparing the maximum AP[0.5:0.05:0.95] segmentation performance achieved

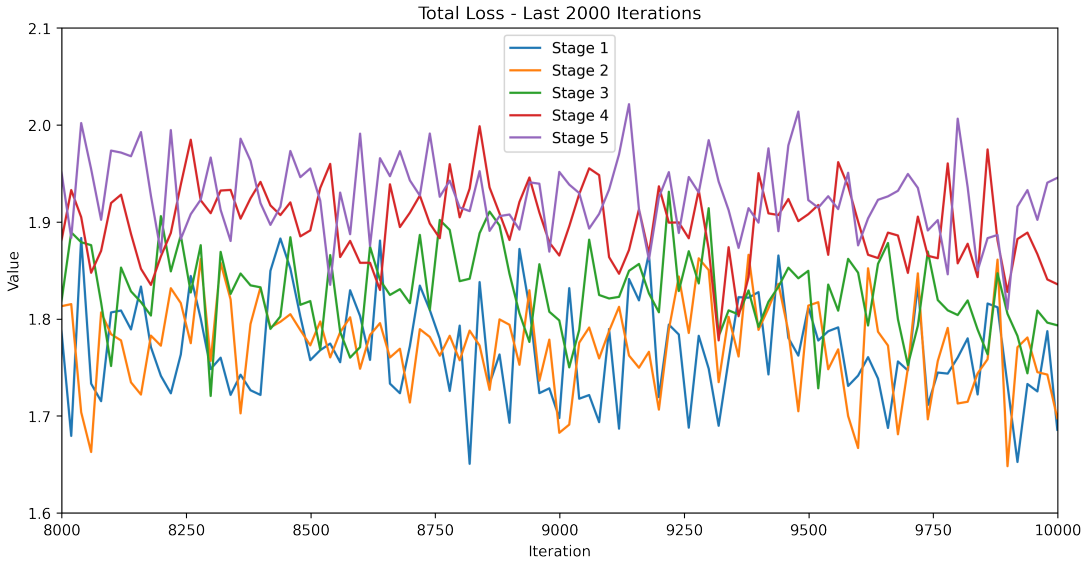

Figure 39: Mask R-CNN model's total loss during the last 2000 iterations training on the artificial car damage dataset

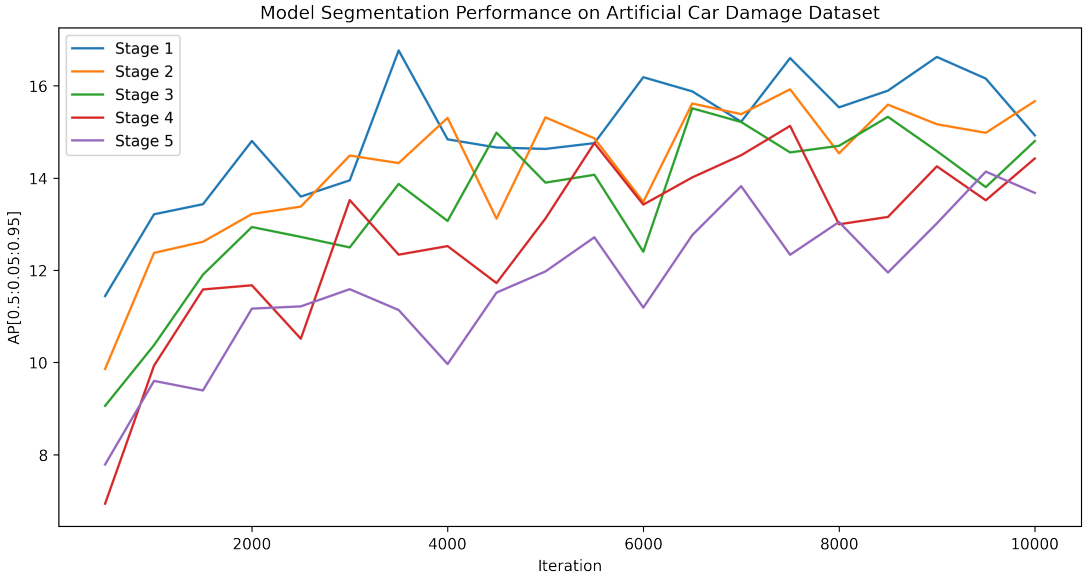

Figure 40: Each model's AP[0.5:0.05:0.95] segmentation performance on the artificial validation dataset

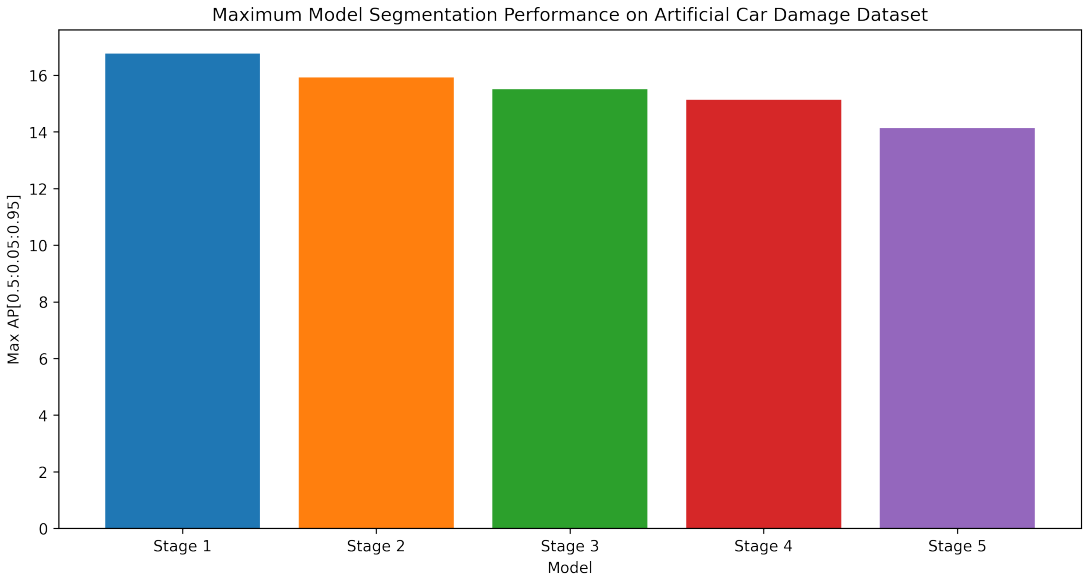

Figure 41: The maximum of each model's AP[0.5:0.05:0.95] segmentation performance on the artificial validation dataset

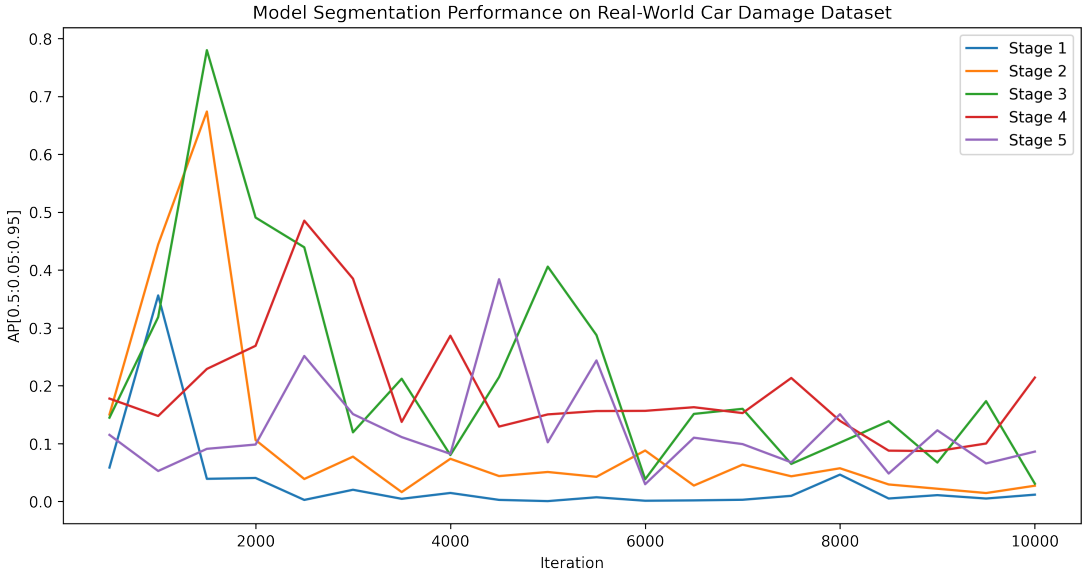

Figure 42: Each model's AP[0.5:0.05:0.95] segmentation performance on the real-world car damage validation dataset

by each of the five models (see Figure 43) shows us that freezing stages 1 through 3 of the ResNet-50 based backbone performed the best out of all the models. This observation implies

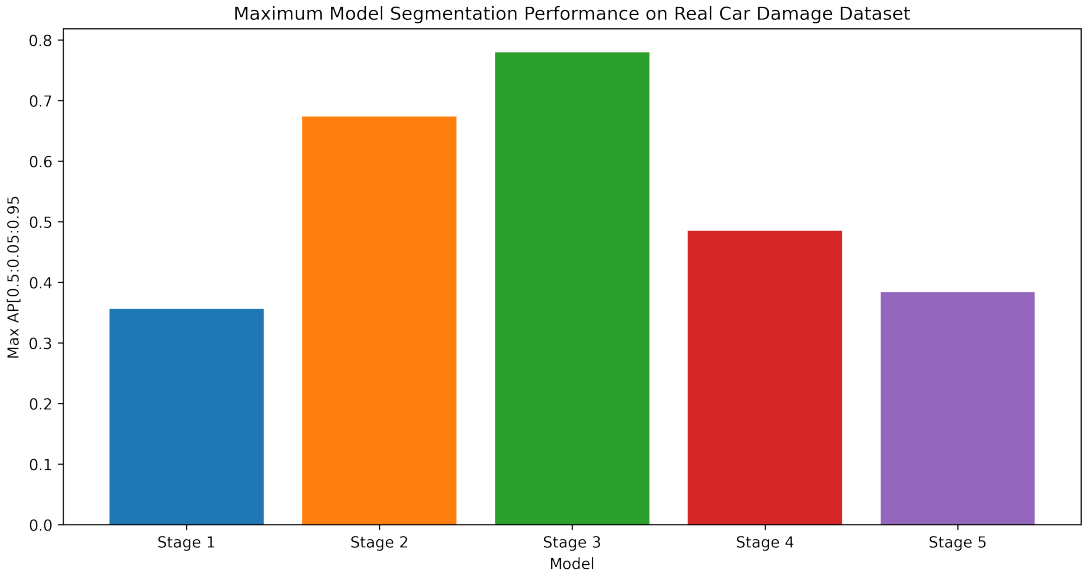

Figure 43: The maximum of each model's AP[0.5:0.05:0.95] segmentation performance on the real-world car damage validation dataset

Comparing the segmentation performance of all models on the real-world car damage dataset to their performance on the artificial validation dataset shows a disparity between the two. Even comparing the Stage 3 model's segmentation performance against the worst performing model on the artificial validation data, Stage 5, there seems to be no significant transfer of performance from the artificial domain to the real-world domain.

4.6.6 Prediction Examples

Figure 44 shows several examples of the final (iteration 10,000) Stage 1 model predictions of the artificial validation data compared to their associated ground truth labels.

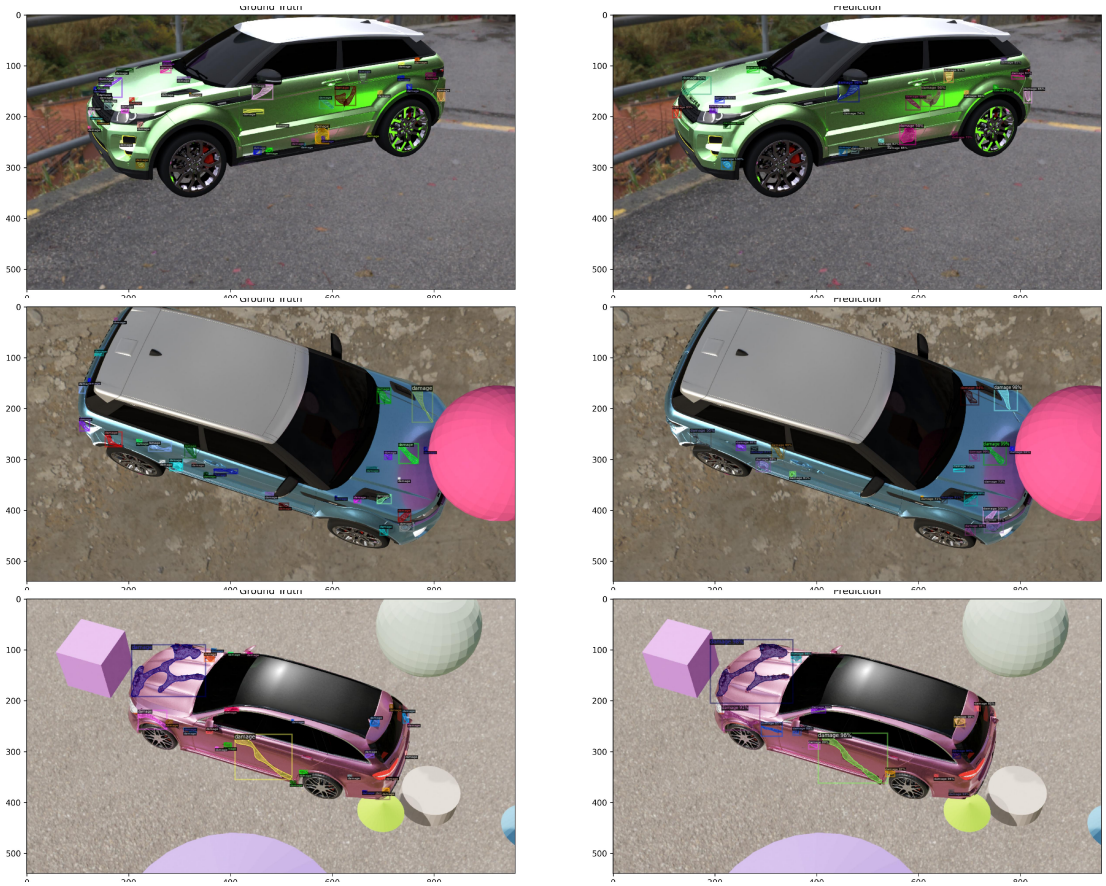

Figure 44: Examples of ground truth references (left column) and final Stage 1 model predictions (right column) of the artificial validation data

Figure 45 shows several examples of the iteration 1500 Stage 3 model predictions of the real-world car damage data compared to their associated ground truth labels.

Figure 45: Examples of ground truth references (left column) and iteration 1500 Stage 3 model predictions (right column) of the real-world car damage data

4.7 Conclusion

Our hypothesis that the minimal unfreezing of the backbone stages before fine-tuning on our artificially generated dataset providing the best segmentation performance on the real-world car damage dataset turned out to be false. Instead, freezing the first three stages of Mask R-CNN's backbone provided the greatest real-world performance, when fine-tuning the model on the artificial dataset.

Although fine-tuning the Mask R-CNN models, which were pre-trained on COCO did not translate to good performance on the "Coco Car Damage Detection Dataset," other research [78, 82, 94, 81] has shown successful implementations of artificial datasets with domain randomization for real-world computer vision problems. We believe the degradation in real-world performance experienced when pre-training a model on both the COCO and artificial datasets is due to not being able to train on the minimal real-world car damage dataset, due to it's small size. Further performance gains should be possible with a larger real-world car damage dataset of several thousand images to allow for further fine-tuning of the model.

Comparing the examples of each dataset given in Figure 36 and Figure 37, there are many obvious differences between the images in each dataset. For one, the artificially generated dataset contains much more damage instances per image than the real-world car damages dataset. Second, the instances of damage in the artificial dataset are typically much smaller than those in the real-world car damage dataset. Also, the distance of the camera from the vehicles are significantly greater in the artificial data than the real-world car damage dataset. These significant differences between the two datasets, among others, could have contributed to the notable performance gap in their evaluation. Further refinement of the artificial data generation process could close this gap and give better performance transfer from artificial to the real-world domain.

4.8 Future Work

Pre-training on COCO and artificial data may provide faster training times; however, the performance boost will be limited given such a small real-world dataset to train and verify on. We simply need more real-world data for training a CNN to aid in damage detection for subjects like cars and aircraft. As an avenue for future research, we suggest collecting an aircraft damage segmentation dataset and analyzing the effects of transfer learning and real-world dataset size on model performance.

Furthermore, to reflect how such a model might be used by maintenance personnel in the Air Force, future research could create a graphical user interface with an integrated damage detection model to notify the user of areas of potential damage areas in aircraft images. A trained user can then verify the segmentation mask and make edits as necessary, where corrections further refine the model through human-in-the-loop [95] online learning [96], increasing future model performance.

V. Conclusions

This chapter recounts the purpose of this thesis as well as summary of the research that was performed. Key results and conclusions from the experiment are reviewed. The chapter ends by suggesting possible avenues for follow on research and final remarks.

5.1 Summary

Our hypothesis that the minimal unfreezing of the backbone stages before fine-tuning on our artificially generated dataset providing the best segmentation performance on the real-world car damage dataset turned out to be false. Instead, freezing the first three stages of Mask R-CNN's backbone provided the greatest real-world performance, when fine-tuning the model on the artificial dataset.

Although fine-tuning the Mask R-CNN models, which were pre-trained on COCO did not translate to good performance on the "Coco Car Damage Detection Dataset," other research [78, 82, 94, 81] has shown successful implementations of artificial datasets with domain randomization for real-world computer vision problems. We believe the degradation in real-world performance experienced when pre-training a model on both the COCO and artificial datasets is due to not being able to train on the minimal real-world car damage dataset, due to it's small size. Further performance gains should be possible with a larger real-world car damage dataset of several thousand images to allow for further fine-tuning of the model.

Comparing the examples of each dataset given in Figure 36 and Figure 37, there are many obvious differences between the images in each dataset. For one, the artificially generated dataset contains much more damage instances per image than the real-world car damages dataset. Second, the instances of damage in the artificial

dataset are typically much smaller than those in the real-world car damage dataset. Also, the distance of the camera from the vehicles are significantly greater in the artificial data than the real-world car damage dataset. These significant differences between the two datasets, among others, could have contributed to the notable performance gap in their evaluation. Further refinement of the artificial data generation process could close this gap and give better performance transfer from artificial to the real-world domain.

5.2 Future Work

- **Collect aircraft damage dataset:** Pre-training on COCO and artificial data may provide faster training times; however, the performance boost will be limited given such a small real-world dataset to train and verify on. We simply need more real-world data for convolutional neural networks (CNNs) to aid in damage detection for subjects like cars and aircraft. As an avenue for future research, we suggest collecting an aircraft damage segmentation dataset and analyzing the effects of transfer learning and real-world dataset size on model performance.

- **Create prototype graphical user interface (GUI) with human-in-the-loop learning:** To reflect how such a model might be used by maintenance personnel in the Air Force, future research could create a GUI with an integrated damage detection model to notify the user of areas of potential damage areas in aircraft images. A trained user can then verify the segmentation mask and make edits as necessary, where corrections further refine the model through human-in-the-loop [95] online learning [96], increasing future model performance.

Bibliography

1. IBM. What are Neural Networks? https://www.ibm.com/cloud/learn/neural-networks, August 2021.

2. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv:1703.06870 [cs]*, January 2018.

3. Hiroto Honda. Digging into Detectron 2. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd, January 2020.

4. Zia Rehman, Muhammad Khan, Fawad Ahmed, Robertas Damasevicius, Syed Naqvi, Muhammad Nisar, and Kashif Javed. Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: An application for smart agriculture. *IET Image Processing*, 15, March 2021.

5. Qing Guo, Xueguang Ma, James Ni, and Yuanxin Wang. Mask RCNN - statwiki, December 2020.

6. Precision and recall. https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=1050491609, October 2021.

7. Richard Gordon. Visual Inspection for Aircraft. Advisory Circular 43-204, Federal Aviation Administration, August 1997.

8. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*, February 2015.

9. Jeremy Norman. The Inspiration for Artificial Neural Networks, Building Blocks of Deep Learning : History of Information, January 2022.

10. Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018:e7068349, February 2018.

11. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524 [cs]*, October 2014.

12. S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, January 1997.

13. Thomas Kautz, Benjamin H. Groh, Julius Hannink, Ulf Jensen, Holger Strubberg, and Bjoern M. Eskofier. Activity recognition in beach volleyball using a Deep Convolutional Neural Network. *Data Mining and Knowledge Discovery*, 31(6):1678–1705, November 2017.

14. Alexander Toshev and Christian Szegedy. DeepPose: Human Pose Estimation via Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, June 2014.

15. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, USA, November 2016.

16. Kunihiko Fukushima and Sei Miyake. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In Shun-ichi Amari and Michael A. Arbib, editors, *Competition and Cooperation in Neural*

Nets, Lecture Notes in Biomathematics, pages 267–285, Berlin, Heidelberg, 1982. Springer.

17. Kunihiko Fukushima. Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, October 1969.

18. Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

19. Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. page 8.

20. Sasank Chilamkurthy. A 2017 Guide to Semantic Segmentation with Deep Learning, July 2017.

21. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

22. Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, April 2019.

23. Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. Facebook's AI Research Lab, November 2021.

24. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*, January 2016.

25. Gabriel de Souza Pereira Moreira. CHAMELEON: A Deep Learning Meta-Architecture for News Recommender Systems [Phd. Thesis]. *arXiv:2001.04831 [cs, stat]*, December 2019.

26. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015.

27. Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. *arXiv:1612.03144 [cs]*, April 2017.

28. Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

29. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

30. David P. Hughes and Marcel Salathe. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv:1511.08060 [cs]*, April 2016.

31. Silvia Liberata Ullo, Amrita Mohan, Alessandro Sebastianelli, Shaik Ejaz Ahamed, Basant Kumar, Ramji Dwivedi, and Ganesh R. Sinha. A New Mask R-CNN-Based Method for Improved Landslide Detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:3799–3810, 2021.

32. Hao Wu, Wenbin Gao, and Xiangrong Xu. Solder Joint Recognition Using Mask R-CNN Method. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 10(3):525–530, March 2020.

33. Farzan Shenavarmasouleh, Farid Ghareh Mohammadi, M. Hadi Amini, Thiab Taha, Khaled Rasheed, and Hamid R. Arabnia. DRDrV3: Complete Lesion Detection in Fundus Images Using Mask R-CNN, Transfer Learning, and LSTM. *arXiv:2108.08095 [cs, eess]*, August 2021.

34. Automatic lung segmentation in CT images using mask R-CNN for mapping the feature extraction in supervised methods of machine learning using transfer learning - IOS Press.

35. A. Broggi, A. Fascioli, P. Grisleri, T. Graf, and M. Meinecke. Model-based validation approaches and matching techniques for automotive vision based pedestrian detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pages 1–1, September 2005.

36. J. Marín, D. Vázquez, D. Gerónimo, and A. M. López. Learning appearance in virtual scenarios for pedestrian detection. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 137–144, June 2010.

37. Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. *arXiv:1406.2227 [cs]*, December 2014.

38. Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. *arXiv:1608.02192 [cs]*, August 2016.

39. Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding Real World Indoor Scenes With Synthetic Data.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4077–4085, 2016.

40. Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *International Journal of Computer Vision*, 126(9):961–972, September 2018.

41. Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, Boston, MA, USA, June 2015. IEEE.

42. DMI Car 3D Models. `https://www.dmi-3d.net/#`, 2022.

43. Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On Pre-Trained Image Features and Synthetic Images for Deep Learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

44. Mark Rice, Liyuan Li, Ying Gu, Marcus Wan, Eng Lim, Gao Feng, Jamie Ng, Melissa Jin-Li, and Shana Babu. Automating the Visual Inspection of Aircraft. February 2018.

45. Julien Miranda, Stanislas Larnier, Ariane Herbulot, and Michel Devy. UAV-based Inspection of Airplane Exterior Screws with Computer Vision:. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 421–427, Prague, Czech Republic, 2019. SCITEPRESS - Science and Technology Publications.

46. Donecle. `https://en.wikipedia.org/w/index.php?title=Donecle&oldid=984869363`, October 2020.

47. Julien Miranda, Jannic Veith, Stanislas Larnier, Ariane Herbulot, and Michel Devy. Machine learning approaches for defect classification on aircraft fuselage images aquired by an UAV. page 10, July 2019.

48. Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *arXiv:1904.05046 [cs]*, March 2020.

49. Anıl Doğru, Soufiane Bouarfa, Ridwan Arizar, and Reyhan Aydogan. Using Convolutional Neural Networks to Automate Aircraft Maintenance Visual Inspection. *Aerospace*, 7, December 2020.

50. Soufiane Bouarfa, Anil Doğru, Ridwan Arizar, Reyhan Aydoğan, and Joselito Serafico. Towards Automated Aircraft Maintenance Inspection. A use case of detecting aircraft dents using Mask R-CNN. In *AIAA Scitech 2020 Forum*, Orlando, FL, January 2020. American Institute of Aeronautics and Astronautics.

51. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs]*, May 2016.

52. Uniting Aviation. The future of MRO: Emerging technologies in aircraft maintenance, August 2019.

53. T. Sattar, S. Chen, B. Bridge, and J. Shang. ROBAIR: Mobile Robotic System to Inspect Aircraft Wings and Fuselage. *undefined*, 2003.

54. Igor Jovancevic, Stanislas Larnier, Jean-José Orteu, and Thierry Sentenac. Automated exterior inspection of an aircraft with a pan-tilt-zoom camera mounted on a mobile robot. *Journal of Electronic Imaging*, 24(6):061110, November 2015.

55. Donecle. Evaluate your aircraft paint quality automatically with Donecle's drone, December 2019.

56. Aircraft Drone Inspections — Mainblades B.V. `https://mainblades.com/`, February 2022.

57. Pandia Rajan Jeyaraj and Edward Rajan Samuel Nadar. Computer-assisted medical image classification for early diagnosis of oral cancer employing deep learning algorithm. *Journal of Cancer Research and Clinical Oncology*, 145(4):829–837, April 2019.

58. Ashnil Kumar, Jinman Kim, David Lyndon, Michael Fulham, and Dagan Feng. An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification. *IEEE Journal of Biomedical and Health Informatics*, 21(1):31–40, January 2017.

59. Samir S. Yadav and Shivajirao M. Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1):113, December 2019.

60. Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. Medical image classification with convolutional neural network. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 844–848, December 2014.

61. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*, May 2015.

62. Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016.

63. D. Vazquez, A. M. Lopez, and D. Ponsa. Unsupervised domain adaptation of virtual and real worlds for pedestrian detection. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3492–3495, November 2012.

64. R. Barth, J. IJsselmuiden, J. Hemming, and E. J. Van Henten. Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset. *Computers and Electronics in Agriculture*, 144:284–296, 2018.

65. German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, Las Vegas, NV, USA, June 2016. IEEE.

66. A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, September 2013.

67. Ton Roosendaal. Blender. Blender Foundation.

68. David Helgason, Nicholas Francis, and Joachim Ante. Unity. Unity Technologies, January 2022.

69. CGTrader - 3D Model Store. `https://www.cgtrader.com/`, 2022.

70. Sketchfab. `https://sketchfab.com/`, February 2022.

71. 3D Models for Professionals :: TurboSquid. `https://www.turbosquid.com/`, February 2022.

72. Patrick Silberberg. Aircraft Inspection by Multirotor UAV Using Coverage Path Planning. Master's thesis, Air Force Institution of Technology, Wright-Patternson Air Force Base, Ohio, March 2021.

73. Balakrishnan Ramalingam, Vega-Heredia Manuel, Mohan Rajesh Elara, Ayyalusami Vengadesh, Anirudh Krishna Lakshmanan, Muhammad Ilyas, and Tan Jun Yuan James. Visual Inspection of the Aircraft Surface Using a Teleoperated Reconfigurable Climbing Robot and Enhanced Deep Learning Technique. *International Journal of Aerospace Engineering*, 2019:1–14, September 2019.

74. C. J. Alberts, C. W. Carroll, W. M. Kaufman, C. J. Perlee, and M. W. Siegel. Automated Aircraft inspection, April 1998.

75. U. S. Government Accountability Office. Weapon System Sustainment: Aircraft Mission Capable Rates Generally Did Not Meet Goals and Cost of Sustaining Selected Weapon Systems Varied Widely. Report to Congressional Requesters GAO-21-101SP, U. S. Government Accountability Office, November 2020.

76. Gaudenz Boesch. Object Detection in 2022: The Definitive Guide, July 2021.

77. Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic Data for Text Localisation in Natural Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

78. Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. *arXiv:1804.06516 [cs]*, April 2018.

79. Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The

Cityscapes Dataset for Semantic Urban Scene Understanding. *arXiv:1604.01685 [cs]*, April 2016.

80. Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape Open Dataset for Autonomous Driving and its Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2702–2719, October 2020.

81. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, September 2017.

82. Rawal Khirodkar, Donghyun Yoo, and Kris M. Kitani. Domain Randomization for Scene-Specific Car Detection and Pose Estimation. *arXiv:1811.05939 [cs]*, November 2018.

83. Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. PyTorch. Facebook's AI Research Lab, October 2021.

84. Jonas Aust, Sam Shankland, Dirk Pons, Ramakrishnan Mukundan, and Antonija Mitrovic. Automated Defect Detection and Decision-Support in Gas Turbine Blade Inspection. *Aerospace*, 8(2):30, February 2021.

85. Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias, and Guowei Cai. Aircraft Inspection Using Unmanned Aerial Vehicles. In *Proceedings of the International Micro Air Vehicle Conference and Competition*, page 7, October 2016.

86. Anil Doğru, Soufiane Bouarfa, Ridwan Arizar, and Reyhan Aydoğan. Using Convolutional Neural Networks to Automate Aircraft Maintenance Visual Inspection. *Aerospace*, 7(12):171, December 2020.

87. Vasileios Tzitzilonis, Konstantinos Malandrakis, Luca Zanotti Fragonara, Jose Angel Gonzalez Domingo, Nicolas P. Avdelidis, Antonios Tsourdos, and Kevin Forster. Inspection of Aircraft Wing Panels Using Unmanned Aerial Vehicles. *Sensors*, 19(8):1824, April 2019.

88. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv:1606.00915 [cs]*, May 2017.

89. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015.

90. Georgios Georgakis, Arsalan Mousavian, Alexander C. Berg, and Jana Kosecka. Synthesizing Training Data for Object Detection in Indoor Scenes. *arXiv:1702.07836 [cs]*, September 2017.

91. Guido van Rossum. Python. Python Software Foundation, January 2022.

92. Blender Foundation. Blender 3.0.1 Python API Documentation. `https://docs.blender.org/api/current/index.html`, January 2022.

93. Lalu Prasad Lenka. Coco Car Damage Detection Dataset, October 2020.

94. Lixin Duan, Dong Xu, and Ivor Tsang. Learning with Augmented Features for Heterogeneous Domain Adaptation. *arXiv:1206.4660 [cs]*, June 2012.

95. Samuel Budd, Emma C. Robinson, and Bernhard Kainz. A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis. *Medical Image Analysis*, 71:102062, July 2021.

96. Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online Learning: A Comprehensive Survey. *arXiv:1802.02871 [cs]*, October 2018.

Acronyms

AFIT Air Force Institute of Technology. 43

AI artificial intelligence. 31

ANN artificial neural network. 16, 17

ANT Autonomy and Navigation Technology. 43

AP average precision. 10, 11, 49, 50, 51

API application programming interface. 33, 34, 35

CAD computer-aided design. 27, 59, 60, 61, 62

CNN convolutional neural network. iv, 3, 17, 20, 25, 27, 29, 30, 31, 32, 34, 40, 42, 43, 44, 52, 53, 54, 59, 60, 78, 80, 1

DR domain randomization. 60

FAA Federal Aviation Administration. 2

FAIR Facebook AI Research. 54, 55

FCRN Fully-Convolutional Regression Network. 59

FN false negative. 9, 46

FP false positive. 9, 10, 46, 49

FPN Feature Pyramidal Network. 22, 24, 55, 56, 58, 66, 68

GUI graphical user interface. 80

HDRI high dynamic range image. ix, 62, 64, 65

IoU intersection over union. 9, 10, 11, 48, 49, 50, 51

mAP mean average precision. 10, 11, 51

PR precision-recall. 10, 50

ReLU rectified linear unit. 17, 54

ROI region of interest. 20, 22, 55, 56

RPN Region Proposal Network. 22, 56

TL transfer learning. 24, 56, 58

TN true negative. 9, 46

TP true positive. 9, 10, 46, 48, 49

UAV unmanned aerial vehicle. iv, 28, 29, 30, 33, 42, 43, 1

YOLO You Only Look Once. 28

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
07–03–2022	Master's Thesis	Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE

Automated Aircraft Visual Inspection with Artificial Data Generation Enabled Deep Learning

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Nathan J. Gaul

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-20-M-028

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RXCA
Building 653
WPAFB OH 45433-7765
DSN 937-255-9798
Email: john.welter.2@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RXCA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Aircraft visual inspection, which is essential to daily maintenance of an aircraft, is expensive and time-consuming to perform. Augmenting trained maintenance technicians with automated UAVs to collect and analyze images for aircraft inspection is an active research topic and a potential application of CNNs. Training datasets for niche research topics such as aircraft visual inspection are small and challenging to produce, and the manual process of labeling these datasets often produces subjective annotations. Recently, researchers have produced several successful applications of artificially generated datasets with domain randomization for training CNNs for real-world computer vision problems. The research outlined herein builds upon this idea to create an artificial data generation pipeline inside Blender and generate an artificial dataset to train an instance-segmentation CNN model for car damage detection. This research then evaluates the real-world performance of several models, each pre-trained on the COCO dataset and fine-tuned on custom generated artificial dataset...

15. SUBJECT TERMS

machine learning, deep learning, automated aircraft inspection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Captain Nathan J. Gaul, AFIT/ENG
U	U	U	UU	95	19b. TELEPHONE NUMBER *(include area code)* (208) 805-0468; nathan.gaul@afit.edu